

RTCA, Inc.
1150 18th Street NW, Suite 910
Washington, DC 20036
USA

**Minimum Operational Performance Standards for
Airborne Collision Avoidance System X (ACAS X)
(ACAS Xa and ACAS Xo)**

Volume I and II

Copies of this document may be obtained from

RTCA, Inc.

Telephone: 202-833-9339

Facsimile: 202-833-9434

Internet: www.rtca.org

Please visit the RTCA Online Store for document pricing and ordering information.

FOREWORD

This document was prepared jointly by Special Committee 147 (SC-147) and EUROCAE Working Group 75(WG-75). It was approved by the RTCA Program Management Committee (PMC) on September 20, 2018 and by the EUROCAE Council on October 2, 2018.

RTCA, Incorporated is a not-for-profit corporation formed to advance the art and science of aviation and aviation electronic systems for the benefit of the public. The organization develops consensus-based recommendations on contemporary aviation issues. RTCA's objectives include but are not limited to:

- coalescing aviation system user and provider technical requirements in a manner that helps government and industry meet their mutual objectives and responsibilities;
- analyzing and recommending solutions to the system technical issues that aviation faces as it continues to pursue increased safety, system capacity and efficiency;
- developing consensus on the application of pertinent technology to fulfill user and provider requirements, including development of minimum operational performance standards for electronic systems and equipment that support aviation; and
- assisting in developing the appropriate technical material upon which positions for the International Civil Aviation Organization and the International Telecommunication Union and other appropriate international organizations can be based.

The organization's recommendations are often used as the basis for government and private sector decisions as well as the foundation for many Federal Aviation Administration Technical Standard Orders and several advisory circulars.

Since RTCA is not an official agency of the United States Government, its recommendations may not be regarded as statements of official government policy unless so enunciated by the U.S. government organization or agency having statutory jurisdiction over any matters to which the recommendations relate.

DISCLAIMER

This publication is based on material submitted by various participants during the SC approval process. Neither the SC nor RTCA has made any determination whether these materials could be subject to valid claims of patent, copyright or other proprietary rights by third parties, and no representation or warranty, expressed or implied is made in this regard. Any use of or reliance on this document shall constitute an acceptance thereof "as is" and be subject to this disclaimer.

This Page Intentionally Left Blank

TABLE OF CONTENTS

1 PURPOSE AND SCOPE	1
1.1 Introduction.....	1
1.1.1 Document Structure.....	1
1.1.2 Operational Goals.....	2
1.1.2.1 Operation of ACAS X on the Airport Surface.....	2
1.1.3 Equipment	3
1.1.3.1 ACAS X Equipment Classes and Article Designations.....	3
1.1.3.2 Aircraft Equipment Information Vulnerabilities.....	5
1.2 Background.....	6
1.2.1 Collision Avoidance Systems.....	6
1.2.1.1 TCAS II.....	6
1.2.1.2 ACAS X.....	7
1.2.1.2.1 ACAS Xa/Xo.....	7
1.2.1.2.2 ACAS Xu	8
1.3 System Overview	8
1.3.1 Surveillance and Tracking.....	8
1.3.2 Threat Resolution	9
1.3.3 Coordinating RAs Against ACAS-Equipped Threats	9
1.3.3.1 Future Coordination Schemes.....	10
1.3.3.2 Multiple Threats, Sense and Intent, and Reversals	11
1.3.4 Control of Interference Caused by Active Surveillance.....	11
1.3.5 Provision of RA Information to Ground Systems	12
1.3.6 Hybrid Surveillance and Extended Hybrid Surveillance.....	12
1.3.6.1 Overview.....	12
1.3.6.2 Description.....	12
1.3.6.2.1 Use When Airborne.....	14
1.3.6.2.2 Use When On The Ground.....	15
1.3.7 ACAS Xo functionality	16
1.4 Assumptions and Limitations	16
1.4.1 Operational	16
1.4.1.1 Airworthiness.....	16
1.4.1.2 Improvements or Alternative Techniques.....	16
1.4.2 Environment	16
1.4.2.1 Transponders.....	17
1.4.2.2 ADS-B Systems	17
1.4.2.3 Integration with ASA Systems.....	17
1.4.2.4 Interoperability with Onboard Aircraft Systems.....	17
1.4.2.5 Ownship Capability	17
1.4.2.5.1 Barometric Altimetry System.....	17
1.4.2.5.2 Mode S Transponder	18
1.4.2.5.3 Ownship Position Source	18
1.4.2.5.4 Aircraft Discretes	18
1.4.2.5.5 Radio Altimeter	18
1.4.2.5.6 Heading	18
1.4.3 Human	19
1.4.3.1 Pilot Actions	19
1.4.3.1.1 Existing TCAS Pilot Responsibilities	19

1.4.3.2	Interaction With ATC	20
1.4.3.3	Use of ACAS X With Other Onboard Traffic Alerting Systems.....	20
1.5	Test Procedures.....	20
1.5.1	Environmental Tests.....	20
1.5.2	Bench Tests	21
1.5.3	Installed System Tests	21
1.5.4	Computer Performance Verification Test	21
1.5.5	Interoperability	21
1.6	Configuration Control, Security and Distribution of the ACAS X Electronic Products.....	22
1.6.1	Configuration Control, Security and Distribution of ACAS X Parameter Data Item Files ...	22
1.6.1.1	Purpose.....	22
1.6.1.2	Overview and Guidelines.....	22
1.6.1.3	PDIF Details	23
1.6.2	Additional Electronic Products	23
1.7	Glossary of Terms.....	23
1.8	Abbreviations.....	31
1.9	References.....	34
2	ACAS X EQUIPMENT REQUIREMENTS AND TEST PROCEDURES.....	37
2.1	General Requirements.....	37
2.1.1	Airworthiness	37
2.1.2	General Performance	37
2.1.3	Federal Communications Commission Rules	37
2.1.4	Self-Test	37
2.1.5	Performance Monitoring	37
2.1.6	Interrogation Test Modes	37
2.1.7	Controls	38
2.1.7.1	Operation of Controls	38
2.1.7.2	Minimum Flight Crew Control Functions	38
2.1.8	Display Functions.....	39
2.1.8.1	Minimum Display Functions	39
2.1.9	Equipment Configuration	39
2.1.10	Mode S Transponder Capabilities	39
2.1.10.1	Performance Compatibility with Ownship ACAS X	39
2.1.11	ACAS X Combined with Other Aircraft Systems.....	40
2.1.12	Aural Annunciation	40
2.1.13	Shared Use of 1090 MHz Receiver with an ADS-B Receiving Subsystem.....	40
2.2	Minimum Performance Standards	40
2.2.1	Definition Of Standard Conditions.....	41
2.2.1.1	Measurement Conventions.....	41
2.2.1.2	Operational Environment.....	41
2.2.1.2.1	Aircraft Density.....	41
2.2.1.2.2	Aircraft Equipage	42
2.2.1.2.3	Interrogator Environment	42
2.2.1.2.3.1	Maximum ATCRBS Fruit Rate	43
2.2.1.2.3.2	Maximum Mode S FRUIT Rate	43
2.2.1.2.3.3	Maximum Co-Site Transmitter Rate.....	43
2.2.1.2.3.4	Intruder Transponder Availability	43

2.2.1.2.3.5	Mode S Transponder Compatibility.....	44
2.2.1.2.4	Maximum Level Of ATCRBS Synchronous Garble Interference.....	44
2.2.1.2.5	Maximum Level Of Multipath Interference	44
2.2.1.2.5.1	Transponder Reply.....	44
2.2.1.2.5.2	ACAS X Interrogation.....	45
2.2.1.3	Target-Of-Interest	45
2.2.1.3.1	Maximum Closing Speed.....	45
2.2.1.3.2	Closing Speed As A Function Of Azimuth	45
2.2.1.3.3	Elevation Angle Relative To ACAS X.....	46
2.2.1.3.4	Altitude Relative To ACAS X.....	46
2.2.1.3.5	Minimum AGL Altitude.....	46
2.2.2	System Performance.....	46
2.2.2.1	Surveillance Range	46
2.2.2.1.1	For Generation Of RAs	46
2.2.2.1.2	For Generation Of TAs.....	47
2.2.2.2	Surveillance Performance Objectives	47
2.2.2.2.1	General Surveillance Performance	47
2.2.2.2.1.1	Probability Of Track	47
2.2.2.2.1.2	False Track Rate	47
2.2.2.2.2	Flight Test Surveillance Performance	48
2.2.2.2.3	Range Accuracy	48
2.2.2.2.4	Bearing Accuracy	48
2.2.3	Compatibility With Other Systems	48
2.2.3.1	Radiated Output Power.....	49
2.2.3.2	Unwanted Output Power.....	49
2.2.3.3	Interrogation Spectrum	49
2.2.3.4	Interrogation Jitter.....	50
2.2.3.5	Transmit Frequency And Tolerance	50
2.2.3.6	Interference Limiting	50
2.2.3.6.1	Interference Limiting Formulas.....	50
2.2.3.6.2	Interference Limiting Procedures	53
2.2.3.6.3	Interrogations From ACAS X On The Ground	54
2.2.3.6.4	Interrogations From ACAS X Above 18,000 Ft Barometric Altitude	55
2.2.3.7	Transmit Pulse Characteristics.....	56
2.2.3.7.1	Mode C Transmissions.....	56
2.2.3.7.2	Mode S Transmissions	57
2.2.3.8	Mode S Message Field Formats.....	58
2.2.3.8.1	Data Blocks	58
2.2.3.8.2	Format Structure.....	59
2.2.3.8.2.1	Bit Numbering and Sequence	59
2.2.3.8.2.2	Fields.....	59
2.2.3.8.2.2.1	Essential Fields	59
2.2.3.8.2.2.2	Mission Fields.....	61
2.2.3.8.2.2.3	Subfields	62
2.2.3.8.3	Field Descriptions	62
2.2.3.8.3.1	Fields and Subfields Defined by Reference B	65
2.2.3.8.3.1.1	AA Address, Announced	65
2.2.3.8.3.1.2	AC Altitude Code	65
2.2.3.8.3.1.3	AQ Acquisition, Special	66
2.2.3.8.3.1.4	BDS: Comm-B Data Selector	66
2.2.3.8.3.1.5	CA Capability	67
2.2.3.8.3.1.6	CC Crosslink Capability	67
2.2.3.8.3.1.7	DR Downlink Request	67
2.2.3.8.3.1.8	FS Flight Status.....	68

2.2.3.8.3.1.9	MA Message, Comm-A	68
2.2.3.8.3.1.10	MB Message, Comm-B	68
2.2.3.8.3.1.11	ME Message, Extended Squitter	68
2.2.3.8.3.1.12	MU Message, Comm-U	69
2.2.3.8.3.1.13	MV Message, Comm-V	69
2.2.3.8.3.1.14	RI Air-to-Air Reply Information	69
2.2.3.8.3.1.15	RL Reply Length	70
2.2.3.8.3.1.16	RR Reply Request	70
2.2.3.8.3.1.17	SD Special Designator	70
2.2.3.8.3.1.18	VS Vertical Status	70
2.2.3.8.3.2	ACAS X Fields and Subfields	71
2.2.3.8.3.2.1	Subparagraph Not Used	71
2.2.3.8.3.2.2	MA Fields Used by ACAS X	71
2.2.3.8.3.2.3	MB Fields Used by ACAS X	71
2.2.3.8.3.2.3.1	Subfields in MB for RA Report	71
2.2.3.8.3.2.3.2	Subfields in MB for Data Link Capability Report	77
2.2.3.8.3.2.3.2.1	Coding of Data Link Capability Report	77
2.2.3.8.3.2.4	MU Fields Used by ACAS X	78
2.2.3.8.3.2.4.1	Subfields in MU for a TCAS Resolution Message	78
2.2.3.8.3.2.4.2	Subfields in MU for a TCAS Broadcast Interrogation Message	82
2.2.3.8.3.2.4.3	Subfields in MU for an RA Broadcast Interrogation Message	82
2.2.3.8.3.2.5	MV Fields Used by ACAS X	83
2.2.3.8.3.2.5.1	Subfields in MV for a Coordination Reply Message	83
2.2.3.8.3.2.6	SL TCAS II Sensitivity Level Report	84
2.2.3.8.3.2.7	Unassigned Coding Space	85
2.2.3.8.3.2.8	ME Fields Used by ACAS X for Passive Surveillance	85
2.2.3.8.3.2.8.1	TYPE Subfield of the ME Field	86
2.2.3.8.3.2.8.2	Altitude from Airborne Position Message	86
2.2.3.8.3.2.8.3	CPR Format from Airborne Position Message	87
2.2.3.8.3.2.8.4	Encoded Latitude from Airborne Position Message	87
2.2.3.8.3.2.8.5	Encoded Longitude from Airborne Position Message	87
2.2.3.8.3.2.8.6	ADS-B Version Number from Aircraft Operational Status Message	87
2.2.3.8.3.2.8.7	Navigational Integrity Category (NIC) from Airborne Position Message	87
2.2.3.8.3.2.8.8	Navigation Accuracy Category for Position (NACp)	88
2.2.3.8.3.2.8.8.1	NACp from Aircraft Operational Status Message	88
2.2.3.8.3.2.8.8.2	NACp from Target State and Status Message	88
2.2.3.8.3.2.8.9	Source Integrity Level (SIL)	88
2.2.3.8.3.2.8.9.1	SIL from Aircraft Operational Status Message	88
2.2.3.8.3.2.8.9.2	SIL from Target State and Status Message	88
2.2.3.8.3.2.8.10	System Design Assurance (SDA)	88
2.2.3.8.3.2.8.11	CPR Format from Surface Position Message	88
2.2.3.8.3.2.8.12	Encoded Latitude from Surface Position Message	89
2.2.3.8.3.2.8.13	Encoded Longitude from Surface Position Message	89
2.2.3.8.3.2.9	ME Fields Used by ACAS X for Air-to-Air Coordination	89
2.2.3.8.3.2.9.1	Subfields in ME for Operational Coordination Message	89
2.2.3.8.3.2.9.2	Subfields in ME for Aircraft Operational Status Message	91
2.2.3.8.3.2.10	ME Fields Used by ACAS X for Communication of RA Information	93
2.2.3.8.3.2.10.1	ADS-B TCAS RA Broadcast	93
2.2.3.9	ACAS X Signal Protocol	94
2.2.3.9.1	Mode C Surveillance Signals	94
2.2.3.9.1.1	ATCRBS-Only All-Call Interrogation	94
2.2.3.9.2	Mode S Surveillance Signals	94
2.2.3.9.2.1	Detection	94
2.2.3.9.2.2	Surveillance Interrogations	95

2.2.3.9.2.2.1	Acquisition.....	95
2.2.3.9.2.2.2	Tracking.....	95
2.2.3.9.2.3	Surveillance Replies	95
2.2.3.9.2.3.1	Acquisition.....	95
2.2.3.9.2.3.2	Tracking.....	95
2.2.3.9.2.4	TCAS Broadcast Interrogations.....	96
2.2.3.9.3	Coordination with Other Aircraft	96
2.2.3.9.3.1	Determining Own Coordination Protocol.....	96
2.2.3.9.3.2	Transmission of Coordination Interrogations to Other Active CAS Aircraft.....	97
2.2.3.9.3.3	Capacity and Handling Requirements for Incoming Coordination Interrogations	98
2.2.3.9.3.4	Coordination Delivery Delay Requirements.....	99
2.2.3.9.3.5	Additional Coordination Topics	100
2.2.3.9.3.5.1	Slave Geometric Reversals	100
2.2.3.9.3.5.2	Coordination Delay When Transitioning from TA-Only Mode to TA/RA Mode	100
2.2.3.9.3.6	Use of Extended Squitter ME Field Information in Air-to-Air Coordination ...	101
2.2.3.9.3.6.1	Coordination Information Received by ACAS X in Incoming ADS-B Messages	101
2.2.3.9.3.6.1.1	Operational Coordination Message.....	101
2.2.3.9.3.6.1.2	Aircraft Operational Status Message	101
2.2.3.9.3.6.2	Coordination Information Composed by ACAS X for Inclusion in Outgoing ADS-B Messages	101
2.2.3.9.3.6.2.1	Operational Coordination Message.....	101
2.2.3.9.4	Communication of RA Information	102
2.2.3.9.5	Communication With Mode S Ground Sensors	103
2.2.3.9.5.1	RA Report.....	103
2.2.3.9.5.2	Data Link Capability Report.....	104
2.2.3.9.5.3	ACAS X Unit Part Number and ACAS X Software Part Number	104
2.2.3.9.5.4	TCAS Sensitivity Level Control.....	104
2.2.3.9.6	Communication With Other Ground Equipment	105
2.2.3.9.6.1	RA Broadcast Interrogations	105
2.2.3.9.7	Mode S Crosslink Capability	105
2.2.3.9.8	Extended Squitter With Aircraft Identification Message	105
2.2.3.10	Compatibility With Own Mode S Transponder.....	105
2.2.3.11	Aircraft Suppression Bus	106
2.2.3.12	Interfaces With Other Systems	107
2.2.3.12.1	Mode S Transponder Related Interfaces	107
2.2.3.12.2	ACAS X Interface with Mode S Transponder	107
2.2.3.12.2.1	General Requirements of the Interface to the Mode S Transponder.....	107
2.2.3.12.2.1.1	ACAS X/Transponder System Capability Determination	107
2.2.3.12.2.1.2	Determination of ACAS X-to-Transponder Write Compatibility	108
2.2.3.12.2.1.3	Determination of Transponder OCM Transmit Capability and Setting of CCCB Value	108
2.2.3.12.2.1.4	Data Reception.....	109
2.2.3.12.2.1.5	Data Integrity	109
2.2.3.12.2.2	Data Provided by the ACAS X Equipment to the Mode S Transponder	110
2.2.3.12.2.2.1	Data Provided to Own Transponder for Use in Special Surveillance Replies (DF=0, 16)	110
2.2.3.12.2.2.2	Data Provided to Own Transponder for Use in Altitude and Identity Surveillance and Comm-B Replies (DF=4, 5, 20, 21)	110
2.2.3.12.2.2.3	Data Provided to Own Transponder for Use in Long Special Surveillance Replies (DF=16)	110

2.2.3.12.2.2.4	Data Provided to Own Transponder for Use in Altitude and Identity Comm-B Replies (DF=20, 21)	110
2.2.3.12.2.2.5	Data Provided to Own Transponder for Use in 1090ES Transmissions	111
2.2.3.12.2.2.5.1	Operational Coordination Message Priority.....	112
2.2.3.12.2.3	Data Received by the ACAS X Equipment from the Mode S Transponder	113
2.2.3.12.2.3.1	Data Received in Long Special Surveillance Interrogations (UF=16) from Other TCAS Aircraft via Own Mode S Transponder	113
2.2.3.12.2.3.2	Data Received in Altitude and Identity Comm-A Interrogations (UF=20, 21) from Mode S Ground Stations via Own Mode S Transponder	113
2.2.3.12.3	ACAS X Operating Mode Control.....	113
2.2.3.12.4	Radio Altimeter Interface.....	114
2.2.3.12.5	Flight Test Recording.....	114
2.2.3.12.6	Position Source Interface.....	114
2.2.3.12.7	Heading Interface	114
2.2.3.12.8	Air/Ground Interface	114
2.2.3.12.9	ASA System Interface	114
2.2.3.12.9.1	Receive Self-Test Status	115
2.2.3.12.9.2	Send Temporal Test Pattern.....	115
2.2.3.12.9.3	Send Self-Test Results.....	115
2.2.3.12.9.4	ASA System Interface for ACAS Xo	115
2.2.4	Surveillance Requirements.....	115
2.2.4.1	Surveillance Update Rate.....	116
2.2.4.2	System Delay	116
2.2.4.3	Differential Channel Delay	116
2.2.4.4	Signal Reception	116
2.2.4.4.1	Receiver Sensitivity And Bandwidth	116
2.2.4.4.1.1	In-Band Acceptance.....	116
2.2.4.4.1.2	Out-of-Band Rejection.....	117
2.2.4.4.2	Reply Detection and Decoding.....	117
2.2.4.4.2.1	Mode C Reply Reception.....	117
2.2.4.4.2.2	Mode S Squitter and Reply Reception.....	119
2.2.4.5	Interference Rejection and Control	122
2.2.4.5.1	Multipath Rejection.....	122
2.2.4.5.1.1	Interrogation Link Interference.....	122
2.2.4.5.1.2	Reply Link Interference	122
2.2.4.5.1.2.1	Narrow Pulse Discrimination.....	122
2.2.4.5.1.2.2	TACAN and DME Discrimination	123
2.2.4.5.2	Narrow Pulse Rejection.....	123
2.2.4.5.3	TACAN and DME Signal Rejection	123
2.2.4.5.4	Control of ATCRBS Synchronous Garble	123
2.2.4.5.4.1	Control of Synchronous Garble by Whisper-Shout	123
2.2.4.5.4.1.1	Minimum Basic Whisper-Shout Sequence	124
2.2.4.5.4.1.2	Higher Capability Whisper-Shout Sequences for Improved Degarbling Performance	124
2.2.4.5.4.1.3	Determination of Whisper-Shout Based on Synchronous Garble.....	130
2.2.4.5.4.1.4	Criteria for Selection of a Specific Whisper-Shout Sequence	131
2.2.4.5.4.1.5	Surveillance in Areas of No ATCRBS Intruder Aircraft	132
2.2.4.5.4.2	Control of Synchronous Garble by Directional Interrogation	133
2.2.4.5.4.2.1	Directional Interrogation Beamwidth Control	133
2.2.4.5.4.2.2	Directional Interrogation Radiated Power	133
2.2.4.5.4.2.3	Degarbling Performance as a Function of Azimuth Resolution	134
2.2.4.5.5	ATCRBS and Mode S Fruit Rejection	134
2.2.4.6	Surveillance Tracking Requirements	134
2.2.4.6.1	Surveillance Target Track Capacity	135

2.2.4.6.1.1	Surveillance Overload.....	135
2.2.4.6.2	Intruder Air/Ground Status Determination.....	135
2.2.4.6.2.1	Estimate of Ground Level.....	136
2.2.4.6.2.2	Determination of Intruder Air-Ground Status.....	136
2.2.4.6.2.2.1	Determination for Newly Initiated Tracks	136
2.2.4.6.2.2.2	Determination for Intruders Equipped with Mode S Transponders.....	136
2.2.4.6.2.2.3	Determination for Intruders Equipped with Mode C (ATCRBS) Transponders	137
2.2.4.6.2.2.3.1	Determination for Intruders Equipped with Mode C (ATCRBS) Transponders and Reporting Pressure Altitude.....	137
2.2.4.6.2.2.3.2	Determination for Intruders Equipped with Mode C (ATCRBS) Transponders and NOT Reporting Pressure Altitude	137
2.2.4.6.2.2.4	Summary of Rules for Determining Air-Ground Status	137
2.2.4.6.3	Determination of Ownship Air-Ground Status.....	138
2.2.4.6.4	Range and Altitude Estimation.....	139
2.2.4.6.4.1	Mode C Targets	139
2.2.4.6.4.1.1	Reply Merging	139
2.2.4.6.4.1.2	Track Initiation	139
2.2.4.6.4.1.3	Maintenance of Established Tracks	140
2.2.4.6.4.1.4	Multipath False Tracks	140
2.2.4.6.4.2	Mode S Targets.....	140
2.2.4.6.4.2.1	Squitter Processing.....	140
2.2.4.6.4.2.2	Acquisition.....	141
2.2.4.6.4.2.2.1	Acquisition Using Passive Position Reports	142
2.2.4.6.4.2.2.1.1	Extended Hybrid Surveillance Quality Requirements.....	143
2.2.4.6.4.2.2.1.2	Establishing an Extended Hybrid Surveillance Track	143
2.2.4.6.4.2.2.1.3	Extended Hybrid Surveillance MTL	143
2.2.4.6.4.2.2.1.4	Determination of Estimated Signal Strength	144
2.2.4.6.4.2.2.2	Acquisition Using Interrogations	144
2.2.4.6.4.2.3	Maintenance of Established Tracks	145
2.2.4.6.4.2.3.1	Maintenance of Tracks Using Passive Surveillance	146
2.2.4.6.4.2.3.1.1	Conditions for Extended Hybrid Surveillance.....	146
2.2.4.6.4.2.3.1.2	Persistence of Extended Hybrid Surveillance	147
2.2.4.6.4.2.3.1.3	Transitions to Extended Hybrid Surveillance.....	147
2.2.4.6.4.2.3.1.4	Conditions for Hybrid Surveillance.....	147
2.2.4.6.4.2.3.1.5	Persistence of Hybrid Surveillance.....	147
2.2.4.6.4.2.3.1.6	Hybrid Surveillance Region	148
2.2.4.6.4.2.3.1.7	Transitions to Hybrid Surveillance	148
2.2.4.6.4.2.3.1.8	Track Updates Using Airborne Position Messages	149
2.2.4.6.4.2.3.1.9	Tracking in the Absence of Airborne Position Messages.....	150
2.2.4.6.4.2.3.1.10	Revalidation.....	151
2.2.4.6.4.2.3.1.11	Error Budget Allocated to ACAS X for Slant Range Validation	153
2.2.4.6.4.2.3.2	Maintenance of Tracks Using Active Surveillance.....	154
2.2.4.6.4.2.3.2.1	Conditions for Active Surveillance of a Track	154
2.2.4.6.4.2.3.2.2	Persistence of Active Surveillance	154
2.2.4.6.4.2.3.2.3	Active Surveillance Region	154
2.2.4.6.4.2.3.2.4	Passive to Active Surveillance Transition	155
2.2.4.6.4.2.3.2.5	Track Updates Using Active Surveillance.....	155
2.2.4.6.4.2.3.2.6	Power Programming	157
2.2.4.6.4.2.3.2.7	Validation of Airborne Position Message Data	158
2.2.4.6.4.2.4	NTA3/NTA6 Range Determination for Active CAS On-Ground Intruders.	159
2.2.4.6.5	Non-Altitude-Reporting Aircraft.....	159
2.2.4.6.6	Bearing Estimation.....	159
2.2.4.6.6.1	General Requirements.....	159

2.2.4.6.6.2	Active Surveillance Bearing Accuracy With Standard Ground Plane.....	160
2.2.4.6.6.2.1	Accuracy, -10 Deg. To +10 Deg. Elevation.....	160
2.2.4.6.6.2.2	Accuracy, greater than 10 Deg. To +20 Deg. Elevation	160
2.2.4.6.6.3	Bearing Accuracy in the Presence of Interference.....	160
2.2.4.6.6.3.1	Mode C Interleaved Replies.....	161
2.2.4.6.6.3.2	Mode C Overlapped Replies	161
2.2.4.6.6.3.3	Mode S Overlapped Replies	161
2.2.4.6.6.4	Bearing Filter Performance.....	161
2.2.4.6.6.4.1	Bearing Filter	161
2.2.4.6.6.5	Dynamic Range and Reply Frequency	162
2.2.4.7	Antenna System	162
2.2.4.7.1	Polarization.....	162
2.2.4.7.2	Radiation Pattern	162
2.2.4.7.2.1	Transmit Radiation Pattern	162
2.2.4.7.2.2	Receive Radiation Pattern	163
2.2.4.7.3	Use of a Directional Antenna for Mode S Interrogations.....	163
2.2.4.7.4	Antenna Selection	164
2.2.4.7.4.1	Squitter Listening.....	164
2.2.4.7.4.2	Interrogations and Replies	164
2.2.4.8	Relationship Between Front-End Surveillance and the STM	164
2.2.5	Surveillance and Tracking Module / Threat Resolution Module	165
2.2.5.1	STM – Surveillance and Tracking Module Overview	165
2.2.5.1.1	STM Tracking	165
2.2.5.1.2	STM Track Source Selection	165
2.2.5.1.3	STM Belief State Outputs for the TRM	167
2.2.5.2	TRM - Threat Resolution Module Overview.....	167
2.2.5.2.1	State Estimation.....	169
2.2.5.2.2	Cost Estimation and Update of Intruder Coordination Data	170
2.2.5.2.2.1	Offline Cost Estimation	171
2.2.5.2.2.2	Online Cost Estimation	171
2.2.5.2.2.3	Individual Cost Estimation	172
2.2.5.2.3	Action Selection	172
2.2.5.2.4	Coordination Selection	172
2.2.5.2.5	Track Threat Assessment	172
2.2.5.2.6	Display Logic Determination	172
2.2.5.2.7	Generate TRM Output.....	173
2.2.5.2.8	ACAS X TRM Processing Modes.....	173
2.2.5.3	General Integration Requirements	177
2.2.5.3.1	GenerateSTMReport	177
2.2.5.3.2	VerticalTRMUpdate	177
2.2.5.4	STM / Surveillance Correlation Requirements	177
2.2.5.5	Combined STM/TRM Input Interfaces.....	179
2.2.5.5.1	ReceiveBaroAltObservation	180
§2.2.7.2.5.1)	181
2.2.5.5.2	ReceiveRadAltObservation	181
2.2.5.5.3	ReceiveHeadingObservation	182
2.2.5.5.4	ReceiveWgs84Observation	182
2.2.5.5.5	ReceiveDiscretes	183
2.2.5.5.6	ReceiveDF0	185
2.2.5.5.7	ReceiveModeCReply	186
2.2.5.5.8	ReceiveModeCReplies	187
2.2.5.5.9	ReceiveStateVectorPosition	188

2.2.5.5.10	ReceiveStateVectorVelocity	190
2.2.5.5.11	ReceiveModeStatus	191
2.2.5.5.12	ReceiveTargetDesignation	192
2.2.5.5.13	ReceiveUF16UDS30	193
2.2.5.5.14	ReceiveCapabilityReport.....	194
2.2.5.6	Combined STM/TRM Output Interfaces	195
2.2.5.6.1	Crew Outputs.....	196
2.2.5.6.1.1	Traffic Display Outputs	196
2.2.5.6.1.1.1	STMReport.display	196
2.2.5.6.1.1.2	TRMReport.display.intruder	197
2.2.5.6.1.2	RA Output (TRMReport.display)	198
2.2.5.6.1.3	ACAS Xo Traffic Designation Information (TRMReport.designation)	199
2.2.5.6.1.3.1	TRMReport.Designation.Intruder	199
2.2.5.6.2	RA Information Transmitted by ACAS X	201
2.2.5.6.2.1	Data for TCAS Resolution Message (TRMReport.coordination)	201
2.2.5.6.2.2	Data for the RA Broadcast Interrogation Message (TRMReport.broadcast)....	203
2.2.5.6.3	Data To Be Provided to Own Transponder	204
2.2.5.6.3.1	SL/RI and DLCR (STMReport.transponder)	204
2.2.5.6.3.2	Coordination Data Provided to the Transponder (TRMReport.coordination) ...	205
2.2.5.6.3.3	Data for the RA Report (TRMReport.ground_msg)	205
2.2.5.6.4	STMReport.trm_input	205
2.2.6	Displays and Controls	205
2.2.6.1	Traffic Displays	206
2.2.6.1.1	Functions	206
2.2.6.1.2	Characteristics	206
2.2.6.1.2.1	General Characteristics	206
2.2.6.1.2.1.1	Ownship Symbol	206
2.2.6.1.2.1.2	Range Rings	206
2.2.6.1.2.1.3	Traffic Symbology	206
2.2.6.1.2.1.4	Off-Scale Symbology	207
2.2.6.1.2.1.5	Altitude Data Tag	207
2.2.6.1.2.1.6	Intruder Vertical Speed Arrow	208
2.2.6.1.2.1.7	Non-Altitude Reporting Intruders	208
2.2.6.1.2.1.8	Intruder Bearing	208
2.2.6.1.2.1.9	Display of Traffic	209
2.2.6.1.2.1.10	Altitude Band for the Display	209
2.2.6.1.2.1.11	ACAS X Operating Mode and Selected Display Range Annunciation	210
2.2.6.1.2.1.12	RA Annunciations	210
2.2.6.1.2.1.13	Lighting Control	210
2.2.6.1.2.1.14	Status and Failure Annunciations	210
2.2.6.1.2.2	Fixed Range Displays	211
2.2.6.1.2.2.1	Display Range	211
2.2.6.1.2.2.2	Range Ring	211
2.2.6.1.2.3	Variable Range Displays	211
2.2.6.1.2.3.1	Display Range	211
2.2.6.1.2.3.2	Range Selection	211
2.2.6.1.2.3.3	Range Rings	211
2.2.6.1.2.4	Part Time (Pop-up) Displays	211
2.2.6.1.2.4.1	Activation	211
2.2.6.1.2.4.2	Information Displayed	211
2.2.6.1.2.4.3	Display Format	212
2.2.6.1.2.4.4	Pilot Selection	212
2.2.6.1.2.5	Full Time Displays	212
2.2.6.1.2.5.1	Display Format	212

2.2.6.1.2.6 Dedicated Displays	212
2.2.6.1.2.6.1 Display Format	212
2.2.6.1.2.6.2 Ownship Symbol and Location.....	212
2.2.6.1.2.7 Shared Displays	212
2.2.6.1.2.7.1 Shared Weather (WX) Radar Displays	212
2.2.6.1.2.7.1.1 Display Modes	212
2.2.6.1.2.7.1.2 Ownship Symbol and Location.....	213
2.2.6.1.2.7.2 Shared Navigation Displays (ND) or Electronic Horizontal Situation Indicator (EHSI)	214
2.2.6.1.2.7.2.1 Display Characteristic	214
2.2.6.1.2.7.2.2 Failure Annunciations	214
2.2.6.1.2.7.2.3 Traffic Display	214
2.2.6.1.2.7.2.4 ACAS X and Weather Information.....	214
2.2.6.1.2.7.2.5 ACAS X and Navigation Information.....	214
2.2.6.1.2.7.2.6 HSI Display Mode	215
2.2.6.1.2.7.2.6.1 Display Characteristics.....	215
2.2.6.1.2.7.2.6.2 Ownship Symbol and Location	216
2.2.6.1.2.7.2.7 EXPANDED Mode.....	216
2.2.6.1.2.7.2.7.1 Display Characteristics.....	216
2.2.6.1.2.7.2.7.2 Display Range and Range Ring.....	216
2.2.6.1.2.7.2.7.3 Ownship Symbol and Location	217
2.2.6.1.2.7.2.8 MAP Display Mode	217
2.2.6.1.2.7.2.8.1 Display Characteristics.....	217
2.2.6.1.2.7.2.8.2 Display Range and Range Rings	217
2.2.6.1.2.7.2.8.3 Display Orientation	217
2.2.6.1.2.7.2.8.4 Ownship Symbol and Location	217
2.2.6.1.2.7.2.9 North-up Mode.....	217
2.2.6.1.2.7.2.9.1 Display Characteristics.....	217
2.2.6.1.2.7.2.9.2 Display Range and Range Rings	217
2.2.6.1.2.7.2.9.3 Ownship Symbol	217
2.2.6.1.2.7.2.9.4 Display Orientation	218
2.2.6.1.2.7.2.9.5 TA or RA Annunciation	218
2.2.6.1.2.7.2.9.6 Selection of Heading Up Display	218
2.2.6.1.2.7.3 Shared EICAS/SYSTEMS Displays	218
2.2.6.1.2.7.3.1 Display Characteristics	218
2.2.6.1.2.7.3.2 Ownship Symbol and Location.....	218
2.2.6.1.2.7.3.3 Traffic Display Inhibit	218
2.2.6.1.2.7.3.4 Restrictions	218
2.2.6.1.2.7.4 TA/RA/VSI Displays	219
2.2.6.1.2.7.4.1 General Requirements.....	219
2.2.6.1.2.7.4.1.1 Format	219
2.2.6.1.2.7.4.1.2 VSI Requirements	219
2.2.6.1.2.7.4.2 Traffic Display Function.....	219
2.2.6.1.2.7.4.2.1 Display Characteristics	219
2.2.6.1.2.7.4.2.2 Ownship Symbol and Location	219
2.2.6.1.2.7.4.2.3 Interference with Vertical Speed Scale	219
2.2.6.2 RA Displays	219
2.2.6.2.1 General	219
2.2.6.2.2 RA/VSI (Round dial VSI)	219
2.2.6.2.2.1 RA Display Characteristics.....	220
2.2.6.2.2.1.1 Red Arcs	220
2.2.6.2.2.1.2 Green Arcs	220
2.2.6.2.2.1.3 Black Arcs.....	220
2.2.6.2.2.1.4 Multi-aircraft Encounters.....	220

2.2.6.2.2.2	VSI Characteristics	221
2.2.6.2.2.3	Lighting Control	221
2.2.6.2.3	RA/VSI (Integrated Tape VSI on a Primary Flight Display [PFD])	221
2.2.6.2.3.1	RA Display Characteristics.....	221
2.2.6.2.3.1.1	Red Zone	221
2.2.6.2.3.1.2	Green Zone	221
2.2.6.2.3.1.3	Black Zones	222
2.2.6.2.3.1.4	Multi-aircraft Encounters.....	222
2.2.6.2.3.2	VSI Characteristics	222
2.2.6.2.3.3	VSI Scale	222
2.2.6.2.3.4	Lighting Control	222
2.2.6.2.4	Pitch Cues on the PFD.....	222
2.2.6.2.4.1	No-Fly Pitch Angles	223
2.2.6.2.4.2	Mode Annunciation	224
2.2.6.2.4.3	Multi-aircraft Encounters.....	224
2.2.6.2.5	Flight Director Guidance.....	224
2.2.6.2.6	Heads-up Display	224
2.2.6.2.6.1	No-Fly Zone.....	225
2.2.6.2.6.2	Mode Annunciation	225
2.2.6.2.6.3	Flight Path Target	225
2.2.6.2.6.4	Multi-aircraft Encounters.....	225
2.2.6.2.6.5	Display Decluttering	225
2.2.6.2.6.6	RA Guidance Availability	226
2.2.6.3	Aural Annunciations	226
2.2.6.3.1	General	226
2.2.6.3.1.1	Announcement Methodology	226
2.2.6.3.1.2	Aural Annunciation Generation.....	226
2.2.6.3.1.3	Aural Annunciation Description and Interruption	226
2.2.6.3.2	Quality	226
2.2.6.3.3	Aural Annunciation Inhibit	226
2.2.6.3.4	TAs	226
2.2.6.3.5	RAs.....	227
2.2.6.4	Visual Alerts	227
2.2.6.4.1	TAs	227
2.2.6.4.2	RAs.....	238
2.2.6.4.3	Message on PFD.....	238
2.2.6.4.4	Visual Alert Inhibit.....	238
2.2.6.5	Controls.....	238
2.2.6.5.1	ACAS X/Mode S Controls.....	238
2.2.6.5.2	Traffic Display Controls.....	239
2.2.6.5.3	Traffic Display Controls on Shared Weather Radar.....	239
2.2.6.5.3.1	WX/ACAS X Mode.....	239
2.2.6.5.3.2	WX-and-Traffic Mode.....	239
2.2.6.5.3.3	Optional Mode Selection	239
2.2.6.5.4	ACAS Xo Controls.....	240
2.2.6.6	Status and Failure Annunciations	240
2.2.6.6.1	General	240
2.2.6.6.2	Traffic Display Annunciations	240
2.2.6.6.3	RA Display	240
2.2.6.7	Display and Control Interfaces	241
2.2.6.7.1	Display and Control Outputs	241
2.2.6.7.2	Display and Control Inputs.....	241
2.2.7	Monitor Requirements.....	252
2.2.7.1	General Requirements.....	252

2.2.7.1.1	Failure Response	252
2.2.7.1.2	Noninterference with Normal Operation.....	252
2.2.7.1.3	Self-Test	252
2.2.7.2	Monitoring of ACAS X Components	252
2.2.7.2.1	Computer Monitoring.....	252
2.2.7.2.2	Coordination Monitoring.....	253
2.2.7.2.3	ACAS X/Transponder Interface Monitoring.....	253
2.2.7.2.4	Display Monitoring	253
2.2.7.2.5	Altitude Input Monitoring	256
2.2.7.2.5.1	Barometric Altitude Monitoring	256
2.2.7.2.5.2	Radio Altitude Monitoring - General.....	256
2.2.7.2.5.2.1	Radio Altimeter Input to the Monitor	256
2.2.7.2.5.2.2	Monitor Inputs to the STM and STM Inputs to the TRM	256
2.2.7.2.5.2.3	Monitor Verification of Radio Altitude Source Prior to Issuance of TAs or RAs	256
2.2.7.2.5.2.4	Radio Altitude Credibility.....	257
2.2.7.2.5.2.4.1	Radio Altitude Credibility Determination Test #1	257
2.2.7.2.5.2.4.2	Radio Altitude Credibility Determination Test #2	258
2.2.7.2.5.2.4.2.1	Radio Altitude Track Initiation.....	258
2.2.7.2.5.2.4.2.2	Operation Subsequent to Radio Altitude Track Initiation	258
2.2.7.2.5.2.5	Radio Altitude Source Failure	258
2.2.7.2.6	ICAO 24-Bit Aircraft Address	259
2.2.7.2.7	Monitoring of Transponder Status	259
2.2.7.2.8	Higher Priority Warning Systems	259
2.2.7.2.9	Monitoring of Ownship Position Source , Ground Speed and Heading Input	260
2.2.7.2.10	Monitoring of ACAS X Transmitter	260
2.2.7.3	Monitor Interfaces.....	260
2.2.7.3.1	Data Provided by the Monitor to Surveillance	260
2.2.7.3.2	Data Provided by the Monitor to the STM	260
2.2.7.3.3	Data Received by the Monitor from the STM.....	261
2.2.7.3.4	Data Received by the Monitor from the Control Panel via the Mode S Transponder	261
2.2.7.3.5	Data Received by the Monitor from Aircraft Discretes	261
2.2.7.3.6	Data Provided by the Monitor to Displays and Controls	262
2.2.7.3.8	Data Received by the Monitor from Ownship Position Source	264
2.2.7.3.9	Data Received by the Monitor from Ownship Heading Source	264
2.2.8	ACAS Xo	264
2.2.8.1	ACAS Xo Overview	264
2.2.8.1.1	Xo Parameters	265
2.2.8.2	Xo Controls and Displays	266
2.2.8.2.1	Xo Traffic Designation/Undesignation	267
2.2.8.2.2	Xo Display.....	268
2.2.8.2.3	ACAS Xo Mode Availability	269
2.2.8.2.3.1	DNA.....	269
2.2.8.2.3.2	CSPO-3000	270
2.2.8.3	Traffic Xo Mode Validity	270
2.2.8.3.1	Traffic Valid for DNA.....	271
2.2.8.3.2	Traffic Valid for CSPO-3000	272
2.2.8.3.3	Traffic Validity Reported to the ASA System	272
2.2.8.4	CAS Logic in Use for Designated/Undesignated Traffic	272
2.2.8.4.1	Switching Logic Modes.....	273
2.2.8.4.2	Multi-threat Encounters with DNA Traffic	274
2.2.8.5	Automatic Undesignation	274
2.2.8.5.1	Dropped Traffic	275

2.2.8.6 Logic Requirements	275
2.2.8.6.1 DNA	275
2.2.8.6.1.1 DNA Coordination.....	275
2.2.8.6.2 CSPO-3000.....	276
2.2.8.7 State Tables.....	276
2.2.8.8 Xo / ASA System Interface.....	283
2.2.8.8.1 Xo Mode Availability.....	283
2.2.8.8.2 Send Traffic Xo Mode Validity.....	283
2.2.8.8.3 Send Traffic Designation Information.....	283
2.2.8.8.4 Receive Xo Traffic Designation/Undesignation Request.....	284
2.3 Equipment Performance - Environmental Conditions	284
2.3.1 Environmental Test Conditions	285
2.4 Equipment Test Procedures	289
2.4.1 Definitions of Terms and Conditions of Test	289
2.4.1.1 Standard Test Signals.....	291
2.4.1.2 Test Equipment Capabilities	292
2.4.1.3 Alternative Mode S Transponder Procedures	299
2.4.1.4 Bench Test Coverage of the TCAS Software	299
2.4.2 Detailed Test Procedures	299
2.4.2.1 Surveillance	299
2.4.2.1.1 Transmitter Characteristics.....	299
2.4.2.1.1.1 Transmit Frequency (§2.2.3.5).....	299
2.4.2.1.1.2 Radiated Output Power (§2.2.3.1)	302
2.4.2.1.1.3 Control of Synchronous Interference by Transmitter Power (§2.2.4.5.4.1)	304
2.4.2.1.1.3.1 Control of Synchronous Garble by Whisper-Shout (§2.2.4.5.4, §2.2.4.5.4.1)	304
2.4.2.1.1.3.2 Determination of Whisper-Shout Sequence (§2.2.4.5.4, §2.2.4.5.4.1.3, §2.2.4.5.4.1.5)	306
2.4.2.1.1.4 ACAS X Transmit Pulse Characteristics (§2.2.3.7, §2.2.4.5.4.2.1).....	308
2.4.2.1.1.5 Interrogation Spectrum (§2.2.3.3).....	313
2.4.2.1.1.6 Unwanted Output Power (§2.2.3.2)	315
2.4.2.1.1.7 Aircraft Suppression Bus (§2.2.3.11)	317
2.4.2.1.1.7.1 Suppression Pulse Supplied by ACAS X.....	317
2.4.2.1.1.7.2 Suppression Pulse Supplied by Other Avionics.....	317
2.4.2.1.1.8 Interrogation Repetition Interval (§2.2.4.1) and Jitter (§2.2.3.4)	317
2.4.2.1.2 Receiver Characteristics (§2.2.4.4.1)	317
2.4.2.1.2.1 In-Band Acceptance.....	320
2.4.2.1.2.1.1 Ability to Operate Over the Frequency Band 1087 to 1093 MHz for ATCRBS and 1089 to 1091 MHz for Mode S Signals (§2.2.4.4.1.1)	320
2.4.2.1.2.1.2 Sensitivity and Dynamic Range at 1090 MHz (§2.2.4.4.1.1)	322
2.4.2.1.2.2 Out-of-Band Rejection (§2.2.4.4.1.2)	323
2.4.2.1.3 Reply Link Interference (§2.2.4.5.1.2)	325
2.4.2.1.3.1 Mode C Reply Reception (§2.2.4.4.2, §2.2.4.4.2.1). Narrow Pulse Discrimination (§2.2.4.5.1.2.1) and TACAN and DME Discrimination (§2.2.4.5.1.2.2)	326
2.4.2.1.3.2 Mode S Squitter and Reply Reception (§2.2.4.4.2.2), Narrow Pulse Discrimination (§2.2.4.5.1.2.1) and TACAN and DME Discrimination (§2.2.4.5.1.2.2)	328
2.4.2.1.4 Mode C Reply Reception (§2.2.4.4.2.1)	330
2.4.2.1.4.1 Bracket Detection and Reply Decoding (§2.2.4.4.2, §2.2.4.4.2.1)	330
2.4.2.1.4.2 Wide Pulse Detection and Pulse Position Discrimination (§2.2.4.4.2, §2.2.4.4.2.1(c))	332
2.4.2.1.4.3 Narrow Pulse Rejection (§2.2.4.4.2.1(b), §2.2.4.5.2)	333

2.4.2.1.4.4	Detection of Garbled Replies (§2.2.4.4.2.1(c)).....	334
2.4.2.1.4.5	Detection of Interleaved Replies (§2.2.4.4.2.1(c)).....	336
2.4.2.1.4.6	Phantom Rejection (§2.2.4.4.2.1(d)).....	337
2.4.2.1.4.7	TACAN and DME Pulse Rejection (§2.2.4.5.3).....	338
2.4.2.1.5	Mode S Squitter and Reply Reception (§2.2.4.4.2.2).....	338
2.4.2.1.5.1	Mode S Preamble Reception (§2.2.4.4.2.2(b))	339
2.4.2.1.5.2	Mode S Squitter and Fruit Reply Reception (§2.2.4.4.2.2(c)).....	341
2.4.2.1.5.3	Mode S Extended Squitter Reception (§2.2.3.9.8, §2.2.3.9.2.1, §2.2.4.6.4.2, §2.2.4.4.2.2(c)).....	342
2.4.2.1.5.4	Mode S Error Correction (§2.2.4.4.2.2(d))	344
2.4.2.1.6	Mode C Target Surveillance Performance (§2.2.4.6.4).....	344
2.4.2.1.6.1	Mode C Surveillance Initiation (§2.2.4.6.4.1.2, §2.2.4.5, §2.2.4.6)	345
2.4.2.1.6.2	Mode C Surveillance Extension (§2.2.4.6.4.1.3, §2.2.4.6.4.1.4).....	346
2.4.2.1.6.2.1	Elimination of Multipath False Tracks	346
2.4.2.1.6.2.2	Range Correlation	349
2.4.2.1.6.2.3	Altitude Correlation	352
2.4.2.1.6.2.4	Altitude Code Processing.....	354
2.4.2.1.6.3	Missing Mode C Replies (§2.2.4.6.4.1.3).....	355
2.4.2.1.6.4	Surveillance Target Capacity (Mode C) (§2.2.4.6.1).....	356
2.4.2.1.6.5	Surveillance Overload (§2.2.4.6.1.1)	360
2.4.2.1.7	Mode S Target Surveillance Performance (§2.2.4.6.4.2, §2.2.4.4.2.2)	361
2.4.2.1.7.1	Mode S Surveillance Initiation (§2.2.4.6.2.2.2, §2.2.4.6.4.2.1).....	362
2.4.2.1.7.2	Mode S Range Acquisition (§2.2.4.6.2.2.2, §2.2.4.6.4.2.2.1, §2.2.4.2, §2.2.4.7, §2.2.4.7.4.1)	366
2.4.2.1.7.3	Maintenance of Established Mode S Tracks (§2.2.4.6.4.2.3.2.5, §2.2.4.7, §2.2.5.5.6)	369
2.4.2.1.7.4	Interference Limiting (§2.2.3.6).....	371
2.4.2.1.7.4.1	Interrogation Control of Airborne ACAS X (§2.2.3.6.1, §2.2.3.6.2, §2.2.3.6.4)	371
2.4.2.1.7.4.2	Interrogation Control of ACAS X On The Ground (§2.2.3.6.3).....	375
2.4.2.1.7.4.3	Correct Content of Transmitted TCAS Broadcast Interrogation Messages (§2.2.3.8.3.2.4.2, §2.2.3.9.2.4)	376
2.4.2.1.7.4.4	Surveillance Special Functionality Test (§2.2.3.6.2)	376
2.4.2.1.7.4.5	Interference Limiting and Required Interrogations	378
2.4.2.1.7.5	Surveillance Target Capacity and Overload (Mode S) (§2.2.4.6.1, §2.2.4.6.1.1)	378
2.4.2.1.7.6	Mode S Power Programming (§2.2.4.6.4.2.3.2.6)	383
2.4.2.1.8	Combined Mode S and Mode C Surveillance	383
2.4.2.1.8.1	Surveillance Target Capacity (§2.2.4.6.1)	383
2.4.2.1.8.2	Altitude and Range Tracking of Mode C and Mode S Targets for TRM (§2.2.4.6)	385
2.4.2.1.8.2.1	Range Tracking Accuracy.....	385
2.4.2.1.8.2.2	Altitude Tracking Accuracy	386
2.4.2.1.9	Bearing Estimation	387
2.4.2.1.9.1	Bearing Accuracy with Standard Ground Plane (§2.2.4.6.6.1, §2.2.4.6.6.2)....	387
2.4.2.1.9.2	Reply Processing (§2.2.4.6.6.3)	389
2.4.2.1.9.2.1	Mode C Interleaved Replies (§2.2.4.6.6.3.1).....	389
2.4.2.1.9.2.2	Mode C Overlapped Replies (§2.2.4.6.6.3.2)	390
2.4.2.1.9.2.3	Mode S Overlapped Replies (§2.2.4.6.6.3.3).....	390
2.4.2.1.9.3	Bearing Filter Performance (§2.2.4.6.6.4)	391
2.4.2.1.9.3.1	Bearing Track and Coast (§2.2.4.6.6.4.1)	391
2.4.2.1.9.3.2	Filter Lag (§2.2.4.6.6.4.1)	392
2.4.2.1.9.4	Radiation Pattern (§2.2.4.7.2, §2.2.4.7.2.1)	392
2.4.2.1.9.5	Mode C Azimuth Filtering (§2.2.4.6.4.1.2)	393

2.4.2.1.10 ACAS X Antenna System	394
2.4.2.1.10.1 Use of Directional Interrogations for Mode C Surveillance (2.2.4.5.4.2.1, §2.2.4.7.2.1) and Bearing Receive Radiation Pattern (§2.2.4.5.4.2.2, §2.2.4.7.2.2) 394	
2.4.2.1.10.2 Use of Directional Antenna for TCAS Broadcast Interrogations (§2.2.4.7.3, §2.2.3.9.2.4)	395
2.4.2.1.10.2.1 Total Radiated Power.....	395
2.4.2.1.10.2.2 Interrogation Repetition Interval and Jitter	396
2.4.2.2.1 Encounter Input Files	398
2.4.2.2.1.1 Rounding.....	400
2.4.2.2.1.2 Timing.....	400
2.4.2.2.2 Encounter Output Files.....	401
2.4.2.2.2.1 STM Report	401
2.4.2.2.2.2 TRM Report.....	403
2.4.2.2.2.3 CostFile.....	405
2.4.2.2.3 Pass / Fail Criteria	407
2.4.2.2.3.1 STM Report	407
2.4.2.2.3.2 TRM Report.....	408
2.4.2.2.3.3 CostFile.....	409
2.4.2.2.4 Prescriptive/Mandatory Vs. Suggestive/Optional Algorithms	409
2.4.2.2.7 Detailed Test Suite Encounter List.....	413
2.4.2.3 External Parameter Selection.....	413
2.4.2.3.1 Ground Control of Sensitivity Level	415
2.4.2.3.2 TA-Only Mode Selection	416
2.4.2.4 Coordination	418
2.4.2.4.1 Transmission of RA Report to Mode S Sensor	418
2.4.2.4.2 Transmission of RA Broadcast Interrogation.....	424
2.4.2.4.3 Coordination with Active CAS	426
2.4.2.4.3.1 Sense Selection and Communication (§2.2.3.9.3.1, §2.2.3.9.3.2)	426
2.4.2.4.3.2 Validity Check for ACAS X Communications Link with Another Active CAS	440
2.4.2.4.3.3 ACAS X in Multi-Aircraft Conflict (§2.2.3.8.3.2.3.1, §2.2.3.9.3.2)	441
2.4.2.4.3.4 Coordination Monitor (§2.2.3.8.3.2.4.1, §2.2.7.2.2).....	444
2.4.2.4.3.5 Transponder to ACAS X Interface and Transponder/ACAS X Throughput (§2.2.3.12.2.1.4, §2.2.3.9.3.3).....	446
2.4.2.4.3.6 Coordination Timing (§2.2.3.9.3.4)	451
2.4.2.4.3.7 Use of Received Collision Avoidance Coordination Capability Bits (CCCB) (§2.2.3.9.3.1, §2.2.3.9.3.6.1.2, §2.2.3.9.3.6.2.1).....	452
2.4.2.4.3.8 Transmission of Own Collision Avoidance Coordination Capability Bits (CCCB) (§2.2.3.12.2.1.3).....	463
2.4.2.5 ACAS X Capability and Part Number Reporting (§2.2.3.9.5.3, §2.2.3.12.2.2.4, §2.2.3.12.2.2.1, §2.2.5.5.5)	465
2.4.2.6 Transmission of Low-level Descend Inhibit (LDI) Information in RF Messages (§2.2.3.8.3.2.3.1).....	471
2.4.2.7 System Integration Tests.....	473
2.4.2.8 Display and Controls Stand Alone Tests	475
2.4.2.8.1 Traffic Displays	476
2.4.2.8.1.1 General Characteristics	476
2.4.2.8.1.1.1 Ownship Symbol (§2.2.6.1.2.1, §2.2.6.1.2.6.2, §2.2.6.1.2.7.1.2, §2.2.6.1.2.7.2.6.2, §2.2.6.1.2.7.2.7.3, §2.2.6.1.2.7.2.8.4, §2.2.6.1.2.7.2.9.3, §2.2.6.1.2.7.3.2, §2.2.6.1.2.7.4.2.2)	476

2.4.2.8.1.1.2	Range Rings (§2.2.6.1.2.1.2, §2.2.6.1.2.2.2, §2.2.6.1.2.3.3, §2.2.6.1.2.7.2.7.2, §2.2.6.1.2.7.2.8.2, §2.2.6.1.2.7.2.9.2)	477
2.4.2.8.1.1.3	Threat Symbology (§2.2.6.1.2.1.3, §2.2.6.1.2.1.12)	478
2.4.2.8.1.1.4	Off-Scale Symbology (§2.2.6.1.2.1.4, §2.2.6.1.2.1.11, §2.2.6.1.2.1.12)	478
2.4.2.8.1.1.5	Altitude Data Tag (§2.2.6.1.2.1.5, §2.2.6.1.2.1.7)	479
2.4.2.8.1.1.5.1	Display of Intruder's Relative Altitude	479
2.4.2.8.1.1.5.2	Display of Intruder's Reported Altitude	479
2.4.2.8.1.1.6	Intruder Vertical Speed Arrow (§2.2.4.6.5, §2.2.6.1.2.1.6)	481
2.4.2.8.1.1.7	No-Bearing Advisories (§2.2.4.6.6.1, §2.2.6.1.2.1.8, §2.2.6.1.2.1.12)	482
2.4.2.8.1.1.8	Display of Traffic (§2.2.6.1.2.1.9)	483
2.4.2.8.1.1.9	Altitude Band for the Display (§2.2.6.1.2.1.10, §2.2.6.5.2)	483
2.4.2.8.1.1.10	Display of ACAS X Operating Mode (§2.2.6.1.2.1.11, §2.2.6.6.2, §2.2.6.1.2.1.14)	485
2.4.2.8.1.2	Fixed Range Displays	485
2.4.2.8.1.3	Variable Range Displays	486
2.4.2.8.1.3.1	Display Range (§2.2.6.1.2.3.1)	486
2.4.2.8.1.3.2	Range Selection (§2.2.6.1.2.3.2)	486
2.4.2.8.1.4	Part Time Displays	487
2.4.2.8.1.4.1	Activation and Information Displayed (§2.2.6.1.2.4.1, §2.2.6.1.2.4.2)	487
2.4.2.8.1.5	Full Time Displays	488
2.4.2.8.1.6	Dedicated Displays	488
2.4.2.8.1.7	Shared Displays	488
2.4.2.8.1.7.1	Shared Weather Radar Displays	488
2.4.2.8.1.7.1.1	Available Display Modes (§2.2.6.1.2.7.1.1)	488
2.4.2.8.1.7.1.2	Display Mode Characteristics (§2.2.6.1.2.7.1.1)	489
2.4.2.8.1.7.1.2.1	WX/ACAS X Mode	489
2.4.2.8.1.7.1.2.2	WX-and-Traffic Mode	490
2.4.2.8.1.7.1.2.3	WX-Only Mode	492
2.4.2.8.1.7.1.2.4	Traffic-Only Mode	493
2.4.2.8.1.7.2	Navigation Displays or Electronic Horizontal Situation Indicators	493
2.4.2.8.1.7.2.1	Failure Annunciations (§2.2.6.1.2.7.2.2)	493
2.4.2.8.1.7.2.2	Display of Traffic (§2.2.6.1.2.7.2.3)	494
2.4.2.8.1.7.2.3	ACAS X and Weather Information (§2.2.6.1.2.7.2.4)	494
2.4.2.8.1.7.2.4	ACAS X and Navigation Information (§2.2.6.1.2.7.2.5)	495
2.4.2.8.1.7.2.5	HSI Display Mode (§2.2.6.1.2.7.2.6.1)	495
2.4.2.8.1.7.2.6	EXPANDED Mode (§2.2.6.1.2.7.2.7.1)	496
2.4.2.8.1.7.2.7	MAP Mode	497
2.4.2.8.1.7.2.7.1	Display Characteristics (§2.2.6.1.2.7.2.8.1)	497
2.4.2.8.1.7.2.7.2	Display Range and Orientation (§2.2.6.1.2.7.2.8.2; §2.2.6.1.2.7.2.8.3)	498
2.4.2.8.1.7.2.8	North-up Displays	499
2.4.2.8.1.7.2.8.1	Display Characteristics (§2.2.6.1.2.7.2.9.1, §2.2.6.1.2.7.2.9.3, §2.2.6.1.2.7.2.9.4, §2.2.6.1.2.7.2.9.5)	499
2.4.2.8.1.7.2.8.2	Display Orientation (§2.2.6.1.2.7.2.9.6)	500
2.4.2.8.1.7.3	EICAS or Systems Displays (§2.2.6.1.2.7.3.3)	501
2.4.2.8.1.7.4	TA/RA/VSI	502
2.4.2.8.1.7.4.1	VSI Requirements (§2.2.6.1.2.7.4.1.2)	502
2.4.2.8.1.7.4.2	Interference with Vertical Speed Scale (§2.2.6.1.2.7.4.2.3)	503
2.4.2.8.2	RA Displays	503
2.4.2.8.2.1	Round Dial VSI (§2.2.6.2.2, §2.2.6.2.2.1.1, §2.2.6.2.2.1.2, §2.2.6.2.2.1.3, §2.2.6.2.2.1.4)	503
2.4.2.8.2.2	Vertical Speed Tape (§2.2.6.2.3, §2.2.6.2.3.1.1, §2.2.6.2.3.1.2, §2.2.6.2.3.1.3, §2.2.6.2.3.1.4)	507
2.4.2.8.2.2.1	Vertical Speed Tape Range (§2.2.6.2.3)	511

2.4.2.8.2.3	Pitch Cues on the PFD (§2.2.6.2.4.1, §2.2.6.2.4.2, §2.2.6.2.4.3).....	511
2.4.2.8.2.4	Flight Director Guidance (§3.2.13.2).....	516
2.4.2.8.2.5	HUD (§2.2.6.2.6.1; §2.2.6.2.6.2; §2.2.6.2.6.3; §2.2.6.2.6.4; §2.2.6.2.6.5; §2.2.6.2.6.6)	518
2.4.2.8.3	Controls	523
2.4.2.8.3.1	ACAS X/Mode S Controls (§2.2.6.5.1).....	523
2.4.2.8.3.2	Traffic Display Controls (§2.2.6.5.2).....	524
2.4.2.8.3.3	Shared Weather Radar/Traffic Display Controls (§2.2.6.5.3, §2.2.6.5.3.1, §2.2.6.5.3.2, §2.2.6.5.3.3)	525
2.4.2.8.4	Status and Failure Annunciations (§2.2.6.6)	525
2.4.2.8.4.1	Traffic Display Annunciations (§2.2.6.6.1).....	525
2.4.2.8.4.2	RA Display Annunciations (§2.2.6.6.2).....	525
2.4.2.9	Automatic Performance Monitoring and Self Test (§2.2.7)	526
2.4.2.9.1	Automatic Performance Monitoring (§2.2.7.1.1).....	526
2.4.2.9.2	Self-Test (§2.2.7.1.3).....	527
2.4.2.9.3	Own Transponder ICAO 24-Bit Aircraft Address (§2.2.7.2.6).....	527
2.4.2.10	Performance Compatibility with Ownship's Mode S Transponder	527
2.4.2.10.1	ACAS X to Mode S Transponder Interference (§2.1.10.1).....	527
2.4.2.10.2	Mode S Transponder to ACAS X Interference (§2.2.3.10).....	527
2.4.2.11	Tests Related to Passive Surveillance.....	528
2.4.2.11.1	General Scenario Description.....	528
2.4.2.11.1.1	ACAS X (Own) Aircraft.....	528
2.4.2.11.1.2	Intruder Aircraft.....	528
2.4.2.11.1.3	Test Success Criteria.....	531
2.4.2.11.2	Verification of Shared Use of 1090 MHz Receiver with an ADS-B Receiving Subsystem (§2.1.13).....	532
2.4.2.11.3	Verification of Acquisition and Maintenance of Established Tracks Using Active Surveillance (§2.2.4.6.4.2.3.2)	532
2.4.2.11.4	Verification of Maintenance of Established Tracks Using Passive Surveillance (§2.2.4.6.4.2.3.1)	537
2.4.2.11.5	Verification of Requirements Related to Transitions Between Passive and Active Surveillance.....	558
2.4.2.11.6	Verification of Error Budget in Computing Slant Range from Passive Data (§2.2.4.6.4.2.3.1.11)	574
2.4.2.11.7	Surveillance Overload and Capacity Tests (§2.2.4.6.4.2.3.2.5)	574
2.4.2.11.8	Verification of DF17 Decoding (§2.2.3.8.3.2.8.1, §2.2.3.8.3.2.8.3, §2.2.3.8.3.2.8.4, §2.2.3.8.3.2.8.5).....	575
2.4.2.11.9	Verification of Monitoring Requirements (§2.2.7.2)	579
2.4.2.11.10	Verification of On-Ground ACAS X Range Determination Using ADS-B (§2.2.4.6.4.2.4)	580
2.4.2.11.11	Verification of DF=18 ADS-B and ADS-R Only Passive Surveillance (§2.2.3.8.3.2.8, §2.2.5.5.4, §2.2.5.5.9, §2.2.5.5.10).....	582
2.4.2.11.12	Verification of Intruder ADS-B Reports to STM.....	587
2.4.2.12	ACAS Xo.....	590
2.4.3	Computer Performance Verification	590
2.4.3.1	RAM Pattern Tests.....	590
2.4.3.2	CPU Instruction Tests	590
2.4.3.3	Program Memory Tests	591
2.4.3.4	CPU Input/Output Tests.....	591
2.4.3.5	CPU Timing Tests	591
2.4.4	Cross-Reference of Requirements and Associated Tests	591

3 INSTALLED EQUIPMENT PERFORMANCE.....621

3.1 Test Conditions	621
3.1.1 Power Input	621
3.1.2 Associated Equipment	621
3.1.3 Environment	621
3.1.4 Adjustment of Equipment	621
3.2 Equipment Installation	621
3.2.1 Equipment Accessibility.....	621
3.2.2 Display Visibility	621
3.2.3 Interference.....	621
3.2.4 Physical Installation	621
3.2.5 Aircraft Power Source	622
3.2.6 Transmission Lines.....	622
3.2.7 Antenna Location	622
3.2.7.1 Minimum Distance from Other Antennas.....	622
3.2.8 Altimetry System.....	622
3.2.8.1 Barometric Altimetry Performance Required for ACAS X Operation	623
3.2.8.2 Altimetry System Error Assessment.....	624
3.2.8.3 Definitions	625
3.2.9 Number of Intruders Displayed (§2.2.6.1.2.1.9)	626
3.2.10 Display of Traffic on EICAS/SYSTEMS Displays (§2.2.6.1.2.7.3.3)	626
3.2.11 Use of Flight Director Cues for RA Guidance (§3.2.13.2).....	626
3.2.12 Red Visual Alert (§2.2.6.4.2)	626
3.2.13 Integration with Aircraft Flight Controls	626
3.2.13.1 Pitch Cues on PFD and HUD.....	626
3.2.13.2 Flight Director.....	626
3.2.13.3 Autopilot TCAS Mode to Perform RA Maneuvers	627
3.3 Minimum Installed Equipment Performance Requirements.....	627
3.4 Test Procedures for Installed Equipment Performance.....	627
3.4.1 Conformity Inspection.....	627
3.4.2 General Test Procedures.....	628
3.4.2.1 Equipment Function.....	628
3.4.2.2 Interference Effects (Ground Test)	628
3.4.2.2.1 Suppression Bus	628
3.4.2.3 Accessibility.....	628
3.4.2.4 Integration with Aircraft Flight Controls	628
3.4.2.4.1 Pitch Cues (§3.2.13.1)	628
3.4.2.4.2 Flight Director (§3.2.13.2)	629
3.4.2.4.3 Auto Pilot (§3.2.13.3).....	629
3.4.3 Antenna Gain Performance	629
3.4.3.1 Success Criteria.....	629
3.4.3.2 Full-Scale Anechoic Antenna Range Measurements of Gain.....	629
3.4.3.3 Scaled Model Measurements of Gain	630
3.4.3.3.1 Aircraft Model	630
3.4.3.3.2 Model Tests	630
3.4.3.4 Theoretical Calculations of Antenna Gain.....	631
3.4.3.4.1 Validation of Theoretical Calculations.....	631
3.4.3.5 Antenna Location Using Distance-Area Calculations	631
3.4.4 Certification Flight Test Procedures.....	631
3.4.4.1 Mode C Surveillance Flight Tests.....	632

3.4.4.2	Mode S Surveillance Flight Tests	634
3.4.4.3	Coordination Flight Tests	636
3.4.4.4	Passive Surveillance Flight Tests.....	636
3.4.5	Bearing Estimation Tests.....	638
3.4.5.1	Success Criteria for Bearing Estimation Measurements.....	638
3.4.5.2	Antenna Range Measurements of Bearing Accuracy	638
3.4.5.3	Airborne Measurements of Bearing Accuracy.....	640
3.4.5.4	Scaled Model Measurements of Bearing Accuracy	640
3.4.5.4.1	Aircraft Model	640
3.4.5.4.2	Antenna Model	640
3.4.5.4.3	Model Tests.....	640
3.4.5.5	Theoretical Calculations of Bearing Accuracy	641
3.4.5.5.1	Validation of Theoretical Calculations of Bearing Accuracy	641
4	MEMBERSHIP	643
APPENDIX A SURVEILLANCE PROCEDURES		A-1
A.1	Interference Limiting Procedure	A-1
A.2	Rejection of Reply Link Interference	A-3
A.3	Mode S Error Correction	A-4
A.4	Determination of Synchronous Garble Potential.....	A-10
A.5	Mode C Track Initiation	A-17
A.6	Mode C Track Maintenance	A-18
A.7	Multipath False Targets.....	A-20
A.8	Mode S Squitter Processing.....	A-21
A.9	Non-Altitude Reporting Aircraft.....	A-21
A.10	Bearing Estimation.....	A-24
A.11	Mode S Tracker Using a Five-Second Update Rate	A-25
A.12	Transition from On-The-Ground to Airborne Interference Limiting.....	A-26
APPENDIX B ACAS X RF COMMUNICATION PATHS		B-1
B.1	Introduction	B-1
B.2	Air-to-Air Communication	B-1
B.2.1	Coordination	B-1

B.2.2	TCAS Broadcast Interrogation Messages	B-1
B.2.3	RA Broadcast Interrogation Messages.....	B-3
B.2.4	Surveillance	B-3
B.3	Air-to-Ground, Ground-to-Air Communication.....	B-3
B.3.1	Sensitivity Level Command Messages (Ground-to-Air)	B-3
B.3.2	RA Report (Air-to-Ground)	B-3
B.3.3	Data Link Capability Report (Air-to-Ground)	B-4
B.3.4	Surveillance	B-4

APPENDIX C DEGRADED SURVEILLANCE.....C-1

C.1	Introduction	C-1
C.2	Analysis Overview	C-1
C.3	Analysis of Degraded Surveillance Inputs	C-1
C.3.1	ADS-B	C-2
C.3.1.1	Update Rate	C-2
C.3.1.1.1	Gating	C-2
C.3.1.1.2	Environment	C-2
C.3.1.2	Quality	C-3
C.3.1.2.1	NACp	C-3
C.3.1.2.2	NACv	C-4
C.3.2	Active.....	C-5
C.3.2.1	Bearingless	C-5
C.3.2.2	Sensitivity to Stressing Bearing Error Models	C-5
C.3.2.3	Active Update Delay	C-6
C.3.2.4	Quantization Deception	C-6
C.3.3	Ownship	C-7
C.3.3.1	Ownship Heading	C-7
C.3.3.1.1	Headingless	C-7

C.3.3.1.2	Track Angle.....	C-8
C.3.3.2	Ownship Altitude Quantization.....	C-9
C.3.4	TA Alerting Region	C-10
C.4	Summary	C-10

APPENDIX D CONVERSION OF REPORTED POSITION TO SLANT RANGE.....D-1

D.1	Overview	D-1
D.2	Exact Conversion Equations.....	D-1
D.3	Height Above Ellipsoid vs. Barometric Altitude.....	D-2
D.4	Spherical Earth Approximation.....	D-3
D.5	Approximation Assuming N is the Same for All Aircraft.....	D-4
D.6	Ellipsoidal Differential Approximation.....	D-6
D.7	Spherical Differential Approximation.....	D-9

APPENDIX E ANALYSIS OF VALIDATION/REVALIDATION RANGE TOLERANCE ERROR BUDGET.....E-1

E.1	Introduction	E-1
E.2	ACAS X Active Surveillance Range Error	E-2
E.3	Own and Intruder Position Sensor Error	E-2
E.4	Intruder Message LSB	E-2
E.5	Uncompensated Latency in Intruder Position When Received by ACAS X.....	E-2
E.6	Uncompensated Latency in Ownship Passive Position When Received by ACAS X	E-3
E.7	Error Budget Allocated For Hybrid Surveillance Data Processing.....	E-3
E.8	Sample Allocation of Hybrid Data Processing Errors.....	E-3

APPENDIX F TEST SUITE ENCOUNTER LIST.....F-1

APPENDIX G VARIABLE VALIDATION INTERVALS.....G-1

G.1	Derivation of the Revalidation Time Interval	G-1
-----	--	-----

APPENDIX H 1090 MHz SPECTRUM REDUCTION ANALYSISH-1

H.2 Simulation and Input Data Set.....	H-1
H.3 Evaluate Algorithm Changes.....	H-2
H.3.1 Algorithm Changes to RTCA/DO-185B.....	H-2
H.3.1.1 Reduced Surveillance Volume when Operating on Airport Surface.....	H-2
H.3.1.2 Eliminating Range Monitoring Interrogations to On-Ground TCAS Aircraft when Own TCAS is on the Ground	H-2
H.3.2 Algorithm Changes to RTCA/DO-300	H-3
H.3.2.1 Passively Monitor TCAS Aircraft when Below 2000 ft AGL	H-3
H.3.2.2 Two Validation Attempts	H-3
H.3.2.3 Variable Validation Rate	H-3
H.3.2.4 Use of Short Replies for Validation	H-4
H.3.2.5 Extended Hybrid Surveillance.....	H-4
H.3.2.6 Passive Only Surveillance on the Ground	H-5
H.4 Results	H-5

APPENDIX I ACAS X EQUIPMENT CLASS REQUIREMENTS COMPLIANCE MATRIX ... I-1

I.1 ACAS X Equipment Compliance Matrix.....	I-1
---	-----

**APPENDIX J PDIF LIFE CYCLE PROCESS FLOW AND THE ROLES AND
RESPONSIBILITIES OF THE STAKEHOLDERS.....J-1**

J.1 PDIF Life Cycle Process Flow and the Roles and Responsibilities of the Stakeholders	J-1
---	-----

TABLE OF FIGURES

Figure 1-1: Transition from Passive to Active Surveillance as a Function of Collision Potential.....	13
Figure 1-2: Extended Hybrid Surveillance State Transition Diagram (Ownship Taking Off / Airborne) ..	14
Figure 1-3: Extended Hybrid Surveillance State Transition Diagram (Ownship Operating on Surface) ...	15
Figure 2-1: Mode C-Only All-Call Interrogation Pulse Sequence for ACAS X	56
Figure 2-2: Mode S Interrogation Pulse Sequence For ACAS X	58
Figure 2-3: Mode S Interrogation or Uplink Formats Used by ACAS X	60
Figure 2-4: Mode S Reply or Downlink Formats Used by ACAS X.....	61
Figure 2-5: RF Transmissions Containing RA Information	103
Figure 2-6: Mode C Reply	118
Figure 2-7: Mode S Reply.....	120
Figure 2-8: Minimum Basic Whisper-Shout Sequence	126
Figure 2-9: Timing For Six Power Steps In The Minimum Basic Whisper-Shout Sequence For Top Antenna Forward Beam	127
Figure 2-10: High Resolution Whisper-Shout Sequence	129
Figure 2-11: Surveillance Region State Transition Diagram.....	146
Figure 2-12: Example Coast Timeline	157
Figure 2-13: The Main Blocks of TRM	167
Figure 2-14: State Estimation	168
Figure 2-15: Transformation of Inputs.....	169
Figure 2-16: Estimation of Tau Distribution – Horizontal Entry Table.....	170
Figure 2-18: Offline Cost Estimation – Cost Table Look-up	171
Figure 2-19: Sample Navigation Display Modes.....	216
Figure 2-20: RA Pitch Cues on a PFD	223
Figure 2-21: ASA / ACAS X System Interface	267
Figure 2-22: Example CDTI display with designated traffic symbology and associated data block indicating ACAS Xo mode	269
Figure 2-23: Example state chart for one ACAS Xo mode showing CAS logic and validity for a traffic	271
Figure 2-24: Permitted/Prohibited Xo Mode Changes during RA.....	273
Figure 2-25: Signal Relationship Between ACAS X, Mode S & Test Equipment	293
Figure 2-26: Definition Of Variables Associated With The Linear Movement Of Targets In The Range Dimension.....	297
Figure 2-27: Frequency Test Setup	301
Figure 2-28: Peak Power Test Setup	303
Figure 2-29: Whisper-Shout And Interrogation Rate Jitter Test Setup.....	305
Figure 2-30: Mode S Transmission Test Setup.....	311
Figure 2-31: DPSK Phase Reversal Tolerance Test Setup	312
Figure 2-32: Mode S RF Spectrum Test Setup	314
Figure 2-33: Unwanted Output Power Test Setup	316
Figure 2-34: Geometrical Illustration Of Multipath.....	347
Figure 2-35: Real Multipath.....	347
Figure 2-36: Simulated Multipath	348
Figure 2-37: Reply Sequence For Range Correlation Test Scenario A (Above 10 Kft).....	350
Figure 2-38: Reply Sequence For Range Correlation Test Scenario B (Below 10 Kft)	351
Figure 2-39: Reply Sequence For Altitude Correlation Test	353
Figure 2-40: Range Vs. Time Plot For Intruders 1, 2, 23, And 24	360
Figure 3-1: Altimetry System Elements and Error Components	623
Figure 3-2: Ramp Test Geometry Considerations	639
Figure A-1: Interference Limiting Flow Diagram.	A-2
Figure A-2: Initial Conditions For Error Detection.	A-6
Figure A-3: Final Conditions Of Error Detection For 56 And 112 Bit Replies.....	A-7
Figure A-4: Initialization Of Error Pattern Location Logic For 56 And 112 Bit Replies.....	A-9
Figure A-5: Generation Of Correction Enable Bit.....	A-11

Figure A-6: Correction Of Decision Sequence	A-12
Figure A-7: Final State Of Error Correction Function.....	A-13
Figure A-8: Low Confidence Bit Measurement - Flow Diagram.....	A-14
Figure A-9: Intruder Range Track Comparison - Flow Diagram	A-16
Figure B-1: ACAS X Communications	B-2
Figure C-1: Gating	C-2
Figure C-2: Reduced Update Interval For ADS-B Report.....	C-3
Figure C-3: NACp.....	C-4
Figure C-5: NACv.....	C-4
Figure C-6: Bearingless	C-5
Figure C-7: Active update delay	C-6
Figure C-8: Reported and Actual Quantization Mismatch	C-7
Figure C-9: No Heading Source Available	C-8
Figure C-10: Track Angle Supplied as Ownship Heading.....	C-9
Figure C-11: – Ownship Coarse Altitude Quantization	C-10
Figure C-12 – Non-Nominal Surveillance Region Compared to TA Alerting Region.....	C-10

TABLE OF TABLES

Table 1-1: Equipment Classes and Designations	4
Table 2-1: Assumptions About Aircraft Density	42
Table 2-2: Maximum ATCRBS Fruit Rate.....	43
Table 2-3: Maximum Mode S FRUIT Rates	43
Table 2-4: Aircraft Co-Site Transmissions	43
Table 2-5: Interrogation Spectrum	49
Table 2-6: Mode C Pulse Shapes	56
Table 2-7: Mode S Pulse Shapes.....	57
Table 2-8: Field Index.....	62
Table 2-9: Flight Status Codes.....	68
Table 2-10: Horizontal Sense Bits (HSB field).....	79
Table 2-11: Vertical Sense Bits (VSB field).....	81
Table 2-12: Capability Class (CC) Subfield in Aircraft Operational Status Messages, Airborne Format, in Version 2 Transmitting Subsystems	92
Table 2-13: Operational Mode (OM) Subfield in Aircraft Operational Status Messages, Airborne Format	93
Table 2-14: Coordination Timing Requirements	100
Table 2-15: Out-of-Band Rejection	117
Table 2-16: Illegal Altitude Codes.....	121
Table 2-17: Improvement Factors For Higher Resolution Whisper-Shout Sequences	125
Table 2-18: Summary of Rules for Determining Air-Ground Status.....	138
Table 2-19: Interval Between Revalidations.....	152
Table 2-20: Summary of STM Outputs for Intruder Surveillance Inputs	166
Table 2-21: ACAS X TRM Processing Modes.....	175
Table 2-22: Barometric Altitude Observation - [ADD Table 2-10].....	180
Table 2-23: Radio Altitude Observation – [ADD Table 2-11]	181
Table 2-24: Heading Observation – [ADD Table 2-12]	182
Table 2-25: WGS84 Observation - [ADD Table 2-13].....	183
Table 2-26: Ownship Discretes - [ADD Table 2-8].....	184
Table 2-27: DF0 Message - [ADD Table 2-2]	185
Table 2-28: Receive Mode C Reply Message - [ADD ModeCReply type]	186
Table 2-29: State Vector Position Report - [ADD Table 2-5]	189
Table 2-30: State Vector Velocity Report - [ADD Table 2-6].....	190
Table 2-31: Mode Status Report - [ADD Table 2-7]	192

Table 2-32: Target Designation – [ADD Table 2-9].....	193
Table 2-33: Designation Field Values	193
Table 2-34: UF16UDS30 Message - [ADD Table 2-14].....	194
Table 2-35: Coordination Capability Report	194
Table 2-36: STM /TRM Output Data	195
Table 2-37: StmDisplayStruct.....	197
Table 2-38: TrmIntruderDisplayData	198
Table 2-39: TrmDisplayData	198
Table 2-40: TRMDesignationData.....	199
Table 2-41: TRMIIntruderDesignationData.....	200
Table 2-42: TRMCoordinationInterrogationData.....	202
Table 2-43: Coordination Protocol	203
Table 2-44: TRMRABroadcastData	203
Table 2-45: TRMRACoordinationData	204
Table 2-46: TransponderData	204
Table 2-47: TRMGroundMsgData.....	205
Table 2-48: Recommended Location of No-Bearing Advisories	209
Table 2-49: Mapping Between RA, Aural Annunciation, and ARINC Word 270 Contents	228
Table 2-50: ACAS X Display and Controls Output (Control Panel).....	242
Table 2-51: ACAS X Display and Controls Output (Displays).....	243
Table 2-52: ACAS X Display and Controls Outputs (Aural Annunciations)	243
Table 2-53: ACAS X Display and Controls Inputs (RA Displays).....	244
Table 2-54: ACAS X Display and Controls Inputs (Aural Annunciations).....	248
Table 2-55: ACAS X Display and Controls Inputs (Traffic Display).....	249
Table 2-56: Recommended Display Actions and Monitor Response to “Not Operational” Status Received from ACAS X Displays	255
Table 2-57: ACAS X Display and Controls Inputs (RA Displays).....	262
Table 2-58: ACAS X Display and Controls Inputs (Aural Annunciations).....	262
Table 2-59: ACAS X Display and Controls Inputs (Traffic Display).....	263
Table 2-60: ACAS X Display and Controls Output (Control Panel).....	263
Table 2-61: ACAS X Display and Controls Output (Displays)	264
Table 2-62: ASA System Traffic state information (for each traffic) for display for ACAS Xo	277
Table 2-63: Nominal ASA System DNA States (for each traffic)	278
Table 2-64: Nominal ASA System CSPO-3000 States (for each traffic)	279
Table 2-65: Nominal ACAS Xo System DNA States (for each traffic)	280
Table 2-66: Nominal ACAS Xo System CSPO-3000 States (for each traffic)	282
Table 2-67: Designation Status	283
Table 2-68: Environmental Test Groupings.....	286
Table 2-69: Applicable Test Groups per Environmental Condition	288
Table 2-70: Standard Test Signal Pulse Shapes	291
Table 2-71: Standard Test Signal Total Jitter	292
Table 2-72: Confidence Factor Required For A 90% Confidence Level.....	319
Table 2-73: Minimum Number Of Trials Vs. Number Of Failures For The Required Probability Of Success Values Specified In §2.4.2.1.2	320
Table 2-74: Squitter And Reply Probability Vs. Range And Altitude.....	381
Table 2-75: Functional Areas Covered by Test Suite	397
Table 2-76: Input File Details	398
Table 2-77: STM Report Details.....	401
Table 2-78: TRM Report Details	403
Table 2-79: CostFile Details	405
Table 2-80: STM Report Test Values and Tolerances.....	407
Table 2-81: TRM Report Test Values and Tolerances	408
Table 2-82: Cost Value Tolerances.....	409
Table 2-83: Branches of Prescriptive Algorithms Not Covered By Prescriptive Test Encounters.....	410

Table 2-84: Unreachable Branches (Defensive Code) in Prescriptive Algorithms.....	411
Table 2-85: Unreachable Branches (Defensive Code) in Suggestive Algorithms	412
Table 2-86: Round Dial VSI RA Tests	504
Table 2-87: Vertical Speed Tape RA Tests	508
Table 2-88: RA Tests for Implementations Using Pitch Cues on a PFD.....	513
Table 2-89: RA Depiction Using Flight Director Guidance	517
Table 2-90: Tests for Implementations Displaying RAs on a HUD	520
Table 2-91: Airborne Position Decoding Values With Range and Bearing	576
Table 2-92: Cross-Reference of Requirements to Associated Tests	592
Table 2-93: Cross-Reference of Associated Tests to Requirements	606
Table 3-1: Specification Of Total Altimetry System Performance.....	623
Table A-1: Code Pulse Agreement	A-18
Table C-1: ADS-B Degradations	C-11
Table C-2: Degraded Inputs Affecting Active Tracks	C-11
Table D-1: Comparison of R and N for a Spherical Earth Radius as a Function of Latitude	D-2
Table D-2: Example of Errors in Spherical Earth Approximation	D-4
Table D-3: Example of Errors Using the Same N for Both Aircraft.....	D-5
Table D-4: Example of Errors in Ellipsoidal Differential Approximation with No Longitude Difference	D-7
Table D-5: Example of Errors in Ellipsoidal Differential Approximation with No Latitude Difference .	D-8
Table D-6: Example of Errors in Ellipsoidal Differential Approximation with Combined Latitude, Longitude and Height Differences.....	D-9
Table D-7: Example of Errors in Spherical Differential Approximation with No Longitude Difference.....	D-10
Table D-8: Example of Errors in Spherical Differential Approximation with No Latitude Difference .	D-11
Table D-9: Example of Errors in Spherical Differential Approximation with Combined Latitude, Longitude and Height Differences.....	D-12
Table E-1 Error Sources in the Hybrid Slant Range Calculation and Validation with ACAS X Range ..	E-2
Table E-2: Sample Allocation of Hybrid Data Processing Errors	E-4
Table F-1: Test Group 10F	F-1
Table F-2: Test Group 20.....	F-12
Table F-3: Test Group 30.....	F-17
Table F-4: Test Group 40.....	F-30
Table F-5: Test Group 50.....	F-32
Table F-6: Test Group 60.....	F-36
Table F-7: Test Group 70.....	F-64
Table H-1: Simulated Reduction in 1090 MHz Receiver Occupancy	H-7

1 PURPOSE AND SCOPE

1.1 Introduction

This document sets forth minimum operational performance standards for the Airborne Collision Avoidance System X (ACAS X) equipment, including both Active surveillance (Xa) and special Operations (Xo) functions. Unless otherwise noted, ACAS X references in this document refer to ACAS Xa/Xo.

ACAS X is intended to improve air safety by acting as a last-resort method of preventing mid-air collisions or near mid-air collisions (NMAC) between aircraft. By utilizing surveillance information from Secondary Surveillance Radar (SSR) and ADS-B technology, ACAS X equipment operates independently of ground-based aids and air traffic control (ATC). Aircraft equipped with ACAS X have the ability to interrogate airborne transponders and receive ADS-B Messages to determine the location of other aircraft in the vicinity and assess the risk of collision. ACAS Xa/Xo equipment is not required to detect non-cooperative aircraft.

ACAS X provides Traffic Advisories (TAs) and Resolution Advisories (RAs) in the vertical plane. RAs are indications given to the flight crew recommending maneuvers intended to avoid collisions with all threats, or restrict maneuvers to maintain existing separation. (In this document the term *separation* means physical separation, i.e., absence of NMAC, and should not be confused with the provision of ATC minimum separation.) ICAO standards give ACAS RAs priority over ATC clearances and instructions, i.e., flight crews are instructed to follow RAs even when it conflicts with ATC guidance (Ref. ICAO Doc. 8168 – PANS-OPS, §3.2.c.2). Controllers typically have no knowledge of RAs unless notified by the flight crew via the radio. RA information is provided by ACAS X to Mode S SSRs and ADS-B ground radios, but typically is not presented to controllers. Some alerts (e.g. wind shear warnings, stall warnings, and Ground Proximity Warning System warnings) have higher priority than ACAS RAs. Results of United States and European safety studies show that there is a significant safety benefit to be gained from the widespread carriage and operation of ACAS, specifically, the risk of mid-air collision is reduced by at least a factor of three (Ref. ACASA/WP1.8/210D/V1.1 27-12-01). Globally, operational experience has shown that ACAS systems have significantly improved the safety of flight.

Incorporated within these standards are system characteristics that should be of value to users, designers, manufacturers, and installers. These characteristics are intended to accommodate the requirements of various users.

1.1.1 Document Structure

This document is published in two volumes.

Section 1 of Volume I is intended to provide information needed to understand the rationale for equipment characteristics and requirements stated in the remaining sections. It describes typical equipment applications and operational goals and is the basis for the standards stated in the document. Definitions essential to proper understanding of this document are also provided in Section 1.

Section 2 of Volume I contains the minimum performance standards for the equipment. These standards define the required performance under standard operating conditions and stressed physical environmental conditions. It also details bench test procedures that demonstrate compliance, including specific bench tests for the collision avoidance logic performance.

Section 3 of Volume I describes the performance required of the installed equipment. Tests for the installed equipment are included when performance cannot be adequately determined through bench testing.

Volume II contains the Algorithm Description Document (ADD), which includes required and suggested collision avoidance algorithms for ACAS X. The algorithms are presented in the Julia programming language with informational commentary text.

Both Volumes I and II contain certain standards and performance requirements that ensure that ACAS X is fully interoperable (see §1.5.5) with other airspace elements and equipment. It is mandatory that these interoperability provisions be met without exception or deviation. Other technical and performance requirements stated herein that were derived from the particular implementation approach that formed the basis for the development of this MOPS may be waived if the manufacturer or installer provides evidence of equivalent system performance for an alternative method of implementation and if the alternative implementation does not violate interoperability provisions.

Since the measured values of equipment performance characteristics may be functions of the methods of measurement, standard test conditions and methods of test are recommended in this document.

1.1.2

Operational Goals

The general operational goals of ACAS X are described in the following subparagraphs. Implementation of new ATC system elements under development will change communication, surveillance, and data management. This will increase the effectiveness of the existing ground-based air traffic control system. It is also apparent that a single, ground-based ATC system cannot meet the requirements of all flight conditions because of the absence of coverage in oceanic and other low traffic-density airspace. Thus, there is a need to provide an air-to-air collision avoidance capability on aircraft, which is independent of the ground-based system. Such a system must remain compatible with the ATC system when operating in controlled airspace. As a minimum, equipment providing this capability must achieve the following operational goals:

- a. The system must be capable of providing timely and effective resolution advisories to a pilot to permit avoiding other air traffic equipped with an Air Traffic Control Radar Beacon System (ATCRBS) or Mode S altitude reporting transponder. This capability must be assured throughout airspace having a maximum traffic density of 0.3 transponder-equipped aircraft per square nautical mile.
- b. Operation must be compatible with the existing ATC system and with planned evolution of the ATC system.
- c. The system design must ensure that the system does not have characteristics that could adversely affect the safety of flight or interfere with other aircraft systems on the aircraft or installed on other nearby aircraft.
- d. Maneuvers expected in response to resolution advisories provided by the system should be acceptable to both pilots and air traffic controllers.
- e. The system must contain a comprehensive performance monitor function that will monitor automatically, on a continuous basis, the operating integrity of the system.

1.1.2.1

Operation of ACAS X on the Airport Surface

TCAS and ACAS X are designed as airborne collision avoidance systems and their use when ownship is on the ground is incidental. Originally, regulatory guidance for TCAS was that TCAS should be turned to standby on landing and it should not be switched to

TA/RA mode until entering the active runway. However, these procedures do not always fit well with other procedures and it is observed that, quite frequently, TCAS is operated on the airport surface. For example, guidance is to switch the Mode S transponder on at push-back, so TCAS is switched on at the same time.

The reason for the guidance to turn TCAS to standby on the airport surface is the need to minimize SSR interference caused by superfluous Mode S interrogations. ACAS X makes much more complete use of ADS-B for surveillance. In particular, ADS-B is used and active interrogation is not attempted when ACAS X is operating on the airport surface (i.e., it is known that ownship is on the ground) and the intruder's and own ADS-B data quality is good. This change goes a long way to address a practical regulatory and operational problem.

The exclusive use of ADS-B leads to a practical change in the functionality of ACAS X. TCAS generates TAs when the appropriate tests are applied when operating on the surface. ACAS X will not do so when the track is based on ADS-B data. Since TAs are designed for the airport surface, this change is desirable.

1.1.3

Equipment

"ACAS X" or "ACAS X equipment" as used herein includes all components or units necessary (as determined by the manufacturer or installer) for the equipment to perform its function properly. There are external systems that ACAS X is dependent upon that may be standardized in separate documents, but in each case, these dependencies will be made explicit. It should not be inferred that each ACAS X equipment design will necessarily have all components or units in separate packages. This will depend on the specific design chosen by the manufacturer. The transponder capabilities may also be implemented as part of the ACAS Xa/Xo avionics package but the term "ACAS X equipment" will refer only to the equipment needed to perform the surveillance and threat logic of the ACAS X system. Additional functions and components that may provide expanded equipment capabilities are identified. Other equipment features that are beyond the scope of this document may be described in future RTCA documents.

Since the equipment implementation includes a computer software package, compliance with the guidelines contained in RTCA/DO-178C, Software Considerations in Airborne Systems and Equipment Certification, December 13, 2011, and all changes thereto, is required. Compliance with the guidelines contained in RTCA/DO-254, Design Assurance Guidance for Airborne Electronic Hardware, April 19, 2000, and all changes thereto, is also required.

This document considers an equipment configuration identified in the System Overview, §1.3. Operational performance standards for functions or components that refer to equipment capabilities that exceed the stated minimum requirements are identified as optional features.

1.1.3.1

ACAS X Equipment Classes and Article Designations

Table 1-1 defines the ACAS X equipment classes, article labels and functionality for the ACAS X architecture. There are two classes:

- Class 1 (ACAS Xa) and
- Class 2 (ACAS X, i.e., Class 1 with ACAS Xo capability)

The classes define the capability of the ACAS X System. Class 1 equipment introduces enhanced algorithms compared to the existing Traffic Alert and Collision Avoidance System II (TCAS II) system, with no new traffic symbols, control modes, TA/RA

information, or operational procedures, so existing articles that support TCAS II can be used with the ACAS Xa unit. Class 2 Systems need displays and control panels that support DNA and/or CSPO-3000 functions.

There are seven articles that define the ACAS X system. The active surveillance transponder article is covered under RTCA/DO-181/EUROCAE ED-73 MOPS and specific sections related to these MOPS under article label D. The other six articles are covered in these MOPS. Four of the articles are generic and do not have a class associated with them because the performance standards do not differ. The other articles have both class and article labels. Each article has performance requirements in these MOPS. Appendix I maps the performance requirements for each article. Each article is labeled alphabetically and is further delineated by the ACAS Xo capability (DNA or CSPO-3000). Articles may be combined into one unit and labeled with multiple article markings.

Table 1-1: Equipment Classes and Designations

Equipment Class	Capability	ACAS X Article Label	ACAS X Article Name	Function/Notes
-	ACAS X	A	Directional Antenna	Broadcasts and receives 1030 and 1090 MHz interrogations, respectively. Mounted on the top or bottom of the aircraft.
-	ACAS X	B	Omni-Directional Antenna	Broadcasts and receives 1030 MHz and 1090 MHz interrogations, respectively. Mounted on the bottom of the aircraft in lieu of a bottom directional antenna.
-	ACAS X	C	Resolution Advisory (RA) Guidance and Annunciation Display	Display and annunciation of RA alerting and vertical guidance
1	ACAS Xa Only	D	ACAS Xa Unit	Performs airspace surveillance, intruder tracking, ownship altitude tracking, threat detection, RA maneuver determination and selection, and generation of advisories. Includes STM and TRM.
		E	Control Panel	Mode control for the basic ACAS Xa
		F	Traffic Display	Display of traffic and alerting
2	ACAS X (ACAS Xa)	D1	ACAS X Unit	Includes performance, verification and validation of DNA

Equipment Class	Capability	ACAS X Article Label	ACAS X Article Name	Function/Notes
and ACAS Xo)	D2	ACAS X Unit	Includes performance, verification and validation of CSPO-3000	
		E1	Control Panel	Mode control with DNA
		E2	Control Panel	Mode control with CSPO-3000
		F1	Traffic Display	Display of traffic, alerting, DNA symbols
		F2	Traffic Display	Display of traffic, alerting, CSPO-3000 symbols

1.1.3.2 Aircraft Equipment Information Vulnerabilities

Aircraft equipment information vulnerabilities (such as cybersecurity risks) have been present for digital systems since the development of the Personal Computer (PC) in the late 70s and even longer for Radio Frequency (RF) systems, and the advent of internet connectivity has substantially increased those risks. Internet and Wireless Fidelity (Wi-Fi) connectivity have become popular as a means for aircraft or equipment manufacturers to update installed avionics software, to update databases, or provide an alternate means of communicating with the flight crew or cabin (e.g., in-flight entertainment, weather, etc.).

In most countries, the State provides oversight of safety-of-flight systems (sometimes referred to as “authorized services”), which provide information to aircraft, such as an Instrument Landing System (ILS), VHF Omnidirectional Range Radar (VOR), Global Navigation Satellite System (GNSS), and Distance Measuring Equipment (DME), to name a few. However, the State typically does not provide oversight on “non-trusted”^[1] connectivity such as the internet, Wi-Fi, or manufacturer-supplied equipment interfaces that permit input of externally-supplied data into aircraft systems. A manufacturer may expose aircraft information to vulnerability through equipment design, or introduce vulnerability as a result of being connected to a common interface. Therefore, it is important that manufacturers consider aircraft information security risk mitigation strategies in their equipment design, particularly when the equipment is responsible for an interface between the aircraft and aircraft-external systems.

Apart from any specific aircraft-information-security-related performance requirements contained in these MOPS, it is recommended that manufacturers look at a layered approach to aircraft information security risk mitigation that includes both technical (e.g., software, signal filtering) and physical strategies. From a technical perspective, for example, this could include signal spoofing detection capabilities or more stringent, multi-factored authentication techniques such as passwords, Personal Identification Numbers (PINs), and

^[1] A “non-trusted” connectivity (sometimes referred to as third-party system) is any frequency or service where an Air Navigation Service Provider (ANSP) is not providing direct monitoring/protection.

digital certificates. From a physical perspective, a manufacturer could consider connectors that can be removed only by using special tools to prevent passenger tampering. Finally, manufacturers should consider supply chain risk management; for example, assure that software code development contractors and their staff are properly vetted.

Civil aviation authorities have a regulatory interest when an applicant's design makes use of a non-trusted connectivity where the installation can potentially introduce aircraft information security vulnerability. This requires the applicant to address not only the information security vulnerabilities and mitigation techniques for the new installation, but to also consider how vulnerability could propagate to existing downstream systems. Therefore, it is recommended that manufacturers reference their equipment aircraft information security review and mitigation strategies in the equipment's installation manual so that the applicant can consider them in meeting the installation regulatory requirements.

1.2

Background

1.2.1

Collision Avoidance Systems

A Collision Avoidance System (CAS) identifies short term airborne conflicts (i.e. encounters) and provides guidance to pilots to prevent NMACs. CAS provides collision avoidance protection in areas with no or limited ATC services, and also provides a backup to the ATC service. Three types of CAS were developed in the 1980s. TCAS I is a lower-cost system to provide situation awareness and TAs (but no RAs) for smaller aircraft. TCAS II provides TAs and RA guidance in the vertical plane, and was intended for larger passenger aircraft. TCAS III was expected to provide the addition of RAs in the horizontal plane based on horizontal measurements from a more complex directional antenna system, but this technique was found to be inadequate. The first available CAS system was TCAS II, the precursor to ACAS X.

1.2.1.1

TCAS II

TCAS II uses secondary surveillance radar to interrogate Mode A/C and Mode S transponder-equipped aircraft and obtain information needed to track their position / speed (ID, barometric altitude, range, etc.). The equipment uses this information to predict the time to closest point of approach (CPA), and determine the altitude difference at that point. Depending on time to CPA and the particular threshold violation, TCAS II may issue TAs and RAs. TCAS II only provides vertical RAs. TCAS II employs rule/heuristics-based logic. The measured bearing information provided by TCAS II antenna systems is used for display of traffic and some horizontal filtering.

TCAS II was first introduced in the United States in 1990 with collision avoidance logic Version 6.0 and later 6.04 and 6.04a. Version 7.0 was introduced in the United States and Europe in 2000 and was subsequently mandated by the International Civil Aviation Organization (ICAO) for all commercial turbine-powered transport aircraft worldwide having more than 19 passenger seats or having a maximum take-off weight above 5700 kg. Version 7.1 addressed several shortfalls in Version 7.0 logic. Version 7.1 was required in Europe by December 1st of 2015 (Ref. EU Commission Regulation No. 1332/2011, subsequently amended by Regulation 2018/583). TCAS II is referred to as the Airborne Collision Avoidance System II (ACAS II) in ICAO terminology. Version 7.1 is the most current version and was standardized in RTCA/DO-185B/EUROCAE ED-143. Moreover, ICAO Annex 10 (volume IV) states that all ACAS installations after 1 January 2017 shall be compliant with version 7.1. Finally, Hybrid Surveillance was added as an option for TCAS II installations starting in 2006; this augments active surveillance with passive ADS-

B-based surveillance, to assist with tracking and to reduce 1030/1090 MHz spectrum usage. At the time of this publication, there are no active efforts to modify TCAS II beyond its current minimum operational standards. All references to a TCAS system in this document, unless otherwise noted, refer to a TCAS II system.

1.2.1.2 ACAS X

The evolving global airspace presented challenges to TCAS II that resulted in the development of ACAS X. The ACAS X concept ensures interoperability with current and future CNS/ATM systems (including TCAS II), meaning that advisories provided on different aircraft in the same encounter are complimentary, maneuvers performed by pilots following advisories are compatible, and that CA operations are minimally disruptive to other ATM operations in that airspace.

There are four major improvements of ACAS X over TCAS II. In addition to the spectrum congestion mitigation changes introduced as part of the TCAS II Hybrid Surveillance MOPS update, ACAS X involves the following:

- Improved risk ratio, reduction of unnecessary alerts, and a reduced total number of RAs issued.
- Easier tuning and optimization to match the system's surveillance and tracking capability or to match the aircraft performance capabilities.
- The capability to resolve conflicts using a Decision Theoretic approach, specifically Markov Decision Process (MDP) rather than a rule/heuristics-based approach. The new logic is referred to as Optimized Resolution Logic and is reflected in the Threat Resolution Module (TRM).
- The capability to accept and process new surveillance sources like ownship position source and ADS-B. The new sources, in addition to Secondary Surveillance Radar (SSR), may be used for tracking cooperative aircraft more accurately.

Each variant of ACAS X supports a different set of operational objectives. Together, they will increase safety, reduce unnecessary RAs, and provide collision avoidance protection for new user classes.

This MOPS defines the requirements for ACAS Xa/Xo. Other ACAS X variants (e.g. ACAS Xu) are also planned for development and will be defined in separate standards.

1.2.1.2.1 ACAS Xa/Xo

ACAS Xa uses Active surveillance and coordination, in addition to passive surveillance, in combination with global threat resolution logic. It is intended as a replacement for TCAS II and ACAS II. ACAS Xa will have all of the improvements listed above.

ACAS Xo uses the same surveillance as ACAS Xa (coordination is slightly modified), combined with both global and Operation-specific threat resolution logic. It will facilitate specific ATC procedures (e.g. such as Closely Spaced Parallel Operations) where, for example, the Optimized Resolution Logic may need to behave differently with respect to a “designated target” than it does with respect to other targets.

ACAS Xa/Xo is standardized in this MOPS document. Note that the prescriptive algorithms provided in the ADD encompass ACAS Xa/Xo functionality. If a manufacturer implements only the ACAS Xa system (equipment class 1), the entire Xa/Xo prescriptive code will still be embedded.

1.2.1.2.2 ACAS Xu

ACAS Xu is designed for vehicles with new surveillance technologies and different performance characteristics, such as Unmanned Aircraft Systems (UAS). ACAS Xu standards will be contained in a separate MOPS document.

1.3 System Overview

The ACAS X equipment described in this document is operationally nearly identical to existing TCAS II systems. It provides similar traffic display information, uses similar display symbology, and provides similar TAs and vertical-only RAs. The basic traffic display remains the same, indicating the range, altitude, altitude rate (climbing or descending), and bearing of the intruding aircraft (when available) relative to ownship. Traffic altitude is required to support RAs. Transponder-equipped aircraft that do not provide altitude information are not considered for RAs, but can receive TAs.

ACAS X is designed to allow drop-in replacement of existing TCAS II systems. While the displays and alerts will be familiar to flight crews, the number, timing and type of alerts generated by ACAS X in an encounter may be significantly different from those issued by TCAS II due to improved surveillance, tracking and CA logic methodology. For example, ACAS X will not issue an RA calling for vertical speed limitations other than 0 fpm, whereas TCAS II allowed RAs with vertical speed limitations of 0, 500, 1,000 and 2,000 fpm. Also, some RAs are announced differently; specifically, the words ‘maintain vertical speed’ and ‘maintain vertical speed, crossing maintain’ are not used.

1.3.1 Surveillance and Tracking

Operation of ACAS X is not dependent upon any ground-based systems. ACAS X is most readily understood by visualizing its operation in flight. When airborne, the ACAS X equipment periodically transmits interrogation signals. These interrogations are received by ATCRBS or Mode S altitude reporting transponders. ACAS X will detect the presence of Mode S-equipped aircraft by listening to Mode S transmissions received with DF=11 (squitter transmissions) and DF=17 (Extended Squitter), as described in §2.2.4.6.4.2.1 of this document. In reply to the interrogations, the transponder transmits a signal which reports its altitude. The ACAS X equipment computes the range of the intruding aircraft by using the round-trip time between the transmission of the interrogation and the receipt of the reply. Altitude information is provided by decoding the information in the received reply.

All ACAS X systems will include the use of hybrid surveillance (making use of Mode S Extended Squitter information) to extend display range and limit interference. ACAS X will also use surveillance information provided by ADS-B and ADS-R. ACAS X does not include any of the requirements for ADS-B reception in RTCA/DO-317 as a minimum capability, but does require some capability to receive DF=17 Messages per RTCA/DO-260B/EUROCAE ED-102. The quality of ADS-B surveillance enhances the performance of Collision Avoidance. ACAS X will process ADS-R messages which are re-broadcast by ADS-B ground stations when this service is available, in a manner consistent with the processing of other ADS-B Messages.

ADS-B Message data from other aircraft is received and processed to provide the location and velocity of those aircraft as well. Data from transponder replies and ADS-B messages are correlated and tracked, and the system provides a probabilistic estimate of the location of all intruders. These data are used to determine whether some action must be taken to

provide awareness or a maneuver should be conducted to avoid the intruding aircraft. Each threat aircraft is processed individually to permit selection of the minimum safe resolution advisory based on track data and coordination with other Active CAS-equipped aircraft with a resolution advisory generating capability.

Bearing estimates are used by the collision avoidance algorithm to determine if the predicted horizontal separation at closest point of approach will be sufficient without declaring the intruder a threat. Bearing estimates are also used to display TAs in the cockpit of ownship. See Appendix C, §C.3.2.2 for a discussion on the potential impact of undetected bearing errors on system performance.

ADS-B data is used in ACAS X for both hybrid surveillance and threat resolution. For TCAS II, ADS-B was only used for hybrid surveillance, but not threat resolution. For ACAS X, TAs and RAs can be issued against targets providing ADS-B data as long as it is validated with active surveillance. If ADS-B fails validation, the system reverts back to using active surveillance for threat resolution.

ACAS X has the optional capability to generate TAs against ADS-B only (including DF=18 ADS-B messages) or ADS-R only targets. This feature is referred to as AOTO. Implementation of this option requires different display symbology indicating such TAs cannot progress to RAs. When not implemented, ADS-B only and ADS-R only targets will be displayed only as non-threats and proximate traffic as appropriate.

1.3.2

Threat Resolution

As in TCAS II, RAs indicate vertical maneuvers that are predicted to either increase or maintain the existing vertical separation from threatening aircraft. TAs indicate intruding aircraft that may later cause resolution advisories to be displayed.

The RAs employed in ACAS Xa/Xo equipment do not provide horizontal escape maneuvers and therefore are not intended to increase horizontal separation. Increased horizontal position accuracy available through the use of new surveillance methodology (such as ADS-B) may allow future development of horizontal RAs. This is not being considered for this ACAS Xa/Xo MOPS document, but will be part of the ACAS Xu standards.

If the threat aircraft is itself equipped with Active CAS equipment that generates RAs, a coordination procedure via the air-to-air Mode S data link is performed. This procedure assures that the aircraft RAs are compatible. Coordination is discussed further in §1.3.3.

TAs, with their aural alert and color/symbology change on the traffic display, may assist the flight crew in visual acquisition of the traffic and situation awareness in relation to other traffic, as well as prepare the flight crew to respond to a possible RA.

1.3.3

Coordinating RAs Against ACAS-Equipped Threats

Coordination of RAs is pairwise: ACAS X transmits to each threat its intent with respect to that threat (i.e., whether to pass above it or below it), and receives from each threat the intention of that threat with respect to ownship. Even in multi-aircraft encounters, when ACAS X generates an RA against several threats simultaneously, the negotiation of complementary RAs involves only two aircraft at a time; it results in a contract between only two parties, and coordination takes place with each ACAS-equipped threat individually.

When the threat is TCAS or ACAS X-equipped, ACAS X sends a “TCAS Resolution Message” to the threat. The information is in the form of a negative command, e.g., ‘do not pass above,’ and is encoded as a Vertical Resolution Advisory Complement (VRC)

subfield. ACAS X will send a ‘do not pass above’ VRC to an equipped threat if it has selected a climb sense advisory against that threat. Likewise, if it has selected a descend sense advisory against an equipped threat, it will send a ‘do not pass below’ VRC to that threat.

Before selecting an RA, ACAS X must first check to see if it has received a VRC from the threat. If no VRC has been received, ACAS X will select the optimum RA based on the encounter geometry. If a VRC has been received, in most cases (not always), ACAS X will initially select an RA based on an intent complementary to that of the threat.

In the vast majority of cases the two aircraft see each other as threats at slightly different moments in time. In contrast, “simultaneous” coordination occurs when both aircraft select RAs before a VRC is received from the other aircraft. In the case of simultaneous coordination, the RA intents can be incompatible and one aircraft must give way to the other by reversing its intent. The aircraft that is obliged to give way is referred to as the slave and the aircraft that prevails is referred to as the master.

The only other circumstance in which RA intent is reversed arises when the master detects that its current choice of intent must be reversed in order to resolve the encounter. The master then reverses intent, cancels the previous VRC and sends a new one forcing the slave to reverse its intent.

When both aircraft are ACAS X or TCAS II, both aircraft have discrete 1030/1090 MHz interrogation reply capability and the ICAO aircraft addresses are used for master/slave determination. The aircraft with the lower ICAO address is the master and the aircraft with the higher ICAO address is the slave.

1.3.3.1

Future Coordination Schemes

ACAS X and TCAS are both intended for commercial transport aircraft with good performance characteristics (e.g., an ability to climb at 2500 fpm) and a discrete 1030/1090 MHz interrogation(reply capability. Such aircraft can be considered to be peers of each other. In the future, less capable aircraft or aircraft not linked to a discrete 1030/1090 MHz interrogation(reply capability might equip with CAS. This document includes provisions to enable ACAS X to coordinate RAs with such new CAS, fitted to aircraft that would not be considered peers of ACAS X.

When the two aircraft are not peers, master/slave determination is accomplished according to §2.2.3.9.3.1. Capability information is exchanged that announces the coordination schemes available to each CAS. The options include using ADS-B instead of 1030/1090 MHz interrogation and reply for coordination messages, and also an announcement that an aircraft should not be considered a peer of other ACAS X or TCAS equipped aircraft.

Three coordination schemes are envisaged: Active, Passive, and Responsive Coordination. Active Coordination is that described above for ACAS X and TCAS and involves active 1030/1090 MHz interrogation and reply. Passive Coordination would be between two aircraft both using ADS-B to coordinate and considered to be mutual peers (and using ICAO 24-bit Aircraft Address to determine master/slave status). Responsive Coordination would be used when one aircraft uses 1030/1090 MHz interrogation and reply and the other uses ADS-B to send coordination messages, and also when, for any reason, it is determined that the two aircraft are not peers and one should be the master regardless of the ICAO 24-bit Aircraft Addresses.

Responsive Coordination is the scheme used by the slave when the two aircraft are not peers. The slave must ensure that it selects RAs that are compatible with an intent received from the master. The master, which in this document would be ACAS X, uses Modified Active Coordination or Modified Passive Coordination, the modification being that it is

the master regardless of ICAO 24-bit Aircraft Address and disregards any coordination information from the threat.

1.3.3.2 Multiple Threats, Sense and Intent, and Reversals

In an encounter with a single threat, the concepts of “RA sense” and “RA intent” are equivalent. The sense is up and the intent is to pass above the threat when a descend is prohibited or limited. The sense is down and the intent is to pass below the threat when a climb is prohibited or limited. The word “intent” is particularly useful when both aircraft are ACAS equipped and the RAs are coordinated.

Once an initial RA has been announced, the logic is designed so that the frequency of reversals of RA sense (intent) is minimized, subject to other constraints. In the first operational version of TCAS II, the first action of TCAS II when a threat was identified was to choose the sense of the RA and it could not be changed thereafter. However, it was found that it is essential that RA sense can be reversed *in extremis*.

An RA reversal is issued when the previous RA is not resolving the encounter and a revised course of action is urgent. When the reversal requires the crew to initiate or maintain a climb or descend the annunciation is either "Climb, Climb Now" or "Descend, Descend Now", emphasizing the urgent action required. When the reversal does not require a climb or descend maneuver the annunciation is either "Level Off" or "Monitor Vertical Speed" depending on the advisory sense and the current vertical speed of the aircraft. Reversals to "Level Off" or "Monitor Vertical Speed" are very rare and can only occur as a result of a multiple aircraft encounter or when ownship descend inhibits are active at low altitudes.

The concepts of “RA sense” and “RA intent” become more complex in multi-threat encounters, in particular when the global RA, i.e. the RA announced on the flight deck, is a Multi-Threat Level Off (MTLO), which is displayed using two red arcs on an IVSI to show that both the rate of climb and the rate of descent are limited. However, in multi-threat encounters, the logic first identifies the optimum for each threat in isolation and then conciliates between the resulting pair-wise RAs to form the global RA. The concepts of “RA sense” and “RA intent” still make sense for these pair-wise RAs. The process of coordinating RAs is maintained for threats that are ACAS-equipped and the coordination messages are based on the RA intent in the pair-wise encounters.

1.3.4 Control of Interference Caused by Active Surveillance

Although ACAS X makes much more use of ADS-B than does TCAS, the ability to detect and track aircraft through active SSR interrogation still provides the bedrock for ACAS X. The use of 1030/1090 interrogation and reply inherently increases the population of SSR signals, potentially leading to excessive interference. ACAS X has inherited the interference limiting provisions of TCAS. Although these provisions are unchanged in themselves, it is anticipated that the increased use of ADS-B and hybrid surveillance by ACAS X will reduce the practical impact of the interference limiting algorithms.

The interference limiting algorithms enforce three limits, expressed as inequalities, on the power and frequency of ACAS X interrogations.

- The first inequality limits the “on” time of transponders on other aircraft caused by Active CAS interrogations.
- The second inequality limits the “on” time of own transponder caused by mutual suppression during the transmission of interrogations by own ACAS X.

- The third inequality limits ATCRBS fruit caused by Active CAS Mode C interrogations.

If any of these three inequalities is violated, the power or frequency of interrogations must be reduced. Further guidance is provided in this document, but manufacturers have considerable freedom in how to ensure these inequalities are met.

For early versions of TCAS the precise form of the inequalities involved assuming that Active CAS units (just TCAS in those days) are uniformly distributed in area. However, experience showed that interference remained excessive where the concentration of TCAS units was high, for example where most aircraft were converging on or diverging from an airfield. To allow for variations in the distribution of TCAS units, between uniform in area at one extreme and uniform in range at the other extreme, a new parameter (α) expressing the distribution was introduced and the inequalities were changed to those defined in §2.2.3.6.1.

The inequalities are modified in two circumstances. On the airport surface, interrogations are more severely curtailed by tripling the observed density of Active CAS within 30NM (NTA). Above 18,000 ft, the inequalities are simplified and relaxed by eliminating their dependence on the local density of Active CAS and by assuming the distribution of Active CAS is uniform in area; additionally, the second inequality is not used at all above 18,000 ft.

1.3.5

Provision of RA Information to Ground Systems

ACAS X provides RA information via Mode S Datalink, ADS-B and 1030 MHz 'RA Broadcasts'. This information is used: to monitor ACAS performance; in incident investigations; and some Air Navigation Service providers (ANSPs) use it to provide some RA information to controllers. The RA Report formats have been revised for ACAS X enabling the provision of more detailed information than provided by TCAS II. (Table 2-49 shows ACAS X RA downlink data, together with representative RA displays.)

1.3.6

Hybrid Surveillance and Extended Hybrid Surveillance

1.3.6.1

Overview

The intent of hybrid surveillance is to reduce the ACAS X interrogation rate through the judicious use of the ADS-B data provided via the Mode S extended squitter. The hybrid surveillance function must meet these goals:

- No degradation of the existing ACAS X collision protection.
- Reduced ACAS X interrogation rate along with reduction in associated replies resulting in decreased 1030 MHz and 1090 MHz utilization.
- Maintain tracks that might have been dropped from the display in high density traffic areas due to interference limiting algorithms of Ref. N.

The system must include hybrid surveillance functionality. Loss of the hybrid surveillance function does not degrade the minimum ACAS X functionality.

1.3.6.2

Description

Hybrid surveillance is used as a means to decrease Mode S interrogations. Hybrid surveillance allows ACAS X to determine the interrogation rate needed to validate passive surveillance depending on certain criteria and the intruder's near-term collision threat

potential. Passive surveillance is performed using on-board navigation data (typically based on GNSS) broadcast from the other aircraft through the use of Mode S extended squitter (i.e., 1090 MHz ADS-B, which is described in Ref. G). Active surveillance uses the standard ACAS X/transponder interrogations/replies that provide range, bearing and altitude to the intruder.

There are two passive surveillance techniques: the first is called ‘Extended Hybrid Surveillance’ and can be used when an intruder’s ADS-B position data meets certain quality and power requirements. The second technique, called ‘Hybrid Surveillance’ or ‘standard hybrid surveillance,’ must use ACAS X active interrogations to validate the passive surveillance position. The requirements for hybrid surveillance have been incorporated into these MOPS. These requirements implement the requirements of RTCA/DO-300A with changes to the ‘active’ surveillance region.

Figure 1-1 illustrates how the system transitions from the extended hybrid surveillance region through the hybrid surveillance region to the active surveillance region as a function of collision potential. When the intruder is far from being a threat and meets all conditions for extended hybrid surveillance, the intruder is tracked using ADS-B data exclusively and no active surveillance validation interrogations are required. Otherwise hybrid surveillance is used, and passive surveillance position is validated every 10 to 60 seconds with an ACAS X active interrogation. When the intruder comes close to being a collision threat in altitude and range (referred to in this section as a near-threat), the ADS-B is validated with active surveillance at a 1 Hz interrogation rate. The conditions that determine that an intruder is a near-threat ensure that TAs and RAs only get issued in the active surveillance region. Note that in the active surveillance region, validated ADS-B is used for threat resolution; and if ADS-B fails validation, active interrogation/reply surveillance is used for threat resolution.

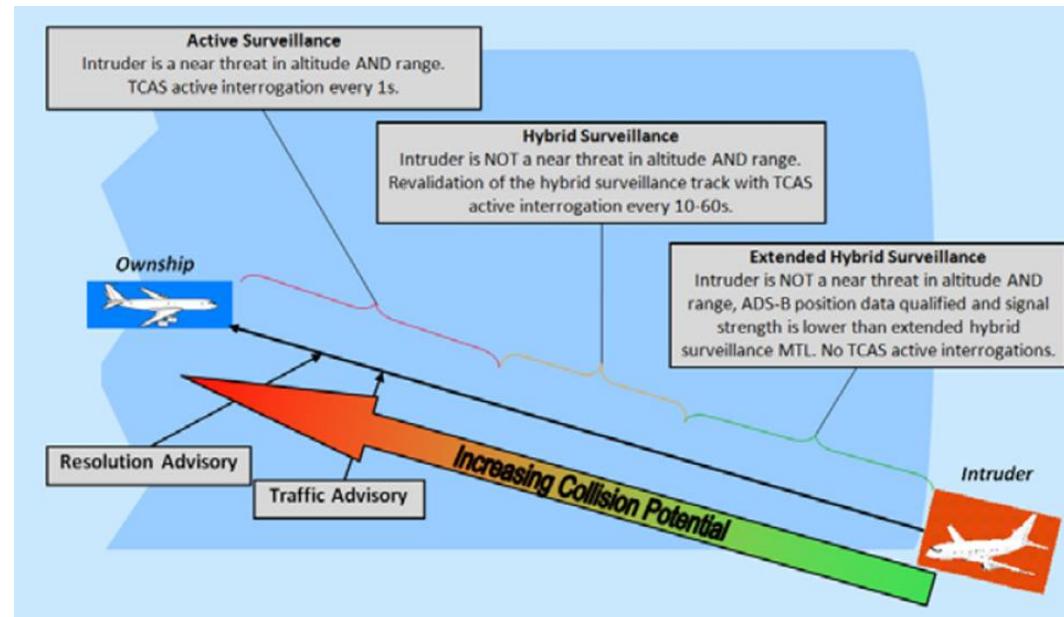


Figure 1-1: Transition from Passive to Active Surveillance as a Function of Collision Potential

Figure 1-2 and Figure 1-3 are decision flow diagrams illustrating possible transitions between surveillance methods for surface and airborne operation.

1.3.6.2.1 Use When Airborne

Figure 1-2 illustrates the transitions between the three surveillance techniques (active, hybrid, and extended hybrid) when ownship is Taking Off/Airborne (§2.2.4.6.3). Intruder tracks can be established either exclusively by extended hybrid surveillance or exclusively by active surveillance. The system initiates an intruder track with extended hybrid surveillance (§2.2.4.6.4.2.2.2) when own and intruder data meet data quality requirements and the intruder's signal strength is lower (weaker) or equal to the extended hybrid surveillance minimum trigger level (MTL). In all other cases the system initiates an intruder track using active surveillance.

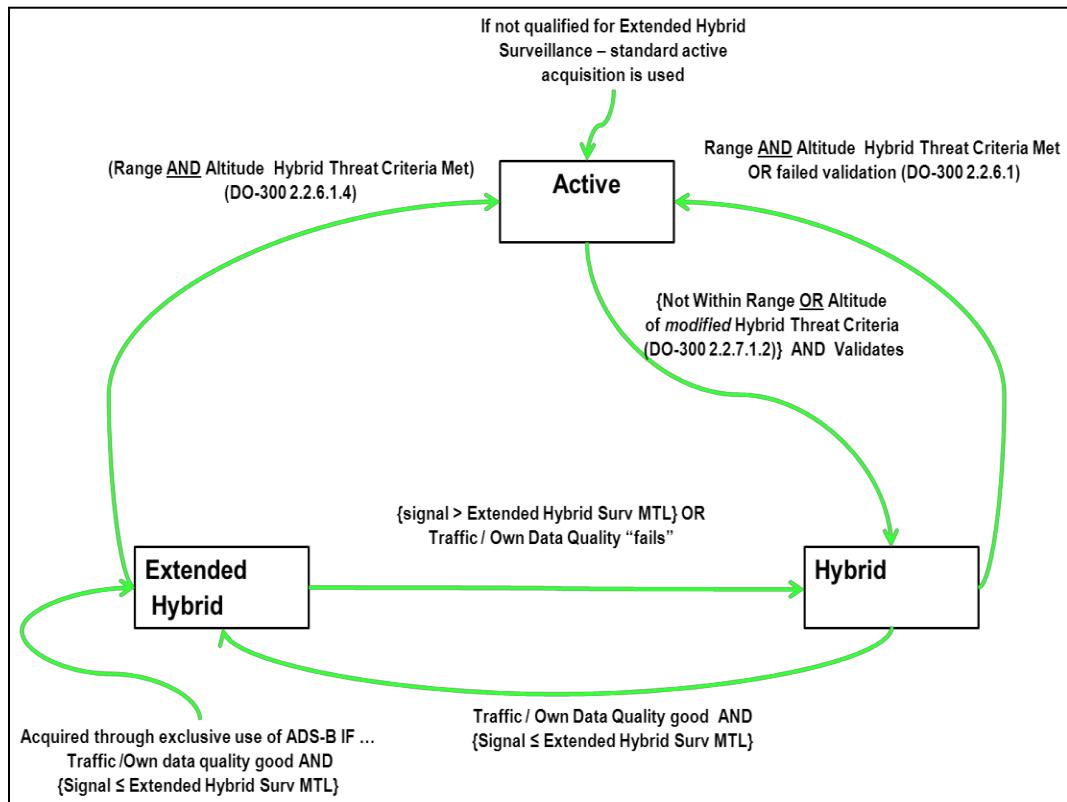


Figure 1-2: Extended Hybrid Surveillance State Transition Diagram (Ownship Taking Off / Airborne)

Once under active surveillance, if an intruder is not a near-threat in both altitude and range and its passive position data is successfully validated (§2.2.4.6.4.2.3.1.7, §2.2.4.6.4.2.3.1.8), the intruder can be tracked using hybrid surveillance, and the passive surveillance data is revalidated regularly (§2.2.4.6.4.2.3.1.10). If revalidation fails, the intruder transitions to active surveillance. The intruder transitions from hybrid to extended hybrid surveillance (§2.2.4.6.4.2.3.1.3) if own and intruder data meet data quality requirements and the intruder's signal strength is lower or equal to the extended hybrid surveillance MTL. Once the intruder is tracked passively (extended hybrid or hybrid surveillance), the hybrid threat status is monitored once per second (§2.2.4.6.4.2.3.2.3). If an intruder becomes a near-threat in range and altitude, it transitions to active surveillance. When the passive data are available but not qualified, or the intruder's signal strength is higher (stronger) than the extended hybrid surveillance MTL, the intruder transitions from extended hybrid surveillance to hybrid surveillance (§2.2.4.6.4.2.3.1.7). A track is maintained under extended hybrid surveillance (§2.2.4.6.4.2.3.1.2) as long as own and intruder data meet data quality requirements, and the intruder's signal strength is lower or

equal to the extended hybrid surveillance MTL and the intruder is not a near-threat in range and altitude.

1.3.6.2.2 Use When On The Ground

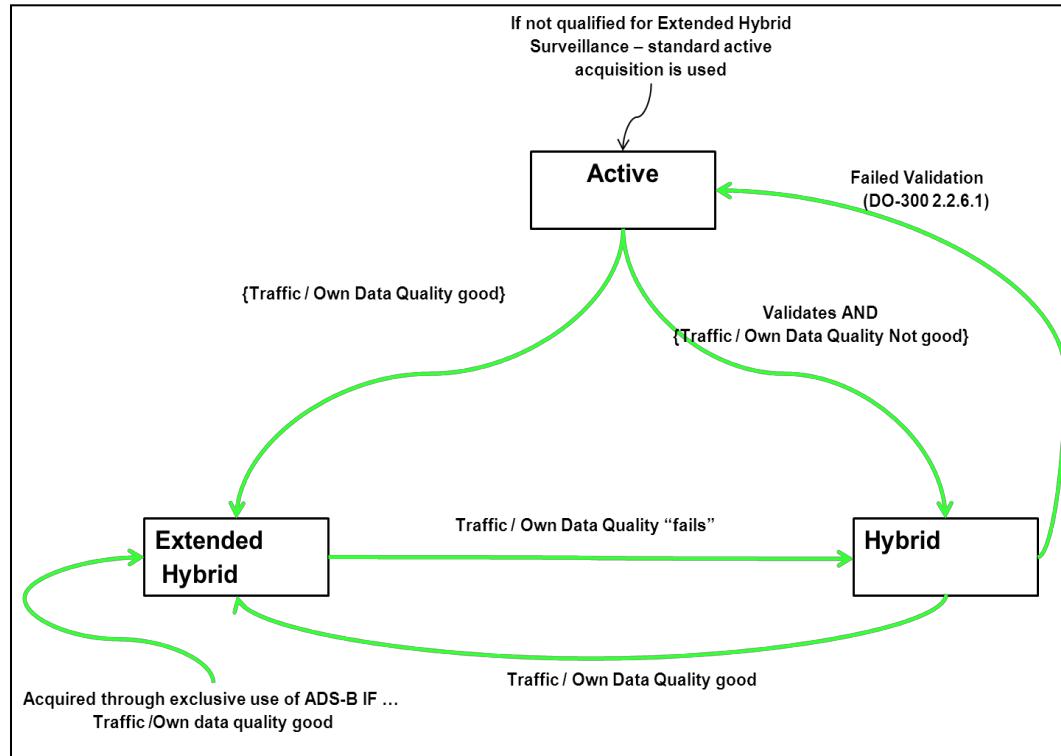


Figure 1-3: Extended Hybrid Surveillance State Transition Diagram (Ownship Operating on Surface)

Figure 1-3 illustrates the transitions between the three surveillance techniques when ownship is operating on the surface (§2.2.4.6.3). In contrast to airborne operation (Figure 1-2), strong signal level does not require active surveillance acquisition (§2.2.4.6.4.2.2.2), and signal level is not a factor in determining transitions between extended hybrid and hybrid surveillance (§2.2.4.6.4.2.3.1.3). Data quality is the only and sufficient condition for transition to extended hybrid surveillance, even directly from active surveillance (§2.2.4.6.4.2.3.1.3). While operating on the surface, the near-threat criteria are not required, and therefore the transition between hybrid and active surveillance is based only on (re)validation results (§2.2.4.6.4.2.3.2.7, §2.2.4.6.4.2.3.1.10). Direct transition from extended hybrid to active is not permitted.

While ACAS X is operating on the airport surface airborne intruders will be displayed as Non-Threat or Proximate traffic but not as TAs while ownship is on the ground. Additionally, this MOPS increases the use of passive surveillance while on the airport surface. This reduces the 1090 MHz channel utilization if ACAS X is used on the ground. Although the operation of TCAS on the airport surface has become common practice, such use was not intended. In particular, TAs and the underlying logic were not designed for use while ownship is operating on the airport surface.

1.3.7 ACAS Xo functionality

ACAS Xo will require all functionality from ACAS Xa systems. ACAS Xo provides the capability of applying different, more operationally appropriate conflict detection and resolution logic to a particular “designated” intruder.

The designation function is specified in the ASA System MOPS requirements. Therefore, aircraft that equip with ACAS Xo will also require integration with an ASA System or equivalent for Xo mode designation and display requirements. ASA System integration provides the flight crew interface necessary for ACAS Xo traffic designation, as well as other CDTI display capabilities such as a unified traffic picture and directional traffic symbology.

This document includes requirements for two initial ACAS Xo modes, DNA and CSPO-3000, and limits the use of ACAS Xo to one mode and one aircraft at a time. Designated No Alerts (DNA) allows suppression of alerts for the designated aircraft while CSPO-3000 includes logic for closely spaced parallel approaches separated by 3000 ft or greater. While use of the ASA System or equivalent is required to provide the controls and displays necessary for ACAS Xo, the initial ACAS Xo modes are designed for stand-alone use (i.e., no corresponding ASA System applications).

Future versions of the ACAS Xa/Xo MOPS are expected to provide additional ACAS Xo modes designed to specifically support future ASA System applications. These applications might designate the ACAS Xo mode automatically. The ACAS Xo system may be required to support designation of multiple aircraft concurrently, and also operation of multiple ACAS Xo modes concurrently.

1.4 Assumptions and Limitations**1.4.1 Operational****1.4.1.1 Airworthiness**

The design and manufacture of the equipment used in ACAS X will provide for installation such that it does not impair the airworthiness of the aircraft.

1.4.1.2 Improvements or Alternative Techniques

It is recognized that developments will continue which may have the potential of producing improved operational performance. It is not feasible to describe a generic way of meeting most ACAS X requirements and also write specific test procedures that should be used to validate them. Thus, specific means of accomplishing each desired result are described by this document.

Manufacturers are reminded that the ACAS X algorithms specified in Volume II have been proven via validation and verification efforts. New features in addition to those specified are permissible, providing that these do not interfere with the functions defined by the existing requirements and are subjected to the same degree of validation and verification.

1.4.2 Environment

The following assumptions describe the environment onboard the ACAS X aircraft and the airspace in which it operates.

1.4.2.1 Transponders

ACAS X systems will encounter targets equipped with ATCRBS transponders and Mode S transponders.

Note: *The intention here is to ensure ACAS X can interrogate and receive data from both types of transponders.*

1.4.2.2 ADS-B Systems

For ACAS X, it is a minimum requirement for the surveillance function to be capable of reception of 1090ES ADS-B broadcasts. This permits the exploitation of hybrid and extended hybrid surveillance techniques. However, neither the surveillance function nor the collision avoidance function require reception of ADS-B as a prerequisite for satisfactory performance. Nevertheless, the use of these techniques by the surveillance function of ACAS X not only significantly improves its performance, and the performance of the collision avoidance function, but also significantly reduces its utilization of the 1030-1090MHz RF environment.

All of the references in this document to hybrid surveillance, extended hybrid surveillance, and ACAS X surveillance assume that ADS-B reception functionality is available. If ADS-B reception capability is not available, ACAS X will conduct surveillance using tracked Mode S and ATCRBS transponder replies.

1.4.2.3 Integration with ASA Systems

ASA systems are not required to be installed in Class 1 (ACAS-Xa only) ACAS-X installations. However, in aircraft equipped with ASA Systems, ACAS X should be integrated with the ASA Systems to provide more complete traffic situation awareness.

The Xo display and control functionality will be performed by the ASA System or equivalent according to RTCA/DO-317B (Ref. T). This is established in order to ensure a uniform means of traffic designation and display. It is also expected that future ASA System applications will automatically designate traffic to ACAS Xo modes, as necessary.

1.4.2.4 Interoperability with Onboard Aircraft Systems

ACAS X may be designed and installed in conjunction with other aircraft systems, including multifunction antennas, displays, or controls. Any such combination is assumed to preserve the general performance requirements of the ACAS X equipment, the general performance requirements of the system with which the ACAS X is combined, any technical performance specifications stated elsewhere, or any technical performance standards for the other system.

1.4.2.5 Ownship Capability

The following sections describe systems that are not part of ACAS X requirements, but provide functionality and/or data to ACAS X for its proper functioning.

1.4.2.5.1 Barometric Altimetry System

It is assumed that ownship is equipped with a barometric altimeter capable of providing information to ACAS X.

It is assumed that the barometric altitude provided in Mode S replies is the same barometric altitude reported in the ADS-B extended Squitter.

1.4.2.5.2 Mode S Transponder

ACAS X design requires that ownship is equipped with a diversity Mode S transponder (whose replies include encoded altitude when appropriately interrogated) and that there is an interface established between the Mode S transponder and ACAS X. The Mode S transponder provides the following important information to ACAS X: ownship Mode S address, ownship Mode A code, ownship altitude, and ownship altitude quantization. Also, the Mode S transponder is used to coordinate RAs with other Active CAS-equipped aircraft and broadcast RA information to the ground. Additionally, the Mode S transponder is responsible for communicating ownship ACAS equipage to other aircraft in the vicinity of ownship and to the ground.

1.4.2.5.3 Ownship Position Source

ACAS X is designed to make use of ADS-B Message data from intruder aircraft for surveillance and tracking. The absolute WGS-84 position of the ownship is required in order to convert these data into relative coordinates. Therefore, it is assumed that ownship has position source, such as GNSS, onboard that is providing the horizontal position in these cases. The ownship aircraft is assumed to have a position source data source of sufficient quality to provide its location.

Note: The loss of ownship position source will not cause ACAS X to fail. Instead, ACAS X will transition to a mode where all surveillance is performed using active interrogations and replies exclusively.

1.4.2.5.4 Aircraft Discretes

Aircraft discretes provide information about ownship capabilities and configuration. These discretes must be used as appropriate per these MOPS and manufacturer specifications.

1.4.2.5.5 Radio Altimeter

The radio altimeter is the device on board the aircraft which provides measurements of an aircraft's height above the ground. ACAS X must know ownship altitude above ground in order to avoid issuing RAs which would cause ownship to approach unsafe levels of separation from the ground. When ownship radio altimeter measurements are available and credible, they must be provided to ACAS X to be incorporated in the threat-resolution decision-making process. If the radio altimeter provides information on its operating status or the credibility of its measurements, that information must be provided to ACAS X.

1.4.2.5.6 Heading

Ownship heading is required for determining the bearing of ADS-B equipped intruders relative to ownship and for determining the appropriate intruder placement on the traffic display. Ownship aircraft is assumed to have a heading source of sufficient quality to be used in this calculation.

1.4.3

Human

ACAS X design incorporates assumptions and constraints arising from the interaction with the flight crew and Air Traffic Controllers. In order to minimize implementation costs and ensure interoperability with existing TCAS and ATC procedures, ACAS X retains most of existing TCAS II assumptions and constraints. Regulatory guidance on the roles and responsibilities of the flight crew and ATC are contained in FAA AC 120-55C, ICAO Annex 6, ICAO Documents 8168 (PANS-OPS) and 4444 (PANS-ATM), and other specific regional guidance documents. ACAS Xo design includes some additional assumptions and constraints; however, although ACAS Xo will require some new flight crew operational procedures, it is not expected that there will be changes to the roles and responsibilities of the flight crew and ATC.

Note: ACAS Xa has intentionally been designed such that a pilot trained about TCAS II alerts, displays, and guidance will be able to operate an aircraft equipped with ACAS Xa. There are no new RAs or visual guidance cues that a pilot will need to learn. However, the aural annunciations associated with Maintain Rate RAs have been changed to “Climb, Climb” or “Descend, Descend” depending on RA sense. To ensure good understanding and confidence of the ACAS Xa system, it might be appropriate to educate flight crews about key differences between the two systems, such as a reduction in the number of alerts and differences in some TA and RA alert timing.

1.4.3.1

Pilot Actions

Extensive monitoring data confirms that pilots often do not respond to TCAS RAs as had been assumed by system design. ACAS X system design retains assumptions regarding desired pilot response to RA events. However, ACAS X optimization includes a pilot model that incorporates imperfect performance; therefore, the selected action explicitly considers that probability that pilot response may not match optimum system design assumptions. Evaluation of ACAS X system performance took into account the optimum desired pilot response, but also evaluated system performance across the expected range of pilot responses, including latency and strength.

The desired pilot RA response assumes the pilot will:

- Initiate response within 5 seconds (initial RA) and 2 ½ seconds for increase rate or reversal RAs.
- Respond with a ¼ G vertical acceleration for an initial RA and up to ⅓ G for strengthening or reversal RAs.
- Maneuver to keep vertical rate out of the red area and in the green area (if present).
- Following RA termination, expeditiously return to current ATC clearance if the RA required deviation from an ATC clearance.

1.4.3.1.1

Existing TCAS Pilot Responsibilities

FAA AC 120-55C provides specific pilot responsibilities for use of existing TCAS. These responsibilities include:

- Upon TA
 - Attempt to visually acquire the traffic
 - Do not deviate from an assigned clearance based solely on TA information
- Upon RA
 - Direct Attention to RA displays and maneuver as indicated

- Respond by:
 - Disconnecting the autopilot and auto-throttles (if necessary)
 - Initiate response within 5 seconds (initial RA) and 2 ½ seconds for increase rate or reversal RAs
 - Response should be ¼ G for an initial RA and up to 1/3 G for strengthening/reversal RAs
 - Maneuver to keep vertical rate in the green area and out of the red area
 - On aircraft with pitch guidance for TCAS RA displays, follow the RA pitch command for initial, increase, reversal, and weakening RAs
- Following RA termination, expeditiously return to current ATC clearance if RA compliance required deviation from clearance

1.4.3.2

Interaction With ATC

It is assumed that controllers will comply with the published standard procedures for TCAS II. It is not expected that those procedures will change for ACAS Xa or ACAS Xo. Additionally, Xa/Xo will provide RA downlink information, as does TCAS II, to support ACAS operational performance monitoring by ground systems; however the bit assignments are not identical. Some ANSPs use this information to implement a display of RA downlink information to controllers. The content and latency of RA information on controller displays will depend upon the implementation by the local ANSP.

1.4.3.3

Use of ACAS X With Other Onboard Traffic Alerting Systems

This MOPS does not consider the safety and operational implications of interaction between ACAS X alerts and any other system (e.g., TIS, TAS, TSAA) that may provide additional TAs or RAs.

1.5

Test Procedures

The test procedures and associated limits specified throughout this document are intended to be used as one means of demonstrating compliance with the minimum acceptable performance parameters. Although specific test procedures are cited, it is recognized that other methods may be preferred.

These alternative procedures may be used if they provide at least equivalent information. In such cases, the procedures cited herein should be used as one criterion in evaluating the acceptability of the alternative procedures.

The order of tests suggests that the equipment be subjected to a succession of different tests as it moves from design and design qualification into operational use. For example, compliance with the requirements of Section 2.0 should have been demonstrated as a precondition to satisfactory completion of the installed system tests of Section 3.0. The tests normally are performed once for each model of equipment to demonstrate compliance with the minimum operational performance standard.

1.5.1

Environmental Tests

Environmental tests are specified in Subsection 2.3. The tests and associated requirements are intended to provide a laboratory means of determining the electrical and mechanical performance of the equipment under environmental conditions expected to be encountered in actual operations. Test results may be used by equipment manufacturers as design guidance, in preparation of installation instructions and, in certain cases, for obtaining formal approval of equipment design and manufacture.

1.5.2 Bench Tests

Bench test procedures are specified in Subsection 2.4. These tests are conducted at the equipment level and are intended to provide a laboratory means of demonstrating compliance with the requirements of Subsections 2.1 and 2.2. Test results may be used by equipment manufacturers as design guidance for monitoring manufacturing compliance and, in certain cases, for obtaining formal approval of equipment design and manufacture.

1.5.3 Installed System Tests

The installed system test procedures and their associated limit requirements are specified in Section 3.0. Although bench and environmental test procedures are not included in the installed system tests, their successful completion is a precondition to completion of the installed tests. In certain instances, however, installed system tests may be used in lieu of bench test simulation of such factors as power supply characteristics, interference from or to other equipment installed on the aircraft, etc. Installed tests are normally performed on one model of each aircraft under two conditions:

- a. With the aircraft on the ground and using simulated or operational system inputs.
- b. With the aircraft in flight using operational system signals appropriate to the equipment under test.

Test results may be used to demonstrate functional performance in the environment in which the equipment is intended to operate.

1.5.4 Computer Performance Verification Test

Bench tests and environmental tests include computer verification tests as specified in §2.4.3.

1.5.5 Interoperability

Interoperability with other elements of the airspace in which ownship is operating is a mandatory requirement for all equipment and is independent of implementation alternatives. Interoperability, for purposes of this MOPS, is defined to include all requirements and provisions stated herein that impact the technical and/or operational relationship of the ACAS X equipment with the following:

ATCRBS: ACAS X is required to interoperate with ATCRBS transponder equipment. This interoperability is ensured by compliance of the ACAS X with all applicable ATCRBS signals-in-space requirements of the Minimum Operational Characteristics-Airborne ATC Transponder Systems, RTCA/DO-144.

Mode S: ACAS X is required to interoperate with Mode S transponder equipment. This interoperability is ensured by compliance of the ACAS X with all applicable Mode S signals-in-space and protocol requirements of the Minimum Operational Performance Standards for the Air Traffic Control Radar Beacon System/Mode Select (ATCRBS/MODE S) Airborne Equipment, RTCA/DO-181A and all revisions thereto.

TCAS: ACAS X is required to interoperate with all RA-generating TCAS equipment. In addition, the ACAS X must interface with its associated Mode S transponder such that all data exchange requirements are met as defined herein.

ADS-B: ACAS X is required to interoperate with ADS-B transmitting and receiving equipment. This interoperability is ensured by compliance of ACAS X with all applicable

1090 MHz ADS-B signals-in-space and protocol requirements of any ADS-B data link system or processing equipment, which may include:

- Minimum Operational Performance Standards for 1090 MHz Extended Squitter Automatic Dependent Surveillance – Broadcast (ADS-B) and Traffic Information Services – Broadcast (TIS-B)", RTCA/DO-260B, and all revisions thereto.

1.6 Configuration Control, Security and Distribution of the ACAS X Electronic Products

The ACAS X electronic products consist of ACAS X Parameter Data Item Files and the ACAS X Test Suite which are further described in the subsequent sections.

1.6.1 Configuration Control, Security and Distribution of ACAS X Parameter Data Item Files

1.6.1.1 Purpose

The purpose of this section is to describe the configuration control, security and distribution of the ACAS X parameter data item files (PDIFs) developed by RTCA SC-147 and EUROCAE WG 75.

Note: RTCA/DO-178C Annex C, Acronyms and Glossary of Terms; Parameter Data Item File – The representation of the parameter data item that is directly usable by the processing unit of the target computer. A Parameter Data Item File is an instantiation of the parameter data item containing defined values for each data element.

1.6.1.2 Overview and Guidelines

The ACAS X design is dependent on the PDIFs, which are used in the ACAS X equipment to aid decision making in real time. It is therefore important that each parameter data file is properly identified, controlled and distributed.

This section establishes requirements to ensure that each parameter data file is properly identified, controlled and distributed. The security aspects of the parameter data files are also addressed.

From the initial creation of the PDIFs, to their eventual loading in ACAS X equipment and subsequent installation on an aircraft, the security, control and distribution responsibilities extend across many stakeholders in the aviation community.

The identified stakeholders are:

- RTCA/EUROCAE
- Regulator (FAA/EASA)
- ACAS X Equipment Manufacturer
- Applicant
- End User/Airline Operator/Aircraft OEM

The ACAS X MOPS documentation is available from RTCA and EUROCAE. In addition to the ACAS X MOPS documentation, the ACAS X PDIFs are only available electronically from RTCA as a supplement to these MOPS. They are not available from EUROCAE.

The ACAS X PDIFs are:

-
1. Minimum blocks index table (Xa and Xo)
 2. Equivalence class cost/score table (Xa and Xo)
 3. Horizontal entry table (Xa and Xo)
 4. Horizontal entry table for active surveillance (Xo only)
 5. Vertical entry table (Xa and Xo)
 6. Parameter file (Xa and Xo)

1.6.1.3 PDIF Details

The six PDIFs each have a unique identifier. The ACAS X PDIFs are only available electronically from RTCA. Changes to a PDIF result in a new unique identifier for that PDIF. The MD5 hashing algorithm is applied to create hash codes of the PDIFs to ensure the integrity of each obtained copy of the PDIFs. It is acceptable to aggregate the PDIFs in one electronic file prior to applying the algorithm. The hash codes are listed in these MOPS (see Volume II, Appendix F).

Changes to the MOPS text should be controlled by corrigendum or revision depending on the extent of the change following existing RTCA and EUROCAE procedures. Any change to the PDIFs results in revision of the MOPS.

For further information relating to the PDIF life cycle process flow and the roles and responsibilities of the stakeholders, refer to Appendix J.

1.6.2 Additional Electronic Products

An additional supplement to these MOPS, containing the ACAS X Test Suite, is available from RTCA and EUROCAE in an electronic format. The supplement is in the form of single zip archive file.

The Test Suite is a set of test cases in the form of aircraft encounter files, i.e., a series of ACAS X sensor inputs and the correct corresponding outputs. These test cases are intended to be used by the implementers of ACAS X to provide assurance that the system was implemented correctly.

Refer to §2.4.2.2 in these MOPS for more information about the Test Suite.

1.7 Glossary of Terms

ACAS X Surveillance - The use of passive ADS-B surveillance (with validation by 1 Hz active interrogations) to update an ACAS X track. Applies to tracks that do not meet the criteria for Hybrid or Extended Hybrid Surveillance.

ACAS Xo Mode – An alternative ACAS X logic. Two ACAS Xo modes are included in this document: DNA and CSPO-3000.

Active CAS - An airborne collision avoidance system that uses interrogations on 1030 MHz to track other aircraft, possibly in addition to other methods of tracking other aircraft, and uses 1030/1090 MHz to coordinate avoidance maneuvers in collision risk bearing encounters with other Active CAS. Such CAS transmit the TCAS Broadcast Interrogation Message.

Notes:

1. *The word "Active" must always be spelled out, lest confusion be caused with the acronym ACAS.*

2. *TCAS II is an Active CAS. TCAS I, as defined in RTCA/DO-197, uses active interrogations on 1030 MHz but is not an Active CAS because it does not (and could not) use 1030/1090 MHz to coordinate avoidance maneuvers. It does not transmit the TCAS Broadcast Interrogation Message; its impact on the 1030/1090 MHz RF spectrum is controlled and limited through its own interference limiting provisions, which are disjoint from those of TCAS II.*

Active Coordination - Coordination of RAs in which both aircraft exchange 1030/1090 MHz discrete coordination interrogations and replies (TCAS Resolution Messages/Coordination Reply Messages). The master aircraft is that with the lower ICAO 24-bit Aircraft Address.

Active Surveillance - The use of ACAS X interrogations and subsequent replies to update or acquire an ACAS X track. Measurements from active surveillance typically include a range, bearing, and altitude measurement.

Actual Xo Mode - The actual Xo mode in use. The Actual Xo Mode is normally the same as the Designated Xo Mode, except when mode transitions are delayed due to RAs or in multi-threat encounters.

ADS-B Report – A report from a Ref. G compliant ADS-B report generator function which contains ICAO 24-bit Aircraft Address, latitude, longitude, altitude, NIC, NACp, SIL, and Air/Ground information.

Advisory - An alert for conditions that require flightcrew awareness and may require subsequent flightcrew response for collision avoidance.

Airborne Collision Avoidance System - An avionics system onboard aircraft that performs collision avoidance.

Airborne Position Message – A Mode S extended squitter (DF=17) that contains aircraft position and altitude or an ADS-B report updated for position.

Aircraft Surveillance Applications System (ASA System) – An avionics system that provides the platform for aircraft-to-aircraft applications using surveillance such as ADS-B, ADS-R and TIS-B. These systems include subsystems for surveillance data processing (ASSAP), and display and control (CDTI).

Air Traffic Control (ATC) - A generic term for a joint civil-military system for controlling traffic within a specific area.

Air Traffic Control Radar Beacon System (ATCRBS) - A secondary surveillance radar system having ground-based interrogators and airborne transponders capable of operation on Modes A and C.

Alert - A flight deck indication meant to attract the attention of and identify to the flightcrew information relevant to collision avoidance.

Altitude Crossing - Encounters in which two aircraft are intended to cross or actually cross in altitude prior to reaching closest point of approach.

Altitude Crossing RA - A resolution advisory is altitude crossing if it is based on an intention to resolve the risk of collision by crossing in altitude prior to reaching closest point of approach. In a multi-threat encounter, each threat is considered separately.

Altitude, Relative - The altitude of the intruder aircraft measured from the ownship aircraft, i.e., relative altitude is positive when own is lower and negative when own is higher (Relative Altitude = Intruder Altitude – Ownship Altitude)

Altitude Separation - The absolute value of relative altitude.

AOTO – ADS-B Only TA Only (target or track) – An option that allows the system to generate TAs on unvalidated high quality ADS-B.

Article - An article can be a material, part, component, process or appliance. In the context of this MOPS, an article refers to a component of the ACAS X system.

ATCRBS - Air Traffic Control Radar Beacon System – Also refers to Mode A/C.

Bearing - The horizontal angle of an intruder aircraft relative to the nose of the ownship aircraft, sometimes expressed as a clock position measured clockwise.

Broadcast – Unsolicited transmission to a non-specific destination.

Cancel Vertical Resolution Advisory Complement (CVC) - Information sent from one Active CAS-equipped aircraft to another via a coordination interrogation to cancel the Vertical Resolution Advisory Complement (VRC) previously sent.

Closest Point of Approach (CPA) - The occurrence of minimum range between own ACAS X aircraft and the intruder. Thus, range at closest approach is the smallest range between the two aircraft and time of closest approach is the time at which this occurs.

Coasted Track - A track that is continued based on previous track characteristics in the absence of surveillance data reports.

Cockpit Display of Traffic Information (CDTI) – A subsystem of the ASA System that provides the flight crew interface. The CDTI subsystem includes the actual display media and the necessary controls to interface with the flight crew. Thus the CDTI consists of all displays and controls necessary to support the applications. The controls may be a dedicated CDTI control panel or it may be incorporated into other controls, (e.g., multifunction control display unit [MCDU] or Electronic Flight Bag [EFB]). Similarly, the CDTI display may be a stand-alone display or displays (dedicated display(s)) or the CDTI information may be present on an existing display(s) (e.g., multi-function display) or an EFB. At a minimum, CDTI includes a graphical plan-view (top down) traffic display, in these MOPS referred to as the Traffic Display, and the controls for the display and applications (as required).

Collision Avoidance - The third layer of conflict management, whereby aircraft in imminent risk of colliding are given traffic situation awareness and possibly maneuvering guidance to avoid a mid-air collision.

Collision Avoidance System (CAS) – The combination of aircraft, pilot (when pilot is present), avionics, and procedures working together to perform the function of collision avoidance.

Coordination - The process of ensuring that two aircraft take complementary (non-conflicting) RA avoidance maneuvers when there is an imminent risk of collision between those two aircraft.

Coordination Interrogation - A Mode S interrogation (uplink transmission UF=16) transmitted by an Active CAS containing a TCAS Resolution Message.

Coordination Reply - A Mode S reply (downlink transmission DF=16) acknowledging the receipt of a coordination interrogation by Active CAS.

Corrective Advisory - A resolution advisory that instructs the pilot to deviate from current vertical rate, e.g., DON'T CLIMB when the aircraft is climbing.

Crosslink Interrogation – A Mode S UF=0 interrogation with RL=1. This interrogation can be used to request the aircraft position data from another aircraft's transponder that would be included if an Airborne Position Message was squittered at that instant.

Designated Xo Mode - The Xo mode for the designated intruder as requested by the flight crew.

Designation Request - ASA System output initiated by the flight crew for Xo mode designation. It includes the ICAO 24-bit Aircraft Address of the intruder and the Designated Xo mode.

Designation Status - Provides information about the status of the Xo mode designation (i.e., normal, delayed, suspended, and undesignated) See Table 2-67 for possible values.

Displayed Resolution Advisory - An RA already displayed to the flight crew.

Encounter - The term used to describe when two or more aircraft are in proximity to each other, such that ACAS X might be activated to provide advisories to avoid collision.

Established Track - A track that has been acquired and subsequently maintained under either active or passive surveillance.

Estimated Signal Strength – The signal strength associated with an aircraft used in determining if an aircraft is above or below the Extended Hybrid Surveillance MTL.

Extended Hybrid Surveillance - A specific surveillance technique that uses unvalidated ADS-B position messages for surveillance.

Extended Hybrid Surveillance MTL – A signal threshold used in determining whether an aircraft qualifies for Extended Hybrid Surveillance.

Extended Hybrid Surveillance Track – A track that is being maintained with qualified Airborne Position Messages without validating active interrogations.

Extended Squitter - A method of determining the position and/or state of an aircraft by having a transponder on board the aircraft periodically broadcast position and/or state information.

False Advisory - An advisory caused by a false track or an ACAS X malfunction.

False Track - A track for which an aircraft does not exist. This can be a result of multipath interference, synchronous garble, fruit, noise, etc..

FRUIT - Transponder replies unsynchronized in time. See Garble, Nonsynchronous.

Garble, Nonsynchronous - Reply pulses received from a transponder that is being interrogated from some other source. Also called FRUIT.

Garble, Synchronous - An overlap of the reply pulses received from two or more transponders answering the same interrogation.

Global RA – The RA announced to the flight crew during a multi-threat encounter.

Note: *In multi-threat encounters, the ACAS Xa logic generates individual RAs for each threat, which are not announced to the flight crew.*

Hash Code – A fixed-length number or message obtained by computing a hash function over an arbitrarily-sized piece of data. Intended to uniquely identify that piece of data.

Horizontal Miss Distance (HMD) - The horizontal range between two aircraft at the point of closest approach.

Horizontal Position Integrity Bounds – Describes an integrity containment region about the reported position, within which the true position of the surveillance position reference point is assured to lie at the reported time of applicability. Horizontal Position Integrity Bounds are communicated in ADS-B messages using the Navigation Integrity Containment (NIC) and Source Integrity Level (SIL) parameters.

Horizontal Position Uncertainty – 95% accuracy bound on horizontal position equivalent to Estimated Position Uncertainty (EPU), which is defined as the radius of a circle, centered on the reported position, such that the probability of the actual position being outside the circle is 0.05. When reported by a GNSS system, EPU is commonly called Horizontal Figure of Merit (HFOM). Horizontal Position Uncertainty is communicated in ADS-B messages using the Navigation Accuracy Category for Position (NACp) parameter.

Hybrid Surveillance – The combined use of active and validated ADS-B passive surveillance data to update a track.

ICAO 24-bit Aircraft Address – A unique 24-bit address assigned to each Mode S equipped aircraft. It is included in Mode S replies and broadcasts in order to uniquely identify the aircraft. It allows Mode S interrogators (SSRs or Active CAS) to interrogate one aircraft at a time by specifying that aircraft's ICAO aircraft address in the interrogation. Historically this was called the Mode S discrete address or Mode S address or Mode S ID.

ICAO Aircraft Identification – The three letter ICAO designator for the aircraft operating agency followed by the flight number (e.g., BAW123); or the Registration Marking of the aircraft (e.g., FGZCF).

Increased Rate Resolution Advisory - A resolution advisory that advises the pilot to increase the altitude rate to a value exceeding that of a previous positive RA.

Intent - The intent to pass above or below a particular threat. For Active CAS-equipped threats, this intent is conveyed in coordination messages that amount to an instruction not to maneuver in a sense that conflicts with ownship's intent.

Intruder - A transponder- or ABS-B-equipped aircraft within the surveillance range of ACAS X for which ACAS X has an established track.

Master - The master is the aircraft that has priority in coordination tie-breaking and is allowed to initiate geometric reversals.

Minimum Trigger Level (MTL) - The minimum input power level that results in a 90% ratio of decoded to received replies.

Mode A - A type of ATCRBS transmission which requests (via Mode A interrogations) or reports (via Mode A replies) aircraft identity information.

Mode C - A type of ATCRBS transmission which requests (via Mode C interrogations) or reports (via Mode C replies) aircraft altitude information. ATCRBS transponders which do not have Mode C capability respond to Mode C interrogations with standard ATCRBS framing pulses but no altitude information.

Mode C Altitude – altitude encoded in accordance with 'SSR automatic pressure-altitude transmission code' as defined in ICAO Annex 10.

Mode S - A type of SSR transmission which contains a unique ICAO 24-bit Aircraft Address, thus allowing interrogations to be addressed to individual aircraft. Mode S transmissions can be short (56 bits) or long (112 bits), with long transmissions containing a 56-bit "message" field.

Mode S Beacon System - A secondary surveillance radar (SSR) system having ground-based interrogators and airborne transponders and capable of using Mode S transmissions. Mode S ground interrogators are evolutionary in that they transmit Mode S interrogations to aircraft equipped with Mode S transponders but also transmit Mode A and C interrogations to aircraft equipped with ATCRBS transponders.

Modified Active Coordination - Coordination of RAs in which ACAS Xa is the master without reference to the ICAO 24-bit Aircraft Address, transmits a TCAS Resolution Message to the threat, and disregards any coordination information from the threat.

Modified Passive Coordination - Coordination of RAs in which ACAS Xa is the master without reference to the ICAO 24-bit Aircraft Address, transmits an ADS-B Operational Coordination Message to the threat, and disregards any coordination information from the threat.

Multipath - Reflections of signals from the ground, or detection of more than one reply to the same interrogation.

NaN - NaN is representation of "not a number" in Julia for floating point types used in the ADD to denote an invalid value.

Near Midair Collision (NMAC) - Two aircraft simultaneously coming within 100 ft vertically and 500 ft horizontally.

Negative Advisory – An advisory that does not require a climb or descent, including Do Not Climb, Do Not Descend and Level Off advisories. A negative advisory can be either preventive or corrective.

Non-Cooperative – An intruder aircraft that is not equipped with a SSR transponder nor ADS-B transmitter, and thus, is not voluntarily making information about its presence or location available to proximate aircraft

Operating Mode – A pilot selected input that determines how ACAS X will operate; ACAS X has four operating modes; off/standby, self-test, TA-Only, and TA/RA.

Note: *The term is not used for Xo modes because these involve only the specific aircraft designated for the Xo mode. (See 'Protected Mode')*

Own / Ownship Aircraft - The ACAS X-equipped reference aircraft.

Passive Coordination - Coordination of RAs in which both aircraft exchange ADS-B messages (Operational Coordination Messages) instead of 1030/1090 MHz discrete coordination interrogations and replies. The master aircraft is that with the lower ICAO 24-bit Aircraft Address.

Passive Surveillance - The use of ADS-B Messages to update an ACAS X track.

Positive Advisory – An advisory that requires either a climb or a descent.

Potential Threat – An intruder about whom ownship has issued a Traffic Advisory (TA) but for which no RA has been issued.

Preventive Advisory - A resolution advisory that instructs the pilot to avoid certain deviations from current vertical rate, as for example a DON'T CLIMB when the aircraft is level.

Processing Cycle - A nominal one second interval which encompasses all ACAS X surveillance, collision avoidance and supporting functions.

Protection Mode – The level of Collision Avoidance for a given target that ACAS X is providing. “Global” is the default protection mode for all traffic (TA/RA mode using ACAS Xa). Specific Xo protection modes (currently limited to “CSPO-3000” and “DNA”) are provided for designated targets based on the Xo mode selected.

Proximate Traffic - Nearby aircraft within ±1,200 feet vertically and less than 6 nautical miles range which do not meet either the threat or the potential threat classification criteria.

Qualification – The function of verifying that the quality of the Airborne Position Messages associated with a passive surveillance track is sufficient for use in Extended Hybrid Surveillance.

RA Intent - The intent to pass above or below a particular threat. For Active CAS-equipped threats, this intent is conveyed in coordination messages that amount to an instruction not to maneuver in a sense that conflicts with ownship's intent.

RA Sense - A direction that a resolution advisory permits or instructs: for ACAS Xa, either up or down.

Range At Co-altitude - Range when own and intruder aircraft are at the same altitude.

Report - A message containing surveillance data on a target aircraft.

Resolution Advisory - A display indication given to the pilot recommending a vertical maneuver, or in some cases no maneuver, to either increase or maintain the existing vertical separation relative to an intruding aircraft. Positive and negative advisories constitute the resolution advisories. A resolution advisory is also classified as corrective or preventive.

Resolution Maneuver – Maneuver in the vertical plane resulting from compliance with a Resolution Advisory.

Responsive Coordination - Coordination of RAs in which the intruder must adjust its choice of RA intent to be complementary with the RA intent of own ACAS Xa. The intruder does not transmit coordination information to ACAS Xa.

Revalidation - The validation that is performed after a track transitions to passive surveillance.

Secondary Surveillance Radar (SSR) - A radar system in which the return signal is radiated from a transmitter on board the target.

Self-Test - Tests of the ACAS X equipment and displays which are initiated by the flight crew and are used to determine the operational status of the equipment. Self-test differs from Performance Monitoring in that it is initiated by the flight crew, may use external stimuli, and is not performed continually or automatically.

Sense Reversal - A change to the sense opposite that of the original Resolution Advisory.

Sensitivity Level (SL) - A set of parameters used to specify the size of the protected volume around TCAS II-equipped aircraft for threat detection. The size of the protected volume and hence the Sensitivity Level varies with altitude. This parameter is not used in ACAS X to make maneuver selection, but is carried forward to maintain interoperability with legacy TCAS avionics in operation.

Slave - The slave is the aircraft that, if necessary, must adjust its choice of RA intent to be complementary with the RA intent of the master.

Squitter - Spontaneous transmission generated once per second by Mode S transponders.

Standard TCAS MTL – -74 ± 2 dBm.

Standby Mode – Operating Mode in which 1030 MHz interrogations are not transmitted, ACAS X tracks are not updated, and advisories are not issued.

Surface Position Message – A Mode S extended squitter (DF=17) that contains aircraft surface position or an ADS-B report updated for surface position.

Surveillance Mode – The data field provided as part of each surveillance report provided to the CAS logic. This field has two values: Reduced and Normal. Reduced implies a 5 sec nominal surveillance update rate and Normal implies a 1 sec nominal surveillance update rate for active tracks. The CAS logic will only generate TAs and RAs for intruders in normal surveillance mode.

Surveillance Update Interval – Nominally 1 sec.

TA-Only Mode – Operating Mode in which TAs are displayed and annunciated when required and RAs are inhibited from display and annunciation.

TA/RA Mode – Operating Mode in which both TAs and RAs are issued as specified.

Target - A transponder- or ABS-B-equipped aircraft within the surveillance range of ACAS X for which ACAS has an established track.

TCAS Broadcast Interrogation Message - A long Mode S air-air surveillance interrogation (UF=16) with the broadcast address.

TCAS Only Traffic Display - A display that presents traffic using a standard TCAS symbology only as originally defined in RTCA/DO-185B without use of RTCA/DO-317 or other ADS-B symbology.

TCAS Resolution Message - The message containing the Vertical Resolution Advisory Complement (VRC).

Threat – An intruder about whom ownship has issued a Resolution Advisory (RA).

Track - Estimated position and velocity of a single aircraft based on surveillance data reports.

Track File – Two uses of this term:

1. The data structure that contains all of the relevant information on a single target aircraft necessary for estimating the state of the target or to create an STM report. This includes the state estimate as well as any flags or timestamps.
2. The data associated with a single target for surveillance outside the STM.

Traffic Advisory - Information given to the pilot pertaining to the position of another aircraft in the immediate vicinity. The information contains no suggested maneuver.

Traffic Density (aircraft per square NM) - The number of transponder-equipped aircraft within R NM of ownship, divided by $\pi * R^2$. Transponder-equipped aircraft include Mode S and ATCRBS Mode A/C, and exclude ownship.

Traffic Designation Information - A block of data that includes any or all of the following: Designated Xo Mode, Actual Xo mode, Designation Status, Xo mode availability.

Transponder - The receiver-transmitter portion of a secondary surveillance radar system that sends an identifying coded signal, in response to a transmitted interrogation.

Transponder Equipage - Indication of type of transponder, whether ATCRBS or Mode S.

Validation - The function of comparing range, bearing, and altitude derived from active surveillance with range, bearing, and altitude derived from passive surveillance in order to determine its suitability for use in Threat Resolution processing.

Vertical Miss Distance - The relative altitude between own and intruder aircraft at closest point of approach.

Vertical Resolution Advisory Complement (VRC) - Information provided by one Active CAS to another via a coordination interrogation to ensure complementary maneuvers by restricting the choice of maneuvers available to the Active CAS receiving the VRC.

Warning Time - The time interval between potential threat or threat detection and closest approach when neither aircraft accelerates.

Whisper-Shout - A sequence of ATCRBS interrogations and suppressions of varying power levels transmitted by Active CAS equipment to reduce the severity of synchronous interference and multipath that suppress transponders that had replied to a previous interrogation.

1.8**Abbreviations**

- AC: Advisory Circular.
- ACARS: Aircraft Communications and Reporting System.
- ACAS X: Airborne Collision Avoidance System X.
- ADS-B: Automatic Dependent Surveillance – Broadcast.
- ADS-R: Automatic Dependent Surveillance – Re-broadcast.
- ANSP: Air Navigation Service Provider.
- AOTO: ADS-B Only TA Only.
- AQ: Acquisition.
- ARP: Aerospace Recommended Practices.
- ARTS: Automated Radar Terminal System.
- ASA System: Aircraft Surveillance Applications System.
- ATC: Air Traffic Control.
- ATCRBS: Air Traffic Control Radar Beacon System.
- BDS: B-Definition Subfield.
- CAS: Collision Avoidance System.
- CC: Crosslink Capable.
- CPA: Closest Point of Approach.
- CPR: Compact Position Report.
- CSPO-3000: Closely Spaced Parallel Operations down to 3,000 ft runway separation.
- CVC: Cancel Vertical RA Complement.
- CW: Continuous Wave.
- dBm: Decibels (dB) Reference One Milliwatt.
- Deg.: Degrees.
- DF: Downlink Format.
- DMTL: Dynamic Minimum Trigger Level.
- DNA: Designated No Alerts.
- DoD: Department of Defense.
- EADS: European Aeronautic Defense and Space Company.
- EASA: European Aviation Safety Agency.
- EFIS: Electronic Flight Instrument System.
- EHSI: Electronic Horizontal Situation Indicator.
- EICAS: Engine Indication and Crew Alerting System.
- ERP: Effective Radiated Power.
- EU: European Union.
- FL: Flight Level.

fpm: Feet per minute.

fps: Feet per second.

FRUIT: False Replies from Unsynchronized Interrogator Transmissions.

ft: Feet.

g: One gravity (nominally 32.2 feet/second/second).

GICB: Ground-Initiated Comm-B.

GNSS: Global Navigation Satellite System

GPWS: Ground Proximity Warning System.

HAE: Height Above Ellipsoid.

HSI: Horizontal Situation Indicator.

HMD: Horizontal Miss Distance.

HUD: Heads-up Display.

Hz: Hertz.

ICAO: International Civil Aviation Organization.

IFF: Identification Friend or Foe.

ILS: Instrument Landing System.

INS: Inertial Navigation System.

kt: Knot.

LOS: Loss of Separation.

LSB: Least Significant Bit.

mbar: Millibar.

ME: Extended Squitter Message.

MHz: Megahertz.

MOPS: Minimum Operational Performance Standards.

MSB: Most Significant Bit.

msec: Millisecond.

MSL: Mean Sea Level.

MTL: Minimum Trigger Level.

NAC: Navigation Accuracy Category.

NACP: Navigation Accuracy Category – Position.

NaN: Not a Number.

NAVSTAR: Navigation Signal Timing and Ranging.

NIC: Navigation Integrity Category.

ND: Navigation Display.

NM: Nautical Miles.

NMAC: Near Midair Collision.

NTA: Number of TCAS or ACAS X-equipped Aircraft.
OCM: Operational Coordination Message
PFD: Primary Flight Display.
PPM: Pulse Position Modulation.
Q-bit: Quantization-bit
RA: Resolution Advisory.
RFLG: Range Flag.
RL: Reply Length.
RMS: Root Mean Square.
RSS: Root Sum Squared.
SAE: Society of Automotive Engineers.
SDA: System Design Assurance.
SEM: Signal Environment Model.
sec: Second(s).
SIL: Source Integrity Level.
SL: Sensitivity Level.
SLC: Sensitivity Level Control.
SSR: Secondary Surveillance Radar.
TA: Traffic Advisory.
T: Time.
TCAS: Traffic Alert and Collision Avoidance System.
TFC: Traffic.
TIS-B: Traffic Information System – Broadcast.
TRP: Total Radiated Power.
UF: Uplink Format.
UUT: Unit Under Test.
VMD: Vertical Miss Distance.
VOR: Very High Frequency Omni-directional Range.
VRC: Vertical Resolution Advisory Complement.
VSI: Vertical Speed Indicator.
VSL: Vertical Speed Limit.
VSWR: Vertical Standing Wave Ratio.
WGS 84: World Geodetic System 1984.
WX: Weather.

1.9**References**

The following documents contain TCAS-related information necessary for the complete definition and understanding of this MOPS. They are referenced by code-letter throughout this document.

<u>Ref.</u>	<u>Title</u>
A.	"Minimum Operational Characteristics-Airborne ATC Transponder Systems", RTCA/DO-144, March 12, 1970, and all revisions thereto.
B.	"Minimum Operational Performance Standards for the Air Traffic Control Radar Beacon System/Mode Select (ATCRBS/MODE S) Airborne Equipment", RTCA/DO-181E, March 17, 2011, and all revisions thereto. "Minimum Operational Performance Specification for Secondary Surveillance Radar Mode S Transponders", EUROCAE ED-73, Revision E, May 1, 2011, and all revisions thereto.
C.	"Software Considerations in Airborne Systems and Equipment Certification", RTCA/DO-178C, December 13, 2011, and all revisions thereto.
D.	"Environmental Conditions and Test Procedures for Airborne Equipment", RTCA/DO-160F, December 6, 2007, and all revisions thereto.
E.	"Radio Altimeter", ARINC Characteristic 707-6, February 6, 1985, and all revisions thereto.
F.	"TCAS II: Design and Validation of the High-Traffic-Density Surveillance Subsystem", DOT/FAA/PM-84/5.
G.	"Minimum Operational Performance Standards for 1090 MHz Extended Squitter Automated Dependent Surveillance – Broadcast (ADS-B) and Traffic Information Services – Broadcast (TIS-B)", RTCA/DO-260B, December 2, 2009, and all revisions thereto. "Minimum Operational Performance Standards for 1090 MHz Extended Squitter Automated Dependent Surveillance – Broadcast (ADS-B) and Traffic Information Services – Broadcast (TIS-B)", EUROCAE ED-102, January 1, 2012, and all revisions thereto.
H.	"Transport Category Airplane - Electronic Display Systems", Advisory Circular 25-11B, October 7, 2014, and all revisions thereto.
I.	"Crew Station Lighting - Commercial Aircraft", SAE Aerospace Recommended Practice ARP 1161A, November 2002, and all revisions thereto.
J.	"Flight Deck Alerting System (FAS)", SAE Aerospace Recommended Practice ARP 4102/4, July 1988, and all revisions thereto.
K.	"Minimum Operational Performance Standards for the Mode S Airborne Data Link Processor", RTCA/DO-218B, June 12, 2001, and all revisions thereto.
L.	"Traffic Computer (TCAS and ADS-B Functionality)", ARINC Characteristic 735B-1, 2012, and all revisions thereto.
M.	"Technical Provisions for Mode S Services and Extended Squitter", ICAO DOC 9871, 2008, and all revisions thereto.
N.	Pratap Misra and Per Enge, Global Positioning System: Signals, Measurements, and Performance, Ganga-Jamuna Press, 2001.
O.	"Atmospheric pressure," Encyclopedia Britannica, Vol. 1, 1998.

-
- P. Paul D. Thomas, Conformal Projections in Geodesy and Cartography, U.S. Dept. of Commerce, Coast and Geodetic Survey, Special Publication No. 251.
 - Q. DOT/FAA/PM-83/36 MTR-83W241. System Safety Study of Minimum TCAS II. 1983.
 - R. "Safety Analysis of Proposed Change to TCAS RA Reversal Logic", RTCA/DO-298, November 8, 2005.
 - S. "Final Report on the Safety of ACAS II in the European RVSM Environment", ASARP/WP9/72/D, Version 1.1, May 11, 2006.
 - T. "Minimum Operational Performance Standards (MOPS) for Aircraft Surveillance Applications (ASA) Systems", RTCA/DO-317B, June 17, 2014, and all revisions thereto.
 - U. "Minimum Operational Performance Standards for Traffic Alert and Collision Avoidance System II (TCAS II) Version 7.1 Attachment A (Syntax of Pseudocode)", June 19, 2008, and all revisions thereto.
 - V. A.D. Panken, et al., "Measurements of the 1030 and 1090 MHz Environments at JFK International Airport", MIT Lincoln Laboratory Project Report ATC-390, September 2012.
 - W. M.J. Kochenderfer, L.P. Espindle, J.K. Kuchar, and J.D. Griffith, "Correlated Encounter Model for Cooperative Aircraft in the National Airspace System," Massachusetts Institute of Technology, Lincoln Laboratory, Project Report ATC-344, 2008.
 - X. SESAR 9.47 D10 (Performance objectives and functional requirements for the use of improved hybrid surveillance in European environment), Appendix A.
 - Y. TCAS Program Office, "Post-FRAC Operational Validation Report", Federal Aviation Administration, ACAS_RPT_18_018_V1R0, August 1, 2018.
 - Z. "Design Assurance Guidance for Airborne Electronic Hardware", RTCA/DO-254, April 19, 2000, and all revisions thereto.
 - AA. "Traffic Alert and Collision Avoidance (TCAS) Program Office Functional Architecture Description for the Airborne Collision Avoidance System X" (FAD), November 30, 2015.
 - BB. "Airworthiness Approval of Traffic Alert and Collision Avoidance Systems (TCAS II), Versions 7.0 & 7.1 and Associated Mode S Transponders", AC 20-151C, July 21, 2017.
 - CC. "Minimum Operational Performance Standards (MOPS) for Automatic Flight Guidance and Control Systems (AFGCS)", RTCA/DO-325, December 8, 2010.

This Page Intentionally Left Blank

2 ACAS X EQUIPMENT REQUIREMENTS AND TEST PROCEDURES

2.1 General Requirements

2.1.1 Airworthiness

Design and manufacture of the equipment **shall** (2000) be such that it can be installed without impairing the airworthiness of the aircraft.

2.1.2 General Performance

The equipment **shall** (2001) perform its intended function, and its use as an airborne collision avoidance system **shall** (2002) not create a hazard to the user of the equipment nor to other users of the National Airspace System or other Air Traffic Management systems.

2.1.3 Federal Communications Commission Rules

The equipment **shall** (2003) comply with all applicable rules of the Federal Communications Commission or equivalent national regulations for non-U.S. entities.

2.1.4 Self-Test

The equipment **shall** (2004) contain a self-test function initiated manually by flight crew action (see §2.2.7.1.3).

2.1.5 Performance Monitoring

The equipment **shall** (2005) contain a performance monitor function. At a minimum, the performance monitor function **shall** (2006):

- c. Automatically monitor for proper operation, on a continual basis, all pertinent ACAS X functions (see §2.2.7).
- d. Not interfere with the performance of the intended function of ACAS X.

2.1.6 Interrogation Test Modes

The equipment **shall** (2007) provide the following interrogation test modes for use with the bench test procedures of §2.4. These modes **shall** (2008) not be flight crew selectable.

a. Mode S Test Mode 1

A Mode S interrogation format with a short P6 pulse, but containing a data block whose bit values are all ONE. The interrogation rate **shall** (2009) be 50 per second, nominal.

b. Mode S Test Mode 2

A Mode S interrogation format with a long P6 pulse, but containing a data block whose bit values are all ONE. The interrogation rate **shall** (2010) be 50 per second, nominal.

c. Mode S Test Mode 3

A Mode S interrogation format without the preamble pulses and with a long P6 pulse containing no internal modulation (i.e., no sync or data phase reversals). The interrogation rate **shall** (2011) be 50 per second, nominal.

d. Mode C Test Mode 1

A standard Mode C only all-call interrogation (§2.2.3.7.1). The interrogation rate **shall** (2158) be 50 per second, nominal.

e. Whisper-Shout Test Mode 1

A standard minimum basic whisper-shout sequence for the minimum TCAS as defined in §2.2.4.5.4.1.1. The interrogation sequence rate **shall** (2012) be 10 per second, nominal.

f. Whisper-Shout Test Mode 2

A standard high resolution whisper-shout sequence for the minimum TCAS as defined in §2.2.4.5.4.1.2. The interrogation sequence rate **shall** (2013) be 10 per second, nominal.

g. No-Interrogation Test Mode

A mode in which the ACAS X transmitter is programmed to transmit no interrogations but otherwise is active.

2.1.7 Controls

2.1.7.1 Operation of Controls

The operation of controls intended for use during flight, in all possible combinations and sequences, **shall** (2016) not result in a condition whose presence or continuation would be detrimental to the continued performance of the equipment or other equipment installed on the aircraft.

Note: The operation of controls intended for use during flight should be evaluated to ensure they are logical and tolerant to human error. In particular, where functions are integrated with other system controls, the manufacturer should ensure that unintentional mode switching (i.e. 'ON' to 'STANDBY' or 'OFF') is minimised. This may take the form of a confirmation of mode switching, required by the flight crew. Typically 'Line Select' Keys, 'Touch Screen' or 'Cursor Controlled/Tracker-ball' methods used to change modes should be carefully designed to minimize crew error.

All possible positions, combinations and sequences of pilot accessible controls should not result in a condition detrimental to the continued performance of the equipment or continued safe flight of the aircraft.

The flight crew should be aware, at all times, of the equipment mode. If either the operational state selection or fail warning of the equipment are not visible to the flight crew any change of transponder mode should be annunciated to the flight crew via suitable means. An installation evaluation may be required to determine the adequacy of the annunciation means.

2.1.7.2 Minimum Flight Crew Control Functions

As a minimum, the following manual controls **shall** (2018) be provided for use by the flight crew.

- a. Means of selecting either a "TA/RA", "TA_Only" or "Standby" mode.

Notes:

1. When the ACAS X function is selected to "TA/RA" mode, traffic and RAs can be issued.
2. When the ACAS X function is selected to "TA-Only" mode, only TAs, not RAs, can be issued.
3. When the ACAS X function is turned off, ACAS X is in a standby condition. The term "Standby" may be used to designate this selection. This does not place the transponder(s) on board the aircraft in a standby condition.

- b. Means of initiating the self test (§2.1.4).

Note: For systems operating in the U.S. NAS, guidance is provided in FAA AC 25.1302-1 ("Installed Systems and Equipment for Use by the Flightcrew") in Section 5-6(C)(3)(b).

2.1.8 Display Functions

2.1.8.1 Minimum Display Functions

As a minimum the following display functions **shall** (2019) be provided:

- a. Means of displaying the RA information required in §2.2.6.2.
- b. Means of displaying to the flight crew that own ACAS X equipment has failed (§2.1.4 and §2.1.5).
- c. Means of displaying to the flight crew that the ACAS X equipment or an ACAS X function has been inhibited either automatically or through flight crew action.
- d. Means of displaying the traffic information required in §2.2.6.1.
- e. Means of displaying TAs for at least three targets including range, altitude, and bearing of intruding aircraft.

2.1.9 Equipment Configuration

It is the intention of this specification to permit a manufacturer to establish interfaces between ACAS X elements and obtain regulatory approval of these components of the total system. It is the intention of this specification to allow interchangeability of system components thus defined. For example, ACAS X with a remote control panel may have several panel configurations available to meet the needs of various aircraft types. The manufacturer is allowed to establish control panel interface standards and receive approval of the various panel designs without performing tests on the total system for each panel design.

2.1.10 Mode S Transponder Capabilities

The ACAS X-equipped aircraft's Mode S transponder must have the capabilities defined in Ref. B.

2.1.10.1 Performance Compatibility with Ownship ACAS X

All of the requirements of a Mode S transponder stated in Ref. B are met when the transponder is operating in conjunction with an operating ACAS X unit except those times that the ACAS X transmitter is active. The active state of the ACAS X transmitter is

defined as either: a) the time interval between 10 microseconds prior to the leading edge of the first transmitted pulse of an interrogation and 10 microseconds after the trailing edge of the last transmitted pulse of that interrogation; or b) the time interval during which a mutual suppression occurs, whichever, a) or b), is greater.

2.1.11

ACAS X Combined with Other Aircraft Systems

This specification does not preclude combining ACAS X with any other aircraft system, or the use of multifunction antennas, displays, or controls. However, any such combination **shall** (2022) not degrade the general performance requirements of the ACAS X equipment, the general performance requirements of the system with which ACAS X is combined, any technical performance specification stated elsewhere in this MOPS nor any technical performance standard for the other system.

2.1.12

Aural Annunciation

Aural annunciations **shall** (2023) be generated when the TA and when the first RA of an encounter is displayed and each time a subsequent change in the RA is displayed (strengthened or weakened), with the exceptions noted in §2.2.6.3.4.

An aural annunciation **shall** (2024) be provided to the flight crew to indicate that ownship is clear of conflict with all threat aircraft. This annunciation **shall** (2025) occur when the RA is cleared.

No aural annunciation **shall** (2026) be issued when ownship is below 400 ft Above Ground Level (AGL) or while a higher priority alert is active (for example GPWS or wind shear).

2.1.13

Shared Use of 1090 MHz Receiver with an ADS-B Receiving Subsystem

ACAS X requires reception of both 1090 MHz TCAS and ADS-B messages. Therefore the requirements for shared use of a 1090 MHz receiver defined in Ref. G, §2.2.4.2 (Receivers Shared With a TCAS II Unit) **shall** (2027) be met.

Note: The cited requirements are designed to ensure (i) that ACAS X does not receive squitters or other asynchronous transmissions from aircraft that are well beyond the maximum range of interest for collision avoidance and (ii) that the Mode S reply processing for ACAS X does not get captured by such transmissions, causing ACAS X to miss higher-power overlapping transmissions from aircraft within the range of interest.

2.2

Minimum Performance Standards

This volume contains many references to specific variable/function names and software structures. (For example, see the "Code" column in Table 2-67.) It is not the intent of this volume to drive the use of these specific names or structures. These are listed to ensure that there is a direct correlation between the requirements called out in this volume and the names and structures defined in the Julia code contained in Volume II. These references are included to ensure that the inputs into the Julia defined algorithms are appropriate and that the outputs from the Julia algorithms are properly interpreted. There is no expectation that the specific names or structures called out in this volume are used. It is not necessary to show that a given structure has been implemented as specified in this volume. It is acceptable to adjust the design elements as appropriate for a given architecture.

2.2.1 Definition Of Standard Conditions

Unless otherwise specified, the signal levels specified in this document are defined at an RF reference point at the antenna end of the cable that connects the ACAS X interrogator/receiver equipment to its antenna. This reference is based on an antenna gain that is 0 dB relative to a matched quarter-wave stub antenna. Specification values in this document are based upon an antenna transmission line loss equal to the maximum for which the ACAS X equipment is designed.

Note: ACAS X may be installed with less than the designed maximum transmission line loss. Nevertheless, the standard conditions of this document are based on the maximum design value. Insertion loss internal to the antenna should be included as part of the net antenna gain.

These performance standards, where applicable, are specified for an avionics configuration that includes both a Mode S transponder and ACAS X equipment. Design specifications relating to any interface between the Mode S transponder and the ACAS X equipment are not covered in detail.

2.2.1.1 Measurement Conventions

Pulse Amplitude is measured at the pulse peak.

Pulse Duration is measured between the half voltage points of the leading and trailing edges.

Pulse Rise Time is measured as the time between 10 percent of peak amplitude and 90 percent of peak amplitude on the leading edge of the pulse.

Pulse Decay Time is measured as the time interval between 90 percent of peak amplitude and 10 percent of peak amplitude on the trailing edge of the pulse.

Pulse-to-Pulse Intervals are measured between the half voltage points of their leading edges.

Phase Reversal Location is the 90-degree point of the phase transition.

Phase Reversal Duration is measured between the 10-degree and the 170-degree points of the phase transition.

Phase Reversal Intervals are measured between the 90-degree points of the phase transition.

Range is defined as the slant range to an intruder aircraft.

Altitude is defined as the absolute pressure altitude of an aircraft referenced to 29.92 inches of mercury.

Bearing is defined as the relative bearing of an intruder aircraft referenced to the nose of ownship.

2.2.1.2 Operational Environment

Note: This section defines the maximum aircraft transponder population and the electromagnetic interference levels under which ACAS X is expected to achieve the surveillance performance specified in §2.2.2.2.

2.2.1.2.1 Aircraft Density

Aircraft traffic density is a function of a Radius of Uniform Density, R_o , and is defined as the number of transponder-equipped aircraft within R_o NM of ownship, $N(R_o)$, divided by

$\pi * (R_o)^2$. Transponder-equipped aircraft include Mode S and ATCRBS aircraft, and exclude ownship. For purposes of this characterization, traffic density is uniform in area within R_o and falls off linearly with increasing range beyond R_o . Specifically, the traffic count within any circle of radius R , where R is greater than R_o , is given by $N(R) = N(R_o) * R / R_o$.

The maximum value of R_o and the maximum traffic density for which ACAS X is expected to achieve the surveillance performance of this document depends on whether ACAS X is operating in lower altitude ($< 10,000$ ft) or higher altitude ($\geq 10,000$ ft) airspace. Lower altitude airspace is characterized by relatively higher traffic densities and lower maximum aircraft speeds. Higher altitude airspace is characterized by lower traffic densities and higher maximum aircraft speeds. Lower altitude airspace is defined as being below 10,000 feet MSL and higher altitude airspace is defined as being at or above 10,000 feet MSL.

Table 2-1: Assumptions About Aircraft Density

Airspace	Max Traffic Density (A/C per sq. NM)	R_o (NM)	# of Aircraft within R_o	# of Aircraft within 30 NM
Lower Altitude	0.3	5	24	141
Higher Altitude	0.06	10	19	56

2.2.1.2.2 Aircraft Equipage

Note: ACAS X surveillance performance is influenced by the number of Active CAS-equipped aircraft within 30 NM of ACAS X and the proportion of transponder-equipped aircraft that carry Mode S transponders. A maximum density of 0.3 aircraft per sq. NM in lower altitude airspace results in 24 aircraft within 5 NM of ACAS X and 141 aircraft within 30 NM of ACAS X. Theoretically, all of these aircraft could be Active CAS-equipped in which case they would also have a Mode S transponder. Analysis has shown that ACAS X, designed according to the requirements of this document, is able to perform satisfactorily in lower altitude airspace in a density of 0.3 aircraft per sq. NM under any mix of ATCRBS, Mode S, and Active CAS-equipped aircraft.

In higher altitude airspace, a density of 0.06 aircraft per sq. NM results in 19 aircraft within 10 NM of ACAS X and 56 aircraft within 30 NM of ACAS X, all of which can be equipped with Active CAS and a Mode S transponder.

The actual distribution of aircraft equipage in lower altitude airspace of 0.3 aircraft per sq. NM density is expected to be at least 60% Mode S transponder-equipped (85 Mode S aircraft within 30 NM of ACAS X) and at least 40% Active CAS-equipped (56 Active CAS aircraft within 30 NM of ACAS X).

2.2.1.2.3 Interrogator Environment

Note: This section defines the maximum design level of electromagnetic interference to ACAS X that results from interrogations caused by ground and airborne interrogators.

2.2.1.2.3.1 Maximum ATCRBS Fruit Rate

Note: The surveillance requirements referenced in §2.2.2.2 are applicable when ACAS X is operating in an ATCRBS fruit environment in which the fruit rate received at the ACAS X RF reference point is defined by the reply power distribution shown in Table 2-2.

Table 2-2: Maximum ATCRBS Fruit Rate

<i>Reply Power Level</i>	<i>Fruit Rate Per Second</i>
$\geq -71 \text{ dBm}$	30,000
$\geq -68 \text{ dBm}$	15,000
$\geq -65 \text{ dBm}$	7,500

2.2.1.2.3.2 Maximum Mode S FRUIT Rate

ACAS X must be capable of operating in a Mode-S interference environment in which the FRUIT rate received at the ACAS X receive reference point is as defined by the reply power distribution shown in Table 2-3.

Table 2-3: Maximum Mode S FRUIT Rates

<i>Receive Power Level</i>	<i>FRUIT Rate per Second</i>
$\geq -84 \text{ dBm}$	21800
$\geq -74 \text{ dBm}$	5900
$\geq -71 \text{ dBm}$	3400
$\geq -68 \text{ dBm}$	2000

Note: FRUIT rates based on an assumed victim receiver height of 30,000 ft with 3000 aircraft within 300 NM of the victim receiver.

2.2.1.2.3.3 Maximum Co-Site Transmitter Rate

Co-Site transmissions are any transmissions near enough to the ownship receiver to warrant inhibiting reception in order to protect receiver components. Sources of co-site transmissions include replies to ATCRBS and Mode-S interrogations, ADS-B transmissions, and DME transmissions. The average number of transmissions per second along with the duration of each transmission is shown in Table 2-4.

Table 2-4: Aircraft Co-Site Transmissions

<i>Co-Site Transmission Type</i>	<i>Average per Second</i>	<i>Duration (us)</i>
ATCRBS Reply	53	58
Mode S Reply	23	100
ADS-B Transmission	6.2	165
DME Transmission	70	44

Note: Durations shown are based on industry standard recommendations for mutual suppression intervals for these co-site transmissions.

2.2.1.2.3.4 Intruder Transponder Availability

Note: In an aircraft density of 0.3 aircraft per sq. NM and except in the most severe interrogator environment in which ACAS X is expected to operate, an intruder

transponder will be subject to a ground beacon radar interrogation rate of 200 per second and a ground beacon sidelobe suppression rate of 1000 per second. These ground transmissions will cause an intruder transponder to be unavailable for a successful ACAS X surveillance interrogation 5% of the time, assuming nominal recovery times for transponder interrogations and suppressions. Taking into account an additional maximum of 2% due to multilateration operation, an intruder transponder will be occupied by ground systems a total of 7% of the time. In addition, the ACAS interference limiting function is intended to limit the unavailability of an intruder transponder to a successful ACAS X interrogation because of other Active CAS interrogations to a maximum of 2%.

The combined ground beacon radar and airborne Active CAS interrogation and suppression rates in the most severe interrogator environment will cause no more than a 9% intruder transponder reply failure rate to ACAS X surveillance interrogations.

RTCA/DO-185B uses 7% transponder reply failure rate. An additional 2% has been added to account for new sources of interrogations, such as multilateration (e.g. Wide Area Multilateration (WAM), Airport Surface Detection Equipment – X (ASDE-X), etc...) systems.

2.2.1.2.3.5 Mode S Transponder Compatibility

Although additional capabilities are supported when ACAS X is installed with ED-73F / RTCA/DO- 181F and ED-102A / RTCA/DO-260C equipment, ACAS X is designed to operate with ED-143 / RTCA/DO- 185A/B compatible transponder versions and RTCA/DO-260B.

2.2.1.2.4 Maximum Level Of ATCRBS Synchronous Garble Interference

Note: The degree of ATCRBS synchronous garble depends on the number of ATCRBS transponder aircraft that are within 1.7 NM of one another relative to the ACAS X interrogator. In an aircraft density of 0.3 aircraft per sq. NM, as specified in §2.2.1.2.1, the maximum level of synchronous garble interference to surveillance of an ATCRBS transponder-equipped aircraft will occur when the aircraft is at approximately 5 NM range to ACAS X. In this density and at a range of approximately 5 NM from ACAS X, the maximum number of other ATCRBS transponder aircraft within ±1.7 NM of the ATCRBS target-of-interest is 22 when 100% of the transponder aircraft are ATCRBS equipped.

In reality, the percentage of aircraft that are ATCRBS-equipped is expected to be much less than 100% (§2.2.1.2.2) and will decrease further as more aircraft are fitted with Mode S transponders, therefore reducing the level of synchronous garble. Also, the percentage of ATCRBS-equipped aircraft in an airspace will vary from region to region. §2.2.4.5.4 addresses requirements that allow ACAS X to optimize its Mode C surveillance based on an estimate of the expected level of synchronous garble in a particular airspace.

2.2.1.2.5 Maximum Level Of Multipath Interference

2.2.1.2.5.1 Transponder Reply

Note: Received transponder replies used for ACAS X surveillance will be subject to interference from delayed and distorted replicas of the same replies reflected from ground and water surfaces. The degree of interference depends on the type of

reflecting surface and whether the ACAS X antenna used for reception and the target transponder antenna used for transmission are top- or bottom-mounted.

Measurements have shown that the largest multipath signal amplitudes, relative to the direct signal, occur when the reply path is from the bottom intruder antenna to the bottom ACAS X antenna and the reflection results from smooth water surfaces. For this situation, the reflected multipath signal level can equal, and occasionally exceed, the level of the direct reply. Use of a top-mounted ACAS X antenna will generally reduce the level of the multipath signal reflected from smooth water surfaces to approximately -15 dB relative to the direct reply.

The surveillance requirements referenced in §2.2.2.2 apply when ACAS X is subjected to this level of multipath interfering signals.

2.2.1.2.5.2 ACAS X Interrogation

Note: ACAS X interrogation of transponders will be subject to interference from delayed replicas of the same interrogating signal reflected from ground and water surfaces. The effect can produce conversion of an ACAS X Mode C interrogation to a transponder suppression signal or a Mode A interrogation.

The relative levels of interrogation multipath signals on the interrogation link are the same as those described in §2.2.1.2.5.1 for the reply link. The surveillance requirements referenced in §2.2.2.2 apply when ACAS X is subjected to these interfering signals.

2.2.1.3 Target-Of-Interest

Note: A target-of-interest is defined as a transponder-equipped intruder aircraft with altitude reporting capability.

A target-of-interest is further defined by the following parameters.

2.2.1.3.1 Maximum Closing Speed

Note: In lower altitude airspace the maximum relative closing speed between a target-of-interest and ACAS X is assumed to be 500 knots and occurs when both aircraft are flying directly towards each other.

In higher altitude airspace the ACAS X surveillance requirements referenced in §2.2.2.2 are applicable for those intruder targets-of-interest that have maximum relative head-on closing speeds of up to 1200 knots. The presence of closing speeds greater than 1200 knots will not affect the performance required against intruders closing at speeds of up to 1200 knots.

Lower altitude airspace is defined as being below 10,000 feet MSL and higher altitude airspace is defined as being at or above 10,000 feet MSL.

2.2.1.3.2 Closing Speed As A Function Of Azimuth

Note: For given target-of-interest and Ownship speeds, the maximum value of the relative closing speed for which ACAS X must satisfy the provisions of §2.2.2.2 will occur only when the intruder aircraft is flying towards ACAS X and the relative velocity vector is aligned at 0 degrees azimuth with respect to ACAS X. As the azimuth of the relative velocity vector changes from 0 degrees, the value of the relative closing speed for which ACAS X surveillance requirements referenced in §2.2.2.2 apply will decrease.

In lower altitude airspace, variable aircraft speeds up to a maximum value of 250 knots will result in an expected relative closing speed of no greater than 229 knots for intruders approaching from the side and no greater than 150 knots for intruders approaching from the rear.

In higher altitude airspace, variable aircraft speeds up to a maximum of 600 knots will result in an expected relative closing speed of no greater than 566 knots for intruders approaching from the side and no greater than 400 knots for intruders approaching from the rear.

2.2.1.3.3 Elevation Angle Relative To ACAS X

Note: The elevation angle associated with a target-of-interest is within ±10 degrees relative to ownship flight path vector.

2.2.1.3.4 Altitude Relative To ACAS X

Note: The altitude of a target-of-interest is within ±3000 feet relative to ACAS X.

2.2.1.3.5 Minimum AGL Altitude

Note: The altitude of the target-of-interest is at least 200 feet above ground level.

2.2 System Performance

Note: When operating within the maximum aircraft transponder population and electromagnetic interference levels defined in §2.2.1.2, ACAS X will provide a level of performance for surveillance of targets-of-interest that will support the requirements for generation of collision advisory information.

Specifically, ACAS X will generate a surveillance track in range and altitude on a target-of-interest at the range and with the track probability and range accuracy specified below. This is to ensure that a correct RA can be issued in time for the pilot to maintain adequate vertical separation at closest-point-of-approach.

ACAS X will also generate, whenever possible, a surveillance track in range and altitude on a target-of-interest at the range and with the track probability and range accuracy specified below such that a correct TA can be issued as a precursor to the RA.

In addition to the surveillance requirements to support generation of RAs and TAs, ACAS X will display the range and, if available, the altitude and bearing position information on targets that generate advisories. The bearing position information will be generated according to the accuracy requirement specified below.

ACAS X will also generate for display, whenever possible, surveillance range, altitude and bearing position information on Mode C and Mode S aircraft that are within the range specified below and within ±10,000 ft altitude relative to ACAS X when airborne, and within 3,000 ft when on the ground.

2.2.2.1 Surveillance Range

2.2.2.1.1 For Generation Of RAs

Note: In higher altitude airspace ACAS X surveillance is designed to be reliable for intruder closing speeds up to 1200 knots. Applying the maximum horizon time

stored in the TRM logic table, the greatest range at which ACAS X will ever be required to generate an RA is 13.33 NM. To allow time to establish a track, the maximum required surveillance range of a target-of-interest in higher altitude airspace and closing head-on is 14 NM. Because of the reduction in the closing speed for intruders approaching from directions other than head-on, the maximum required surveillance range will be less. For example, the maximum required surveillance range for intruders approaching from the side is 8.8 NM and from the rear, 5.0 NM.

In lower altitude airspace the maximum required surveillance range of a target-of-interest with a head-on closing speed of 500 knots is 4.0 NM. For intruders approaching from the side and rear, the maximum required surveillance range is 2.5 NM and 1.4 NM respectively.

ACAS X will be able to maintain active surveillance in range and altitude on targets-of-interest to a minimum range of 1000 feet and on targets-of-interest that transition through closest-point-of-approach (CPA) when CPA equals or exceeds 1000 feet.

2.2.2.1.2 For Generation Of TAs

Note: A TA should be generated prior to an RA. The limit on ACAS X radiated power due to ATC electromagnetic compatibility considerations may prevent reliable surveillance for generation of timely TAs on targets-of-interest that are closing at the maximum rates.

Whenever possible and within the constraints imposed by ACAS X interference reduction power limiting (§2.2.3.6), ACAS X will establish a surveillance track on a target-of-interest at a range that is consistent with the time that a TA is to be generated relative to the time of the RA.

2.2.2.2 Surveillance Performance Objectives

Note: The ACAS X equipment will meet or exceed the surveillance performance for both ATCRBS and Mode S targets as summarized in this section if each of the specific surveillance requirements in §2.2.3 and §2.2.4 and the flight test requirement referenced in §2.2.2.2.2 are satisfied.

2.2.2.2.1 General Surveillance Performance

Note: When operating within the environment defined in §2.2.1.2, ACAS X will provide the following level of surveillance track reliability and accuracy on targets-of-interest (§2.2.1.3) at the ranges specified in §2.2.2.1 for generation of RAs.

2.2.2.2.1.1 Probability Of Track

Note: The design objective for the probability that a surveillance track on an actual aircraft, consisting of either an updated or coasted value for range and altitude, exists on a given cycle is that it be equal to or greater than 90%.

2.2.2.2.1.2 False Track Rate

Note: A false track is defined as a track for which an aircraft does not exist and can be the result of multipath interference, synchronous garble, fruit, noise, etc.

The design objective for the probability that a Mode S transponder-equipped aircraft track exists in the absence of an actual Mode S aircraft is that it be less than 0.1%.

The design objective for the probability that an ATCRBS transponder-equipped aircraft track exists in the absence of an actual ATCRBS aircraft is that it be less than 1.2%.

2.2.2.2.2 Flight Test Surveillance Performance

Note: In addition to the general surveillance performance objectives of §2.2.2.2.1, ACAS X will provide the level of surveillance track performance specified in §3.4.4, “Certification Flight Test Procedures.”

2.2.2.2.3 Range Accuracy

Note: The overall range accuracy achievable by ACAS X depends on the following error contributions to the measurement of reply arrival time:

- a. transponder reply delay and jitter
- b. apparent reply jitter due to the signal delay difference between the diversity transponder top and bottom antenna channel (including cable and antenna location)
- c. apparent reply jitter due to transmit and receive delay differences between the ACAS X top and bottom antenna channel (including cable and antenna location)
- d. ACAS X range clock quantization
- e. ACAS X system noise

The first two error sources are associated with the transponder and are not under control of the ACAS X manufacturer. Never-the-less, for received power levels at least 6 dB above MTL and for transponders meeting the characteristics specified in Mode S MOPS, RTCA/DO-181A or later, ACAS X should be able to achieve an overall range measurement with an error not exceeding 50 ft rms jitter for both Mode C and Mode S reports and 250 ft bias for Mode C reports and 125 ft bias for Mode S reports.

Independent of transponder effects, the ACAS X-only range measurement error (considering only c, d and e above) will not exceed 35 ft rms jitter for either Mode C or Mode S reports for received power levels at least 6 dB above MTL.

2.2.2.4 Bearing Accuracy

Note: The error associated with the ACAS X measurement of target bearing is as specified in §2.2.4.6.6.

2.2.3 Compatibility With Other Systems

The following subparagraphs of §2.2.3 address specific design features that are required of an ACAS X in order to comply with the ATCRBS, Mode S and regulatory standards or MOPS regarding electromagnetic and signal compatibility with elements of the Air Traffic Control System and to ensure standardization of interface signals between ACAS X and other systems on the ACAS X-equipped aircraft.

2.2.3.1 Radiated Output Power

The power-gain product or Effective Radiated Power (ERP) in the forward direction associated with each radiated transmission pulse **shall** (1000) be a maximum of +56 dBm (400 W), and a minimum of +52 dBm (160 W), assuming full power operation.

The power-gain product is related to Total Radiated Power (TRP), a variable used in the interference limiting algorithms, according to the following expression:

$$TRP = P \cdot G \cdot (BW/360 \text{ deg})$$

where P is the net power delivered to the RF reference point, G is the peak azimuth antenna gain at 0 degrees elevation relative to a matched quarter wave stub ($P \cdot G$ is the power-gain product), and BW is the 3 dB azimuth beamwidth in degrees. The specified limit on radiated power is intended to prevent an excessive power transmission from causing premature interference limiting and in turn an unnecessary reduction in the dynamic range of the whisper-shout sequence.

2.2.3.2 Unwanted Output Power

When the ACAS X interrogator is in the inactive state, the RF power at 1030 ± 3 MHz at the terminals of the antenna **shall** (2028) not exceed -70 dBm. The inactive state is defined to include the entire period between ATCRBS and/or Mode S interrogations less 10 microsecond transition periods, if necessary, preceding and following the extremes of the interrogation transmission.

Note: This power restriction is necessary to ensure that ACAS X does not prevent the on-board Mode S transponder from meeting its sensitivity and interference rejection requirements. It assumes that the isolation between the ACAS X antenna and the Transponder antenna exceeds 20 dB.

2.2.3.3 Interrogation Spectrum

The spectrum of a ACAS X Mode S or ATCRBS interrogation **shall** (1001) not exceed that specified in Table 2-5.

Table 2-5: Interrogation Spectrum

Frequency Difference (MHz From Carrier)	Maximum Relative Power (dB Down From Peak)
≥ 4 and < 6	6
≥ 6 and < 8	11
≥ 8 and < 10	15
≥ 10 and < 20	19
≥ 20 and < 30	31
≥ 30 and < 40	38
≥ 40 and < 50	43
≥ 50 and < 60	47
≥ 60 and < 90	50
≥ 90	60

2.2.3.4 Interrogation Jitter

The transmission time of each ATCRBS interrogation sequence and each TCAS Broadcast Interrogation **shall** (2029) be intentionally jittered about their nominal update interval. The jitter **shall** (1002) vary randomly, and be sufficient to prevent synchronous interference with other ground-based and airborne interrogators. The maximum value of the jitter **shall** (1003) not exceed $\pm 10\%$ of the nominal update interval as defined in §2.2.4.1 and §2.2.3.9.2.4.

Note: It may not be necessary to intentionally jitter Mode S surveillance interrogations if the Mode S interrogation scheduling process used has an inherently random nature.

2.2.3.5 Transmit Frequency And Tolerance

The transmission frequency **shall** (1004) be 1030 ± 0.01 MHz.

2.2.3.6 Interference Limiting

The surveillance and collision avoidance functions of the ACAS X equipment will be capable of reliable operation in the traffic densities described in §2.2.1.2. However, the ACAS X equipment must operate in regions of high traffic density without degrading the ATC electromagnetic environment. The following paragraphs specify means for controlling electromagnetic interference to ATC by limiting interrogation rates and powers.

2.2.3.6.1 Interference Limiting Formulas

ACAS X equipment at or below 18,000 ft barometric altitude **shall** (1005) control its interrogation rate or power or both to minimize interference effects; and it **shall** (1006) conform to a set of three specific inequalities, which are a means of ensuring that all interference effects resulting from these interrogations, together with the interrogations from all other Active CAS airborne interrogators in the vicinity, are kept to a low level. The limits on interrogation rate and power are functions of the local airborne environment. In the process of checking for compliance with the limits, ACAS X **shall** (1007) count the number of other Active CAS airborne interrogators in the vicinity. This count (NTA) **shall** (1008) be obtained by monitoring Mode S TCAS Broadcast Interrogation Messages with uplink format UF=16 (§2.2.3.8.3.2.4.2, §2.2.3.9.2.4). This transmission indicates that the aircraft is equipped with an Active CAS interrogator that is currently interrogating. The MU field of this interrogation includes the ICAO 24-bit Aircraft Address of the transponder on the Active CAS aircraft. Each ACAS X that is currently interrogating **shall** (1009) spontaneously transmit these interrogations at nominal 10-second intervals and at full power. Each ACAS X **shall** (1010) monitor the receipt of such interrogations by own Mode S transponder to determine the number of other Active CAS units within detection range (nominally 30 NM). Once each second, NTA **shall** (1011) be updated as the number of distinct Active CAS addresses monitored within the previous 20-second period.

In addition to knowledge of NTA, proper operation of inequality 1 also depends on knowing the distribution of Active CAS interrogators in close vicinity. Each scan, ACAS X **shall** (1012) estimate the distribution of other Active CAS aircraft by using range information derived from Mode S surveillance of those aircraft from which TCAS Broadcast Interrogation Messages have been received. In order to achieve an accurate estimate of the distribution regardless of the level of interference limiting, ACAS X includes only those Active CAS aircraft that are within 6 NM range.

Each scan, the ACAS X distribution function within 30 NM and within 6 NM **shall** (1014) be characterized in the following manner. The distribution of Active CAS within 30 NM **shall** (1015) be computed as the ratio of the number of Active CAS within 30 NM (i.e., NTA) to the number within 6 NM. This ratio is used to calculate an ACAS X distribution factor, α_2 , according to the following formula:

$$\alpha_2 = \log_{10} \left[\frac{NTA}{NTA_6} \right] / \log_{10} 25 ,$$

where NTA is the Active CAS count derived from monitoring all TCAS Broadcast Interrogation Messages and NTA_6 is the number of aircraft from which TCAS Broadcast Interrogation Messages have been received that are estimated to be within 6 NM.

The distribution of Active CAS within 6 NM **shall** (1016) be computed as the ratio of the number of Active CAS within 6 NM to the number within 3 NM. This ratio is used to calculate an ACAS X distribution factor, α_1 , according to the following formula:

$$\alpha_1 = 1/4 [NTA_6/NTA_3],$$

where NTA_3 is the number of aircraft from which TCAS Broadcast Interrogation Messages have been received that are estimated to be within 3 NM.

ACAS X aircraft that are operating on the ground or at or below a radio altitude of 2000 ft AGL **shall** (1017) include both airborne and on-ground Active CAS in the value for NTA_3 and NTA_6 . Otherwise, ACAS X **shall** (1018) include only airborne Active CAS in the value for NTA_3 and NTA_6 . Identification and surveillance of on-ground Active CAS can be accomplished by examining the VS field in the DF=0 acquisition reply to a UF=0 acquisition interrogation and by correlating the ICAO 24-bit Aircraft Address associated with the reply with the address received in the TCAS Broadcast Interrogation.

The maximum value of α_1 and α_2 **shall** (1019) not exceed 1 which defines a uniform-in-area Active CAS distribution and the minimum value of α_1 and α_2 **shall** (1020) not be allowed to decrease below 0.5 which defines a uniform-in-range Active CAS distribution. These constraints on α_1 and α_2 are imposed to ensure that the ACAS X surveillance range is always adequate for collision avoidance and that the maximum allowable interrogation power-rate product is always maintained at a reasonable level regardless of the measured distribution.

The value of α_1 **shall** (1021) be further constrained according to the following logic expressions:

IF $[(NTA_6 \leq 1) \text{ OR } (NTA_6 \leq 4 \text{ AND } NTA_3 \leq 2 \text{ AND } NTA > 25)]$ THEN $\alpha_1 = 1.0$

IF $[(NTA_3 > 2) \text{ AND } (NTA_6 > 2NTA_3) \text{ AND } (NTA < 40)]$ THEN $\alpha_1 = 0.5$

The first expression addresses the case in which the measured Active CAS aircraft count within 6 NM is so low as to provide an unreliable estimate of the distribution factor, α_1 , whereas the total Active CAS count within 30 NM is large enough relative to NTA_6 to indicate a uniform-in-area distribution. The second expression addresses the case in which the measured Active CAS count within 3 and 6 NM indicates a localized uniform-in-area distribution but the total Active CAS count within 30 NM is low enough relative to NTA_3 to indicate an overall uniform-in-range distribution.

In order to minimize scan-to-scan variations in the value of α_1 or α_2 due to track fluctuations, the value of α_1 or α_2 **shall** (1022) be updated each scan using the following recursive filter:

$$\alpha = \alpha_{\text{last scan}} + 0.2(\alpha_{\text{this scan}} - \alpha_{\text{last scan}})$$

The value of α used in interference limiting inequality 1 **shall** (1023) be selected as the minimum of the smoothed values of α_1 and α_2 .

The three inequalities are:

$$1) \sum_{i=1}^I \left[\frac{P(i)}{250 \text{ watts}} \right]^\alpha \leq \text{the smaller of } \left[\frac{280}{NTA+1}, \frac{11}{\alpha^2} \right]$$

$$2) \sum_{i=1}^I M(i) \leq 0.01 \text{ second}$$

$$3) \frac{1}{B} \sum_{k=1}^K \left[\frac{PA(k)}{250 \text{ watts}} \right] \leq \text{the smaller of } \left[\frac{80}{NTA+1}, 3 \right]$$

Variables in these inequalities are defined as follows:

I is the total number of interrogations excluding air-to-air coordination interrogations transmitted by own ACAS X in a surveillance update interval.

i is the index number for all interrogations other than coordination interrogations; $i=1, 2, \dots, I$.

$P(i)$ is the total radiated power (in watts) from the antenna for the i -th interrogation.

NTA is the number of airborne Active CAS interrogators detected with a transponder receiver threshold of -74 dBm.

B is the beam sharpening factor (ratio of interrogation beamwidth to reply beamwidth resulting from interrogation sidelobe suppression).

$M(i)$ is the duration of the mutual suppression interval for own transponder associated with the i -th interrogation.

K is the total number of Mode C interrogations transmitted by own ACAS X in a surveillance update interval.

k is the index number for Mode C interrogations; $k=1, 2, \dots, K$.

$PA(k)$ is the total radiated power (in watts) from the antenna for the k -th Mode C interrogation.

α is a variable derived from surveillance of Active CAS aircraft and is used to match inequality 1 to the measured distribution of nearby Active CAS aircraft.

Notes:

1. *The factor B is approximately 1.2 for a four-beam directional antenna.*
2. *The three inequalities are associated with the following physical mechanisms: 1) reduction in "on" time of other transponders caused by Active CAS interrogations, 2) reduction in "on" time of own transponder caused by mutual suppression during transmission of interrogations, and 3) ATCRBS fruit caused by Active CAS Mode C interrogations.*

Inequality 1 ensures that a "victim" transponder will never detect more than 280 Active CAS interrogations in a one-second period from all the Active CAS interrogators within 30 NM. The left-hand side of the inequality allows an ACAS X unit to increase its interrogation rate if it transmits at less than 250 W since low-power transmissions are detected by fewer transponders. The denominator of the

first term on the right-hand side of this inequality accounts for other Active CAS interrogators in the vicinity and the fact that all Active CAS units must limit their interrogation rate and power in a similar manner so that, as the number of Active CAS units in a region increases, the interrogation rate and power from each of them decreases and the total Active CAS interrogation rate for any nearby transponder remains less than 280 per second. If the victim is taken off the air for 35 microseconds by suppression or reply dead time whenever it receives an Active CAS interrogation, the total "off" time caused by Active CAS interrogations will then never exceed 1%. The term $11/\alpha^2$ on the right-hand side ensures that an individual ACAS X unit with a four-beam directional interrogation antenna and operating in a uniform-in-area Active CAS distribution (i.e., $\alpha=1$) never transmits more average power than it would if there were approximately 26 other Active CAS units nearby or if operating in a uniform-in-range distribution (i.e., $\alpha=0.5$) never transmits more average power than it would if there were approximately 6 other Active CAS units nearby.

Inequality 2 ensures that the transponder on board ownship will not be turned off by mutual suppression signals from the Active CAS unit on the same aircraft more than 1% of the time.

Inequality 3 ensures that a victim ATCRBS transponder will not generate more than 40 ATCRBS replies in a one-second period in response to interrogations from all the Active CAS interrogators within its detection range. Like inequality 1 it includes terms to account for reduced transmit power, to account for the other Active CAS interrogators in the vicinity, and to limit the power of a single Active CAS unit. 40 ATCRBS replies per second is approximately 20% of the reply rate for a transponder operating without Active CAS in a busy area of multiple ATCRBS ground sensor coverage.

3. *The three interference limiting inequalities are designed to check if the current Mode C and Mode S surveillance are aligned with the goals of the ACAS X contribution to the communication environment as a system, which are stated in note 2 above. The inequalities are not meant to restrict individual interrogations, but rather modify Mode C and Mode S surveillance parameters so that the inequalities can be satisfied in the future. The procedures above describe how to modify Mode C and Mode S surveillance for interference limiting.*

2.2.3.6.2

Interference Limiting Procedures

At the beginning of each surveillance update interval, α and *NTA shall* (1024) be determined as indicated in §2.2.3.6.1. The value α **shall** (1025) then be used to:

- a. modify each normalized Mode C and Mode S interrogation value in the left hand limits of inequality 1 prior to its summation over the surveillance update interval, and
- b. modify the fixed term in the right-hand limits of inequality 1.

NTA shall (1026) then be used to evaluate the current right-hand limits in inequalities 1 and 3. Smoothed values (8 second averages) of the Mode S contributions to $P(i)$ and $M(i)$ **shall** (1027) also be used to calculate the left hand limits of inequalities 1 and 2. Air-to-air coordination interrogations **shall** (1028) be transmitted at full power. These interrogations need not be included in the summations of Mode S interrogations in the left-hand terms of these inequalities. If the smoothed value of the left-hand side of either inequality 1 or 2 equals or exceeds the current limit and ownship is below 18,000 ft barometric altitude, both the Mode S and the Mode C surveillance parameters **shall** (1029) be modified to satisfy the inequalities. If the left-hand side of inequality 3 exceeds the

current limit and own ACAS X is below 18,000 ft barometric altitude, Mode C surveillance **shall** (1030) be modified to satisfy the inequality. In both cases, hysteresis of ± 500 ft relative to the 18,000 ft altitude boundary **shall** (1031) be applied to prevent fluctuations in interference limiting due to small altitude deviations.

Mode C surveillance sensitivity **shall** (1032) be modified by reducing the interrogation power and by increasing the receiver MTL during the Mode C listening period. If a high resolution whisper-shout interrogation sequence is used (§2.2.4.5.4.1.2), Mode C surveillance **shall** (1033) be modified by sequentially eliminating steps from the whisper-shout sequence. If the minimum basic whisper-shout interrogation sequence or a single interrogation is used (§2.2.4.5.4.1.1 or §2.2.4.5.4.1.5 respectively), Mode C surveillance **shall** (1034) be modified by reducing the power level of each step within the whisper-shout sequence or each single interrogation sequentially for each beam. Each whisper-shout or power reduction step is uniquely associated with a receiver MTL setting. Thus, the receiver sensitivity in Mode C surveillance periods will be automatically tailored to match these power reductions.

The overall surveillance sensitivity for Mode S targets **shall** (1035) be reduced by reducing the interrogation power and by increasing the receiver MTL during all Mode S squitter listening periods. Once such a change has been made the only change allowed during the ensuing 8 seconds is a reduction in the number of whisper-shout steps if such is needed to satisfy Inequality 3. This 8-second freeze allows the effect of the Mode S changes to become apparent since the 8-second averages used in Inequalities 1 and 2 then will be determined by the behavior of the system since the change.

Note: This will indirectly reduce the Mode S interrogation rate by reducing the target count. Many Mode S interrogations are acquisition interrogations transmitted to targets of unknown range. It is thus not effective to control the Mode S interrogation rate directly simply by dropping long-range targets from the track file.

The Mode C and Mode S surveillance sensitivity reductions **shall** (1036) be accomplished such that the ACAS X equipment is not prematurely limited and has the capability of using at least 75% of the allowance specified in the three limiting equations for all mixes of target types and for all densities up to the maximum density capability of the system. When the value of any of the smoothed limits is exceeded, the appropriate action **shall** (1037) be taken to limit interference within one surveillance update interval. Means **shall** (1038) be provided for gradually restoring the surveillance sensitivity when the environment subsequently improves enough to allow the interference limits to be relaxed.

The maximum allowed interference limiting power reduction for each top antenna beam for an airborne ACAS X **shall** (1039) be 10 dB for Mode S interrogations and 7 dB for Mode C interrogations.

Section A.1 in Appendix A describes one possible implementation of the interference limiting procedure.

2.2.3.6.3

Interrogations From ACAS X On The Ground

Whenever ownership determines that it is on the ground, ACAS X interrogations **shall** (1040) be limited by setting the *NTA* count in the interference limiting inequalities to a value three times the measured value. This value will ensure that an ACAS X operating on the ground does not add unnecessary interference by transmitting at power levels greater than is necessary to provide local area surveillance prior to departure. The modified value of *NTA* will provide an approximate surveillance range of 3 NM in the highest density terminal areas to support reliable ground ACAS X surveillance of local airborne traffic and a 14

NM range in very low density airspace to provide wide area surveillance in the absence of an SSR. When ACAS X transitions from standby to TA or TA/RA mode and ownship determines that it is on the ground, it **shall** (1041) initialize to the maximum allowed airborne interference limiting values, 10 dB attenuation for Mode S interrogations and 7 dB for Mode C interrogations

The interference limiting adjustment of interrogation power and receiver MTL for ACAS X on the ground **shall** (1042) be accomplished as described in §2.2.3.6.2 except that the Mode C interrogation power and sensitivity in the forward top beam **shall** (1043) be reduced first until the forward whisper-shout sequence matches the sequence in the right and left beams. The forward, right and left beam interrogation powers and sensitivities **shall** (1044) then sequentially be reduced until they match the rear beam interrogation power and sensitivity. Further interference limiting **shall** (1045) be accomplished by sequentially reducing the forward, side and rear beam interrogation power and sensitivity.

The maximum allowed interference limiting power reduction for an ACAS X on the ground **shall** (1046) be as follows:

- a. Forward beam: 13 dB for Mode S and 10 dB for Mode C.
- b. Side beam: 13 dB for Mode S and 6 dB for Mode C
- c. Rear beam: 13 dB for Mode S and 1 dB for Mode C

An ACAS X aircraft **shall** (1047) recover the surveillance range capability that would ordinarily result using interference limiting with a direct value of NTA and an alpha (α) value equal to one within three surveillance update intervals following the transition from an on-the-ground status to an airborne status. The value of alpha **shall** (1048) then be allowed to achieve its normal value according to §2.2.3.6.1.

Appendix A, Section A.12 describes one acceptable method of ensuring that ACAS X recovers its normal airborne surveillance range capability in a timely manner following a transition from on-the-ground status to an airborne status.

2.2.3.6.4

Interrogations From ACAS X Above 18,000 Ft Barometric Altitude

Whenever ownship is flying above 18,000 ft barometric altitude, ACAS X **shall** (1049) not be subject to interference limiting inequality 2 nor to the increased interference limiting provided by inequalities 1 and 3 due to the added presence of other nearby Active CAS-equipped aircraft (NTA). A revised form of interference limiting is still necessary in order to prevent each ACAS X from transmitting unlimited power during each surveillance update interval. To retain the fixed upper limit on interrogation power and rate contained within inequalities 1 and 3 in §2.2.3.6.1, ACAS X equipment above 18,000 ft barometric altitude **shall** (1050) conform to the following interference limiting inequalities. These are simpler than inequalities 1 and 3: the terms $280/(NTA+1)$ and $80/(NTA +1)$ are not used and α equals 1 regardless of the distribution of Active CAS:

$$4) \sum_{i=1}^I \left[\frac{P(i)}{250 \text{ watts}} \right] \leq 11$$

$$5) \frac{1}{B} \sum_{k=1}^K \left[\frac{PA(k)}{250 \text{ watts}} \right] \leq 3$$

2.2.3.7 Transmit Pulse Characteristics

2.2.3.7.1 Mode C Transmissions

Mode C interrogations from ACAS X equipment **shall** (1051) employ the "Mode-C-Only All-Call" format which consists of three pulses P_1 , P_3 , and P_4 . This **shall** (1052) normally be preceded by a Mode C "whisper-shout" suppression pulse designated S_1 (see §2.2.4.5.4.1). Sidelobe suppression is accomplished by transmitting a P_2 pulse via a separate control pattern. These formats are illustrated in Figure 2-1. The pulses **shall** (1053) have shapes and spacings as tabulated below except that the rise and decay time may be less than shown in the table, providing the side-band radiation does not exceed the spectral limits tabulated in §2.2.3.3. The amplitude of P_3 **shall** (1054) be within 0.5 dB of the amplitude of P_1 , and the amplitude of P_4 **shall** (1055) be within 0.5 dB of the amplitude of P_3 . The amplitude of S_1 is specified in §2.2.4.5.4.1.

Table 2-6: Mode C Pulse Shapes

(All values in microseconds)

Pulse Designator	Pulse Duration	Duration Tolerance	Rise Time		Decay Time	
			Min.	Max.	Min.	Max.
S_1, P_1, P_2, P_3, P_4	0.8	± 0.05	0.05	0.1	0.05	0.2

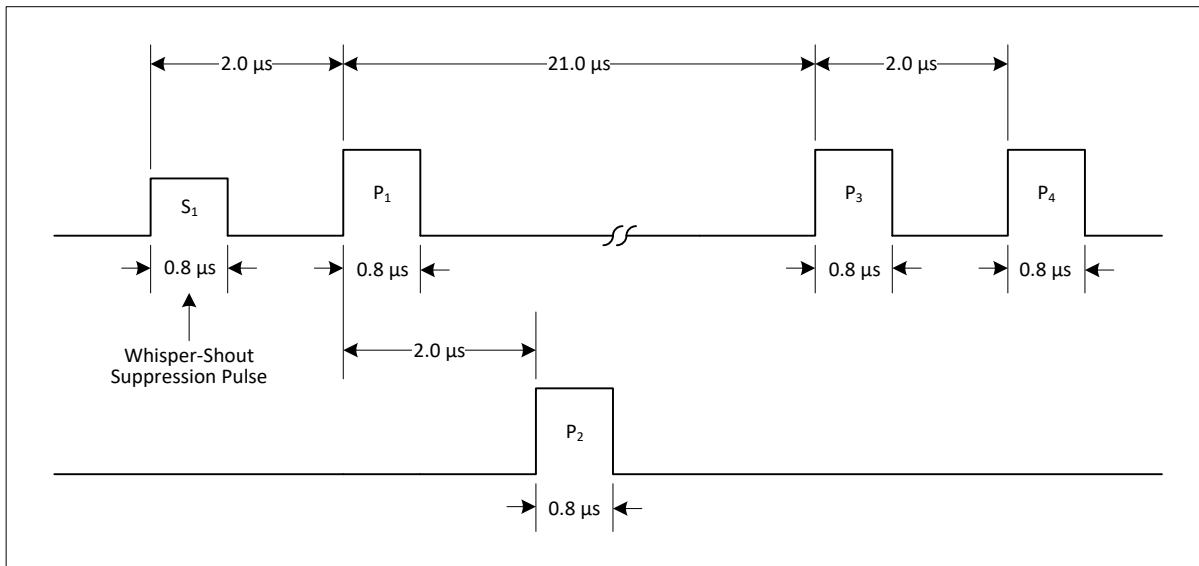


Figure 2-1: Mode C-Only All-Call Interrogation Pulse Sequence for ACAS X

The pulse spacing tolerances **shall** (1056) be as follows:

- S₁ to P₁ 2 ±0.10 microseconds;
- P₁ to P₂ 2 ±0.10 microseconds;
- P₁ to P₃ 21 ±0.10 microseconds;
- P₃ to P₄ 2 ±0.04 microseconds.

Note: The tolerance values on these pulse widths, spacings, and amplitudes are smaller than the signal-in-space tolerance values defined in Ref. B in order to provide margin for waveform distortion due to multipath.

2.2.3.7.2 Mode S Transmissions

Mode S transmissions **shall** (1057) consist of P₁, P₂, and P₆ pulses as shown in Figure 2-2. The pulses **shall** (1058) have shapes and spacings as tabulated below except that the rise and decay time may be less than shown in the table, providing the side-band radiation does not exceed the spectral limits tabulated in §2.2.3.3.

Table 2-7: Mode S Pulse Shapes
(All values in microseconds)

Pulse Designator	Pulse Duration	Duration Tolerance	Rise Time		Decay Time	
			Min.	Max.	Min.	Max.
P ₁ , P ₂	0.8	±0.05	0.05	0.1	0.05	0.2
P ₆ (Short)	16.25	±0.125	0.05	0.1	0.05	0.2
P ₆ (Long)	30.25	±0.125	0.05	0.1	0.05	0.2

The short (16.25-microsecond) and long (30.25-microsecond) P₆ pulses **shall** (1059) have internal modulation consisting of possible 180-degree phase reversals of the carrier at designated times. The first phase reversal in the P₆ pulse is the sync phase reversal and is always present. The presence or absence of a subsequent phase reversal indicates a one or zero in the transmitted code respectively.

Note: The sync phase reversal is the timing reference provided to identify chip positions to Mode S interrogation decoders.

The duration of a phase reversal in P₆ **shall** (1060) be less than 0.08 microseconds as measured between the 10-degree and 170-degree points of the phase transition. The interval between the 80-percent points of the amplitude transient associated with the phase reversal **shall** (1061) be less than 0.08 microseconds.

The tolerance on the 0- and 180-degree phase relationships in P₆ **shall** (1062) be ±5 degrees.

The 90-degree point of each data phase reversal in P₆ **shall** (1063) occur only at a time (N × 0.25) ±0.02 microseconds (N ≥ 2) after the 90-degree point of the sync phase reversal.

Note: 56 or 112 data phase reversals can occur in the 16.25 and 30.25-microsecond P_6 pulses respectively. This results in a 4 Mbit/sec data rate within the P_6 pulses.

The spacing from P_1 to P_2 shall (1064) be 2 ± 0.04 microseconds between leading edges. The spacing from the leading edge of P_2 to the 90-degree point of the sync phase reversal of P_6 shall (1065) be 2.75 ± 0.04 microseconds. The leading edge of P_6 shall (1066) occur 1.25 ± 0.04 microseconds before the sync phase reversal.

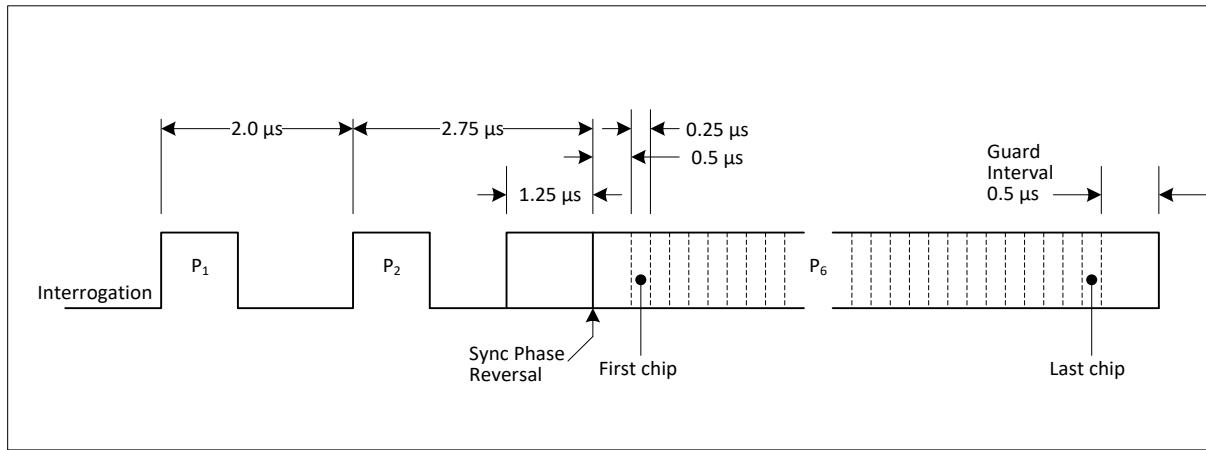


Figure 2-2: Mode S Interrogation Pulse Sequence For ACAS X

Note: The P_1 - P_2 pair preceding P_6 suppresses replies from ATCRBS transponders to avoid synchronous garble due to random triggering of ATCRBS transponders by Mode S interrogations. A series of “chips” containing the information within P_6 starts 0.5 microseconds after the sync phase reversal. Each chip is of 0.25 microsecond duration and is preceded by a possible phase reversal. If preceded by a phase reversal, a chip represents a logic “1”. There are either 56 or 112 chips. The last chip is followed by a 0.5 microsecond guard interval which prevents the trailing edge of P_6 from interfering with the demodulation process.

The radiated amplitudes of P_2 and the initial first microsecond of P_6 shall (1067) be greater than the radiated amplitude of P_1 minus 0.25 dB. The maximum envelope amplitude variation between successive phase modulation chips in P_6 shall (1068) be less than 0.25 dB.

Note: The tolerance values on these pulse widths and spacings and the location of the sync phase reversal are smaller than the signal-in-space tolerance values defined in Ref. B in order to provide margin for waveform distortion due to multipath reflections.

2.2.3.8 Mode S Message Field Formats

Note: Ref. B describes the message formats and coding that are used in specified Mode S transmissions. The following paragraphs further define Mode S formats and information coding used for the transmission of ACAS X messages.

2.2.3.8.1 Data Blocks

The Mode S interrogation and reply data blocks can contain either 56 or 112 bits.

2.2.3.8.2 Format Structure

Mode S transmission formats consist of 56 or 112 bits of which 24 bits are used for address/parity information and the remainder are used for data transfer. The interrogation and reply formats that are used by ACAS X or can contain ACAS X information are presented in Figure 2-3 and Figure 2-4.

Note: Fields shown in these figures that do not specifically pertain to ACAS X are not described in this document, but are described in Ref. B.

2.2.3.8.2.1 Bit Numbering and Sequence

In the description of the information fields contained in Mode S transmissions, bits are numbered consecutively with bit 1 being transmitted first. Numerical values encoded by groups of bits are encoded with the most significant bit (MSB) transmitted first. Bits designated as "not assigned" are transmitted as all 0s, and are reserved for assignment in the future.

2.2.3.8.2.2 Fields

Information is coded in fields that consist of one or more bits. The decimal equivalent of the binary code formed by the bit sequence within a field is used as the designator of the field function.

Note: As an example, the 5-bit UF field is used to designate the uplink format type. A surveillance interrogation requesting identity has in its UF field position the bit sequence 00101. Since $00101_2 = 5_{10}$, this format is designated as UF=5.

Codes designated as "not assigned" or "unassigned" are reserved for future assignment. Fields containing numeric data are encoded in positive binary notation unless otherwise noted. The least significant bit (LSB) value and the allowed range of values are included for all numeric quantities. Unless otherwise noted, when the value of a numeric quantity is outside the indicated range of values, it will be encoded as equal to the nearest end point of the range. Unless otherwise noted, the coded value of a numeric quantity is the quantized value nearest the actual value of the quantity (i.e., the value is rounded off rather than truncated). If the actual value is half way between the two nearest quantized values, the larger of the two quantized values **shall** (1069) be used.

2.2.3.8.2.2.1 Essential Fields

Each Mode S transmission contains two essential fields. The first essential field is a format descriptor which occurs at the beginning of the transmission. This field is designated UF (Uplink Format) for interrogations and DF (Downlink Format) for replies. The UF and DF codes pertinent to this MOPS are given in Figure 2-3 and Figure 2-4 respectively. The second essential field is a 24-bit field which contains either the ICAO 24-bit Aircraft Address or the ground interrogator identity overlaid on parity as described in Ref. B, §2.2.16.2.1. This field is either designated AP (Address/Parity) or PI (Parity/Identity) and is located at the end of the transmission.

Format

No.

0 UF (0 0000) --3-- (RL: 1) --4-- (AQ: 1) (BDS: 8) --10-- (AP: 24) Short Special
Surveillance

4 (0 0100) (PC: 3) (RR: 5) (DI: 3) (SD: 16) (AP: 24) Surveillance,
Altitude Request

5 (0 0101) (PC: 3) (RR: 5) (DI: 3) (SD: 16) (AP: 24) Surveillance,
Identity Request

11 (0 1011) (PR: 4) (II: 4) --19-- (AP: 24) All-Call
Interrogation

16 (1 0000) --3-- (RL: 1) --4-- (AQ: 1) --18-- (MU: 56) (AP: 24) . . . Long Special
Surveillance

20 (1 0100) (PC: 3) (RR: 5) (DI: 3) (SD: 16) (MA: 56) (AP: 24) . . . Comm-A,
Altitude Request

21 (1 0101) (PC: 3) (RR: 5) (DI: 3) (SD: 16) (MA: 56) (AP: 24) . . . Comm-A,
Identity Request

Notes: (1) (XX: M) denotes a field designated "XX" which is assigned M bits.
 (2) ---N--- denotes unassigned coding space, N bits; must be set to all ZEROS.
 (3) The format number in the above uplink formats (UF) corresponds to the binary code in the first 5 bits of the interrogation.

Figure 2-3: Mode S Interrogation or Uplink Formats Used by ACAS X

**Format
No.**

0 DF
 (0 0000) (VS: 1) (CC: 1) -1- (SL: 3) -2- (RI: 4) -2- (AC: 13) (AP: 24). Short
Special Surveillance

4 (0 0100) (FS 3) (DR: 5) (UM: 6) (AC: 13) (AP: 24).....
Surveillance, Altitude

5 (0 0101) (FS: 3) (DR: 5) (UM: 6) (ID: 13) (AP: 24).
Surveillance, Identity

11 (0 1011) (CA: 3) (AA: 24) (PI: 24)..... All-
Call Reply

16 (1 0000) (VS: 1) --2-- (SL: 3) --2-- (RI: 4) --2-- (AC: 13) (MV: 56) (AP: 24)... Long
Special Surveillance

17 (10001) (CA: 3) (AA: 24) (ME: 56) (PI: 24)
Extended Squitter

20 (1 0100) (FS: 3) (DR: 5) (UM: 6) (AC: 13) (MB: 56) (AP: 24) Comm-
B, Altitude

21 (1 0101) (FS: 3) (DR: 5) (UM: 6) (ID: 13) (MB: 56) (AP: 24) Comm-
B, Identity

Notes: (1) (XX: M) denotes a field designated "XX" which is assigned M bits.
 (2) ---N--- denotes unassigned coding space, N bits; must be set to all ZEROs.
 (3) The format number in the above downlink formats (DF) corresponds to the binary code in the first 5 bits of the reply.

Figure 2-4: Mode S Reply or Downlink Formats Used by ACAS X

2.2.3.8.2.2.2 Mission Fields

The remaining coding space is used to transmit the mission fields. For specific missions, a specific set of fields is prescribed.

Note: All mission fields described in this MOPS are assigned two-letter designators.

2.2.3.8.2.2.3 Subfields

Subfields may appear within mission fields. The name, length (number of bits) and coding of a defined subfield are the same wherever the subfield appears; however, the location and protocol for use may vary.

Note: Subfields of mission fields are labeled with three-character alphanumeric designators within this MOPS. Subfields of three-character designator subfields are labeled with four-character alphanumeric designators within this MOPS.

2.2.3.8.3 Field Descriptions

For all mission fields and subfields processed by the airborne ACAS X equipment, the location and coding are described in the following paragraphs. This description duplicates the description found in Ref. B. Mission fields and subfields defined in Ref. B are described in §2.2.3.8.3.1. Mission fields and subfields that are further defined in this document are described in §2.2.3.8.3.2. Table 2-8 provides an index of all described fields and the three-character designator subfield groups (if any) within them.

Table 2-8: Field Index

Field	Sub-Field	Bits		Formats		Reference Paragraphs	
		No.	Position	Up	Down	Content	Protocol
AA	AA	24	9-32		X	2.2.3.8.3.1.1	2.2.3.9.2.1
	AC	13	20-32		X	2.2.3.8.3.1.2	2.2.4.4.2.2(c)
	AP	24	33-56	X	X	2.2.3.8.2.2.1	
	AQ	1	14	X		2.2.3.8.3.1.3	2.2.3.9.2.2.1; 2.2.3.9.2.2.2
	BDS	8	15-22	X		2.2.3.8.3.1.4	2.2.3.9.7
	CA	3	6-8		X	2.2.3.8.3.1.5	2.2.3.9.2.1
	CC	1	7		X	2.2.3.8.3.1.6	2.2.3.9.7
	DF	5	1-5		X	2.2.3.8.2.2.1	
	DI	3	14-16	X		B:2.2.14.4.11*	2.2.3.8.3.1.17
	DR	5	9-13		X	2.2.3.8.3.1.7	2.2.3.12.2.2.2
	FS	3	6-8		X	2.2.3.8.3.1.8	2.2.3.9.2.1
	ID	13	20-32		X	B:2.2.14.4.17*	Figure 2-4
	MA	56	33-88	X		2.2.3.8.3.1.9	
<u>TCAS II Sensitivity Level Command Message</u>							
MB	Subfields of MA:					2.2.3.8.3.2.2	2.2.3.9.5.4
	ADS	8	33-40				
	SLC	4	41-44				
	(-n-)	44	45-88				
<i>Note: The notation (-n-) is used to denote bits that are not assigned.</i>							
MB		56	33-88		X	2.2.3.8.3.1.10	
	<u>Data Link Capability Report</u>					2.2.3.8.3.2.3.2	2.2.3.9.5.2
	Subfields of MB:						
BDS 8 33-40							

Field	Sub-Field	Bits		Formats		Reference Paragraphs		
		No.	Position	Up	Down	Content	Protocol	
<i>Note: Coding of Data Link Capability Report (See 2.2.3.8.3.2.3.2.1)</i>								
<i>Bits 42: OCM Transmit Capability</i>								
ME	43-46: ACAS X Version							
	48: Operational Status							
	69: HS Status							
	70: TA vs TA+RA							
	72,71: TCAS Version							
	<u>RA Report</u>							
	Subfields of MB (RTCA/DO-185A,B systems):					2.2.3.8.3.2.3.1	2.2.3.9.5.1	
	BDS	8	33-40					
	ARA	10	41-50					
	LDI	2	51-52					
	RMF	2	53-54					
	RAC	4	55-58					
	RAT	1	59					
	MTE	1	60					
	CNT	1	61					
	TTI	1	62					
	TID	24	63-86					
	TIDA	11	63-73					
	TIDR	7	74-80					
	TIDB	6	81-86					
	DSI	1	87					
	SPI	1	88					
		56	33-88	X		2.2.3.8.3.1.11		
<u>Operational Coordination Message</u>								
ME	Subfields of ME:					2.2.3.8.3.2.9.1	2.2.3.9.3.6.2.1	
	TYPE	5	33-37					
	Subtype	3	38-40					
	(-n-)	1	41					
	MTB	1	42					
	CVC	2	43-44					
	VRC	2	45-46					
	CHC	3	47-49					
	HRC	3	50-52					
	HSB	5	53-57					
	VSB	4	58-61					
	(-n-)	3	62-64					
	TAA	24	65-88					
<u>Aircraft Operational Status Message</u>								
ME	Subfields of ME:					2.2.3.8.3.2.9.2	2.2.3.9.3.6.1.2	
	TYPE	5	33-37					
	Subtype	3	38-40					
	CC	16	41-56					
	(-n-)	2	41-42					
	CA Op.	1	43					
	(-n-)	11	44-54					

Field	Sub-Field	Bits		Formats		Reference Paragraphs	
		No.	Position	Up	Down	Content	Protocol
MU	DAA	2	55-56				
	OM	16	57-72				
	(-n-)	2	57-58				
	CA RA Act.	1	59				
	(-n-)	5	60-64				
	CCCB	7	65-71				
	Inter. Act. Op.	1	72				
	(-n-)	16	73-88				
		56	33-88	X		2.2.3.8.3.1.12	
	<u>TCAS Resolution Message</u>						
MV	Subfields of MU:					2.2.3.8.3.2.4.1	2.2.3.9.3.1
	UDS	8	33-40				
	(-n-)	1	41				
	MTB	1	42				
	CVC	2	43-44				
	VRC	2	45-46				
	CHC	3	47-49				
	HRC	3	50-52				
	(-n-)	3	53-55				
	HSB	5	56-60				
RA	VSB	4	61-64				
	MID	24	65-88				
	<u>TCAS Broadcast Interrogation Message</u>						
	Subfields of MU:					2.2.3.8.3.2.4.2	2.2.3.9.2.4
	UDS	8	33-40				
	(-n-)	24	41-64				
	MID	24	65-88				
	<u>RA Broadcast Interrogation Message</u>						
	Subfields of MU:					2.2.3.8.3.2.4.3	2.2.3.9.6.1
	UDS	8	33-40				
VDS	ARA	10	41-50				
	LDI	2	51-52				
	RMF	2	53-54				
	RAC	4	55-58				
	RAT	1	59				
	MTE	1	60				
	SPI	1	61				
	(-n-)	1	62				
	AID	13	63-75				
	CAC	13	76-88				
ARA		56	33-88		X	2.2.3.8.3.1.13	
	<u>Coordination Reply Message</u>						
	Subfields of MV (RTCA/DO-185A,B systems):					2.2.3.8.3.2.5.1	
VDS	VDS	8	33-40				
	ARA	10	41-50				

Field	Sub-Field	Bits		Formats		Reference Paragraphs	
		No.	Position	Up	Down	Content	Protocol
	LDI	2	51-52				
	RMF	2	53-54				
	RAC	4	55-58				
	RAT	1	59				
	MTE	1	60				
	(-n-)	28	61-88				
	<u>Mode S Crosslink Message</u>					2.2.3.8.3.1.13	2.2.3.9.7
	Subfields of MV: Contains contents of the requested Mode S register.						
PI		24	33-56		X	2.2.3.8.2.2.1	
RI		4	14-17		X	2.2.3.8.3.1.14	2.2.3.9.2.3.1; 2.2.3.9.2.3.2;
RL		1	9	X		2.2.3.8.3.1.15	2.2.3.12.2.2.1 2.2.3.9.2.2.1; 2.2.3.9.2.2.2;
RR		5	9-13	X		2.2.3.8.3.1.16	2.2.3.9.2.3 2.4.2.3; 2.4.2.4; 2.4.2.5
SD		16	17-32	X		2.2.3.8.3.1.17	
	Subfields of SD:						
	IIS	4	17-20			2.2.3.8.3.1.17	2.2.3.9.5.4
	LAS	3	30-32			2.2.3.8.3.1.17	
SL		3	9-11		X	2.2.3.8.3.2.6	2.2.3.9.2.3; 2.2.3.12.2.2.1
UF		5	1-5	X		2.2.3.8.2.2.1	
UM		6	14-19		X	B: 2.2.14.4.41*	Figure 2-4
VS		1	6		X	2.2.3.8.3.1.18	2.2.3.9.2.1

* The notation "B:" indicates that the cited paragraph is in Ref. B.

2.2.3.8.3.1 Fields and Subfields Defined by Reference B.

Location and coding of mission fields and subfields that are required for transmission of ACAS X information and that are specified by Ref. B are described in alphabetical order in the following paragraphs.

Note: Certain codes in the mission fields described below that are designated "not assigned" in Ref. B are assigned for ACAS X use and are specified in this section. ACAS X mission fields not defined by Ref. B are defined in §2.2.3.8.3.2.

2.2.3.8.3.1.1 AA Address, Announced

This 24-bit (9-32) downlink field contains the aircraft address in the clear and is used in DF=11, the All-Call reply, and in DF=17, the Extended Squitter.

2.2.3.8.3.1.2 AC Altitude Code

When operated in the Mode S system, the 13-bit AC field (bits 20-32 transmitted in the short and long special surveillance reply, the altitude surveillance reply, and the altitude

Comm-B reply) contains the encoded altitude of the aircraft. The altitude is encoded as follows:

- a. Bit 26 is designated as the M bit, and is ZERO if the altitude is reported in feet. M equals ONE is reserved for possible future use to indicate that the altitude reporting is in metric units.

Note: Use of the M bits as defined here does not alter any conventions regarding the X bits in ATCRBS replies.

- b. If M equals ZERO, bit 28 is designated as the Q bit. Q equals ZERO is used to indicate that the Mode S altitude is reported in 100 ft increments as defined in c. below. Q equals ONE is used to indicate that the altitude is reported in 25 ft increments as defined in d. below.

Note: Bit 28 (Q) corresponds to the D1 pulse of a Mode C reply and is not used in the ATCRBS altitude code.

- c. If M and Q both equal ZERO, the altitude is coded according to the pattern of Ref. A, §2.7.13.2.5. Starting with bit 20, the sequence is C1, A1, C2, A2, C4, A4, ZERO, B1, ZERO, B2, D2, B4, D4.

- d. If M equals ZERO and Q equals ONE, the 11-bit field represented by bits 20 to 25, 27 and 29 to 32 represents a binary-coded field whose least significant bit has a value of 25 feet. The binary value of the decimal number N is used to report pressure altitudes in the range (25xN-1000 ±12.5) feet.

Note: The most significant bit of this field is bit 20. This code is able to provide code values only between -1000 ft and +50,175 ft. The coding used for Mode C replies in c. above must be used to report pressure altitudes greater than 50,175 ft.

- e. ZERO is transmitted in each of the 13 bits of the AC field if altitude information is not available.

2.2.3.8.3.1.3 AQ Acquisition, Special

This 1-bit (14) uplink field appears in special surveillance formats UF=0, 16. This field is used to designate the UF=0 or 16 interrogation as an acquisition transmission (§2.2.3.9.2.2.1).

The codes are:

- | | |
|---|----------------------------------|
| 0 | = Non-acquisition interrogation. |
| 1 | = Acquisition interrogation. |

This bit is repeated as received by the transponder in bit 14 of the RI field of DF=0, 16 (§2.2.3.8.3.1.14).

2.2.3.8.3.1.4 BDS: Comm-B Data Selector

This 8-bit (15-22) uplink field appears in short special surveillance format UF=0. It contains the identity of the ground-initiated Comm-B register whose contents are requested to appear in the MV field of the corresponding reply.

Note: A GICB request containing BDS=8 will cause the resulting reply to contain the aircraft identification message in its MV field.

2.2.3.8.3.1.5 CA Capability

This 3-bit (6-8) downlink field reports transponder capability and is used in DF=11, the All-Call reply, and in DF=17, the Extended Squitter.

The codes reported by RTCA/DO-181 transponders are:

- 0 = No extended capability report available.
- 1 = Comm A/B and extended capability report available.
- 2 = Comm A/B/C and extended capability report available.
- 3 = Comm A/B/C/D and extended capability report available.
- 4-7 = Not assigned.

The codes reported by RTCA/DO-181A or later transponders are:

- 0 = Signifies no communications capability (surveillance only), no ability to set CA code 7, either on the ground or airborne.
- 1 = Not used.
- 2 = Not used.
- 3 = Not used.
- 4 = Signifies at least Comm-A and Comm-B capability, ability to set CA code 7, on the ground.
- 5 = Signifies at least Comm-A and Comm-B capability, ability to set CA code 7, airborne.
- 6 = Signifies at least Comm-A and Comm-B capability, ability to set CA code 7, either on the ground or airborne.
- 7 = Signifies DR is NOT equal to zero (Ref. B, §2.2.14.4.8), or FS equals 2, 3, 4, 5, either on the ground or airborne (§2.2.3.8.3.1.8).

2.2.3.8.3.1.6 CC Crosslink Capability

This 1-bit (7) downlink field appears in short special surveillance format DF=0. It is used to indicate the ability of the transponder to support the crosslink capability, i.e., decode the contents of the BDS field in a UF=0 interrogation and respond with the contents of the specified ground-initiated Comm-B register in the MV field of the corresponding DF=16 reply.

The codes are:

- 0 = aircraft cannot support the crosslink capability.
- 1 = aircraft supports the crosslink capability.

2.2.3.8.3.1.7 DR Downlink Request

This 5-bit (9-13) downlink field appears in surveillance and Comm-B formats DF=4, 5, 20, 21. This field is used to request extraction of downlink messages from the transponder by the interrogator.

The codes are:

- 0 = No downlink request.
- 1 = Request to send Comm-B message (B bit set).
- 2 = TCAS bit set.
- 3 = TCAS bit set and B bit set.
- 4 = Comm-B broadcast 1 available
- 5 = Comm-B broadcast 2 available
- 6 = TCAS bit set and Comm-B broadcast 1 available
- 7 = TCAS bit set and Comm-B broadcast 2 available
- 8-15 = Not assigned
- 16-31 = Comm-D protocol (Ref. B, §2.2.17).

Note: The B bit is explained in Ref. B, §2.17.1.12.4.

2.2.3.8.3.1.8 FS Flight Status

This 3-bit (6-8) downlink field reports the flight status of the aircraft and is used in formats DF=4, 5, 20, 21

Table 2-9: Flight Status Codes

Code	Alert	SPI	Airborne	On the Ground
0	no	no	yes	no
1	no	no	no	yes
2	yes	no	yes	no
3	yes	no	no	yes
4	yes	yes		Either
5	no	yes		Either
6 and 7 are not assigned				

2.2.3.8.3.1.9 MA Message, Comm-A

This 56-bit (33-88) uplink field appears in Comm-A formats UF=20, 21 and is used by Mode S sensors to transmit a TCAS Sensitivity Level Command Message (§2.2.3.8.3.2.2) to airborne ACAS X equipment.

Note: This field is not used by ACAS X systems.

2.2.3.8.3.1.10 MB Message, Comm-B

This 56 bit (33-88) downlink field appears in Comm-B formats DF=20, 21 and is used by airborne ACAS X equipment to transmit RA Reports (§2.2.3.8.3.2.3.1) and Data Link Capability Reports (§2.2.3.8.3.2.3.2) to Mode S sensors.

2.2.3.8.3.1.11 ME Message, Extended Squitter

This 56-bit (33-88) downlink field appears in Extended Squitter format DF=17 and is used to transmit Aircraft Position Messages, Aircraft Identification Messages, Operational Coordination Messages, and Aircraft Operational Status Messages.

Note: The contents of ground-initiated Comm-B register 08 {HEX} (Ref. G) is inserted in the ME field of an Extended Squitter containing an aircraft identification message.

2.2.3.8.3.1.12 MU Message, Comm-U

This 56-bit (33-88) uplink field appears in long special surveillance format UF=16 and is used by airborne ACAS X equipment to transmit three types of messages: (1) TCAS Resolution Messages (§2.2.3.8.3.2.4.1) for air-to-air coordination, (2) TCAS Broadcast Interrogation Messages (§2.2.3.8.3.2.4.2) to indicate own presence and identity for the purpose of controlling interference caused by ACAS X interrogations, and (3) RA Broadcast Interrogation Messages (§2.2.3.8.3.2.4.3) to allow RA activity to be monitored in areas where Mode S ground station surveillance coverage does not exist by using special RA Broadcast signal receivers on the ground.

2.2.3.8.3.1.13 MV Message, Comm-V

This 56-bit (33-88) downlink field appears in long special surveillance format DF=16 and is used by airborne ACAS X equipment to transmit Coordination Reply Messages (§2.2.3.8.3.2.5) to requesting Active CAS-equipped aircraft.

This field is also used to contain the contents of the ground-initiated Comm-B register as requested in the BDS field of the short special surveillance format UF=0 interrogation that elicited the reply.

2.2.3.8.3.1.14 RI Air-to-Air Reply Information

This 4-bit (14-17) downlink field appears in special surveillance formats DF=0, 16. This field contains information pertaining to replying aircraft. Where airspeed is reported (§2.2.3.9.2.3.1), the maximum true airspeed flown in normal operations is given using the coding shown below.

The RI field reports "No On-board TCAS" (RI=0) if the TCAS unit has failed or is in standby. The RI field reports "On-Board TCAS with no resolution capability" (RI=2) if RAs are inhibited (unit is in TA-Only mode).

Note: For ACAS X systems, RI=2 means (a) TA-Only mode, when ownship is airborne, or (b) neither TAs nor RAs, with surveillance tracks being sent to the display, when ownship is on-the-ground.

The codes are:

- 0 = No on-board TCAS.
- 1 = Not assigned.
- 2 = On-board TCAS with no resolution capability.
- 3 = On-board TCAS with vertical-only resolution capability.
- 4 = On-board TCAS with vertical and horizontal resolution capability
- 5-7 = Not assigned.
- 8 = No maximum airspeed data available.
- 9 = Airspeed \leq 75 knots.
- 10 = Airspeed $>$ 75 knots and \leq 150 knots.
- 11 = Airspeed $>$ 150 knots and \leq 300 knots.
- 12 = Airspeed $>$ 300 knots and \leq 600 knots.
- 13 = Airspeed $>$ 600 knots and \leq 1200 knots.
- 14 = Airspeed $>$ 1200 knots.
- 15 = Not assigned.

Bit 14 of this field replicates the AQ bit (§2.2.3.8.3.1.3) of the interrogation. That is, codes 0-7 indicate that this is a reply to an air-to-air non-acquisition interrogation; codes 8-15 indicate that this is an acquisition reply.

2.2.3.8.3.1.15 RL Reply Length

This 1-bit (9) uplink field appears in special surveillance formats UF=0, 16. This field commands a specific air-to-air reply format (short or long) regardless of the uplink format.

The codes are:

- | | |
|---|--|
| 0 | = Reply with short special surveillance format DF=0. |
| 1 | = Reply with long special surveillance format DF=16. |

2.2.3.8.3.1.16 RR Reply Request

This 5-bit (9-13) uplink field appears in surveillance and Comm-A formats UF=4, 5, 20, 21. This field contains length and content of the reply requested by the interrogator. The RR code that airborne ACAS X equipment responds to is RR=19. This RR code elicits a Comm-B reply whose MB field content is provided by the ACAS X equipment.

The codes are:

<u>RR Code</u>	<u>Requested MB Content</u>
19	RA Report (§2.2.3.8.3.2.3.1)

2.2.3.8.3.1.17 SD Special Designator

When the Designator Identification field (DI, bits 14-16) has a code value of 1 or 7, the Special Designator field (SD, bits 17-32) of each Comm-A interrogation is used to obtain the Interrogator Identifier Subfield (IIS) and the Linked Comm-A Subfield (LAS).

Subfields in SD:

- IIS: Interrogator Identifier - This 4-bit (17-20) subfield contains the self-identification code of the ground interrogator and may take on the values 0 through 15. IIS=0 is not a valid interrogator identifier code for multisite purposes.
- LAS: Linked Comm-A Subfield - This 3-bit (30-32) subfield contains coding for the linking of Comm-A message segments. LAS=0 indicates a single segment Comm-A message. A TCAS Sensitivity Level Command (§2.2.3.8.3.2.2) is transmitted only in a single segment Comm-A message; i.e., ACAS X should ignore Comm-A messages if LAS≠0.

Note: The 4-bit TMS subfield previously used for linking of Comm-A messages is now composed of a spare bit (bit 29) and the 3-bit LAS subfield (bits 30-32). When checking for LAS=0, care must be taken to examine only the three LAS bits, not bit 29 also. See Ref. K.

2.2.3.8.3.1.18 VS Vertical Status

This 1-bit (6) downlink field appears in special surveillance formats DF=0, 16.

The codes are:

0	=	aircraft airborne
1	=	aircraft on the ground

2.2.3.8.3.2 ACAS X Fields and Subfields

The following paragraphs describe the location and coding of those mission fields and subfields that are not defined in Ref. B, but are used by aircraft equipped with ACAS X.

2.2.3.8.3.2.1 Subparagraph Not Used

2.2.3.8.3.2.2 MA Fields Used by ACAS X

ACAS X does not use the MA field of Comm-A interrogations.

2.2.3.8.3.2.3 MB Fields Used by ACAS X

ACAS X uses the MB field (§2.2.3.8.3.1.10) of Comm-B replies to transmit RA Reports and Data Link Capability Reports to Mode S sensors. This field always contains the 8-bit BDS (B-Definition Subfield).

2.2.3.8.3.2.3.1 Subfields in MB for RA Report

BDS: B-Definition Subfield - This 8-bit (33-40) subfield defines the data contained in the remainder of MB. For convenience in coding, BDS is expressed in two groups of 4 bits each, BDS1, 33 through 36, and BDS2, 37 through 40. An RA Report is indicated by BDS1=3 and BDS2=0, the combination of which is equivalent to BDS=48.

Notes:

1. *Bits 41-88 of the RA Report are identical to bits 41-88 of the ADS-B TCAS RA Broadcast (§2.2.3.8.3.2.10.1). ACAS X deposits these bits into register 30₁₆ of its associated Mode S transponder. The transponder will insert the bits into DF20, 21 replies for an RA Report to a Mode S ground sensor and into 1090 MHz Extended Squitters for an ADS-B TCAS RA Broadcast.*
2. *This section (§2.2.3.8.3.2.3.1) defines the RA Report subfields as they are transmitted by the Mode S transponder. §2.2.3.9.5.1 gives requirements for communication of RA Report information from ACAS X to its associated Mode S transponder.*

ARA: Active RA - This 10-bit (41-50) subfield indicates the currently active RA (if any) generated by own ACAS X unit against one or more threat aircraft.

The ARA subfield is further divided into:

AVRA: Vertical RA – This 7-bit (41-47) subfield contains the vertical component of the ARA as defined below.

AHRA: Horizontal RA – This 3-bit (48-50) subfield contains the horizontal component of the ARA. For systems designed to these MOPS, the AHRA=0.

Bits 41-47 have the following meanings:

<u>Bit</u>	<u>Coding</u>
41	<p>0 = Different vertical senses have been generated in a multi-threat encounter (when MTE=1); or no RA has been generated (when MTE=0)</p> <p>1 = The same vertical sense has been generated in a single or multi-threat encounter</p>
42	<p>0 = RA is not crossing</p> <p>1 = RA is crossing</p>
	<p><i>Note:</i> An RA is considered crossing if ownship is expected to cross the altitude of the intruder before closest approach; e.g., pass above a threat currently above own. An RA is crossing regardless of whether "Crossing" is included in the aural annunciation. See Table 2-49 Note 4 for more details.</p>
43	<p>0 = Upward sense RA has been generated, i.e., ownship intent is to pass above the threat</p> <p>1 = Downward sense RA has been generated; i.e., ownship intent is to pass below the threat</p>
	<p><i>Note:</i> When bit 41=0 and MTE=1, i.e., different vertical senses have been generated in a multi-threat encounter:</p> <p style="padding-left: 20px;">if strength bits = 14 (preventive MTLO; issued while level), the advisory has neither up nor down sense, and by convention, bit 43=0;</p> <p style="padding-left: 20px;">if strength bits = 15 (corrective MTLO; issued while climbing/descending), bit 43=1 while climbing and bit 43=0 while descending.</p>
44	Strength Bit 1
45	Strength Bit 2
46	Strength Bit 3
47	Strength Bit 4

An RA is considered crossing if ownship is expected to cross the altitude of the intruder before closest approach, e.g., pass above a threat currently above own. An RA is crossing

regardless of whether “Crossing” is included in the aural annunciation. See Table 2-49 for more details.

The Strength bits are defined as follows:

<u>Strength Bits</u>	<u>Value</u>	<u>Meaning</u>
<u>1 2 3 4</u>		
0 0 0 0	= 0	Clear of Conflict
0 0 0 1	= 1	Monitor Vertical Speed
0 0 1 0	= 2	Level-off; weakening of positive RA
0 0 1 1	= 3	Level-off; corrective when climbing/descending
0 1 0 0	= 4	Climb/descend at 1500 fpm
0 1 0 1	= 5	Reversal to climb/descend
0 1 1 0	= 6	Increase climb/descend
0 1 1 1	= 7	Maintain rate; at current rate > 1500 fpm
1 0 0 0	= 8	Reversal to maintain rate; at current rate > 1500 fpm
1 0 0 1	= 9	Level-off; reversal to corrective negative RA
1 0 1 0	= 10	Monitor Vertical Speed; following descend, descend inhibited
1 0 1 1	= 11	Monitor Vertical Speed; reversal to preventive negative RA
1 1 0 0	= 12	Not Assigned
1 1 0 1	= 13	Not assigned
1 1 1 0	= 14	Preventive MTLO while level*
1 1 1 1	= 15	Corrective MTLO while climbing/descending*

Note: *Level for MTLO is ownship -500 fpm to +500 fpm;
Climbing for MTLO is ownship vertical rate > 500 fpm
Descending for MTLO is ownship vertical rate < -500 fpm

LDI: Low-level Descend Inhibit – This 2-bit (51-52) subfield indicates whether low level descend inhibit costs are being applied. The coding is as follows:

<u>Bits</u>	<u>Value</u>	<u>Meaning</u>
<u>51 52</u>		
0 0	= 0	No RAs are inhibited
0 1	= 1	Increased rate descend RAs are inhibited – but see Note 1
1 0	= 2	All positive descend RAs are inhibited
1 1	= 3	All RAs are inhibited

Notes:

1. When ownship is descending, the continuation of active increased rate descend RAs is not inhibited unless all positive descend RAs are inhibited. The generation of new increased rate descend RAs is inhibited as indicated.
2. The TRM sets the descend inhibits once per cycle by reference to radar altitude as shown below. All these radar altitude values have ± 100 ft hysteresis applied, i.e., add 100 ft when ownship is climbing and subtract 100 ft when ownship is descending.

No RAs are inhibited

radar altitude > 1550 ft AGL

Increased rate descend RAs are inhibited

1550 ft AGL \geq radar altitude > 1100 ft AGL

<i>All positive descend RAs are inhibited</i>	$1100 \text{ ft AGL} \geq \text{radar altitude} > 1000 \text{ ft AGL}$
<i>All RAs are inhibited</i>	$1000 \text{ ft AGL} \geq \text{radar altitude}$

RMF: RA Message Format – This 2-bit (53-54) subfield indicates the CA system used to generate bits 41-88 of the RF message and is coded as follows:

<u>Bits</u>	<u>Value</u>	<u>Meaning</u>
<u>53</u>	<u>54</u>	
0 0	= 0	TCAS II
0 1	= 1	ACAS Xa
1 0	= 2	ACAS Xu
1 1	= 3	Not assigned

RAC: RA Complements - This 4-bit (55-58) subfield indicates the currently active RA complements (if any) received from all other ACAS X or TCAS aircraft equipped with an on-board resolution capability.

The bits in RAC have the following meanings:

<u>Bit</u>	<u>RAC</u>
55	Do not pass below
56	Do not pass above
57	Reserved for horizontal coordination
58	Reserved for horizontal coordination

A bit set to '1' indicates that the associated condition is active. A bit set to '0' indicates that the associated condition is inactive.

RAT: RA Terminated indicator - This 1-bit (59) subfield indicates when an RA previously generated by ACAS X has ceased being generated.

Coding:

- 0 = The RA indicated by the ARA subfield is currently active.
- 1 = The RA indicated by the ARA subfield has been terminated.

Notes:

1. *After an RA has been terminated by ACAS X, it is still required to be reported by the Mode S transponder for 18 ± 1 seconds (RA Report) or 24 ± 1 seconds (ADS-B TCAS RA Broadcast). The RA Terminated indicator may be used, for example, to permit timely removal of an RA indication from an air traffic controller's display, or for assessments of RA duration within a particular airspace.*
2. *RAs may terminate for a number of reasons: normally, when the conflict has been resolved and the threat is diverging in range; or when the threat's Mode S transponder for some reason ceases to report altitude during the conflict. The RA terminated indicator is used to show that the RA has been removed in each of these cases.*

MTE: Multiple Threat Encounter - This 1-bit (60) subfield indicates whether two or more simultaneous threats are currently being processed by the ACAS X threat resolution logic.

Coding:

- 0 = One threat is being processed by the resolution logic (when ARA bit 41=1); or no threat is being processed by the resolution logic (when ARA bit 41=0).
- 1 = Two or more simultaneous threats are being processed by the resolution logic.

CNT: Continuation bit – This 1-bit (61) subfield indicates whether a follow-on RF message is being generated to report additional RA or TA related information.

Coding:

- 0 = No follow-on RF message exists
- 1 = A follow-on RF message exists

Note: In this MOPS the CNT bit is always set to '0'. The capability to transmit a follow-on message may be incorporated into future systems.

TTI: Threat Type Indicator - This 1-bit (62) subfield defines the type of identity data contained in the TID subfield.

Coding:

- 0 TID contains altitude, range, and bearing data
- 1 TID contains an ICAO 24-bit Aircraft Address

TID: Threat Identity Data - This 24-bit (63-86) subfield contains the ICAO 24-bit Aircraft Address of the threat or the altitude, range, and bearing if the threat is not Mode S equipped. If two or more threats are simultaneously processed by the ACAS X resolution logic, TID contains the identity or position data for the most recently declared threat. If TTI=1, TID contains in bits 63-86 the ICAO 24-bit Aircraft Address of the threat. If TTI=0, TID contains the following three subfields.

TIDA: Threat Identity Data Altitude - This 11-bit (63-73) subfield contains the most recent threat altitude estimated by ACAS X, expressed in binary to a resolution of 100 ft as follows:

Coding:

- | | |
|-------|-------------------------|
| 0 | No altitude data |
| 1 | Altitude below -950 ft |
| 2 | -950 ft ≤ Alt < -850 ft |
| 3 | -850 ft ≤ Alt < -750 ft |
| 4.... | |

TIDR: Threat Identity Data Range - This 7-bit (74-80) subfield contains the most recent threat range estimated by ACAS X (horizontal range).

Coding(n):

- | | |
|---|-----------------------------|
| n | <u>Estimated range (NM)</u> |
| 0 | No range estimate available |
| 1 | Less than 0.05 |

2-126	$(n-1)/10 \pm 0.05$
127	Greater than 12.55

TIDB: Threat Identity Data Bearing - This 6-bit (81-86) subfield contains the most recent estimated bearing of the threat aircraft, relative to the ACAS X aircraft heading.

Coding(n):

<u>n</u>	<u>Estimated bearing (degrees)</u>
0	No bearing estimate available
1-60	Between $6(n-1)$ and $6n$
61-63	Not assigned

DSI: Designation Indicator – This 1-bit (87) subfield is coded as follows:

- =0 the threat defined in TID is not designated for Xo
- =1 the threat defined in TID is designated for Xo, and the designation is in force

For multi-threat encounters, the SPI Indication is used to indicate additional designation information as shown below.

SPI: Suppression Indicator – This 1-bit (88) subfield is coded as follows:

For single-threat encounters:

- =0 the RA is not suppressed
- =1 the RA is suppressed (not announced to the flight crew)

For multi-threat encounters, suppression does not apply, so the SPI subfield is used to indicate designation:

- =0 no threat other than the one defined in TID is designated for Xo
- =1 another threat is designated for Xo, and the designation is in force

Note: The structure of MB for an RA Report for RTCA/DO-385 compatible systems is:

<u>Position</u>	<u># of Bits</u>	<u>Subfield</u>	<u>Remarks</u>
33-36	4	BDS1	=3
37-40	4	BDS2	=0
41-50	10	ARA	
41-47	7	AVRA	
48-50	3	AHRA	
51-52	2	LDI	
53-54	2	RMF	
55-58	4	RAC	
59	1	RAT	=1 RA not active
60	1	MTE	
61	1	CNT	
62	1	TTI	
63-86	24	TID	
63-73	11	TIDA	

74-80	7	<i>TIDR</i>
81-86	6	<i>TIDB</i>
87	1	<i>DSI</i>
88	1	<i>SPI</i>

2.2.3.8.3.2.3.2 Subfields in MB for Data Link Capability Report

BDS: B-Definition Subfield - This 8-bit (33-40) subfield defines the data contained in the remainder of MB. For convenience in coding, BDS is expressed in two groups of 4 bits each, BDS1, 33 through 36, and BDS2, 37 through 40. All Data Link Capability Reports use BDS1=1. The basic Data Link Capability Report used by ACAS X uses BDS2=0.

2.2.3.8.3.2.3.2.1 Coding of Data Link Capability Report

Bits 43-46, 48, and 69-72 are used to convey TCAS/ACAS X capability information as follows:

<u>Bit(s)</u>	<u>Coding</u>
43-46	0000 = non-future version 0001 = ACAS Xa version 1 0010 = ACAS Xu version 1 0011- 1111 = reserved
48	0 = TCAS/ACAS X failed or on standby 1 = TCAS/ACAS X operating
69	0 = Hybrid surveillance not fitted 1 = Hybrid surveillance fitted
70	0 = TCAS/ACAS X generating TAs only 1 = TCAS/ACAS X generating TAs and RAs
72,71	TCAS/ACAS X Version ¹ 0,0 = RTCA/DO-185 0,1 = RTCA/DO-185A 1,0 = RTCA/DO-185B and ED-143 1,1 = Future version ²

¹ See transponder registers 0xE5 and 0xE6 in Ref L for CAS unit part number and CAS unit software revision, respectively.

² For ACAS X, set 72,71 = 1,1 and fill in bits 43-46.

Note: In the Data Link Capability Report format above, bit 69=1 indicates that ‘hybrid surveillance is fitted and operational’. ACAS X software includes hybrid surveillance functionality; thus hybrid surveillance can be considered to be

'fitted.' Vendors are reminded that in order to set bit 69=1, hybrid surveillance must also be 'operational', i.e., the installed ACAS X system must be correctly receiving appropriate GNSS position and status information both from ownship and from intruder aircraft that are transmitting such information. Hybrid surveillance is not operational unless all the required data are made available and provided to ACAS X.

2.2.3.8.3.2.4 MU Fields Used by ACAS X

ACAS X uses the MU field (§2.2.3.8.3.1.12) of a long special surveillance interrogation (UF=16) to transmit three types of messages: (1) TCAS Resolution Messages for air-to-air coordination, (2) TCAS Broadcast Interrogation Messages to indicate own presence and identity for the purpose of controlling interference caused by ACAS X interrogations, and (3) RA Broadcast Interrogation Messages to allow monitoring of ACAS X RA activity.

2.2.3.8.3.2.4.1 Subfields in MU for a TCAS Resolution Message

UDS: U-Definition Subfield - This 8 bit (33-40) subfield defines the data content and coding in the remainder of MU. For convenience in coding, UDS is expressed in two groups of 4 bits each, UDS1, 33 through 36, and UDS2, 37 through 40. TCAS Resolution Messages are identified by UDS1=3 and UDS2=0, the combination of which is equivalent to UDS=48.

MTB: Multiple Threat Bit - This 1-bit (42) subfield indicates a multiple threat.

The codes are:

- 0 = Interrogating ACAS X has no more than one threat.
- 1 = Interrogating ACAS X has more than one threat.

CVC: Cancel Vertical Resolution Advisory Complement - This 2-bit (43-44) subfield is used by airborne ACAS X equipment to cancel a vertical RA complement sent to an equipped threat aircraft.

The codes are:

- 0 = No cancellation.
- 1 = Cancel, do not pass below.
- 2 = Cancel, do not pass above.
- 3 = Not assigned.

VRC: Vertical Resolution Advisory Complement - This 2-bit (45-46) subfield is used by airborne ACAS X equipment to send a vertical RA complement (do not pass above or do not pass below) to the equipped threat aircraft.

The codes are:

- 0 = No vertical RA complement sent.
- 1 = Do not pass below.
- 2 = Do not pass above.
- 3 = Not assigned.

CHC: Cancel Horizontal Resolution Advisory Complement - This 3-bit (47-49) subfield is used by ACAS X with horizontal on-board resolution equipment to cancel a horizontal

RA complement sent to an equipped threat aircraft. In TCAS Resolution Messages transmitted by ACAS X without horizontal resolution capability, CHC is set to the zero code.

The codes are:

- 0 = No cancellation.
- 1 = Reserved for horizontal coordination.
- 2 = Reserved for horizontal coordination.
- 3-7 = Not assigned.

HRC: Horizontal Resolution Advisory Complement - This 3-bit (50-52) subfield is used by ACAS X with horizontal on-board resolution equipment to send a horizontal resolution maneuver complement to the equipped threat aircraft. In TCAS Resolution Messages transmitted by ACAS X without horizontal resolution capability, HRC is set to the zero code.

The codes are:

- 0 = No horizontal RA complement sent
- 1-7 = Reserved

HSB: Horizontal Sense Bits - This 5-bit (56-60) subfield is a parity coding field used to protect the six horizontal sense bits (47-52). The originating Active CAS will include bits 56-60 in all TCAS Resolution Messages sent. The receiving ACAS X will examine bits 56-60 in TCAS Resolution Messages received. If bits 47-52 are not in agreement with bits 56-60 as shown in the table below, the receiving ACAS X will assume there is an error in the message and will not use the message contents.

Table 2-10: Horizontal Sense Bits (HSB field)

Subfield Bits	CHC			HRC			HSB				
	47	48	49	50	51	52	56	57	58	59	60
0 0 0	0	0	0	0	0	0	0	0	0	0	0
0 0 0	0	0	0	0	0	1	0	1	0	1	1
0 0 0	0	0	1	0	1	0	1	0	0	1	1
0 0 0	0	0	0	0	1	1	1	1	0	0	0
0 0 0	0	0	0	1	0	0	1	1	1	0	0
0 0 0	0	0	0	1	0	1	1	0	1	1	1
0 0 0	0	0	0	1	1	0	0	1	1	1	1
0 0 0	0	0	0	1	1	1	0	0	1	0	0
0 0 1	0	0	0	0	0	0	0	1	1	0	1
0 0 1	0	0	0	0	1	0	0	0	1	1	0
0 0 1	0	0	1	0	0	0	1	1	1	1	0
0 0 1	0	0	1	0	1	1	1	0	1	0	1
0 0 1	0	0	1	1	0	0	1	0	0	0	1
0 0 1	0	0	1	1	0	1	1	1	0	1	0
0 0 1	0	0	1	1	1	0	0	0	0	1	0
0 1 0	0	1	0	0	0	0	1	0	1	0	1
0 1 0	0	1	0	0	0	1	1	1	1	1	0
0 1 0	0	1	0	0	1	0	0	0	1	1	0

Subfield Bits	CHC			HRC			HSB				
	47	48	49	50	51	52	56	57	58	59	60
	1	1	1	1	1	1	0	0	1	0	1

Note: The rule used to generate the HSB subfield bit setting is a distance 3 Hamming code augmented with a parity bit, producing the ability to detect up to three errors in the eleven transmitted bits.

VSB: Vertical Sense Bits - This 4-bit (61-64) subfield is a parity coding field used to protect the four vertical sense bits (43-46). The originating Active CAS will include bits 61-64 in all TCAS Resolution Messages sent. The receiving ACAS X will examine bits 61-64 in TCAS Resolution Messages received. If bits 43-46 are not in agreement with bits 61-64 as shown in the table below, the receiving ACAS X will assume there is an error in the message and will not use the message contents.

Table 2-11: Vertical Sense Bits (VSB field)

Subfield Bits	CVC		VRC		VSB			
	43	44	45	46	61	62	63	64
0 0	0	0	0	0	0	0	0	0
0 0	0	0	1	1	1	1	0	
0 0	1	0	0	1	1	1	1	
0 0	1	1	1	0	0	0	1	
0 1	0	0	1	0	1	0	1	1
0 1	0	1	0	1	0	1	0	1
0 1	1	0	1	1	0	0	0	
0 1	1	1	0	0	0	1	0	
1 0	0	0	1	1	0	1	0	1
1 0	0	1	0	0	0	0	1	1
1 0	1	0	1	0	1	0	1	0
1 0	1	1	0	1	0	1	0	0
1 1	0	0	0	1	1	1	0	
1 1	0	1	1	0	1	0	0	0
1 1	1	0	0	0	0	0	0	1
1 1	1	1	1	1	1	1	1	1

Note: The rule used to generate the VSB subfield bit setting is a distance 3 Hamming code augmented with a parity bit, producing the ability to detect up to three errors in the eight transmitted bits.

MID: ICAO aircraft address - This 24-bit (65-88) subfield contains the ICAO 24-bit Aircraft Address of the interrogating ACAS X aircraft.

Note: The structure of MU for a TCAS Resolution Message is:

<u>Position</u>	<u># of Bits</u>	<u>Subfield</u>	<u>Remarks</u>
33-36	4	UDS1	= 3
37-40	4	UDS2	= 0
41	1	-	<i>Not assigned</i>
42	1	MTB	-
43-44	2	CVC	-
45-46	2	VRC	-
47-49	3	CHC	-
50-52	3	HRC	-
53-55	3	-	<i>Not assigned</i>
56-60	5	HSB	-
61-64	4	VSB	-
65-88	24	MID	-

2.2.3.8.3.2.4.2 Subfields in MU for a TCAS Broadcast Interrogation Message

The following subfields appear in MU for a TCAS Broadcast Interrogation Message.

UDS: **U-Definition Subfield** - This 8-bit (33-40) subfield indicates a TCAS Broadcast Interrogation Message by UDS1=3 and UDS2=2, the combination of which is equivalent to UDS=50.

MID: ICAO aircraft address - This 24-bit (65-88) subfield contains the ICAO 24-bit Aircraft Address of the interrogating ACAS X aircraft.

Note: The structure of MU for a TCAS Broadcast Interrogation Message is:

<u>Position</u>	<u># of Bits</u>	<u>Subfield</u>	<u>Remarks</u>
33-36	4	UDS1	= 3
37-40	4	UDS2	= 2
41-64	24	-	<i>Not assigned</i>
65-88	24	MID	-

2.2.3.8.3.2.4.3 Subfields in MU for an RA Broadcast Interrogation Message

The following subfields appear in MU for an RA Broadcast Interrogation Message.

UDS: U-Definition Subfield - This 8-bit (33-40) subfield indicates an RA Broadcast Interrogation Message by UDS1=3 and UDS2=1, the combination of which is equivalent to UDS=49.

ARA: Active RA – The coding of this 10-bit (41-50) subfield is as defined in §2.2.3.8.3.2.3.1.

LDI: Low-level Descend Inhibit – The coding of this 2-bit (51-52) subfield is as defined in §2.2.3.8.3.2.3.1.

RMF: RA Message Format – The coding of this 2-bit (53-54) subfield is as defined in §2.2.3.8.3.2.3.1.

RAC: RA Complements – The coding of this 4-bit (55-58) subfield is as defined in §2.2.3.8.3.2.3.1.

RAT: RA Terminated indicator – The coding of this 1-bit (59) subfield is as defined in §2.2.3.8.3.2.3.1.

MTE: Multiple Threat Encounter – The coding of this 1-bit (60) subfield is as defined in §2.2.3.8.3.2.3.1.

SPI: Suppression Indication – The coding of this 1-bit (61) subfield is as defined in §2.2.3.8.3.2.3.1.

AID: Mode A Identity code - This 13-bit (63-75) subfield denotes the Mode A identity code of the reporting aircraft.

Coding:

Bit No.	63	64	65	66	67	68	69	70	71	72	73	74	75
Mode A code bit	A4	A2	A1	B4	B2	B1	0	C4	C2	C1	D4	D2	D1

CAC: Mode C Altitude Code - This 13-bit (76-88) subfield denotes the Mode C altitude code of the reporting aircraft.

Coding:

Bit No.	76	77	78	79	80	81	82	83	84	85	86	87	88
Mode C code bit	C1	A1	C2	A2	C4	A4	0	B1	D1	B2	D2	B4	D4

Note: The structure of MU for an RA Broadcast Interrogation Message is:

<u>Position</u>	<u># of Bits</u>	<u>Subfield</u>	<u>Remarks</u>
33-36	4	UDS1	= 3
37-40	4	UDS2	= 1
41-50	10	ARA	
51-52	2	LDI	
53-54	2	RMF	
55-58	4	RAC	
59	1	RAT	
60	1	MTE	
61	1	SPI	
62	1	-	Not assigned
63-75	13	AID	
76-88	13	CAC	

2.2.3.8.3.2.5 MV Fields Used by ACAS X

ACAS X uses the MV field of a long special surveillance reply (§2.2.3.8.3.1.13) to transmit a Coordination Reply Message to the requesting ACAS X or TCAS aircraft.

2.2.3.8.3.2.5.1 Subfields in MV for a Coordination Reply Message

VDS: V-Definition Subfield - This 8-bit (33-40) subfield defines the data content and coding in the remainder of MV. For convenience in coding, VDS is expressed in two groups of 4 bits each, VDS1, 33 through 36, and VDS2, 37 through 40. The airborne ACAS X / TCAS equipment is a source of long special surveillance reply MV messages containing the VDS1=3 code. A Coordination Reply Message is indicated by VDS1=3 and VDS2=0, the combination of which is equivalent to VDS=48.

ARA: Active RA - The coding of this 10-bit (41-50) subfield is as defined in §2.2.3.8.3.2.3.1.

LDI: Low-level Descend Inhibit- The coding of this 2-bit (51-52) subfield is as defined in §2.2.3.8.3.2.3.1.

RMF: RA Message Format - The coding of this 2-bit (53-54) subfield is as defined in §2.2.3.8.3.2.3.1.

RAC: RA Complements - The coding of this 4-bit (55-58) subfield is as defined in §2.2.3.8.3.2.3.1.

RAT: RA Terminated Indicator - The coding of this 1-bit (59) subfield is as defined in §2.2.3.8.3.2.3.1.

MTE: Multiple Threat Encounter - The coding of this 1-bit (60) subfield is as defined in §2.2.3.8.3.2.3.1.

Note: The structure of MV for a Coordination Reply Message in RTCA/DO-385 compatible systems is:

<u>Position</u>	<u># of Bits</u>	<u>Subfield</u>	<u>Remarks</u>
33-36	4	VDS1	= 3
37-40	4	VDS2	= 0
41-50	10	ARA	
51-52	2	LDI	
53-54	2	RMF	
55-58	4	RAC	
59	1	RAT	
60	1	MTE	
61-88	28	-	<i>Not assigned</i>

2.2.3.8.3.2.6 SL TCAS II Sensitivity Level Report

This 3-bit field (9-11) appears in special surveillance reply formats DF=0, 16. This field reports the sensitivity level at which the TCAS II unit is currently operating. ACAS X does not use sensitivity level but will provide the appropriate sensitivity level code, based on operating mode, to the transponder for population of this field in outgoing replies.

The codes provided by ACAS X are:

- 0 = No report.
- 1 = ACAS X in Standby.
- 2 = ACAS X in TA only mode.
- 3 = ACAS X in TA/RA mode.

Note: The sensitivity level for TCAS II systems also includes codes 4 – 7. ACAS X will never indicate operation at a level greater than 3 (as specified by the logic of Volume II). This ensures that ACAS X will not impact TCAS II advisory determination that may be affected by intruder sensitivity levels greater than 3. TCAS II Systems set an ‘encounter sensitivity level’ to be the greater of ownship and intruder sensitivity level. This ‘encounter sensitivity level’ is used by TCAS II systems in selecting its collision avoidance logic parameters, e.g., tau, DMOD, ZTHR (altitude threshold), ALIM. Since the SL code indicated by the ACAS X system is never greater than 3, it will have no impact on a TCAS II system ‘encounter sensitivity level’ in TCAS II/ACAS X encounters.

2.2.3.8.3.2.7 Unassigned Coding Space

Unassigned coding space, as indicated in Figure 2-3 and Figure 2-4, is reserved for future use. Zeroes are to be transmitted by interrogators and transponders in these bit positions.

Note: This rule ensures that future assignments in this coding space define a "zero-block" as a default code, i.e., no message is sent and/or no capability exists.

2.2.3.8.3.2.8 ME Fields Used by ACAS X for Passive Surveillance

ADS-B reports (DF=17 ADS-B and DF=18 ADS-B, ADS-R) may be received from either an RTCA/DO-260B (Ref. G) compliant receiver, RTCA/DO-317B (Ref. T) AIRB qualified tracks, or by decoding the messages directly per the requirements in this section and subsections. DF=18 TIS-B messages are not used by ACAS X.

If a Mode S transmission is accepted as a valid extended squitter (DF=17, DF=18, and an appropriate value for the PI field), the ME message field in bits 33 through 88 of the extended squitter is decoded as indicated in §2.2.3.8.3.2.8.1 through §2.2.3.8.3.2.8.13 in order to support passive surveillance.

Note: The format of the DF=17, 18 extended squitter and its fields and subfields are specified in Ref. G, §2.2.3.2 (ADS-B and TIS-B Message Baseline Format and Structures), and its subparagraphs. DF=18 ADS-B messages are identified with a CF=0 or 1, where the CF value indicates whether the DF=18 ADS-B data is identified by an ICAO 24-bit Aircraft Address or by an Anonymous 24-bit Address. DF=18 ADS-R messages are identified with a CF=6 and the IMF value indicates whether the DF=18 ADS-R data is identified by an ICAO 24-bit Aircraft Address or by an Anonymous 24-bit Address. DF=18 ADS-R messages use the same TYPE Codes and Message Formats as are defined for DF=17 ADS-B Messages, with the exception of bits modified as identified in Ref. G, §2.2.18. ACAS X with hybrid surveillance capability uses the Capability (CA), ICAO 24-bit Aircraft Address (AA), and Parity/Identity (PI) fields of all DF=17 extended squitter transmissions for purposes of Mode S target detection and address determination, as described in §2.2.4.6.4.2.1 of this document. ACAS X with hybrid surveillance capability may use the data in the ME field for purposes of altitude determination and passive surveillance when that field contains an Airborne Position Message. The type of message in the ME field is determined from the TYPE subfield of the ME field, located in bits 1 to 5 of the ME field (bits 33 to 37 of the overall extended squitter).

Horizontal position and altitude are determined from receipt of Airborne Position Messages by extracting the appropriate data elements per §2.2.3.8.3.2.8.2 - §2.2.3.8.3.2.8.5. Horizontal position data **shall** (2570) be initialized in accordance with Ref. G, Appendix A, using the appropriate CPR decoding algorithm, from receipt of an Airborne Position Message per §2.2.3.8.3.2.8.3. The computed position **shall** (1081) be checked for reasonableness per the following requirements:

- 1) A global unambiguous CPR decode and validation per Ref. G §2.2.10.3.1.
- 2) The local unambiguous decode from the receipt of the Airborne Position Message completing the global unambiguous CPR decode in 1) above validated per Ref. G §2.2.10.6.2 subparagraph c.

Note: Horizontal position data using ADS-B Reports from RTCA/DO-260B compliant receivers is initialized by performing a globally unambiguous CPR decode per Ref. G §2.2.10.3.1 and undergoes an additional reasonableness test per Ref. G §2.2.10.6.2. Since the range of extended squitter reception for ACAS X will typically be much less than 180 NM, a local CPR decode can be used for

initialization without the need to decode both even and odd reports, as noted in Ref. G, Appendix A, §A.1.7.1 (Principle of the CPR Algorithm), Note 4. Otherwise, a global unambiguous CPR decode should be used consistent with an RTCA/DO-260B or later report generator. Regardless of the use of a local or global CPR decode for initialization, a globally unambiguous CPR decode is computed as part of the above reasonableness tests.

Horizontal position for aircraft reporting the surface formats of extended squitter is determined from receipt of Surface Position Messages by extracting the appropriate data elements per §2.2.3.8.3.2.8.11 - §2.2.3.8.3.2.8.13.

2.2.3.8.3.2.8.1 TYPE Subfield of the ME Field

If the TYPE subfield of the ME field has the value 0, the Altitude subfield of that ME field **shall** (1082) be decoded. If the TYPE subfield has one of the values in the range 9 to 18, the Altitude, CPR Format, Encoded Latitude and Encoded Longitude subfields of that ME field **shall** (1083) be decoded. If the TYPE subfield has one of values in the range 5 to 8, the CPR Format, Encoded Latitude and Encoded Longitude subfields of that ME field **shall** (1085) be decoded when required to satisfy the requirements of §2.2.4.6.4.2.2. If the TYPE subfield has the value of 29, the NACp and SIL subfields of that ME field **shall** (1086) be decoded. If the TYPE subfield has the value of 31, the ADS-B Version Number, NACp, SIL, and SDA subfields of that ME field **shall** (1087) be decoded.

Note: If the TYPE field has the value 0, the ME field contains an Airborne Position Message without valid position data, but may contain either a barometric altitude or no altitude data. If the TYPE field has one of the values in the range 9 to 18, the ME field contains an Airborne Position Message with valid position data and either a barometric altitude or no altitude data. If the TYPE field has one of the values in the range 5 to 8, the ME field contains a Surface Position Message. If the TYPE field has the value of 29, the ME field contains a Target State and Status Message. If the TYPE field has the value of 31, the ME field contains an Aircraft Operational Status Message.

Note: ACAS X passive surveillance does not make use of Airborne Position Messages with TYPE field values in the range 20 to 22.

If the TYPE field has the value 1, 2, 3 or 4, the ME field contains an Aircraft Identification and Type message. This may optionally be decoded as specified in §2.2.3.9.8 (Extended Squitter With Aircraft Identification Message).

Note: The remaining ME field message types, 19, 23 to 28 and 30 are Airborne Velocity Messages, Extended Squitter Aircraft Status Messages (Emergency/Priority Status), Test Messages, or reserved message types. None of these message types are used by ACAS X passive surveillance.

2.2.3.8.3.2.8.2 Altitude from Airborne Position Message

The Altitude subfield of an Airborne Position Message is located in bit positions 9 to 20 of the ME field (bit positions 41 to 52 of the extended squitter). If the ME field of a valid extended squitter transmission has a TYPE subfield with the value 0 or with a value in the range 9 to 18, then the Altitude subfield of the ME field **shall** (1088) be decoded. If all bits in the Altitude subfield have the value 0, then the message contains no valid altitude data, and the Altitude subfield **shall** (1089) be ignored. Otherwise, the Altitude subfield contains altitude data encoded as specified in Ref. G, §2.2.3.2.3.4.3 (“Altitude Encoding” in ADS-

B Airborne Position Messages), and **shall** (1090) be decoded into an appropriate value for use in initial altitude determination and in passive surveillance.

2.2.3.8.3.2.8.3 CPR Format from Airborne Position Message

The CPR Format subfield of an Airborne Position Message is located in bit position 22 of the ME field (bit position 54 of the extended squitter). This field indicates whether the Encoded Latitude and Encoded Longitude subfields of that Airborne Position Message use the even or odd encoding method specified in Ref. G. If the ME field of a valid extended squitter transmission has a TYPE subfield with a value in the ranges 9 to 18, then the CPR Format subfield **shall** (1091) be used in conjunction with the Encoded Latitude and Encoded Longitude fields to decode them into latitude and longitude in accordance with Ref. G, Appendix A, §A.1.7.5 (Computations for Airborne, TIS-B and Intent Lat/Lon).

Note: After position initialization and the reasonableness tests in §2.2.3.8.3.2.8, a locally unambiguous CPR decode is used to update the position.

The computed position for any Airborne Position Message received subsequent to successful reasonableness test validation per §2.2.3.8.3.2.8 **shall** (1092) be checked for reasonableness per Ref. G §2.2.10.6.3.

Note: An erroneous local unambiguous CPR decode could result in an incorrect position that potentially remains incorrect for the life of the track..

2.2.3.8.3.2.8.4 Encoded Latitude from Airborne Position Message

The Encoded Latitude subfield of an Airborne Position Message is located in bit positions 23 to 39 of the ME field (bit positions 55 to 71 of the extended squitter). If the ME field of a valid extended squitter transmission has a TYPE subfield with a value in the ranges 9 to 18, then the Encoded Latitude is valid and **shall** (1093) be decoded into a latitude value in accordance with Ref. G, Appendix A, §A.1.7.5 (Computations for Airborne, TIS-B and Intent Lat/Lon).

2.2.3.8.3.2.8.5 Encoded Longitude from Airborne Position Message

The Encoded Longitude subfield of an Airborne Position Message is located in bit positions 40 to 56 of the ME field (bit positions 72 to 88 of the extended squitter). If the ME field of a valid extended squitter transmission has a TYPE subfield with a value in the ranges 9 to 18, then the Encoded Longitude is valid and **shall** (1094) be decoded into a longitude value in accordance with Ref. G, Appendix A, §A.1.7.5 (Computations for Airborne, TIS-B and Intent Lat/Lon).

2.2.3.8.3.2.8.6 ADS-B Version Number from Aircraft Operational Status Message

The ADS-B “Version Number” (VN) subfield of an Aircraft Operational Status Message is located in bit positions 41 to 43 of ME field bit positions 73 to 75 of the extended squitter). If the ME field of a valid extended squitter transmission has a TYPE subfield with a value of 31, then the ADS-B Version Number is valid and **shall** (1095) be decoded in accordance with Ref. G, §2.2.3.2.7.2.5.

2.2.3.8.3.2.8.7 Navigational Integrity Category (NIC) from Airborne Position Message

The Navigation Integrity Category (NIC) **shall** (1096) be decoded from the TYPE subfield in the Airborne Position Messages in accordance with Ref. G, §2.2.3.2.7.2.6. The NIC Supplement subfields are not required for ACAS X passive surveillance.

2.2.3.8.3.2.8.8 Navigation Accuracy Category for Position (NACp)

The NACp field is encoded in the Aircraft Operational Status Message and the Target State and Status Message.

2.2.3.8.3.2.8.8.1 NACp from Aircraft Operational Status Message

The NACp subfield of an Aircraft Operational Status Message is located in bit positions 45 to 48 of the ME field (bit positions 77 to 80 of the extended squitter). If the ME field of a valid extended squitter transmission has a TYPE subfield with a value of 31, then the NACp is valid and **shall** (1097) be decoded in accordance with Ref. G, §2.2.3.2.7.2.7.

2.2.3.8.3.2.8.8.2 NACp from Target State and Status Message

The NACp subfield of a Target State and Status Message is located in bit positions 40 to 43 of the ME field (bit positions 72 to 75 of the extended squitter). If the ME field of a valid extended squitter transmission has a TYPE subfield with a value of 29, then the NACp is valid and **shall** (1098) be decoded in accordance with Ref. G, §2.2.3.2.7.1.3.8.

2.2.3.8.3.2.8.9 Source Integrity Level (SIL)

The SIL field is encoded in the Aircraft Operational Status Message and the Target State and Status Message.

2.2.3.8.3.2.8.9.1 SIL from Aircraft Operational Status Message

The SIL subfield of an Aircraft Operational Status Message is located in bit positions 51 to 52 of the ME field (bit positions 83 to 84 of the extended squitter). If the ME field of a valid extended squitter transmission has a TYPE subfield with a value of 31, then the SIL is valid and **shall** (1099) be decoded in accordance with Ref. G, §2.2.3.2.7.2.9.

2.2.3.8.3.2.8.9.2 SIL from Target State and Status Message

The SIL subfield of a Target State and Status Message is located in bit positions 45 to 46 of the ME field (bit positions 77 to 78 of the extended squitter). If the ME field of a valid extended squitter transmission has a TYPE subfield with a value of 29, then the SIL is valid and **shall** (1100) be decoded in accordance with Ref. G, §2.2.3.2.7.1.3.10.

2.2.3.8.3.2.8.10 System Design Assurance (SDA)

The SDA subfield of an Aircraft Operational Status Message is located in bit positions 31 to 32 of the ME field (bit positions 63 to 64 of the extended squitter). If the ME field of a valid extended squitter transmission has a TYPE subfield with a value of 31, then the SDA is valid and **shall** (1101) be decoded in accordance with Ref. G, §2.2.3.2.7.2.4.6.

2.2.3.8.3.2.8.11 CPR Format from Surface Position Message

The CPR Format subfield of a Surface Position Message is located in bit position 22 of the ME field (bit position 54 of the extended squitter). This field indicates whether the Encoded Latitude and Encoded Longitude subfields of that Surface Position Message use the even or odd encoding method specified in Ref. G. If the ME field of a valid extended squitter transmission has a TYPE subfield with a value in the ranges 5 to 8, then the CPR Format subfield **shall** (1102) be used in conjunction with the Encoded Latitude and

Encoded Longitude fields to decode them into latitude and longitude in accordance with Ref. G, Appendix A, §A.1.7.6, Locally Unambiguous Decoding for Surface Position.

2.2.3.8.3.2.8.12 Encoded Latitude from Surface Position Message

The Encoded Latitude subfield of a Surface Position Message is located in bit positions 23 to 39 of the ME field (bit positions 55 to 71 of the extended squitter). If the ME field of a valid extended squitter transmission has a TYPE subfield with a value in the ranges 5 to 8, then the Encoded Latitude is valid and **shall** (1103) be decoded into a latitude value in accordance with Ref. G, Appendix A, §A.1.7.6, Locally Unambiguous Decoding for Surface Position.

2.2.3.8.3.2.8.13 Encoded Longitude from Surface Position Message

The Encoded Longitude subfield of a Surface Position Message is located in bit positions 40 to 56 of the ME field (bit positions 72 to 88 of the extended squitter). If the ME field of a valid extended squitter transmission has a TYPE subfield with a value in the ranges 5 to 8, then the Encoded Longitude is valid and **shall** (1104) be decoded into a longitude value in accordance with Ref. G, Appendix A, §A.1.7.6, Locally Unambiguous Decoding for Surface Position.

2.2.3.8.3.2.9 ME Fields Used by ACAS X for Air-to-Air Coordination

If a Mode S transmission is accepted as a valid extended squitter (DF=17, DF=18, and an appropriate value for the PI field), the ME message field in bits 33 through 88 of the extended squitter is decoded as indicated in §2.2.3.8.3.2.9.1 and §2.2.3.8.3.2.9.2.

Note: The formats of the DF=17, 18 extended squitter and its fields and subfields are specified in Ref. G, §2.2.3.2 (ADS-B and TIS-B Message Baseline Format and Structures) and its subparagraphs. The type of message in the ME field is determined from the TYPE subfield of the ME field, located in bits 1 to 5 of the ME field (bits 33 to 37 of the overall extended squitter).

ADS-B Messages with TYPE code = 28 (ADS-B Operational Coordination Message) and TYPE code = 31 (ADS-B Aircraft Operational Status Message) are used in air-to-air coordination.

2.2.3.8.3.2.9.1 Subfields in ME for Operational Coordination Message

In the subfields defined below, the bit number is relative to the start of the extended squitter, where bit 33 is the start of the ME message field.

TYPE: This 5-bit (33-37) subfield defines the type of extended squitter. For the Operational Coordination Message, TYPE = 28

Subtype: This 3-bit (38-40) subfield further defines TYPE. For the Operational Coordination Message, Subtype = 3.

MTB: Multiple Threat Bit - This 1-bit (42) subfield indicates a multiple threat. Codes are as defined in §2.2.3.8.3.2.4.1.

CVC: Cancel Vertical Resolution Advisory Complement - This 2-bit (43-44) subfield is used by airborne ACAS X equipment to cancel a vertical RA complement sent to an equipped threat aircraft. Codes are as defined in §2.2.3.8.3.2.4.1.

VRC: Vertical Resolution Advisory Complement - This 2-bit (45-46) subfield is used by airborne ACAS X equipment to send a vertical RA complement (Do Not Pass Above or Do Not Pass Below) to the equipped threat aircraft. Codes are as defined in §2.2.3.8.3.2.4.1.

CHC: Cancel Horizontal Resolution Advisory Complement - This 3-bit (47-49) subfield is used by ACAS X with horizontal on-board resolution equipment to cancel a horizontal RA complement sent to an equipped threat aircraft. In TCAS Resolution Messages transmitted by ACAS X without horizontal resolution capability, CHC is set to the zero code. Codes are as defined in §2.2.3.8.3.2.4.1.

HRC: Horizontal Resolution Advisory Complement - This 3-bit (50-52) subfield is used by ACAS X with horizontal on-board resolution equipment to send a horizontal resolution maneuver complement to the equipped threat aircraft. Operational Coordination Messages transmitted by ACAS X without horizontal resolution capability, HRC is set to the zero code. Codes are as defined in §2.2.3.8.3.2.4.1.

HSB: Horizontal Sense Bits - This 5-bit (53-57) subfield is a parity coding field used to protect the six horizontal sense bits (47-52). The originating Active CAS will include bits 53-57 in all Operational Coordination Messages sent. The receiving ACAS X will examine bits 53-57 in Operational Coordination Messages received. If bits 47-52 are not in agreement with bits 53-57, the receiving ACAS X will assume there is an error in the message and will not use the message contents. Codes are as defined in §2.2.3.8.3.2.4.1.

VSB: Vertical Sense Bits - This 4-bit (58-61) subfield is a parity coding field used to protect the four vertical sense bits (43-46). The originating Active CAS will include bits 58-61 in all Operational Coordination Messages sent. The receiving ACAS X will examine bits 58-61 in Operational Coordination Messages received. If bits 43-46 are not in agreement with bits 58-61, the receiving ACAS X will assume there is an error in the message and will not use the message contents. Codes are as defined in §2.2.3.8.3.2.4.1.

TAA: Threat Identity Aircraft Address - This 24-bit (65-88) subfield contains the ICAO 24-bit Aircraft Address of the threat aircraft that is the intended recipient of the OCM.

Note: The structure of ME for an Operational Coordination Message is:

<u>Position</u>	<u># of Bits</u>	<u>Subfield</u>	<u>Remarks</u>
33-37	5	TYPE	= 28
38-40	3	Subtype	= 3
41	1	-	<i>Not assigned</i>
42	1	MTB	-
43-44	2	CVC	-
45-46	2	VRC	-
47-49	3	CHC	-
50-52	3	HRC	-
53-57	5	HSB	-
58-61	4	VSB	-
62-64	3	-	<i>Not assigned</i>
65-88	24	TAA	-

2.2.3.8.3.2.9.2 Subfields in ME for Aircraft Operational Status Message

In the subfields defined below, the bit number is relative to the start of the extended squitter, where bit 33 is the start of the ME message field.

TYPE: This 5-bit (33-37) subfield defines the type of extended squitter. For the Aircraft Operational Status Message, TYPE = 31.

Subtype: This 3-bit (38-40) subfield further defines TYPE. Subtype = 0 for airborne aircraft; subtype =1 for surface aircraft. For ACAS X air-to-air coordination purposes, Subtype will always = 0.

CC: Airborne Capability Class Code - This 16-bit (41-56) subfield appears in Subtype=0 messages and is defined in Table 2-12 below.

Bits 41,42: For ACAS X air-to-air coordination purposes, these two bits will always = 0,0.

CA Operational: Collision Avoidance Operational - This 1-bit (43) subfield, when set to 1, indicates that a collision avoidance system is operational and capable of issuing RAs. When CA Operational=1, the Collision Avoidance Coordination Capability Bits may be examined to provide detailed coordination information.

Note: For TCAS and ACAS X, the associated Mode S transponder sets the CA Operational bit=1 when RI=3 or 4.

DAA : Detect and Avoid - This 2-bit (55-56) subfield is defined as follows:

Bits	Value	Meaning
55 56		

0 0	= 0	No DAA capability or no capability of DAA system to receive CA coordination information
0 1	= 1	Aircraft has a DAA system capable of receiving TCAS Resolution Messages and ADS-B OCMs
1 0	= 2	Aircraft has a DAA system capable of receiving only ADS-B OCMs
1 1	= 3	Not defined

Note: The DAA bits indicate whether – and what type of – coordination information needs to be provided to the aircraft so that the DAA system can listen and provide guidance that is interoperable with CA systems. These bits are independent of the CA Coordination Capability Bits, as aircraft with a DAA system may or may not have a CA system.

The type of coordination message transmitted [TCAS Resolution Message or Operational Coordination Message (OCM)] depends both on the receive capability of the DAA system and on the transmit capability of the CA system. If the DAA system can receive both the TCAS Resolution Message and the OCM, a CA system with 1030 MHz transmit capability is required to transmit the TCAS Resolution Message (§2.2.3.9.3.1).

The remainder of the CC subfield (bits 44-54) is not used by the ACAS X air-to-air coordination process.

Table 2-12: Capability Class (CC) Subfield in Aircraft Operational Status Messages, Airborne Format, in Version 2 Transmitting Subsystems

Msg Bit #	41	42	43	44	45	46	47	48	49	50	51	52	53-54	55-56
“ME” Bit #	9	10	11	12	13	14	15	16	17	18	19	20	21-22	23-24
Content	0,0	CA Operational	1090ES IN	Reserved = 0,0	ARV	TS	TC	UAT IN	Reserved for ADS-R	Reserved [2]		Reserved	DAA	
	0,1	Reserved												
	1,0	Reserved												
	1,1	Reserved												

OM: Airborne Operational Mode - This 16-bit (57-72) subfield appears in Subtype=0 messages and is defined in Table 2-13 below.

Bits 57,58: For ACAS X air-to-air coordination purposes, these two bits will always = 0,0.

CCCB: Collision Avoidance Coordination Capability Bits – This 7-bit (65-71) subfield is defined as follows:

Bits 65-66: Sense -Vertical and Horizontal

- 0 = vertical
- 1 = horizontal
- 2 = blended
- 3 = reserved

Bits 67-69: Aircraft CAS Type/Capability

- 0 = Active CAS (TCAS II)
- 1 = Active CAS (not TCAS II)
- 2 = Active CAS (not TCAS II) with OCM transmit capability
- 3 = Responsive CAS
- 4 = Passive CAS with 1030 MHz TCAS Resolution Message receive capability
- 5 = Passive CAS with only OCM receive capability
- 6-7 = Reserved

Note: An aircraft reporting ‘Responsive CAS’ has active 1030/1090 MHz interrogation/reply capability but is required by regulatory authorities (e.g., due to limited performance capability) to perform responsive coordination in encounters with an aircraft reporting ‘Active CAS.’ That is, the Responsive CAS ensures that its RA sense is compatible with that of the Active CAS, and the Responsive CAS does not transmit coordination information to the Active CAS. (See §2.2.3.9.3.1.)

Bits 70-71: Reserved

Note: These two bits were envisioned for UAS use, to serve as a priority field to distinguish among users with different levels of capability or as directed by regulatory authorities.

The remainder of the CC subfield (bits 59-64 and bit 72) is not used by the ACAS X air-to-air coordination process.

Table 2-13: Operational Mode (OM) Subfield in Aircraft Operational Status Messages, Airborne Format

Msg Bit #	57	58	59	60	61	62	63-64	65-71	72
“ME” Bit #	25	26	27	28	29	30	31-32	33-39	40
OM Format	0 0		CA RA Active [1]	IDENT Switch Active [1]	Reserved for Receiving ATC Services [1]	Single Antenna Flag [1]	System Design Assurance [2]	CCCB	Interrogation Active Operation [1]
	0 1	Reserved							
	1 0	Reserved							
	1 1	Reserved							

Note: The associated Mode S transponder sets (=1) the ‘CA Operational’ bit when RI \geq 3 and clears (=0) the bit when RI=0, 1, or 2. Also, for transponder versions 6 and later, the associated Mode S transponder sets (=1) the ‘Interrogation Active Operation’ bit when RI \geq 2 and clears (=0) the bit when RI= 0 or 1.

The newly-added ‘Interrogation Active Operation’ bit provides extra information that is expected to allow ADS-B messages to: (a) be used for ground monitoring of ACAS X performance, and (b) potentially replace the 1030 MHz TCAS Broadcast Interrogation Messages that are currently used in ACAS X Interference Limiting algorithms.

The ‘Interrogation Active Operation’ bit allows a ground user monitoring ACAS X performance via ADS-B messages to know whether ACAS X is operating in TA-Only mode. TA-Only mode is indicated by bit 43=0 (CA Operational Bit) AND bit 40=1 (Interrogation Active Operation) in the ADS-B Operational Status Message. ACAS X operating in TA/RA mode would be indicated by bit 43=1 AND bit 40=1.

2.2.3.8.3.2.10 ME Fields Used by ACAS X for Communication of RA Information

2.2.3.8.3.2.10.1 ADS-B TCAS RA Broadcast

The ADS-B TCAS RA Broadcast is identified by TYPE=28 and Subtype=2 in bits 33-40 of the 112-bit Extended Squitter transmission; (bits 33-40 are the first 8 bits of the 56-bit ME field).

Bits 41-88 (i.e., the last 48 bits of the ME field) of the ADS-B TCAS RA Broadcast are identical to bits 41-88 of the RA Report (§2.2.3.8.3.2.3.1). ACAS X deposits these bits into register 30₁₆ of its associated Mode S transponder. The transponder will insert the bits into DF20, 21 replies for an RA Report to a Mode S ground sensor and into 1090 MHz Extended Squitters for an ADS-B TCAS RA Broadcast.

Note: Bit 61 of the ADS-B TCAS RA Broadcast, which is set equal to CNT (§2.2.3.8.3.2.3.1), should be disregarded when using the ADS-B TCAS RA Broadcast to monitor ACAS X performance because it reports whether or not a further register can be interrogated by Mode S ground stations.

2.2.3.9 ACAS X Signal Protocol

2.2.3.9.1 Mode C Surveillance Signals

2.2.3.9.1.1 ATCRBS-Only All-Call Interrogation

For surveillance of aircraft equipped with ATCRBS transponders, the ATCRBS-Only All-Call interrogation **shall** (1105) be used (Ref. B, §2.1.11.3.2).

Note: This interrogation format causes all ATCRBS-only equipped aircraft to reply and all Mode S equipped aircraft not to reply. The absence of replies from Mode S transponders is desirable since such replies would add to the interference environment. Absence of these replies from the ACAS X synchronous interference environment is particularly beneficial.

2.2.3.9.2 Mode S Surveillance Signals

Note: The following paragraphs describe the interactions between an airborne ACAS X and a Mode S equipped aircraft required for the proper functioning of the ACAS X Mode S surveillance tasks.

2.2.3.9.2.1 Detection

ACAS X detects the presence of Mode S-equipped aircraft by listening at 1090 MHz to Mode S transmissions with DF=11 (All-Call reply). ACAS X may also detect the presence of Mode S-equipped aircraft by optionally listening to Mode S transmissions with DF=17 (Extended Squitter). Mode S transponder transmissions with DF=11 may occur in response to Mode S ground sensor All-Call interrogations or as squitters at a maximum period of 1.2 seconds. Mode S transponder transmissions with DF=17 occur as extended squitters at a maximum period of 0.6 seconds for airborne aircraft.

Note: Although optional, listening to DF=17 (Extended Squitter) transmissions for Mode S transponder acquisition is recommended and may become a requirement in the future.

The Capability field, CA, in DF=11 transmissions from RTCA/DO-181A or later transponders (§2.2.3.8.3.1.5) indicates whether the vertical status of the Mode S equipped aircraft is airborne, on the ground, or unknown. ACAS X **shall** (1106) make use of the CA field information from RTCA/DO-181A or later transponders to prevent unnecessary interrogations to aircraft that are on the ground.

Note: For example, it is necessary to interrogate some on-ground ACAS for NTA3 and NTA6 computations.

The altitude of Mode S-equipped aircraft not indicated to be on the ground or aircraft with RTCA/DO-181 transponders is then determined by passively monitoring transmissions received from that aircraft with DF=4 (surveillance reply with altitude) or with DF=0 (short special surveillance), or by actively interrogating the aircraft to elicit a short special surveillance reply.

ACAS X derives the ICAO 24-bit Aircraft Address of Mode S-equipped aircraft from the AA field of DF11 or DF17 messages. Additionally, ACAS X **shall** (1107) examine the Vertical Status field, VS, in the DF=0 transmissions from the aircraft in order to minimize interrogations and prevent tracking of aircraft on the ground.

Note: If the aircraft is on the ground and is under Mode S ground surveillance, ACAS X may also be able to determine that the aircraft is on the ground by monitoring the FS field in downlink formats DF=4, 5, 20, 21 (§2.2.3.8.3.1.8).

2.2.3.9.2.2 Surveillance Interrogations

2.2.3.9.2.2.1 Acquisition

ACAS X **shall** (1108) use the short special surveillance interrogation, UF=0, for range acquisition. ACAS X **shall** (1109) set AQ=1 in acquisition interrogations. This serves as an aid in distinguishing the reply to own interrogation from replies elicited by other Active CAS equipment. In addition, ACAS X **shall** (1110) set RL=0 in acquisition interrogations to command short acquisition replies, DF=0.

2.2.3.9.2.2.2 Tracking

ACAS X **shall** (1111) use the short special surveillance interrogation, UF=0, for tracking (non-acquisition) interrogations. ACAS X **shall** (1112) set AQ=0 and RL=0 (to command a short reply, DF=0) in tracking interrogations.

2.2.3.9.2.3 Surveillance Replies

On receipt of a short special interrogation, UF=0, or a long special interrogation, UF=16, the interrogated Mode S transponder replies with a short special reply, DF=0, or a long special reply, DF=16, depending on the code contained in the RL field of the interrogation. In this reply, the sensitivity level and the air-to-air reply information are reported in the SL and RI fields. Operating ACAS X equipment **shall** (1113) cause the SL field to be set appropriately to indicate the sensitivity level at which it is currently operating (§2.2.3.8.3.2.6). Non-operating ACAS X equipment **shall** (1114) cause RI=0 to be set. If RI=0 is set, the SL field has no meaning.

2.2.3.9.2.3.1 Acquisition

On receipt of a short or long special surveillance interrogation containing a '1' in the AQ field (acquisition interrogation), the interrogated Mode S transponder reports only the maximum airspeed capability of the aircraft in the RI field (§2.2.3.8.3.1.14) of the reply; codes 0 through 7 are not used in this reply.

Note: The protocol described above for the RI field implicitly identifies the reply as an acquisition reply through RI because the first bit of RI is effectively an echo of AQ(=1) received in the acquisition interrogation.

2.2.3.9.2.3.2 Tracking

On receipt of a short or long special surveillance interrogation containing a '0' in the AQ field (non-acquisition interrogation), the interrogated aircraft sets, as appropriate, a code in the 0 through 7 block of codes in the RI field of the reply; codes 8 through 15 are not used in this reply.

Note: The protocol described above for the RI field implicitly identifies the reply as a tracking (non-acquisition) reply through RI because the first bit of RI is effectively an echo of AQ(=0) received in the tracking interrogation.

2.2.3.9.2.4 TCAS Broadcast Interrogations

ACAS X **shall** (1115) use the long special surveillance interrogation, UF=16, with UDS=50 and a broadcast address to transmit periodic TCAS Broadcast Interrogation Messages (§2.2.3.8.3.2.4.2) to other Active CAS equipment within its range. A broadcast address with UF=16 causes transponders associated with Active CAS equipment to accept the transmission without replying and to present the interrogation content at the transponder/TCAS interface (Ref. B, §2.2.17.1.11). TCAS Broadcast Interrogation Messages contain the ICAO 24-bit Aircraft Address of the interrogating Active CAS aircraft and allow the interrogating Active CAS to determine the number of other Active CAS-equipped aircraft within its detection range for the purpose of limiting interference (§2.2.3.6.1). TCAS Broadcast Interrogation Messages **shall** (1116) be transmitted at full power and **shall** (1117) be transmitted such that, for any other Active CAS aircraft within 30 NM and at any azimuth, the nominal rate of own TCAS Broadcast Interrogation Messages arriving at that Active CAS is 1 per 8 to 10 seconds.

2.2.3.9.3 Coordination with Other Aircraft

This section describes the actions required for ACAS X to carry out the air-to-air coordination function. Corresponding requirements are placed upon the Mode S transponder associated with each ACAS X. These transponder requirements appear in Ref. B, §2.2.22. An overview of the coordination function is given in §1.3.3, Coordinating RAs Against ACAS-Equipped Threats.

2.2.3.9.3.1 Determining Own Coordination Protocol

ACAS X **shall** (2091) use intruder information in the sequence below to determine whether to coordinate with the intruder and if so, what type of coordination protocol to perform (Active, Modified Active, or Modified Passive) – see glossary. ACAS X transmits coordination information only when a requirement in (1) or (2) below is met.

- (1) If DF=0 reply is available and RI = 3 or 4, ACAS X **shall** (2094) perform Active coordination, meaning that ACAS X will determine the master/slave setting from the two ICAO 24-bit Aircraft Addresses and will transmit a TCAS Resolution Message if ACAS X declares the intruder to be a threat.

Notes:

1. *Aircraft installations that do not include a 1030MHz transmitter are not allowed to set RI=3 or 4. Setting RI=3 or 4 indicates that the aircraft can fully participate in active coordination.*
2. *The aircraft with the lower ICAO 24-bit Aircraft Address is the master; the aircraft with the higher ICAO 24-bit Aircraft Address is the slave.*

- (2) Otherwise (no DF=0 reply is available or RI ≠ 3 or 4),

- A. If CA Operational = 1,

ACAS X **shall** (2095) use the intruder's CCCB to determine whether the intruder is performing responsive coordination with respect to ownship.

Note: The intruder is performing responsive coordination if its CCCB Aircraft CAS Capability/Type equals any of the following: 'Responsive CAS,' 'Passive CAS with 1030 TCAS Resolution Msg receive capability,' or 'Passive CAS with only OCM receive capability.'

If the intruder is performing responsive coordination with respect to own, then:

- i. ACAS X **shall** (2096) disregard any received TCAS Resolution Messages from this intruder; and
- ii. If the intruder can receive TCAS Resolution Messages, i.e., if the intruder is reporting 'Responsive CAS' or 'Passive CAS with 1030 TCAS Resolution Message receive capability', ACAS X **shall** (2097) perform Modified Active coordination, meaning that ACAS X will be the master regardless of ICAO 24-bit Aircraft Addresses and will transmit a TCAS Resolution Message if ACAS X declares the intruder to be a threat; and
- iii. If the intruder can receive only OCMs, i.e., if the intruder is reporting 'Passive CAS with only OCM receive capability', ACAS X **shall** (2098) perform Modified Passive coordination, meaning that ACAS X will be the master and will send an OCM to the ownship Mode S transponder for ADS-B transmission if ACAS X declares the intruder to be a threat.

B. If CA Operational=0 and DAA= non-zero,

If DAA = 1, ACAS X **shall** (2099) transmit a TCAS Resolution Message if ACAS X declares the intruder to be a threat.

If DAA = 2, ACAS X **shall** (2100) send an OCM to the ownship Mode S transponder for ADS-B transmission if ACAS X declares the intruder to be a threat.

2.2.3.9.3.2 Transmission of Coordination Interrogations to Other Active CAS Aircraft

When ACAS X declares another aircraft reporting RI=3 or 4 to be a threat, interrogations to that aircraft **shall** (1118) be transmitted for RA coordination. Coordination interrogations **shall** (1119) use the TCAS Resolution Message format. Coordination interrogations **shall** (1120) be transmitted each update period to the other aircraft as long as the other aircraft remains a threat.

In the absence of a reply to a coordination message, ACAS X **shall** (1121) re-interrogate to accomplish successful transmission of the TCAS Resolution Message. Such re-interrogations **shall** (1122) be transmitted within a 100 ± 5 millisecond period and **shall** (1123) be repeated not less than six times and not more than twelve times nominally spaced at equal intervals over the duration of this period. Interrogations **shall** (1124) be randomly jittered about the nominal spacing by at least 360 microseconds in order to prevent synchronous interference with other ground-based and airborne interrogators.

If the maximum number of attempts is made and no reply is received, ACAS X **shall** (1125) continue its resolution process without further attempts to coordinate with this threat during the current update period.

A TCAS Resolution Message sent by ACAS X **shall** (1126) use the long special surveillance interrogation, UF=16, with the MU field containing the fields specified in §2.2.3.8.3.2.4.1 and the following codes:

- | | | |
|-----|-----|----------------------------------|
| AQ | = 0 | (non-acquisition interrogation). |
| RL | = 1 | (respond with long reply). |
| HRC | = 0 | (no horizontal complement sent). |

- CHC = 0 (no horizontal complement to be canceled).
HSB = 0 (Parity associated with combined HRC, CHC fields).

Coordination interrogations **shall** (1127) also be transmitted to an aircraft declassified from threat status within one update period following cancellation of the RA with respect to that threat. In this case, the CVC subfield **shall** (1128) contain a cancellation code for the previously sent VRC. Such coordination interrogations **shall** (2233) be transmitted for six cycles or until a Coordination Reply Message is received, whichever occurs first.

Notes:

1. *Coordination interrogations transmitted to an aircraft declassified from threat status adhere to the re-interrogation requirements above (i.e., 6-12 re-interrogations within a 100 msec interval are transmitted each cycle as necessary).*
2. *The six cycle re-transmit in the above sentence is consistent with VRC timeout specified in §2.2.3.9.3.3.*

2.2.3.9.3.3 Capacity and Handling Requirements for Incoming Coordination Interrogations

ACAS X **shall** (1129) have the capacity to maintain information corresponding to at least ten concurrent threats for any mix of Mode C- and Mode S-equipped threats. For coordination purposes, this means that ACAS X **shall** (1130) be able to store incoming TCAS Resolution Messages corresponding to at least ten unique ICAO 24-bit Aircraft Addresses.

ACAS X **shall** (1131) prevent concurrent access of intruder coordination data (e.g. VRC, CVC, etc.) in the ADD Design Note (Volume II, Appendix K).

Note: Care must be taken to prevent simultaneous reading and writing by concurrent processes. A process that is interrupted or suspended while another task executes is considered concurrent with the interrupting or suspending task. The potential for simultaneous data access exists because incoming TCAS Resolution Messages are received asynchronously to ACAS X processing, effectively interrupting this processing.

The ACAS X processor **shall** (2101) receive TCAS Resolution Messages from the transponder and **shall** (2102) be able to process these messages at a continuous rate of 60 per second (16.6 milliseconds per message).

Notes:

1. *The 60 message-per-second requirement derives from the requirement for ACAS X to maintain information for at least ten concurrent threats. Of the 60 messages per second, at most 10 messages per second are unique. If all of these threats were Active CAS-equipped, and if each threat transmitted the maximum of 12 TCAS Resolution Messages per second, theoretically 120 TCAS Resolution Messages per second could be received. The requirement for reception of TCAS Resolution Messages was set at half the theoretical maximum, which provides sufficient capacity for all credible scenarios.*
2. *Reference B requires that the transponder be able to receive long Mode S interrogations (UF=16) and generate long Mode S replies (DF=16) at a continuous rate of 60 per second. Assuming the 60-per-second message input rate,*

the transponder is required to deliver TCAS Resolution Messages to Active CAS-equipped aircraft within 0.01 second of receipt by the transponder.

The ACAS X Processor **shall** (2193) clear an intruder VRC (i.e., a received VRC associated with a specific intruder track file) in each of the following three circumstances:

- (1) immediately upon receipt of a CVC (i.e., VRC cancellation) from that intruder,
- (2) immediately upon dropping of that intruder track, and
- (3) on the 6th cycle in which no VRC is received from that intruder.

Notes:

1. *The timeline for clearing a VRC mirrors the approach taken for dropping a Mode S intruder from which no replies have been received as described in §2.2.4.6.4.2.3.2.5 (See Figure 2-12).*
2. *Timely clearing of VRCs is necessary to prevent potentially damaging limitations in ownship sense selection against an intruder.*

2.2.3.9.3.4 Coordination Delivery Delay Requirements

In an equipped-equipped encounter, coordination delivery delay is defined as the elapsed time between sense selection in one aircraft and use of the transmitted VRC in sense selection by the second aircraft. The coordination delivery delay is divided into five segments:

T1: Time from sense selection to transmission of VRC via UF16.

Note: The T1 time delay segment is accumulated on the aircraft transmitting the VRC, while delays T2-T5 occur on the aircraft receiving and acting upon the VRC.

T2: Time from receipt of incoming VRC by transponder to delivery to ACAS X processor.

Note: T2 is defined in the Reference B to be 10 msec.

T3: Time from receipt of incoming VRC by ACAS X processor to its availability for use by the TRM (Threat Resolution Module).

T4: Time required by TRM to reach sense selection stage after VRC is made available.

Note: The ADD "reach-back" concept allows the TRM to request the most recent VRC at any time following receipt of the STM Report up until the beginning of "Online Cost Estimation." Thus, the VRC is not locked-in during this time, and T4=0 for purposes of coordination delivery delay.

T5: Time that sense selection requires to complete.

Note: This represents the processing time of 'OnlineCostEstimation' in the TRM. The VRC is locked-in once sense selection begins.

The requirement for each time segment **shall** (2109) be as defined in Table 2-14 below. Each requirement is given in terms of a value to be met at least 95% of the time and a value to be met 100% of the time.

Table 2-14: Coordination Timing Requirements

	95%	100%
T1	25 msec	40 msec
T2	10 msec	10 msec
T3 +T4 + T5	25 msec	40 msec
Total	60 msec	90 msec

Notes:

1. *Manufacturers are encouraged to minimize these values as they will result in lower probability of quick reversals.*
2. *T2 is a requirement placed upon the Mode S transponder installed onboard the ACAS X aircraft [Ref B, §2.2.22.2.4, “Communication Timing”].*

2.2.3.9.3.5 Additional Coordination Topics

2.2.3.9.3.5.1 Slave Geometric Reversals

Beginning with TCAS V7.0 and continuing into ACAS X, a master is permitted to initiate ‘geometric reversals’: Each cycle, the master reassesses its RA intent and can reverse its intent based on geometric considerations. The master is limited to one geometric reversal per intruder per encounter. If the master reverses intent, then the slave is forced to reverse its intent in order to maintain compatibility with the master. A slave reversal to maintain compatibility with the master is referred to as a ‘coordination reversal.’

Prior to ACAS X, a slave could never initiate geometric reversals. An ACAS X slave **shall** (2226) be permitted to initiate a ‘slave geometric reversal’ if warranted by geometric considerations when there is no currently valid VRC from the master.

Note: ‘Currently valid’ includes VRCs received in the current cycle or coasted.

Like the master, the slave **shall** (2227) be limited to one geometric reversal per intruder per encounter.

Note: The terms ‘master’ and ‘slave’ have meaning only in coordinated encounters and are used only in coordinated encounters, i.e., in encounters in which both aircraft are collision avoidance equipped.

2.2.3.9.3.5.2 Coordination Delay When Transitioning from TA-Only Mode to TA/RA Mode

When a track is established, the TRM imposes a 3-second track stabilization delay [Ref: ADD InitializationCost]. In addition, whenever ACAS X transitions from TA-Only mode to TA/RA mode and ACAS X is in an encounter with an intruder equipped with Active CAS (e.g., TCAS or ACAS X), ACAS X **shall** (2228) delay issuing an RA for 3 seconds unless it receives a VRC from the intruder [Ref: ADD CoordinationDelayCost]. The counters for the track stabilization delay and the coordination delay are independent and run concurrently when both apply.

If the intruder is not equipped with Active CAS, ACAS X **shall** (2229) apply NO coordination delay.

Notes:

- 1: *The transition could occur in a number of circumstances, e.g., when the pilot switches manually from TA-Only to TA/RA mode, or when the aircraft passes above the RA inhibit altitude.*
- 2: *The reason for the coordination delay is that own ACAS X, reacting to an encounter already in progress, should not unnecessarily reverse the sense of the intruder's RA (if ownship is the master) or unnecessarily display an RA that may be immediately reversed (if ownship is the slave).*

The delay is not applied if the intruder is unequipped or equipped with a CAS other than Active CAS. An unequipped intruder would not send any VRC, and ACAS X would disregard any VRC received from a CAS other than an Active CAS, thus applying the 3-second delay and potentially degrading ACAS X performance.

- 3: *The maximum delay is 3 seconds because surveillance studies have shown virtually 100% probability of VRC reception within 3 seconds, given that the two aircraft are close to one another, the coordination interrogations are transmitted at full power, and coordination interrogations are retried if necessary 6-12 times over a 100ms interval.*

2.2.3.9.3.6 Use of Extended Squitter ME Field Information in Air-to-Air Coordination

2.2.3.9.3.6.1 Coordination Information Received by ACAS X in Incoming ADS-B Messages

2.2.3.9.3.6.1.1 Operational Coordination Message

If the ME field of a valid extended squitter has TYPE subfield = 30 and Subtype = 0, then the ME field contains an Operational Coordination Message. ACAS X **shall** (2103) disregard incoming Operational Coordination Messages.

2.2.3.9.3.6.1.2 Aircraft Operational Status Message

If the ME field of a valid extended squitter has TYPE subfield = 31 and Subtype = 0, then the ME field contains an Aircraft Operational Status Message from an airborne aircraft, and ACAS X **shall** (2104) decode the subfields of that ME field according to the format in §2.2.3.8.3.2.9.2.

ACAS X uses the intruder's CA Operational bit, DAA bits, and CCCB in determining whether to coordinate with the intruder and if so, what type of coordination protocol to perform as described in §2.2.3.9.3.1.

2.2.3.9.3.6.2 Coordination Information Composed by ACAS X for Inclusion in Outgoing ADS-B Messages

2.2.3.9.3.6.2.1 Operational Coordination Message

Each cycle in which the TRM issues an RA with respect to an intruder that has been determined by the requirements in §2.2.3.9.3.1 to be EITHER

(a) 'Passive CAS with only OCM receive capability' OR

(b) CA Operational=0 and DAA=10,

ACAS X **shall** (2105) compose an Operational Coordination Message according to the format in §2.2.3.8.3.2.9.1.

Note: Operational Coordination Messages are sent to the transponder per §2.2.3.12.2.2.5 for inclusion in outgoing ADS-B Messages.

2.2.3.9.4

Communication of RA Information

ACAS X communicates its RA to the pilot of own ACAS X aircraft by means of the RA display and aural annunciation subsystems. ACAS X also communicates RA information in the 56-bit message fields of four RF transmissions:

- (a) In a 1090 MHz RA Report [§2.2.3.8.3.2.3.1 and §2.2.3.12.2.2.4] to a Mode S ground sensor when the ACAS X-associated transponder indicates that RA information is available for read-out by the ground sensor.
- (b) In a 1090ES ADS-B TCAS RA Broadcast Message [§2.2.3.12.2.2.5], transmitted at approximately 0.8 second intervals during the period that an RA is active and intended for reception by ADS-B ground monitoring stations.
- (c) In a 1090 MHz Coordination Reply Message [§2.2.3.8.3.2.5.1 and §2.2.3.12.2.2.3] sent by TCAS or ACAS X-equipped aircraft in response to a received TCAS Resolution Message (coordination interrogation). The Coordination Reply Message is not used for any purpose within the collision avoidance logic. The purpose of the reply is to acknowledge to the originating aircraft that its TCAS Resolution Message was received. (TCAS Resolution Messages are required to be re-transmitted until a reply is received up to a maximum of 6-12 retries.)
- (d) In a 1030 MHz RA Broadcast Interrogation Message [§2.2.3.8.3.2.4.3 and §2.2.3.9.2.4]. The RA Broadcast Interrogation Message, transmitted at approximately one second intervals, is intended for reception by ground-based systems that may monitor RAs in airspaces not covered by full-capability Mode S ground sensors.

As highlighted in Figure 2-5, all four RF messages contain the same core subfields, including the Active RA (ARA) subfield, which contains detailed information on the sense, strength and classification of the displayed RA; the RA Complements (RAC) subfield, a composite of all currently active RA complements (if any) received from other CAS-equipped aircraft; the RA Terminated Indicator (RAT) subfield, which is set to one when the RA display ceases in the cockpit; the Multi-threat Encounter (MTE) subfield, which indicates whether this RA addresses only one threat or several threats; and the RA Message Format (RMF), which indicates whether this message contains information for a TCAS, ACAS Xa, or ACAS Xu system.

The RA Report and ADS-B TCAS RA Broadcast Message also include the Threat Type Indicator (TTI) subfield, distinguishing between a Mode S or ATCRBS threat. For Mode S threats, TID contains the ICAO 24-bit Aircraft Address of the most recent threat; for ATCRBS threats, TID contains the relative range, altitude and bearing of the most recent threat. The RA Broadcast Interrogation Message does NOT identify the threat aircraft. Because it is a 1030 MHz interrogation, the RA Broadcast Interrogation Message does not include ownership address in the last 24 bits of the 112-bit transmission; thus ownership identity must be included within the 56-bit message field. The ownership is identified by

Mode A identity code (AID) and Mode C altitude code (CAC), allowing the RA information to be correlated with information from an ATCRBS ground sensor.

# of bits	Subfield
4	BDS1
4	BDS2
10	ARA
2	LDI
2	RMF
4	RAC
1	RAT
1	MTE
1	CNT
1	TTI
24	TID*
1	DSI
1	SPI

# of bits	Subfield
5	Type
3	Subtype
10	ARA
2	LDI
2	RMF
4	RAC
1	RAT
1	MTE
1	CNT
1	TTI
24	TID*
1	DSI
1	SPI

# of bits	Subfield
4	VDS1
4	VDS2
10	ARA
2	LDI
2	RMF
4	RAC
1	RAT
1	MTE
28	Not Assigned

# of bits	Subfield
4	UDS1
4	UDS2
10	ARA
2	LDI
2	RMF
4	RAC
1	RAT
1	MTE
1	SPI
1	Not Assigned
13	AID
13	CAC

*TID = 24-bit ICAO address

OR
11-bit TIDA
7-bit TIDR
6-bit TIDB

Figure 2-5: RF Transmissions Containing RA Information

2.2.3.9.5

Communication With Mode S Ground Sensors

Note: The ACAS X processor provides information to its associated Mode S transponder for inclusion in replies to a Mode S ground sensor. ACAS X requirements associated with downlinking information to a Mode S ground sensor are stated in §2.2.3.9.5.1 and §2.2.3.9.5.2 and in §2.2.3.12.2.2. Corresponding transponder requirements are given in Ref. B, §2.2.22.1.2.1, §2.2.22.1.2.2, and §2.2.22.3.4, a. and b.

Information transmitted by a Mode S ground sensor to ACAS X is received by ACAS X's associated Mode S transponder and passed to ACAS X via the transponder/ACAS X interface. ACAS X requirements associated with receiving and processing uplinked information from a Mode S sensor are stated in §2.2.3.9.5.4 and in §2.2.3.12.2.3. Corresponding transponder requirements are given in Ref. B, §2.2.22.1.1.

2.2.3.9.5.1

RA Report

When ACAS X provides RA information to the transponder (§2.2.3.12.2.4), it **shall** (1140) provide a means to determine whether an RA is active. An active RA **shall** (1141) be indicated by use of an RA Indicator (RAI) bit. RAI = '0' indicates that there is an active RA; RAI = '1' indicates that there is no active RA.

DO-385

© 2018 RTCA, Inc.

Each cycle, the TRM outputs subfield values for the RA Report in a data block referred to as *Ground_Msg*. ACAS X constructs the RA Report from these *Ground_Msg* subfields. The *Ground Msg* data block contains a bit that is referred to as RAT (although its setting differs from that of the RAT bit in the 1090 MHz RA Report that is transmitted by the transponder). ACAS X **shall** (2605) replace this RAT bit with the RAI bit. ACAS X **shall** (1138) clear RAI (RAI=0) when the TRM output contains both RAT=0 and non-zero ARA strength bits. Otherwise ACAS X **shall** (2606) set RAI=1.

In the TRM output, RAT=0 on every cycle except the one cycle that terminates an RA. On the one cycle that RAT=1, the TRM outputs all of the other subfields with the values they had on the previous cycle (when the RA was last active).

Notes:

1. *The shalls in the above paragraphs are expected to be carried out by the ACAS X SPM/Front End.*
2. *The transponder MOPS [RTCA/DO-181E §2.22.1.2.1.3] states that “Termination of the RA is indicated by the transition of RAI from ZERO to ONE.”*
3. *Following the end of an RA, the transponder retains the RA Report for an additional 18±1 seconds in order to allow read-out by a ground sensor's rotating antenna.*

2.2.3.9.5.2 Data Link Capability Report

The ACAS X processor **shall** (1143) provide TCAS capability information to its associated Mode S transponder for inclusion in a Data Link Capability Report (§2.2.3.8.3.2.3.2).

2.2.3.9.5.3 ACAS X Unit Part Number and ACAS X Software Part Number

ACAS X **shall** (1144) determine whether or not to transfer its part numbers to its associated Mode S transponder. If so, ACAS X **shall** (1145) transmit its ACAS X unit part number to transponder register E5 hex and **shall** (1146) transmit its ACAS X software part number to transponder register E6 hex. ACAS X **shall** (1147) initiate the E5 and E6 transmissions to the transponder once per 10 seconds with a tolerance of plus one or minus one second.

Formats for transponder registers E5 and E6 are defined in Attachment 19L of ARINC Characteristic 735B [Ref. L].

2.2.3.9.5.4 TCAS Sensitivity Level Control

For TCAS II systems, control of the TCAS sensitivity level can be accomplished by one or more ground-based Mode S sensors through the transmission of Comm-A interrogations, UF=20, 21, containing TCAS Sensitivity Level Command Messages (§2.2.3.8.3.2.2) to the TCAS II aircraft. The interrogator identification information required to correlate the sensitivity level command (SLC) with a particular originating ground-based Mode S sensor site is contained in the IIS subfield (§2.2.3.8.3.1.17) of SD of the same Comm-A.

ACAS X **shall** (1149) not use the Sensitivity Level value obtained from a Sensitivity Level Command Message to modify the value of ownship Sensitivity Level.

2.2.3.9.6 Communication With Other Ground Equipment

2.2.3.9.6.1 RA Broadcast Interrogations

ACAS X **shall** (1152) use the long special surveillance interrogation, UF=16, with UDS=49 and a broadcast address (i.e., 24 ONES) to transmit RA Broadcast Interrogations. RA Broadcast Interrogations **shall** (1153) be transmitted at full power from the bottom antenna at jittered, nominally 1-second, intervals for the period that the RA is active. Additionally, one interrogation, indicating RA termination, **shall** (1154) be transmitted immediately after the RA is concluded. The RA Broadcast Interrogation **shall** (1155) include the MU field as specified in §2.2.3.8.3.2.4.3 and **shall** (1156) describe the most recent RA that existed during the preceding 1-second period. Installations using directional antennas **shall** (1157) operate such that complete circular coverage is provided nominally every 1 second and the same RA sense and strength is broadcast in each direction.

Note: This allows RA activity to be monitored in areas where Mode S ground station surveillance coverage does not exist by using special RA broadcast signal receivers on the ground. RA broadcasts are normally destined for ground equipment but are defined as uplink transmissions.

2.2.3.9.7 Mode S Crosslink Capability

Mode S crosslink capability provides the capability to request and receive ground-initiated Comm-B register information from Mode S transponders able to provide this information. The following paragraph describes the interaction required between an airborne ACAS X which utilizes crosslink interrogations and a Mode S-equipped aircraft to properly carry out the Mode S crosslink function. At a minimum, crosslink interrogations would need to be accounted for by the Interference Limiting algorithms as specified in §2.2.3.6, and additionally impact to ACAS X range performance would need to be considered.

ACAS X identifies those Mode S transponders able to support the crosslink capability by examining the CC field in short special surveillance format DF=0 replies. ACAS X can then request the contents of a specific Comm-B register in a Crosslink-capable Mode S transponder by setting RL=1 and designating the identity of the register in the BDS field of a short special surveillance format UF=0 interrogation addressed to that transponder. In response to this request, the Mode S transponder will reply with a long special surveillance format DF=16 in which the MV field contains the requested Comm-B register data.

In particular, in response to a UF=0 with RL=1 and BDS=8, the transponder will reply with a DF=16 in which the MV field contains the aircraft identification message (Ref. B: Appendix 1, Section 4.0). In response to a UF=0 with RL=1 and BDS=0, the transponder will reply with a DF=16 with an MV field of all 0s.

2.2.3.9.8 Extended Squitter With Aircraft Identification Message

ACAS X may optionally be implemented to receive and decode the content of the ME field in format DF=17 Extended Squitters containing the aircraft identification message. The aircraft identification message is identified by format type code 1, 2, 3 or 4 in the first five bits of the message field, and the contents of the message are defined in Ref. G.

Note: The ME field provides callsign information.

2.2.3.10 Compatibility With Own Mode S Transponder

The ACAS X equipment will operate in conjunction with and in close proximity to a Mode S transponder. The functions of ACAS X equipment **shall** (1158) not in any way

cause the Mode S transponder to fail to meet its performance requirements as specified in Ref. B. Similarly, the Mode S transponder should not cause ACAS X to fail to meet its requirements stated herein. It is the responsibility of the manufacturer to identify all areas where incompatibilities may arise and to design and construct his equipment so that the requirements of Ref. B. and this document are met when both equipment are operating.

Two general areas of potential incompatibility are identified below as examples; others may be found to exist in accordance with specific equipment designs.

- a. The first area of possible incompatibility concerns the methods used to interface the transponder and the ACAS X equipment. It is not within the scope of this document to control or test specific interface design details within ACAS X. It is the responsibility of the manufacturer to verify that his equipment does in fact interface in such a way that all overall logical, protocol, timing and other requirements or constraints imposed on the combined operation of the two functional entities have been fulfilled under any possible combination of operating conditions.

This may be accomplished by conducting all testing with both functions fully operative and performing their normal functions by means of simulated inputs and/or outputs. For example, when tests are under way to verify that the ACAS X equipment requirements of this document are met, the Mode S transponder could be periodically interrogated by simulated input signals near its MTL level.

- b. The second area of potential incompatibility concerns electromagnetic interference. Since each unit generates transmission signals at the receiver frequency of the other, it is possible for signals from one to degrade the performance of the other.

All of the ACAS X requirements stated in this document **shall** (1159) be met when the ACAS X equipment is operating in conjunction with an operating Mode S transponder with the possible exception of those times that the Mode S transponder is active. The active state of the Mode S transmitter is defined as either the time interval between the leading edge of the first transmitted pulse of a reply minus 10 microseconds and the trailing edge of the last transmitted pulse of that reply plus 10 microseconds, or the time interval during which a mutual suppression occurs, whichever is greater.

2.2.3.11

Aircraft Suppression Bus

The ACAS X equipment **shall** (1160) issue a 70 ± 1 microsecond suppression pulse to other on-board aircraft equipment beginning at each interrogation transmitted from the top-mounted antenna. The ACAS X equipment **shall** (1161) issue a 90 ± 1 microsecond suppression pulse to other equipment for each interrogation from the bottom antenna.

The ACAS X equipment **shall** (1162) be designed to accept and respond to interference suppression pulses from other electronic equipment in the aircraft (to disable it while the other equipment is transmitting). The ACAS X equipment **shall** (1163) regain normal sensitivity, within 3 dB, not later than 15 microseconds after the end of the applied interference suppression pulse.

Note: The suppression duration must be longer than the interrogation to ensure the on-board transponder does not respond to reflections of ACAS X interrogations from the ground. The durations specified above have been determined experimentally to be adequate for this purpose. This document does not establish the design parameters of the interference suppression system other than the durations. However, it is recommended that all sources of interference suppression pulses be DC coupled and sinks be AC coupled. This standardization will prevent source or sink failures from disabling all users of the interference suppression pulses.

2.2.3.12 Interfaces With Other Systems

2.2.3.12.1 Mode S Transponder Related Interfaces

The ACAS X equipment **shall** (1164) accept the following from its associated Mode S transponder:

- a. The aircraft ICAO 24-bit Aircraft Address.
 - b. The aircraft Pressure Altitude from the source that is the basis for own Mode S altitude replies. If coarse altitude is indicated, ACAS X **shall** (1165) round the altitude to the nearest 100 feet.
 - c. Quantization for pressure altitude (fine [10 ft or less] or coarse [more than 10 ft]).
- Note: RTCA/DO-181A and later versions state that: (1) when selecting the altitude source used for Mode S replies and for TCAS, the transponder shall use the source that is valid and provides the finest quantization, and (2) the altitude data shall be provided to TCAS at the finest quantization available.*
- d. The aircraft Mode A Identity Code.
 - e. Mode S subnetwork version (from register 10₁₆).

2.2.3.12.2 ACAS X Interface with Mode S Transponder

The ACAS X equipment **shall** (1166) receive messages from the Mode S transponder via the transponder/ACAS X interface.

Notes:

1. *It is recommended that the processing of TCAS Resolution Messages take priority over the processing of TCAS Broadcast Interrogation Messages (§2.2.3.9.3.4).*
2. *The manufacturer may combine the functions of the Mode S transponder with those of the ACAS X system in a single unit with internal interfacing or may implement these functions in two separate units connected via a remote interface. The requirements below apply in either case.*

2.2.3.12.2.1 General Requirements of the Interface to the Mode S Transponder

2.2.3.12.2.1.1 ACAS X/Transponder System Capability Determination

ACAS X **shall** (1169) annunciate to its associated Mode S transponder that it is RTCA/DO-185A,B,ACAS X compatible. Likewise, ACAS X **shall** (1170) receive from its associated Mode S transponder an annunciation stating either that the transponder is RTCA/DO-185A,B compatible or that it is FAA TSO-C119A compatible. The system **shall** (1171) only inter-operate with RTCA/DO-185A,B compatible transponders. ACAS X **shall** (1469) annunciate an ACAS failure if the paired transponder indicates that it is FAA TSO-C119A compatible.

Note: This two-way annunciation or handshake, performed at power-up, is meant to determine/confirm the interface protocol that is used for ACAS X-to-transponder and transponder-to-ACAS X data transfers. The terms 'RTCA/DO-185A,B compatible' and 'RTCA/DO-185A,B,ACAS X compatible' mean that the interface protocol makes use of the general-purpose TGD (TCAS to transponder) and XGD (transponder to TCAS) protocols (see Ref. L). With this protocol, ACAS X can load 56 bits into a specified transponder register and/or read 56 bits from a

specified transponder register. ACAS X can operate only with an associated transponder that uses this general-purpose interface protocol.

2.2.3.12.2.1.2 Determination of ACAS X-to-Transponder Write Compatibility

ACAS X **shall** (2192) be permitted to write into transponder registers 0x31 through 0x3F if and only if the Mode S Subnetwork Version Number (bits 17-23 of transponder register 0x10) indicates version 6 or later. [Ref: ICAO Doc 9871, Table A-2-16]

Notes:

1. *ACAS X is permitted to write into transponder registers 0x0F, 0x10, 0x30, 0xE5, and 0xE6. ACAS X can read from any transponder register.*
2. *Attempting to write into an unauthorized transponder register will cause some transponders to fail and consequently cause ACAS X to fail. ACAS X can operate properly without writing into registers 0x31 through 0x3F, although some future interoperability capability will be less than ideal; in that case ACAS X would be recognized by other aircraft as being TCAS-equipped, as opposed to being ACAS X-equipped.*

2.2.3.12.2.1.3 Determination of Transponder OCM Transmit Capability and Setting of CCCB Value

The requirements of this section apply if and only if ACAS X has determined per §2.2.3.12.2.1.2 that its associated Mode S transponder is version 6 or later.

Each cycle ACAS X **shall** (2571) provide to its associated Mode S transponder the following 7-bit CCCB value (§2.2.3.8.3.2.9.2): 00 001 00, indicating “vertical” sense and “Active CAS (not TCAS II).”

Notes:

1. *On ARINC platforms, the existing ARINC TX label 274 is an appropriate periodic message for this purpose.*
2. *ACAS X systems built to these MOPS do not have the capability to transmit OCMs. The capability to transmit OCMs may be incorporated into a future version of these MOPS and will be announced when all three of the following conditions are true:
 - a) The transponder “OCM Transmit Status” = “ON.” This indicates that both the transponder static OCM transmit capability is true, i.e., the transponder’s DO-181 and DO-260 MOPS versions support OCM transmission, and the transponder dynamic OCM transmit capability is true, i.e., the equipment is functioning properly and the pilot has not selected “ADS-B Out OFF.”
 - b) The ACAS X implementation meets or exceeds the (static) criteria for successfully associating a 24-bit address with a Mode C intruder.*

- c) ACAS X determines that its (dynamic) capability to receive incoming ADS-B Operational Status Messages is true, i.e., ACAS X will know to send an OCM to the intruder.

When these three conditions are satisfied, CCCB = “Active CAS (not TCAS II) with OCM transmit capability.”

Otherwise, CCCB = “Active CAS (not TCAS II).”

[The setting “Active CAS (not TCAS II)” makes no assertion with regard to the ability to send OCMs and the aircraft might or might not send OCMs in practice.]

3. *The transponder OCM Transmit Status, condition a above, is communicated each second/cycle from the transponder to ACAS X; on ARINC platforms, the existing ARINC XT label 276 is an appropriate periodic message for this purpose.*

4. *Condition b above is assumed false until such time as the criteria are determined.*

Regarding condition b: OCMs are used to communicate coordination information to threats equipped with passive CA systems (§2.2.3.9.3.1).

OCMs identify the threat via a 24-bit aircraft address. In order to obtain a 24-bit address for (for example) a threat that is Mode C- and UAT-equipped, ACAS X must be able to associate the active Mode C track with a 24-bit aircraft address, received directly via a UAT receiver or via ADS-R.

Otherwise, an OCM will not be composed or transmitted.

2.2.3.12.2.1.4 Data Reception

ACAS-X **shall** (1172) meet the timing requirements of §2.2.3.9.3.4 delivering every received TCAS Resolution Message to the TRM, regardless of processing load and message traffic on the ACAS-X/transponder interface. Incoming TCAS Resolution Messages **shall** (1173) be processed in the order in which they are received.

2.2.3.12.2.1.5 Data Integrity

The interface between the ACAS X equipment and the transponder **shall** (1175) be designed to provide communication in the normal operational aircraft environment for that class of ACAS X equipment while assuring error rates of less than one detected error in 10^3 messages and less than one undetected error in 10^7 messages for transfers in both directions. Compliance with this requirement **shall** (1176) be demonstrated either by direct test in a simulated operational environment or by analysis based on the known characteristics of proven interface techniques.

The ACAS X equipment **shall** (1177) provide a means to acknowledge all non-periodic messages sent to or received from the Mode S transponder. A failure in the acknowledgment procedure **shall** (1178) be recognized by ACAS X as a loss of integrity and **shall** (1179) be treated as an ACAS X failure.

Note: Incomplete data transmission can result in erroneous execution of the collision avoidance logic algorithms.

2.2.3.12.2.2 Data Provided by the ACAS X Equipment to the Mode S Transponder

2.2.3.12.2.2.1 Data Provided to Own Transponder for Use in Special Surveillance Replies (DF=0, 16)

a. Contents of the SL Field

The ACAS X equipment **shall** (1180) provide to the Mode S transponder an SL value as specified in §2.2.3.8.3.2.6. ACAS X **shall** (1181) report a changed value of sensitivity level to the transponder within 500 msec after the change occurs within ACAS X. The requirements for setting SL and RI are defined in §2.2.3.9.2.3.

b. Contents of the RI Field

The ACAS X equipment **shall** (1182) provide to the Mode S transponder an RI value as specified in §2.2.3.8.3.1.14. ACAS X **shall** (1183) report a changed resolution capability within 500 msec after the change occurs within ACAS X.

2.2.3.12.2.2.2 Data Provided to Own Transponder for Use in Altitude and Identity Surveillance and Comm-B Replies (DF=4, 5, 20, 21)

a. Contents of the DR Field

Whenever an RA exists, the ACAS X equipment **shall** (1184) indicate this fact to its associated Mode S transponder. This will cause the transponder to set the 'TCAS' bit in the DR field in appropriate replies.

Note: Per §2.2.3.12.2.2.4, ACAS X indicates the existence of an RA by means of the RA Report, sent by ACAS X to the transponder within 200 msec of the RA Report subfields being output by the TRM in the Ground_Msg data block.

2.2.3.12.2.2.3 Data Provided to Own Transponder for Use in Long Special Surveillance Replies (DF=16)

a. Coordination Reply Message

The ACAS X equipment **shall** (1186) provide to the Mode S transponder MV subfield values as specified in §2.2.3.8.3.2.5.

Note: MV subfield values transmitted by the transponder in a Coordination Reply Message reflect the most recent MV subfield values received from ACAS X.

2.2.3.12.2.2.4 Data Provided to Own Transponder for Use in Altitude and Identity Comm-B Replies (DF=20, 21)

a. RA Report

Each surveillance update interval, the ACAS X equipment **shall** (1187) provide to the Mode S transponder MB subfield values as specified in §2.2.3.8.3.2.3.1. ACAS X **shall** (2604) provide this information to the transponder within 200 msec of the RA Report subfields being output by the TRM in the Ground_Msg data block.

Notes:

1. §2.2.3.9.5.1 gives requirements for communication of RA Report information from ACAS X to its associated Mode S transponder.

2. Following the end of an RA, the transponder retains the RA Report for an additional 18 ± 1 seconds in order to allow read-out by a ground sensor's rotating antenna.
- b. Data Link Capability Report

Each surveillance update interval, the ACAS X equipment, when operating correctly, **shall** (1188) provide to the Mode S transponder ACAS X capability information as specified in §2.2.3.8.3.2.3.2. If the performance monitoring function determines that ACAS X is no longer able to operate or the ACAS X capability is turned off, ACAS X **shall** (1189) be capable of conveying this fact to the transponder no later than 1 second after such failure.

Note: Ref. B, §2.2.19.1.12.6.3 specifies that the transponder will recognize a change in ACAS X capability and will automatically initiate a revised basic data link capability report by Comm-B broadcast.

2.2.3.12.2.2.5 Data Provided to Own Transponder for Use in 1090ES Transmissions

- a. ADS-B TCAS RA Broadcast Message

Note: The ADS-B TCAS RA Broadcast Message is identical in content (Bits 9-56) to the RA Report (§2.2.3.12.2.2.4). No action is required by ACAS X beyond writing the contents of the RA Report into transponder register 0x30. The transponder takes bits 9-56 of register 0x30 and loads them into 1090ES TCAS RA Broadcast Message ME bits 9-56. The transponder fills in ME bits 1-8 with the Type Code and Subtype Code.

- b. ADS-B Aircraft Operational Status Message

Notes:

1. Every second, per §2.2.3.12.2.1.3, ACAS X provides the 7-bit CCCB value to the transponder for inclusion in the ADS-B Aircraft Operation Status Message.
2. The transponder will insert the CCCB into the proper locations for transmission in outgoing Aircraft Operational Status Messages.
3. There are other bits related to ACAS X that appear in the Aircraft Operational Status Message: CA Operational bit, CA RA Active bit, and Interrogation Active bit. These are filled in by own transponder based on information (RI field and RA Report) sent by ACAS X to the transponder. See the note at the end of §2.2.3.8.3.2.9.2 for explanation of these bits.

- c. ADS-B Operational Coordination Message

The requirements of this section apply if and only if the determination from §2.2.3.12.2.1.3 is “Active CAS (not TCAS II) with OCM transmit capability.”

The ACAS X equipment **shall** (2590) provide to the Mode S transponder ME subfield values as specified in §2.2.3.8.3.2.9.1 within 200 msec of the OCMs being updated in the TRM Output Report. In a given processing cycle, ACAS X **shall** (2572) store up to five OCMs in transponder registers 0x33 to 0x37, as appropriate. In the case of five or fewer concurrent OCMs, each OCM (i.e., each unique taa subfield) **shall** (2591) remain in its original register until no further OCMs are to be transmitted to this

aircraft. If there are more than five concurrent OCMs, and the five stored in the transponder are not the highest priority, then the lowest priority OCM(s) stored in the transponder **shall** (2592) be replaced by new higher priority OCM(s), until the five highest priority OCMs are stored in transponder registers 0x33 through 0x37.

Each cycle, as a part of the storing of OCMs into registers 0x33 through 0x37, ACAS X **shall** (2573) zero any registers 0x33 through 0x37 that no longer contain current information, as the transponder will continue to transmit an OCM for any non-zero register.

Notes:

1. *There is no provision for storage and/or transmission of OCMs for more than five intruders per cycle. §2.2.3.12.2.2.5.1 below gives requirements for OCM prioritization if there are more than five intruders requiring OCMs.*
2. *When an OCM is to be transmitted, the transponder takes bits 9-56 of the appropriate register and loads them into a 1090ES OCM ME bits 9-56. The transponder fills in ME bits 1-8 with the Type Code and Subtype Code. ACAS X needs only to write into registers 0x33 to 0x37 as appropriate, and the transponder will compose and transmit the OCM(s).*

2.2.3.12.2.2.5.1 Operational Coordination Message Priority

In multi-threat encounters, ACAS Xa coordinates with each threat individually, so many OCMs might need to be broadcast on the same cycle. When there is a limit on the number of OCMs that may be broadcast each second [§2.2.3.12.2.2.5], the OCMs to be broadcast **shall** (2574) be selected according to the following priorities, ordered highest first, which are determined by the nature of the RA against the particular threat to which the OCM is addressed:

- RA reversals for which an OCM has not been broadcast;
- Initial crossing RAs for which an OCM has not been broadcast;
- Other initial RAs for which an OCM has not been broadcast;
- RA reversals for which an OCM has not been broadcast twice;
- Initial crossing RAs for which an OCM that has not been broadcast twice;
- Other initial RAs for which an OCM has not been broadcast twice;
- RAs for which an OCM has not been broadcast for the longest period of time.

When the number of OCMs with equal priority exceeds the number with that priority that can be selected, those with the highest track display score [§2.2.5.6.1.1.2] **shall** (2575) be selected.

Notes:

1. *In a multi-threat encounter, the collision avoidance logic selects an RA for each threat independently of the other threats before reconciling these pair-wise RAs in a single global RA that is announced to the flight crew. In the above list of priorities, the term “RA” refers to the pair-wise RA generated against the threat addressed in the OCM.*

2. *The objective is to ensure that two attempts are made to coordinate every RA before the frequency of its OCM broadcasts is reduced to comply with the overall limit.*
3. *Reversals of RA intent are to be coordinated before other RAs. They are most urgent because the need to reverse RA intent indicates that the present, coordinated, RA is not working and the CAS on both aircraft need to reverse RA intent, i.e., cancel the previous VRC and send the opposite VRC.*
4. *Crossing RAs are given priority over other initial RAs because it is more urgent to secure compliance on both aircraft with crossing RAs than it is for non-crossing RAs, and because there is a greater risk that the CAS on the threat aircraft will select a non-complementary RA when own CAS chooses a crossing RA than when it chooses a non-crossing RA. Crossing and non-crossing reversal RAs are given equal priority.*
5. *For the purposes of these requirements, (2574 and 2575 above), an RA is considered an “initial RA” on the cycle during which the threat is first recognized and subsequently until the corresponding OCM has been broadcast twice.*
6. *Most changes in the RA do not result in a change in the OCM and do not merit increased priority, because the only information conveyed in an OCM is that the threat should not pass below ownship or that the threat should not pass above ownship.*

2.2.3.12.2.3.3 Data Received by the ACAS X Equipment from the Mode S Transponder

2.2.3.12.2.3.3.1 Data Received in Long Special Surveillance Interrogations (UF=16) from Other TCAS Aircraft via Own Mode S Transponder

- a. TCAS Resolution Message

ACAS X **shall** (1190) receive from the Mode S transponder values for MTB, CVC, VRC, CHC, HRC, HSB, VSB, and MID as specified in §2.2.3.8.3.2.4.1.

- b. TCAS Broadcast Interrogation Message

ACAS X **shall** (1191) receive from the Mode S transponder values for MID as specified in §2.2.3.8.3.2.4.2.

2.2.3.12.2.3.3.2 Data Received in Altitude and Identity Comm-A Interrogations (UF=20, 21) from Mode S Ground Stations via Own Mode S Transponder

The system **shall** (1192) accept the Sensitivity Level Command from the transponder.

Note: ACAS X is required both to accept SLC and to not modify sensitivity based on an SLC.

2.2.3.12.3 ACAS X Operating Mode Control

The equipment **shall** (1193) have an interface for accepting the operating mode control which includes the following states: Standby, TA-Only, or TA/RA mode.

2.2.3.12.4 Radio Altimeter Interface

The equipment **shall** (2113) have an input of height above ground from a radio altimeter, i.e., radio altitude. In addition, if the radio altimeter has its own integral failure monitor, e.g., as required by Ref. E, the outputs from such an integral failure monitor **shall** (1194) also be provided (§2.2.7.2.5.2.1). The minimum operational range for radio altimeters used with ACAS X must be [0 ft, 1,850 ft] AGL.

2.2.3.12.5 Flight Test Recording

The equipment **shall** (2114) have track file output of surveillance track data for use in evaluating the success or failure of test procedures specified in §2.4.

2.2.3.12.6 Position Source Interface

ACAS X **shall** (1196) receive ownship position source (e.g. GNSS) data from onboard sources. In an unsynchronized installation, time of receipt may be used to represent time of applicability.

Time of applicability of the ownship position **shall** (1197) be determined with an accuracy of 0 to 0.25 seconds. If time of receipt is used as time of applicability then time of receipt **shall** (1198) be determined within 0.050 seconds.

The equipment **shall** (1199) receive the Horizontal Position based on references to WGS 84 latitude/longitude from the ownship position sources.

The equipment **shall** (1200) receive the N/S and E/W velocities from the ownship position sources when available.

The equipment **shall** (1201) receive from the ownship position sources indication of Horizontal Position Uncertainty and Horizontal Position Integrity.

2.2.3.12.7 Heading Interface

ACAS X **shall** (1202) receive True Heading from an ownship source when available.

ACAS X uses ownship heading to improve the relative cross range velocity estimate of an intruder and to compute relative bearing for the display of ADS-B intruders.

The system **shall** (1203) establish time of applicability (receipt by ACAS X) for the true heading that is accurate to within 0.25 seconds.

Heading accuracy is not critical to the system as TCAS active bearing errors will dominate those of the heading source.

See §2.2.7.2.9 for associated monitoring requirements.

2.2.3.12.8 Air/Ground Interface

As an option, the system may receive the aircraft air/ground indicator.

2.2.3.12.9 ASA System Interface

When integrated with an ASA System, ACAS X **shall** (2581) send prioritized CAS tracks to the ASA System (or equivalent) that contain ALL of the following when available:

- a. A unique identifier that identifies the traffic for which data is being provided (CAS Track ID)
- b. Traffic 24-bit Address
- c. Traffic Bearing relative to ownship
- d. Traffic Horizontal Range relative to ownship
- e. Traffic Pressure Altitude
- f. Traffic TCAS Vertical Sense or Traffic Pressure Altitude Rate
- g. Traffic TCAS Alert Status (i.e., no threat, proximity traffic, TA, RA)

When the ACAS X system has more than 30 tracks, ACAS X **shall** (2142) send a minimum of the 30 highest priority tracks to the ASA System (or equivalent).

The system **shall** (2584) provide to a TCAS Only Traffic Display only TCAS active tracks, hybrid surveillance tracks and extended hybrid surveillance tracks. By definition, the AOTO option cannot be enabled with a TCAS Only Traffic Display. This excludes ADS-R Only traffic and ADS-B Only traffic that does not qualify for extended hybrid surveillance from being presented on a TCAS Only Traffic Display.

If an RTCA/DO-317 display is utilized then the limitation of the previous requirement does not apply.

2.2.3.12.9.1 Receive Self-Test Status

The ACAS X system **shall** (2198) receive indication that the flight crew has initiated a CAS self-test routine.

2.2.3.12.9.2 Send Temporal Test Pattern

During initiated self-test the ACAS X system **shall** (2150) send a temporal test pattern for RA display evaluation purposes to the display components.

2.2.3.12.9.3 Send Self-Test Results

The ACAS X system **shall** (2147) send the results of the initiated self-test routine to the display components.

Note: It is recommended that manufacturers also incorporate an extended self-test or other means to access unit and software part numbers needed for certain maintenance or regulatory activities.

2.2.3.12.9.4 ASA System Interface for ACAS Xo

ASA System interface with ACAS Xo is covered in §2.2.8.8.

2.2.4 Surveillance Requirements

The function that implements the requirements specified by this section of the MOPS is referred to as Front End Surveillance. This section uses the terms associated with tracking including “track”, “coasting”, “established track”, etc. These tracking terms do not refer to tracking performed by the STM (Surveillance and Tracking Module). Front End Surveillance may satisfy the requirements of this section by using the tracking of the STM or using independent surveillance trackers for these purposes:

- Maintaining history and estimated range, range rate, altitude, altitude rate, signal level, etc. so that the appropriate surveillance technique can be selected:
 - Normal – 1 Hz Active
 - Reduced – 0.2 Hz Active
 - Hybrid – with variable validation rates
 - Extended Hybrid - with no interrogations
- Provide tracking satisfying tracking reliability requirements.
- Support the input requirements for other non-ADD functionality such as Interference Limiting.

The ACAS X surveillance function **shall** (2236) be capable of reception of 1090ES ADS-B broadcasts in order to meet the passive surveillance requirements of §2.2.4 and interface requirements in §2.2.5.

2.2.4.1 Surveillance Update Rate

The nominal surveillance update rate for all aircraft that satisfy the requirements for generation of a TA or RA **shall** (1204) be 1 per second. That is, the nominal "surveillance update interval" is one second.

The total interrogation rate **shall** (1205) be controlled by the interference limiting procedures of §2.2.3.6.

2.2.4.2 System Delay

The ACAS X system **shall** (1206) display the correct RA within 1.5 seconds after receipt of the first reply used in updating a target's track that indicates the need for an RA. This reply can be an active reply used to update the track or a reply used to validate the ADS-B position.

2.2.4.3 Differential Channel Delay

The total difference in mean transmitter delay between the top and bottom antenna channels (including the ACAS X-to-antenna transmission lines) **shall** (1207) not exceed 0.05 microseconds.

The total difference in mean receiver delay between the top and bottom antenna channels (including the ACAS X-to-antenna transmission lines) **shall** (1208) not exceed 0.05 microseconds for equal amplitude replies received at signal levels between MTL + 3 dB and - 21 dBm.

2.2.4.4 Signal Reception

2.2.4.4.1 Receiver Sensitivity And Bandwidth

2.2.4.4.1.1 In-Band Acceptance

Given a valid transponder reply signal in the absence of interference or overloads, the minimum trigger level (MTL) is defined as the minimum input power level that results in a 90% ratio of decoded to received replies.

- a. The MTL for ATCRBS and Mode S signals over the frequency range of 1087 to 1093 MHz **shall** (1209) be $-74 \text{ dBm} \pm 2 \text{ dB}$.

The MTL for Mode S signals over the frequency range of 1089 to 1091 MHz **shall** (1210) be $-74 \text{ dBm} \pm 2 \text{ dB}$.

Notes:

1. *This provides adequate link margin for reliable detection of near-co-altitude aircraft in level flight at a range of 14 NM.*
2. *To accommodate a shared ACAS X/ADS-B receiver, a narrow band (1089 to 1091 MHz) receiver for Mode S signals is acceptable.*

- b. For an input signal power of level -78 dBm or less, no more than 10% of ATCRBS and Mode S signals **shall** (1211) be decoded.

- c. The decoding ratio **shall** (1212) be at least 99% for ATCRBS and Mode S signals between MTL +3 dB and -21 dBm.

If the antenna gain is not as specified in §2.2.4.7, the MTL **shall** (1213) be adjusted to account for the antenna gain.

Note: *For example, if the antenna gain were 3 dB higher than specified, the nominal MTL would be raised by 3 dB to -71 dBm .*

2.2.4.4.1.2 Out-of-Band Rejection

The selectivity of the receiver **shall** (1214) be such that an RF CW signal at the receiver input **shall** (1215) result in the receiver output levels relative to center frequency as a function of input signal frequency offset that are specified in Table 2-15.

Table 2-15: Out-of-Band Rejection

Input Signal Frequency Difference From 1090 MHz	Output Signal Level Relative to 1090 MHz
$\pm 5.5 \text{ MHz}$	$\leq -3 \text{ dB}$
$\pm 10 \text{ MHz}$	$\leq -20 \text{ dB}$
$\pm 15 \text{ MHz}$	$\leq -40 \text{ dB}$
$\pm 25 \text{ MHz}$	$\leq -60 \text{ dB}$

2.2.4.4.2 Reply Detection and Decoding

The following pulse decoder characteristics **shall** (1216) apply over the RF input signal level range from MTL to -21 dBm:

2.2.4.4.2.1 Mode C Reply Reception

All performance requirements **shall** (1217) be met for pulses having the following characteristics:

Pulse amplitude variation: up to $\pm 2 \text{ dB}$, relative to F_1 amplitude.

Pulse rise time: 0.05 to 0.1 microseconds.

Pulse decay time: 0.05 to 0.2 microseconds.

- a. Description of Mode C Received Signals. The Mode C received signal is illustrated in Figure 2-6. The Mode C received signal consists of a pair of framing pulses spaced 20.3 ± 0.1 microseconds apart. The code pulse positions begin 1.45 microseconds after the leading edge of the first framing pulse, and are spaced at 1.45-microsecond intervals thereafter. Each code pulse position has a tolerance of ± 0.1 microseconds relative to the first framing pulse and ± 0.15 microseconds relative to every other pulse within the reply. All pulses have a width of 0.45 ± 0.1 microseconds. A one or zero in the reply code is indicated by the presence or absence of a code pulse, respectively.

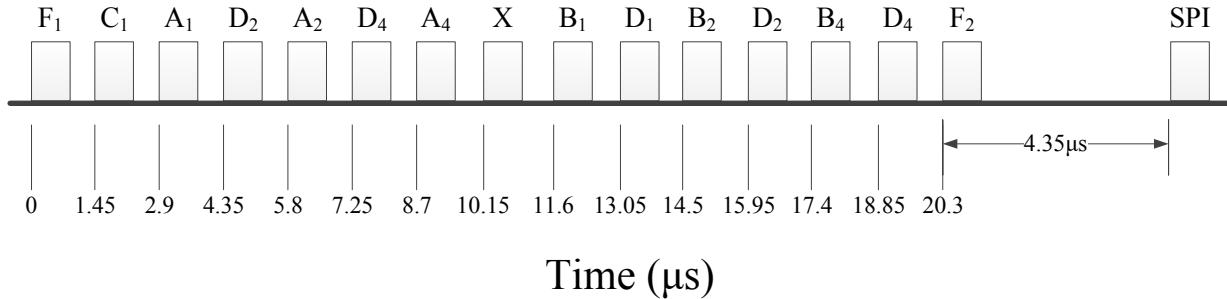


Figure 2-6: Mode C Reply

- b. Criteria for Mode C Pulse Detection. Mode C decoding **shall** (1218) be based on pulse leading edges. The occurrence of a leading edge **shall** (1219) be determined directly from the positive slope of a clear leading edge, or inferred from the pulse width and trailing edge positions associated with overlapping pulses. An actual leading edge is defined as an event for which: the signal rises at a rate exceeding 48 dB per microsecond to a level above the receiver threshold, AND 0.121 microseconds later the rate of rise is less than 48 dB per microsecond. An inferred leading edge is defined as an event in which a leading edge is assumed to exist in order to account for a pulse whose width implies the existence of overlapping pulses. The first qualifying criterion for reception of a Mode C signal **shall** (1220) be the occurrence of a pair of bracket pulse leading edges spaced 20.3 ± 0.121 microseconds apart. The pulses **shall** (1221) not be accepted as a bracket pair if their spacing deviates from 20.3 microseconds by 0.242 microseconds or more, OR if either of the bracket pulses occurs within 0.242 microseconds of a previous bracket pulse, OR if the width of either of the pulses is 0.242 microseconds or less. A Mode C code pulse **shall** (1222) be accepted if its leading edge occurs within 0.121 microseconds of a nominal code pulse position relative to the leading edge of the first bracket pulse. The code pulse **shall** (1223) be rejected if the time of occurrence of its leading edge deviates from a nominal code pulse position by 0.242 microseconds or more, OR if the width of the pulse is 0.242 microseconds or less.
- c. Criteria for Acceptance of Garbled Mode C Replies. The ACAS X equipment **shall** (1224) be capable of correctly determining arrival times and detecting reply brackets and reply codes for at least three valid overlapping Mode C replies whose code and bracket pulses are interleaved, i.e., they fall into the spaces between the other reply code pulses. The equipment **shall** (1225) also be capable of determining arrival times and detecting brackets and reply codes for valid Mode C replies with overlapping pulses. As a minimum, the probability of correctly decoding three valid overlapping replies **shall** (1226) be as specified below for the following conditions of reply pulse overlap:

Reply 1

Reply Code	= 6020
Received Amplitude	= -60 dBm
RF Frequency	= 1090 MHz
Probability of Correct Decode	= 60%*

Reply 2

Reply Code	= 4030
Received Amplitude	= -63 dBm
RF Frequency	= 1087 MHz
Range	= Reply 1 range plus 4.6 microseconds
Probability of Correct Decode	= 50%*

Reply 3

Reply Code	= 4420
Received Amplitude	= -57 dBm
RF Frequency	= 1093 MHz
Range	= Reply 1 range minus 21.3 microseconds to Reply 2 range plus 21.3 microseconds
Probability of Correct Decode	= 60%*

* - When averaged over all relative arrival times between reply 3 and replies 1 and 2.

Note: Attempting to detect overlapping reply pulses requires that the ACAS X equipment be capable of resolving pulses in situations where overlapped pulse edges are clearly distinguishable, and reconstructing the position of hidden pulses in situations where overlapping pulses of nearly the same amplitude cause the following pulses to be obscured. The ACAS X equipment should be designed to reliably decode overlapping replies with any number of code pulses. The reason for basing this specific quantitative requirement on replies with 3 code pulses is that a realizable decoding probability has been determined experimentally for the specific case of 3 code pulses.

- d. **Phantom Rejection.** The ACAS X equipment **shall** (1227) reject all replies whose brackets could possibly be pulses of preceding or following replies.

Note: It is recommended that the Mode C reply processing function be disabled for the duration of a long Mode S reply if a Mode S reply preamble is received during a Mode C listening period. This will prevent the data block of the asynchronous Mode S reply from generating a string of false Mode C fruit replies during the Mode C listening window.

2.2.4.4.2.2 Mode S Squitter and Reply Reception

All performance requirements **shall** (1228) be met for pulses having the following characteristics:

Pulse amplitude variation: up to ± 2 dB, relative to the amplitude of the first preamble pulse.

Pulse rise time: 0.05 to 0.1 microseconds

Pulse decay time: 0.05 to 0.2 microseconds

- a. **Description of Mode S Received Signals.** The Mode S received signal, illustrated in Figure 2-7, consists of a preamble and a data block. The preamble consists of four 0.5 ± 0.05 -microsecond pulses. The second, third, and fourth pulses are spaced 1, 3.5, and 4.5 microseconds respectively from the first transmitted pulse. The data block begins 8 microseconds after the first preamble pulse. Either 56 or 112 one-microsecond bit intervals are assigned to each data block. A pulse with a width of 0.5 ± 0.05 microseconds is transmitted in the first half of each interval to indicate a binary one and the second half of each interval to indicate a binary zero. If a pulse transmitted in the second half of one bit interval is followed by a pulse transmitted in the first half of the next bit interval, the two pulses are merged and a 1 ± 0.05 microseconds pulse is received.
- b. **Criteria for Preamble Acceptance.** The first qualifying criterion for reception of a Mode S signal **shall** (1229) be the detection of a Mode S preamble. A preamble **shall** (1230) be accepted if each of the four pulse positions of the preamble waveform contains a pulse that is above the receiver threshold for at least 75% of its nominal duration, AND the last three pulses are within ± 0.125 microseconds of their nominal positions relative to the first pulse, AND at least two of the four preamble pulses have actual leading edges (as defined in §2.2.4.4.2.1b.) that occur within ± 0.125 microseconds of their nominal edge positions and there are no earlier leading edges associated with those pulses.

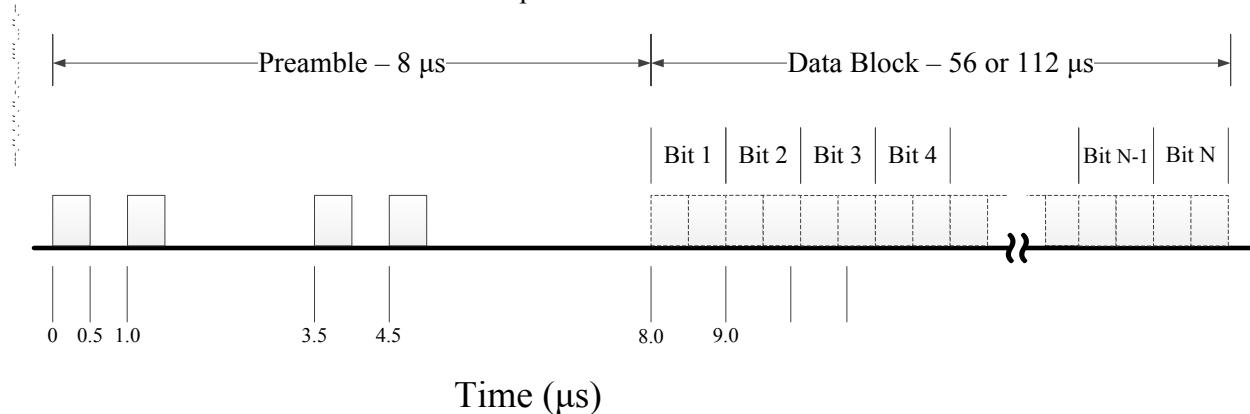


Figure 2-7: Mode S Reply

- c. **Criteria for Data Block Acceptance in Squitter and Asynchronous Transmissions.** The initial detection of the Mode S target address is accomplished by monitoring Mode S squitter transmissions, in the Mode S All-Call format. These Mode S squitters contain 56 data bits. In addition to monitoring squitters in the All-Call format, initial detection of the Mode S target address may also be accomplished by optionally monitoring Mode S squitter transmissions in the Mode S extended squitter format. These Mode S squitters contain 112 data bits. Each bit of the Mode S data block **shall** (1231) be decoded by comparing the received signal with a 0.5-microsecond delayed replica of

itself to determine the difference between the signal amplitudes at the centers of the two possible pulse positions for that bit.

In the All-Call or extended squitter format, the target address is protected by an independent parity field. The ACAS X equipment **shall** (1232) make use of the parity coding in the Mode S squitter to detect and correct squitter messages that are received in error. One means of implementing error correction is described in Appendix A, Section A.3. Except as specified in the following sentence, the ACAS X equipment **shall** (1233) accept only those Mode S squitters that contain the code 01011 in the first 5 bits of the data block and that contain the correct Mode S All-Call address. If the ACAS X equipment optionally chooses to monitor Mode S squitter transmissions in the Mode S extended squitter format, it **shall** (1234) also accept those Mode S squitters that contain the code 10001 in the first 5 bits of the data block and that contain the correct Mode S All-Call address. Detection of the Mode S altitude is accomplished by either actively interrogating to elicit a DF=0 reply or by passively monitoring Mode S asynchronous reply transmissions in the DF=0 or DF=4 formats. An asynchronous reply transmission **shall** (1235) be accepted as a valid Mode S altitude reply if: the first 5 bits of the data block contain either the code 00000 or the code 00100, AND the encoded altitude bits conform to either of the encoding standards specified in §2.2.3.8.3.1.2 AND, if the 26th bit and the 28th bit are ZERO, bits 20, 22 and 24 do not assume one of the illegal code combinations shown in Table 2-16.

Table 2-16: Illegal Altitude Codes

Bit No.	20 (C1)	22 (C2)	24 (C3)
0	0	0	0
1	0	0	1
1	1	1	1

AND no more than 34 data bits, of which no more than seven are consecutive, fail the following high confidence test:

Sample the received signal eight times during the one-microsecond bit interval to determine if the amplitude of the received signal is above or below the dynamic minimum triggering level of the receiver. The data bit **shall** (1236) be declared a high-confidence bit if, between the first and second of the two possible pulse positions for that bit, the difference in the number of samples for which the signal is above DMTL is at least three AND the sign of this difference agrees with the decoded value of the bit.

Note: These detection criteria for Mode S replies provide a means of pre-filtering asynchronous Mode S replies to minimize the time spent searching the track file for non-existent ICAO 24-bit Aircraft Addresses.

- d. Additional Criterion for Data Block Acceptance in Discrete Transmissions. Responses to Mode S interrogations include known addresses protected by an independent parity field. The ACAS X equipment **shall** (1237) make use of the parity field to correct discrete messages that are received in error. One means of implementing error correction is described in Appendix A, Section A.3. The ACAS X equipment **shall** (1238) confirm that the address of the Mode S reply is correct.

Note: If two or more acquisition replies requiring error correction are received within the Mode S range acquisition window, it may be impractical to apply

error correction to more than the first received reply. None of the acquisition replies need to be corrected when this occurs.

2.2.4.5 Interference Rejection and Control

ACAS X **shall** (1239) employ the necessary means of interference rejection and control in order to achieve the system performance requirements specified in §2.2.2 on targets-of-interest as defined in §2.2.1.3 when operating in the environment defined in §2.2.1.2.

The following subparagraphs define specific requirements associated with each of the interference conditions defined in §2.2.1.2.

2.2.4.5.1 Multipath Rejection

2.2.4.5.1.1 Interrogation Link Interference

The equipment **shall** (1240) employ means of preventing the multipath signals defined in §2.2.1.2.5.2 from causing Mode conversion or transponder suppression on the interrogation link.

The use of the whisper-shout interrogation sequence specified in §2.2.4.5.4.1, is one means of reducing the effect of uplink multipath to an acceptable level by causing transponders to be interrogated at power levels close to their MTL values. Since most signal-to-multipath ratios are high, most multipath signals will be reduced to levels below transponder MTL by use of the whisper-shout interrogation sequence.

2.2.4.5.1.2 Reply Link Interference

The equipment **shall** (1241) employ means of rejecting the low level multipath signals defined in §2.2.1.2.5.1. Specifically, the equipment **shall** (1242) be able to successfully detect and decode valid replies under the following simultaneous conditions of reply signal level and multipath signal level:

a. ATCRBS Reply Link

$$\begin{aligned}\text{Minimum reply signal level} &= \text{MTL} + 11 \text{ dB} \\ \text{Maximum multipath signal level} &= \text{Reply level} - 10 \text{ dB}\end{aligned}$$

b. Mode S Reply Link

$$\begin{aligned}\text{Minimum reply signal level} &= \text{MTL} + 8 \text{ dB} \\ \text{Maximum multipath signal level} &= \text{Reply level} - 7 \text{ dB}\end{aligned}$$

Section A.2 in Appendix A describes one means of rejecting low level multipath signals on the reply link.

2.2.4.5.1.2.1 Narrow Pulse Discrimination

If the means for rejecting low level multipath signals involves receiver desensitization, this means **shall** (1243) not be responsive to pulses that have a width of less than 0.3 microseconds.

2.2.4.5.1.2.2 TACAN and DME Discrimination

If the means for rejecting low level multipath signals involves receiver desensitization, this means **shall** (1244) not be responsive to pulses that have a rise time exceeding 0.5 microseconds.

2.2.4.5.2 Narrow Pulse Rejection

The ACAS X equipment **shall** (1245) reject any received ATCRBS bracket or code pulses that have a width of 0.242 microseconds or less.

2.2.4.5.3 TACAN and DME Signal Rejection

The ACAS X equipment **shall** (1246) reject any received pulse with a rise time exceeding 0.5 microseconds.

2.2.4.5.4 Control of ATCRBS Synchronous Garble

The equipment **shall** (1247) provide a means of controlling ATCRBS synchronous garble to a level that will enable ACAS X to achieve the system requirements of §2.2.2 on a target-of-interest as defined in §2.2.1.3 when operating in the environment defined in §2.2.1.2.

ATCRBS synchronous interference can be controlled by the use of a Mode C whisper-shout interrogation sequence and by the use of a directional transmitting antenna. The degree to which synchronous interference can be reduced depends on the ATCRBS transponder density, the resolution of the whisper-shout interrogation sequence and the azimuth directionality of the transmitting antenna. For a given antenna directionality and degarble performance, the required degree of resolution is directly proportional to the ATCRBS transponder density; i.e., the lower the ATCRBS density the less resolution is required to provide an equivalent reduction in the level of synchronous interference. ACAS X equipment **shall** (1248) employ at least a four-beam top-mounted directional interrogation antenna as specified in §2.2.4.5.4.2. In conjunction with a four-beam antenna, ACAS X **shall** (1249) use the Minimum Basic 6-level whisper-shout sequence (§2.2.4.5.4.1.1), the high resolution whisper-shout sequence (§2.2.4.5.4.1.2) or a single interrogation (§2.2.4.5.4.1.5) according to the selection criteria specified in §2.2.4.5.4.1.4 or §2.2.4.5.4.1.5. Alternative whisper-shout designs that are either more or less capable than those specified above may be implemented so long as the combination of the whisper-shout resolution and the azimuth directionality result in degarbling performance that equals or exceeds the minimum required to satisfy the system requirements of §2.2.2.

The following subparagraphs define the important parameters of a whisper-shout interrogation sequence and an antenna interrogation pattern that will enable a manufacturer to select an acceptable design for control of synchronous garble. The improvements in degarbling performance of various whisper-shout sequences and antenna beam widths are presented relative to the degarbling performance of the required minimum whisper-shout interrogation sequence specified in §2.2.4.5.4.1.1 and to the degarbling performance of an omnidirectional interrogation antenna, respectively.

2.2.4.5.4.1 Control of Synchronous Garble by Whisper-Shout

To control ATCRBS synchronous interference and also to reduce the severity of multipath effects on the interrogation link (§2.2.4.5.1.1), a sequence of interrogations at different power levels **shall** (1250) be transmitted during each surveillance update period. Each of the interrogations in the sequence, other than the one at lowest power, **shall** (1251) be preceded by a suppression pulse (designated S_1) 2 microseconds preceding the P_1 pulse.

The combination of S₁ and P₁ **shall** (1252) serve as a suppression transmission. S₁ **shall** (1253) be at a power level lower than that of P₁. The minimum time between successive interrogations **shall** (1254) be 1 millisecond. All interrogations in the sequence **shall** (1255) be transmitted within a single surveillance update interval.

Note: Because the suppression transmission in each step is always at a lower power level than the following interrogation, this technique is referred to as whisper-shout. The intended mechanism is that each aircraft replies to only one or two of the interrogations in a sequence. The lowest power interrogation is not preceded by an S₁ suppression pulse to ensure that each transponder will respond to at least one of the interrogations in the sequence. A typical population of ATCRBS transponders at any given range may have a large spread in effective sensitivity due to variations in receivers, cable losses, and antenna shielding. Typically, each transponder in the population will respond to two interrogations in the sequence, and will be turned off by the higher power suppression transmissions accompanying higher-power interrogations in the sequence. Given a situation in which several aircraft are near enough to each other in range for their replies to synchronously interfere, it is unlikely they would all reply to the same interrogation and, as a result, the severity of synchronous interference is reduced.

2.2.4.5.4.1.1 Minimum Basic Whisper-Shout Sequence

Figure 2-8 defines the minimum basic whisper-shout sequence that **shall** (1256) be used. Figure 2-9 illustrates the timing for the sequence in the forward beam.

This design is the shortest whisper-shout sequence that has been developed, tested and shown to be effective, when used with a four-beam top-mounted directional antenna, in providing an acceptable level of degarbling performance in a moderate ATCRBS transponder density of approximately 0.05 ATCRBS-equipped aircraft per sq. NM. Five distinct sub-sequences are defined for use in the four beams of the top-mounted antenna and for the bottom-mounted omnidirectional antenna in accordance with §2.2.4.5.4.1.4. The interrogations may be transmitted in any order. When interrogation power is reduced to limit interference (§2.2.3.6.2), each interrogation and its related receiver MTL, as indicated in the last column of Figure 2-8 **shall** (1257) be reduced by 1 dB in the order shown in the column labeled *Interference Limiting Priority* with the lowest number reduced first.

Note: Most of the interrogations are transmitted from the top antenna because it is less susceptible to multipath interference from the ground.

2.2.4.5.4.1.2 Higher Capability Whisper-Shout Sequences for Improved Degarbling Performance

The extent to which a whisper-shout interrogation sequence reduces synchronous garble depends on the resolution of its interrogation steps. The resolution of a whisper-shout sequence is best described in terms of "bin-width," which is the difference in dB between an interrogation and the associated suppression. For example, the bin-width of the minimum basic 6-level sequence illustrated in Figure 2-8 is 10 dB. Table 2-17 provides improvement factors for higher resolution whisper-shout sequences relative to the minimum basic 6-level sequence.

Table 2-17: Improvement Factors For Higher Resolution Whisper-Shout Sequences

Bin-Width (dB)	Relative Degrabbling Improvement Factor
10	1.0
9	1.1
8	1.2
7	1.4
6	1.5
5	1.8
4	2.0
3	2.5
2	3.3
1	4.4

As an example, a whisper-shout sequence with a bin-width of 2 dB will provide somewhat greater than 3 times the degarbling capability provided by the 6-level sequence.

STEP NUMBER	MINIMUM EFFECTIVE RADIATED INTERROGATION POWER LEVEL (dBm)	INTERFERENCE LIMITING PRIORITY	MTL (-dBm)
Top Antenna Forward Direction			
1	S.....I	52	
2	S.....I	48	
3	S.....I	44	
4	S.....I	40	
5	S.....I	36	
6I	32	
Top Antenna Left & Right Direction			
7,8	S.....I	48	
9,10	S.....I	44	
11,12	S.....I	40	
13,14	S.....I	36	
15,16I	32	
Top Antenna Aft Direction			
17	S.....I	43	
18	S.....I	39	
19	S.....I	35	
20I	31	
Bottom Omni			
21	S..I	34	
22	S..I	32	
23	S..I	30	
24I	28	
MINIMUM EFFECTIVE RADIATED POWER LEVEL (dBm)			
	22	32	42
			52

Note: "I" indicates ERP of P_1 , P_3 , and P_4 Interrogation Pulses.

"S" indicates ERP of S_1 Suppression Pulse.

"S.I" means that the S_1 ERP is 2 dB less than the interrogation ERP.

"S..I" means that the S_1 ERP is 3 dB less than the interrogation ERP.

"S.....I" means the S_1 ERP is 10 dB less than the interrogation ERP.

In the last steps of each quadrant no S_1 pulses are transmitted.

Figure 2-8: Minimum Basic Whisper-Shout Sequence

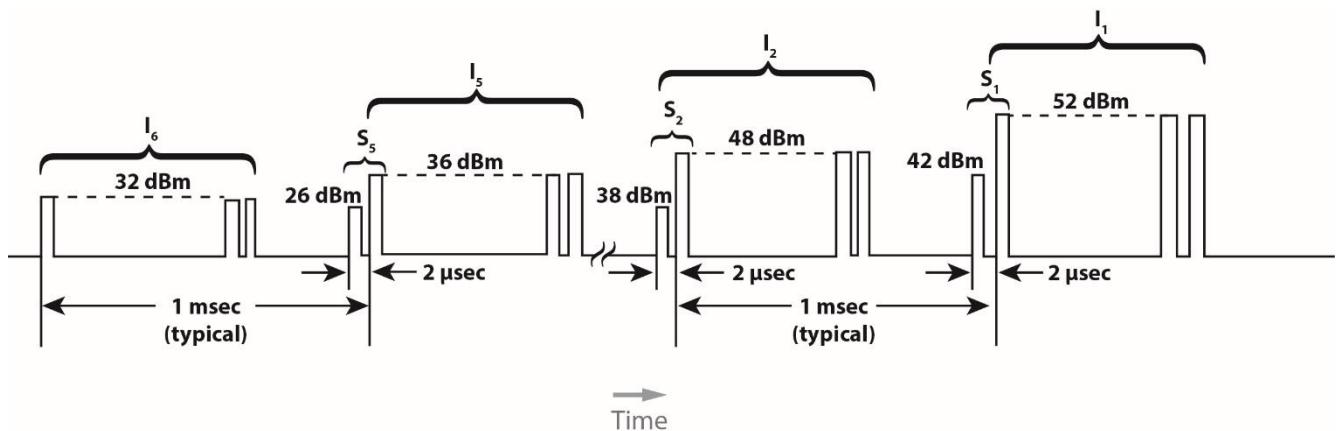


Figure 2-9: Timing For Six Power Steps In The Minimum Basic Whisper-Shout Sequence For Top Antenna Forward Beam

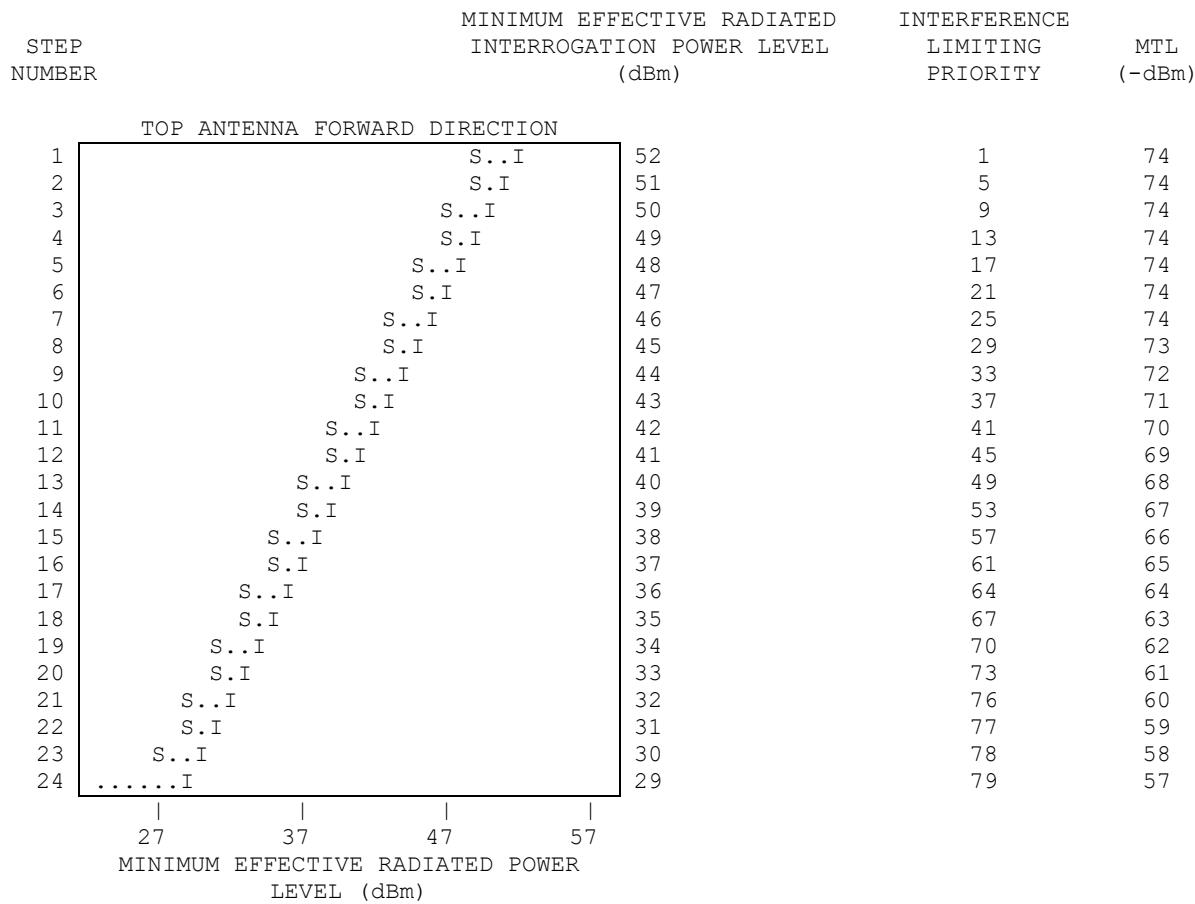
First Pulse Of Interrogation Serves As Second Pulse Of Suppression.

Figure 2-10 defines the high resolution whisper-shout sequence that **shall** (1258) be used in conjunction with a four-beam top-mounted directional antenna for high density ATCRBS surveillance. This sequence, when used with a top mounted four-beam directional antenna, has been verified to operate successfully in densities up to 0.3 aircraft per square NM. Five distinct sub sequences are defined for use in the four beams of the top-mounted antenna and the bottom-mounted omnidirectional antenna in accordance with §2.2.4.5.4.1.4. The interrogations may be transmitted in any order. When the sequence is truncated to limit interference (§2.2.3.6.2), the steps **shall** (1259) be dropped in the order shown in the column labeled *Interference Limiting Priority*. The lowest numbered steps **shall** (1260) be dropped first. The timing of individual pulses and steps in the sequence **shall** (1261) be as defined in Figure 2-9.

All whisper-shout sequence designs must satisfy the following requirements:

- a. The total extent of the sequence in the forward direction must span a dynamic range of at least 24 dB where dynamic range is defined as the product of the number of whisper-shout steps and step increment. For example, for a dynamic range of 24 dB, the 6-level whisper-shout sequence specified in §2.2.4.5.4.1.1 has a step increment of 4 dB.
- b. The nominal power level of each of the interrogation pulses, if arranged in a monotonic sequence, **shall** (1262) increment linearly throughout the entire power range. The transmission of the interrogation steps can actually occur in any order without affecting the degarbling performance of the whisper-shout sequence.
- c. The tolerance associated with each of the interrogation pulses **shall** (1263) be the smaller of ± 2 dB or 1/2 of the nominal step increment size of a monotonically arranged sequence. The tolerance associated with the nominal bin-width **shall** (1264) be the smaller of ± 2 dB or 1/4 of the nominal bin-width.
- d. The power level of the highest powered interrogation **shall** (1265) be such as to provide adequate coverage at the cross-over points of adjacent beams of a directional antenna. For a four-beam antenna the Effective Radiated Power of the highest powered interrogation **shall** (1266) be at least +52 dBm.
- e. The level of the suppression pulse relative to the preceding interrogation pulses **shall** (1267) be within ± 0.5 dB of the difference between the step increment value and the bin-width value.

- f. The MTL used in the reply listening period following each interrogation **shall** (1268) be related to the interrogation power in such a fashion as to maintain a balance between the uplink and downlink surveillance performance.
- g. The MTL values are based on the assumption that replies to all interrogations are received omnidirectionally. If a directional-receive antenna is used, the MTL values must be adjusted to account for the antenna gain. For example, for a net antenna gain of +3 dB, all MTL values would be raised by 3 dB relative to the values associated with an omnidirectional antenna.
- h. Although the steps in the sequence may be transmitted in any order, the steps **shall** (1269) be dropped in order of decreasing power level when the sequence is truncated as a result of interference limiting.



(Figure continues on next page)

STEP NUMBER		MINIMUM EFFECTIVE RADIATED INTERROGATION POWER LEVEL (dBm)	INTERFERENCE LIMITING PRIORITY	MTL (-dBm)
TOP ANTENNA LEFT&RIGHT DIRECTIONS				
25, 26	S..I	48	2, 3	74
27, 28	S.I	47	6, 7	74
29, 30	S..I	46	10, 11	74
31, 32	S.I	45	14, 15	73
33, 34	S..I	44	18, 19	72
35, 36	S.I	43	22, 23	71
37, 38	S..I	42	26, 27	70
39, 40	S.I	41	30, 31	69
41, 42	S..I	40	34, 35	68
43, 44	S.I	39	38, 39	67
45, 46	S..I	38	42, 43	66
47, 48	S.I	37	46, 47	65
49, 50	S..I	36	50, 51	64
51, 52	S.I	35	54, 55	63
53, 54	S..I	34	58, 59	62
55, 56	S.I	33	62, 63	61
57, 58	S..I	32	65, 66	60
59, 60	S.I	31	68, 69	59
61, 62	S..I	30	71, 72	58
63, 64I	29	74, 75	57
TOP ANTENNA AFT DIRECTION				
65	S.I	43	4	71
66	S..I	42	8	70
67	S.I	41	12	69
68	S..I	40	16	68
69	S.I	39	20	67
70	S..I	38	24	66
71	S.I	37	28	65
72	S..I	36	32	64
73	S.I	35	36	63
74	S..I	34	40	62
75	S.I	33	44	61
76	S..I	32	48	60
77	S.I	31	52	59
78	S..I	30	56	58
79I	29	60	57
BOTTOM OMNI				
80	S..I	34	80	62
81	S..I	32	81	60
82	S..I	30	82	58
83I	28	83	56
MINIMUM EFFECTIVE RADIATED POWER LEVEL (dBm)				
	27	37	47	57

Note: "I" indicates ERP of P_1 , P_3 , and P_4 Interrogation Pulses.

"S" indicates ERP of S_1 Suppression Pulse.

"S.I" means that the S_1 ERP is 2 dB less than the interrogation ERP.

"S..I" means that the S_1 ERP is 3 dB less than the interrogation ERP.

In the last steps of each quadrant no S_1 pulses are transmitted.

Figure 2-10: High Resolution Whisper-Shout Sequence

2.2.4.5.4.1.3 Determination of Whisper-Shout Based on Synchronous Garble

Each scan, ACAS X **shall** (1270) determine the appropriate whisper-shout sequence (i.e., the high resolution sequence or the minimum basic sequence) to be used with each top antenna directional beam according to criteria based on the expected level of synchronous garble within the reliable ACAS X surveillance range of that beam.

The reliable ACAS X surveillance range in the absence of interference limiting is considered to be nominally 14 NM in the forward beam, 8.8 NM in the left and right beams and 5.0 NM in the aft beam (§2.2.2.1.1). When interference limiting is imposed, the reliable surveillance range used in each beam to determine the relative range of Mode C aircraft and Mode C aircraft density (see below) is reduced by a factor equivalent to the square root of the interference limiting power reduction for that beam. For example, if the largest interrogation in the forward beam is reduced 6 dB because of interference limiting, the reliable surveillance range is reduced from 14 NM to 7 NM. The reliable surveillance range used to determine the level of low confidence bits in each received Mode C reply is always 14 NM in the forward beam, 8.8 NM in the side beams and 5.0 NM in the aft beam regardless of the level of interference limiting.

The following information on Mode C aircraft within each antenna beam and within the reliable ACAS X surveillance range defined above, **shall** (1271) be determined each scan in order to select the proper whisper-shout sequence for that beam:

- a. The level of low confidence bits in each received Mode C reply.
- b. The relative range of Mode C aircraft.
- c. The Mode C aircraft density.

The whisper-shout sequence for a specific top directional beam is determined and selected solely on the garble situation in that beam and is independent of the situation or use in the other beams. In other words, beams can simultaneously have different levels of degarbling capability. When interference limiting calls for the elimination of a whisper-shout step or a 1-dB reduction in the interrogation power level, the order of priority in a mixed-beam-interrogation-sequence follows the order specified in Figure 2-8 and Figure 2-10 for the specific interrogation sequence and beam. For example, regardless of sequence used, the forward beam priority always follows 1, 5, 9, 13, 17, 21 etc., the side beam priorities always follow 2&3, 6&7, 10&11, 14&15 etc., the rear beam priority always follows 4, 8, 12, 16 etc., and the bottom always follows 80, 81, 82, and 83.

- d. Low Confidence Bit Measurement (Cl)

Each scan, ACAS X **shall** (1272) examine successive Mode C replies from an intruder for the existence of low-confidence altitude code bits in order to provide an early detection of synchronous garble, particularly during Mode C intruder acquisition, and to indicate whether a switch to an appropriate whisper-shout interrogation sequence is necessary. The existence of low-confidence code bits in an intruder reply must persist for two or more successive scans to qualify for synchronous garble and to eliminate fruit interference.

If two successive replies from an intruder contain at least one low-confidence altitude code bit, regardless of its position in each reply, a flag **shall** (1273) be set for that beam to indicate synchronous garble. This information is then used in §2.2.4.5.4.1.4 to select the appropriate interrogation sequence for that beam.

Note: During reply decoding, the confidence flag for a reply pulse position is set LOW whenever there exists another received reply (either real or phantom)

that could have had a pulse within $\pm 0.121 \mu s$ of the same position. Otherwise, the confidence flag is set HIGH.

Appendix A, Section A.4 describes one acceptable algorithm for the low-confidence bit measurement of synchronous garble.

e. Intruder Range Track Comparison (T_C)

Each scan, ACAS X **shall** (1274) compare the tracked range of Mode C intruders within the reliable surveillance range of each directional beam to determine whether synchronous garble in that beam is possible, and if so, to indicate whether a switch to a less capable whisper-shout interrogation sequence is acceptable.

If two Mode C intruders within the same beam are determined to be separated in range by less than 1.7 NM, a flag **shall** (1275) be set for that beam to indicate synchronous garble. This information is then used in §2.2.4.5.4.1.4 to select the appropriate interrogation sequence for that beam.

Appendix A, Section A.4 describes one acceptable algorithm for the intruder track comparison.

f. Mode C Aircraft Density Measurement (ρ)

Each scan, ACAS X **shall** (1276) determine the density of ATCRBS aircraft within the reliable surveillance range of each directional interrogation beam. The density (ρ) for N_{ATCRBS} ATCRBS aircraft within the reliable surveillance range of a beam is:

$$\rho = \frac{N_{ATCRBS}}{r_N}$$

where r_N is the range to the most distant (N^{th}) aircraft within the reliable surveillance range.

The ATCRBS density (ρ) is then compared to the threshold shown below to determine which whisper-shout sequence is acceptable for a specific directional beam. This information is then used in §2.2.4.5.4.1.4 to select the appropriate interrogation sequence for that beam.

A switch is acceptable from High Resolution W-S to Minimum Basic W-S when $\rho \leq 6/14$ per NM.

2.2.4.5.4.1.4 Criteria for Selection of a Specific Whisper-Shout Sequence

Each scan, ACAS X **shall** (1277) use the results of the measurements described above to provide an indication of ATCRBS synchronous garble activity or potential garble activity within each directional beam. This information **shall** (1278) be used to select the appropriate whisper-shout sequence to meet the surveillance performance requirements referenced in §2.2.

The switching criteria must be able to quickly recognize the current and anticipated ATCRBS aircraft density and distribution and select for use the next scan the most capable degarbling interrogation sequence for that situation. Specifically, each scan ACAS X **shall** (1279) select a whisper-shout sequence for a beam according to the following criteria:

- a. Select the high resolution whisper-shout sequence if the (C_1) flag is set.
- b. Select the Minimum Basic whisper-shout sequence if:
 - 1. the (C_1) flag is not set AND no more than one ATCRBS aircraft exists within the reliable surveillance range, OR

2. the (C_p) flag is not set AND the (T_C) flag is not set AND (ρ) indicates a switch to the next lower level is acceptable.

Exceptions to the above rules for whisper-shout selection occur under the following circumstances:

- a. When ACAS X is powered on, the high resolution whisper-shout sequence **shall** (1280) be used for all top directional beams and the bottom omni for the first 60 seconds to allow track acquisition and stabilization.
- b. When a TA is issued on a Mode C intruder, the high resolution whisper-shout sequence **shall** (1281) be used in the beam providing surveillance on the threat for the duration of the advisory.
- c. When a RA is issued on a Mode C intruder, the high resolution whisper-shout sequence **shall** (1282) be used in all beams for the duration of the advisory.

2.2.4.5.4.1.5 Surveillance in Areas of No ATCRBS Intruder Aircraft

When there are no ATCRBS aircraft within the ACAS X surveillance region of a directional or omnidirectional beam, it is not necessary to transmit a whisper-shout interrogation sequence in that beam for degarbling purposes. However, it is still necessary to transmit a single interrogation in a beam in order to detect the presence of an ATCRBS-equipped aircraft and to initiate whisper-shout interrogations for Mode C acquisition and tracking.

ACAS X **shall** (1283) transmit a single Mode C interrogation from a top-mounted directional beam or from the bottom omnidirectional beam only if an established or candidate Mode C track does not exist within the full ACAS X surveillance range of that beam. The power level of the single interrogation in a specific beam and the corresponding MTL **shall** (1284) be equivalent to the highest whisper-shout level transmitted in that beam if using the high resolution sequence defined in Figure 2-10. When interrogation power is reduced to limit interference (§2.2.3.6.2), each single interrogation and its related receiver MTL **shall** (1285) be reduced by 1 dB in the order shown for the beams in the column labeled *Interference Limiting Priority* in Figure 2-8 with the lowest number reduced first. The timing of individual pulses in each interrogation is as defined in Figure 2-1 except that a whisper-shout suppression pulse is not transmitted.

ACAS X **shall** (1286) switch from a single interrogation within a beam to the Minimum Basic whisper-shout sequence in that beam whenever:

- a. two successive Mode C replies correlate in range such that the second Mode C reply occurs within a 5000 ft range correlation window centered either at the measured range of the previous Mode C reply or at a range offset by $\pm 13 \mu\text{sec}$ from the measured range of the previous reply, or
- b. an established Mode C track or a Mode C track in the process of being acquired traverses into that beam from another beam.

Mode C track initiation (§2.2.4.6.4.1.2) **shall** (1287) only be accomplished through use of a whisper-shout interrogation sequence. ACAS X **shall** (1288) switch back to the single interrogation after 10 surveillance update intervals in which no established Mode C track or Mode C track in the process of being acquired was present within the beam and no two successive correlating replies, indicating the potential for initiation of a Mode C track, were received within the beam.

Note: Single Mode C interrogations are susceptible to uplink mode conversion due to multipath and may result in a mixture of Mode A and Mode C replies from an

intruder that are separated by 13 μ sec. A switch from a single Mode C interrogation to a whisper-shout sequence is based on a successful range correlation between two consecutive replies in order to reduce the possibility of a switch due to a fruit reply. Mode C tracks are established only through use of whisper-shout interrogations in order to prevent multipath induced mode conversion.

2.2.4.5.4.2 Control of Synchronous Garble by Directional Interrogation

A minimum four-beam directional interrogation antenna mounted on the top of the aircraft is required for reliable ACAS X surveillance of ATCRBS targets in aircraft densities up to 0.3 aircraft per square NM. The required azimuth resolution of the directional antenna for reply degarbling depends on the specific whisper-shout sequence used. A high-resolution whisper-shout sequence will reduce the need for high resolution in azimuth and vice-versa. The following discussion of antenna requirements and definition of antenna characteristics will enable the manufacturer to determine the degarbling benefits to be gained as a function of azimuth resolution.

2.2.4.5.4.2.1 Directional Interrogation Beamwidth Control

In order to reduce synchronous garble, the interrogation beamwidth **shall** (1289) be limited by transmission of a P₂ sidelobe suppression pulse following each P₁ interrogation pulse by 2 microseconds. The P₂ pulse **shall** (1290) be transmitted on a separate control pattern and **shall** (1291) have the same shape as the other pulses specified in §2.2.3.7. The azimuth and elevation patterns of all beams **shall** (1292) be as specified in §2.2.4.7.

Interrogation beamwidth limiting, using the P₂ suppression pulse, **shall** (1293) be controlled to prevent transponder reply gaps between adjacent directional beams. The minimum required reply beamwidth is defined as the maximum amount of beamwidth limiting that still causes a maximum-suppression transponder located at any azimuth angle and between +20 degrees and -15 degrees elevation to reply to interrogations from at least one of the directional beams. A maximum-suppression transponder is defined as one that, when interrogated by a whisper-shout interrogation sequence, replies only when the received ratio of P₁ to P₂ exceeds the whisper-shout bin-width as defined in §2.2.4.5.4.1.2. When the relative P₂ power level is adjusted to satisfy the above conditions, the required reply beamwidth is optimally minimized resulting in a maximum amount of degarbling.

2.2.4.5.4.2.2 Directional Interrogation Radiated Power

The ACAS X effective radiated power (ERP) level in any azimuth direction **shall** (1294) be sufficient to satisfy the system requirements of §2.2.2. Effective radiated power is defined as the product of the net power delivered to the antenna terminal times the gain of the antenna at 0 degrees elevation and at the azimuth of interest, where antenna gain is specified relative to a matched quarter-wave stub.

For a directional interrogation top antenna, the ERP at the azimuth peak of each beam **shall** (1295) be selected to maintain adequate range coverage, as defined by the product of the appropriate closing speed and time to closest-point-of-approach, at the crossover point of two adjacent beams.

For a four-beam directional top antenna with a 90 degree azimuth beamwidth, the ERP associated with the largest whisper-shout step in the forward beam **shall** (1296) be at least +52 dBm in order to provide the necessary coverage at the crossover points of the two adjacent beams. The ERP associated with the largest whisper-shout step in each of the side beams **shall** (1297) be -4 dB relative to the forward beam and the ERP associated with the

largest whisper-shout step in the rear beam **shall** (1298) be -9 dB relative to the forward beam in order to account for the lower closing speeds in those directions.

In terms of effective radiated power level, the largest interrogation transmitted from the bottom omni-directional antenna **shall** (1299) be no larger than -18 dB relative to the largest whisper-shout step in the forward beam of the top antenna in order to minimize multipath interference and to reduce the impact of interference limiting.

2.2.4.5.4.2.3 Degarbling Performance as a Function of Azimuth Resolution

The degarbling improvement factor associated with directional interrogations can be estimated by determining the "average beamwidth" which is defined as that azimuth beamwidth over which an average transponder will reply to an interrogation. Since the minimum and maximum reply beamwidths are bounded by $P1/P2 = \text{bin-width}$ and 0 dB, the average transponder can be considered to reply when its received P1 to P2 ratio equals one-half of the bin-width. Although the average directional beamwidth is a good indicator of the quantity of valid replies received, it does not predict the total quantity actually observed through experience. Use of whisper-shout means that transponders are being interrogated near their threshold level. This fact occasionally causes the P2 pulse and P4 pulse of an interrogation that occurs outside of the main beam to trigger the transponder into replying with a delayed Mode C reply. These unwanted replies, which account for 20% of the valid replies, add to the synchronous garble problem and must be accounted for. An "effective beamwidth" is defined which is 1.2 times the average beamwidth described above. The degarbling improvement factor associated with a specific directional antenna system relative to an omni antenna is then:

$$\text{Improvement Factor} = \frac{360 \text{ Degrees}}{\text{Effective Beamwidth}}$$

2.2.4.5.5 ATCRBS and Mode S Fruit Rejection

The equipment **shall** (1300) provide means for rejection of ATCRBS and Mode S replies received in response to interrogations by other ACAS X interrogators and by ground-based beacon interrogators.

Effective methods of rejecting ATCRBS and Mode S fruit are represented by the surveillance requirements in §2.2.4.6.4 and the associated implementations suggested in Appendix A, Sections A.5 through A.8.

2.2.4.6 Surveillance Tracking Requirements

In order to support the system performance specified in §2.2.2, ACAS X **shall** (1301) provide the following surveillance functions in order to generate position reports on Mode C and Mode S targets that are:

- a. Closing at speeds of up to 1200 kt and at relative altitude rates of up to 10,000 ft/min.
- b. Within 14 NM of ACAS X.
- c. Within ± 3000 ft and whenever possible within $\pm 10,000$ ft altitude relative to ACAS X.

The maximum closing rate capability may be increased beyond 1200 kt and/or the surveillance range capability may be increased beyond 14 NM provided that the number or power of interrogations are not currently reduced by the interference limiting algorithms of §2.2.3.6. When the algorithms of §2.2.3.6 indicate that the number or power of

interrogations must be reduced, the number or power of interrogations to aircraft closing at rates in excess of 1200 kt and/or to aircraft beyond 14 NM must be reduced first.

2.2.4.6.1 Surveillance Target Track Capacity

The ACAS X tracking function **shall** (1302) have the capacity for active surveillance of at least 30 aircraft which can include any mix of Mode C- and Mode S-equipped targets, which might also be ADS-B-equipped.

Note: A density of 0.06 aircraft per sq. NM in higher altitude airspace will result in 26 targets within the required surveillance range of 14 NM. In lower altitude airspace, a density of 0.3 aircraft per sq. NM will result in 24 targets within the required surveillance range of 5 NM. A peak capacity of 30 targets will allow for sufficient margin.

*These target numbers (24 and 26) are derived using methodology in §2.2.1.2.1, Aircraft Density, which states: "For purposes of this characterization, traffic density is uniform in area within R_o and falls off linearly with increasing range beyond R_o . Specifically, the traffic count within any circle of radius R , where R is greater than R_o , is given by $N(R)=N(R_o)*R/R_o$." Therefore, in higher airspace, $N(5) = \pi 25 * .3 = 24$. In lower airspace, $N(14) = N(10) * 14/10 = \pi 100 * .06 * 14/10 = 26$.*

2.2.4.6.1.1 Surveillance Overload

If the number of targets under active surveillance exceeds the surveillance target capacity specified in §2.2.4.6.1, the excess targets **shall** (1303) be deleted in order of decreasing range without regard to target type.

ACAS X **shall** (1396) not interrogate an aircraft for which DF=17 extended squitters are being received as long as that aircraft qualifies for Extended Hybrid Surveillance.

Notes:

1. *Passive data received from an aircraft that qualifies for Extended Hybrid Surveillance is used to determine whether that track should be placed in the track file. If the track capacity is exhausted ACAS X will delete an existing track in order to add a new track in accordance with the other requirements in this section.*
2. *This requirement can be met if the maximum number of Extended Hybrid Surveillance tracks that are retained at any time is one less than the maximum that the system can accommodate.*

This avoids the need for an active interrogation on the sole grounds that it is not possible to use Extended Hybrid Surveillance for one more intruder.

2.2.4.6.2 Intruder Air/Ground Status Determination

This section defines how an intruder's air/ground status is determined for Mode C and Mode S surveillance. Determination of a Mode C transponder-equipped intruder requires an estimate of own ground level.

2.2.4.6.2.1 Estimate of Ground Level

ACAS X **shall** (1304) use radio altitude data when available, along with barometric altitude data to estimate ground level in order to reduce interrogations to and prevent advisories against aircraft that are on the ground.

If radio altitude data is available, an estimate of ground level **shall** (1305) be determined each scan as follows:

$$\text{ground level} = \text{own tracked barometric altitude} - \text{own tracked radio altitude}$$

Radio altitude data is used for this purpose only if:

- a. The data is credible (§2.2.7.2.5.2.2), and
- b. The data is less than 10 seconds old, and
- c. The radio altitude value indicates less than 1700 ft.

Notes:

1. *Own barometric altitude is received from the ACAS X own altitude tracker (§2.2.5.5.1)*
2. *In order to prevent erratic radio altitude input from causing undue jitter in the availability of the ground level estimate, at least ±50 ft of hysteresis should be applied in making a determination as to declaring the radio altitude value above or below 1700 ft.*

If the criteria for establishing a ground level is no longer satisfied, ACAS X **shall** (1306) declare a ground level estimate to be unavailable.

2.2.4.6.2.2 Determination of Intruder Air-Ground Status

2.2.4.6.2.2.1 Determination for Newly Initiated Tracks

All newly initiated tracks **shall** (1307) be placed in the ON-GROUND state until ACAS X determines their true Air-Ground condition. ACAS X **shall** (1308) determine the true Air-Ground state for a newly initiated track at the time the track is declared established as in §2.2.4.6.4.1.2 and §2.2.4.6.4.2.2.

2.2.4.6.2.2.2 Determination for Intruders Equipped with Mode S Transponders

A Mode S-equipped intruder **shall** (1309) be declared ON-GROUND by ACAS X if its CA field, FS field, or VS field indicates on the ground. Otherwise, a Mode S-equipped intruder **shall** (1310) be declared IN-AIR by ACAS X.

Notes:

1. *On rare occasions, an airborne Mode S intruder may incorrectly report on-the-ground status in its VS, FS, and/or CA fields due to a failure in the source of the air-ground discrete. In such cases, if the Mode S intruder is also equipped with an Active CAS that is RTCA/DO-185A,B compatible or ACAS X, the equipped intruder can remain fully operational since it uses radio altitude to determine its own air-ground status instead of the air-ground discrete used by the transponder.*
2. *As a strongly recommended option, ACAS X may do additional monitoring to determine if an Active CAS- equipped intruder reporting on-the-ground status in its VS, FS, and/or CA fields is, in fact, likely to be airborne. ACAS X may monitor*

Mode S DF=0 replies from the intruder. If the intruder's RI field indicates RA capability, then the intruder can be assumed to be airborne.

3. *ACAS X may additionally determine the ON-GROUND status of an intruder through the reception of DF=17 transmissions that are decoded to be Surface Position Messages. ACAS X is not intended to track and display other aircraft that are on-the-ground. Therefore Surface Position Messages may only be used to determine on-ground status and to assist in the calculation of NTA3/NTA6 as described in §2.2.4.6.4.2.4.*

2.2.4.6.2.2.3 Determination for Intruders Equipped with Mode C (ATCRBS) Transponders

2.2.4.6.2.2.3.1 Determination for Intruders Equipped with Mode C (ATCRBS) Transponders and Reporting Pressure Altitude

For intruders equipped with such Mode C transponders:

An intruder that is in the ON-GROUND state **shall** (1311) transition to the IN-AIR state if any of the following occur:

- a1. The intruder's tracked barometric altitude is ≥ 400 ft above the ground level estimate determined in §2.2.4.6.2.1.
- b1. The intruder's tracked barometric altitude is ≥ 250 ft above the ground level estimate determined in §2.2.4.6.2.1 AND there has been at least one positive altitude transition that has persisted for three consecutive reports or longer AND the intruder's reported barometric altitude is greater than 100 ft above its lowest reported barometric altitude.
- c1. There is no ground level estimate.

Otherwise, the intruder **shall** (1312) remain in the ON-GROUND state.

An intruder that is in the IN-AIR state **shall** (1313) transition to the ON-GROUND state if either of the following occurs:

- a2. Ownership is in the IN-AIR state AND the intruder's tracked barometric altitude has been less than 360 ft above the ground level estimate determined in §2.2.4.6.2.1 for 20 seconds AND its reported barometric altitude has not increased by more than 100 ft from its lowest reported barometric altitude for the same 20-second period.
- b2. The intruder's tracked barometric altitude is less than 150 ft above the ground level estimate determined in §2.2.4.6.2.1.

Otherwise the intruder **shall** (1314) remain in the IN-AIR state.

2.2.4.6.2.2.3.2 Determination for Intruders Equipped with Mode C (ATCRBS) Transponders and NOT Reporting Pressure Altitude

A Mode C intruder not reporting altitude **shall** (1315) be in the IN-AIR state.

2.2.4.6.2.2.4 Summary of Rules for Determining Air-Ground Status

The following summarize the rules for determining Air-Ground status.

A Mode C-equipped intruder's status is determined in accordance with Table 2-18.

Table 2-18: Summary of Rules for Determining Air-Ground Status

Current Intruder Status	Own Status	New Intruder Status
G	•	a1 OR b1 OR c1 = A NOT(a1 OR b1 OR c1) = G
A	A	a2 OR b2 = G NOT(a2 OR b2) = A
A	G	b2 = G NOT(b2) = A
A (No Pressure Altitude Reported)	•	A

Note: G=ON-GROUND; A=IN-AIR; a1, b1, c1, a2 and b2 as specified in §2.2.4.6.2.2.3.1

A newly initiated track's status is determined in accordance with the requirements of §2.2.4.6.2.2.1.

A Mode S transponder-equipped intruder's status is determined in accordance with §2.2.4.6.2.2.2.

2.2.4.6.3

Determination of Ownship Air-Ground Status

This section specifies when ownship is operating in one of two conditions:

1. Taking Off/Airborne
2. Operating on the surface

Note: The operating on the surface state or condition is used in determining when the exclusive use of 1090 MHz ADS-B reports is permitted. The taking off/airborne condition is used to detect when ownship is either airborne or about to be airborne so that the exclusive use of ADS-B report is terminated.

Ownship **shall** (1319) be considered to be taking off/airborne when any of the following are true:

- Ground speed is invalid
- Ground speed input is valid AND is ≥ 35 knots
- ACAS X own.on_ground variable indicates in air

Once ownship has satisfied the tests above to be considered to be taking off/airborne, it **shall** (1320) continue to be considered to be taking off/airborne until the tests for considering it be operating on the surface are satisfied.

The system **shall** (1321) be designed to use a ground speed that remains valid even when ownship is stationary.

Note: The above requirement is to prevent a system from deriving ground speed from the GNSS receiver reported North South/ East West velocities which do become invalid when ownship is stationary or moving slowly.

Ownship **shall** (1322) be considered to be operating on the surface when all these conditions are true:

- Ground speed input is valid AND is < 25 knots
- ACAS X own.on_ground variable indicates on-ground

At power up, ownship **shall** (1323) be assumed to be taking off/airborne until the operating on surface state is true.

Note: own.on_ground is the ACAS X STM variable in the Julia code (see Volume II) which defines the ownship air/ground state for ownship.

2.2.4.6.4 Range and Altitude Estimation

2.2.4.6.4.1 Mode C Targets

2.2.4.6.4.1.1 Reply Merging

The ACAS X equipment **shall** (1324) merge into one surveillance target report all replies that correlate with each other in their range, bearing (when available) and altitude code bits. This is necessary because aircraft may respond to more than one interrogation during each surveillance update interval. MergeModeCReplies algorithm specified in Appendix C of Volume II has been shown to meet the Mode C surveillance flight test requirements with data provided by at least one manufacturer.

Note: The requirements contained in this subsection are met by TCAS II systems that are compliant with RTCA/DO-185B and consider bearing in the determination of reply correlation.

2.2.4.6.4.1.2 Track Initiation

The ACAS Xa equipment **shall** (1325) initiate surveillance on Mode C transponder-equipped aircraft if and only if the conditions in a., b., and c. below are satisfied.

- a. Initially, a Mode C reply is received from the aircraft in each of three consecutive surveillance update intervals, and:
 1. The replies do not correlate with established tracks.
 2. The magnitude of the range rate indicated by the two most recent replies is less than 1200 kt.
 3. The replies correlate with each other in their range, bearing (when available) and altitude code bits.

Appendix A, Section A.5 describes successful methods that are recommended for assessing correlation of reply code bits and determining the initial altitude code estimate for a target.

- b. A fourth reply is received that correlates as described in a.(3) above within five surveillance update intervals following the third reply of the three consecutive replies in a. above.
- c. The altitude indicated by the four correlating replies is within ±10,000 ft relative to ACAS Xa.

ACAS Xa **shall** (1326) not use a reply to satisfy a. and b. above if the direction of arrival for that reply is outside of the required interrogation beamwidth as defined in §2.2.4.5.4.2.1 for a maximum suppression transponder.

A track is considered to be established for an aircraft when b. above is satisfied. All established tracks except as follows, including the information in the four correlating replies used for establishment, are made available to the STM and TRM. Mode C target reports that are identified as being formed by replies from aircraft that are on the ground (§2.2.4.6.2.2) **shall** (1328) be removed from any list of tracks that is provided to the TRM. If the correlation processing algorithms specified by Appendix C of the ADD (Volume II) are implemented in the STM then those on ground Mode C tracks **shall** (1327) be supplied to the STM.

2.2.4.6.4.1.3 Maintenance of Established Tracks

Established tracks **shall** (1329) be updated using replies that meet this criterion as a minimum:

The reply range and altitude correlate to the range and altitude predicted from previous reply history. Bearing may also be used as a criterion.

ACAS X **shall** (1330) not use a reply to satisfy the criterion above if the direction of arrival for that reply is outside of the required interrogation beamwidth as defined in §2.2.4.5.4.2.1 for a maximum suppression transponder.

The ACAS X equipment **shall** (1331) delete the established track on a Mode C transponder-equipped aircraft after the sixth surveillance update interval following receipt of the last valid correlating reply.

Appendix A, Section A.6 describes acceptable sets of algorithms for maintenance of Mode C target tracks.

2.2.4.6.4.1.4 Multipath False Tracks

Mode C target reports that are identified as having been formed by replies specularly reflected from the ground **shall** (1332) be used to initiate and maintain tracks but **shall** (1333) not be made available to the TRM.

Appendix, Section A.7 describes acceptable sets of algorithms that have been successfully used to identify image tracks.

Note: Image tracks should be evaluated each second to see if reclassification as a real track is warranted.

2.2.4.6.4.2 Mode S Targets

Note: In accordance with the aircraft address allocation scheme established by ICAO, each aircraft equipped with a Mode S transponder will have a unique ICAO 24-bit Aircraft Address, and addresses consisting of all 0s or all 1s are not permitted.

2.2.4.6.4.2.1 Squitter Processing

The identity of Mode S targets **shall** (1334) be determined by passively monitoring transmissions received with DF=11 (squitter transmissions). As an option, the identity of Mode S targets may also be determined by passively monitoring transmissions received with DF=17 (Extended Squitter). Error detection and correction (§2.2.4.4.2.2c) **shall** (1335) be applied to the received squitters to reduce the number of unnecessary interrogations. Squitters that contain an all 0s or all 1s ICAO 24-bit Aircraft Address or an address identical to that of ownship transponder **shall** (1336) be rejected.

The CA field in squitters from RTCA/DO-181A or later transponders **shall** (1337) be used to determine whether the aircraft is on the ground as described in §2.2.4.6.2.2. The equipment **shall** (1338) not interrogate an aircraft for acquisition in the following three cases: (1) the aircraft's CA field indicates that the aircraft is currently on the ground and the aircraft does not have an operational Active CAS, (2) the aircraft's CA field indicates that the aircraft is currently on the ground, the aircraft has an operational Active CAS, but own ACAS X is above 2000 feet AGL, or (3) the aircraft qualifies for acquisition using passive position reports as specified in §2.2.4.6.4.2.2.2.

For Mode S targets with RTCA/DO-181 transponders from which a squitter has been received or Mode S targets with RTCA/DO-181A or later transponders from which a squitter has been received that indicates the target to be airborne, altitude **shall** (1339) be determined by monitoring transmissions received with any, some, or all of DF=0, DF=4, or DF=17 messages; or, in the absence of such reply transmissions, by actively interrogating to elicit a short special surveillance reply. Validity of extended squitter transmissions and of altitude data is specified in §2.2.3.8.3.2.8 and §2.2.3.8.3.2.8.2 respectively.

The equipment **shall** (1340) monitor squitters and altitude replies to other interrogations whenever it is not transmitting or receiving replies to its own interrogations. Monitoring **shall** (1341) be accomplished using an antenna capable of simultaneous reception over 360 degrees of azimuth.

To reduce the number of ineffectual interrogations, no interrogations **shall** (1342) be transmitted to a target if so few squitters and altitude replies are received that the RF link seems unreliable. Also, the equipment **shall** (1343) not interrogate for acquisition a target whose altitude information indicates that it is not within $\pm 10,000$ ft of own altitude when ownership is airborne, or not within $\pm 3,000$ ft of own altitude when ownership is on the ground.

Targets that are within $\pm 10,000$ ft of own altitude when ACAS X is airborne, or within $\pm 3,000$ ft of own altitude when ACAS X is on the ground, and that indicate a reliable RF link are called valid targets. In order to provide timely acquisition of targets that transition the altitude surveillance boundary, the altitude of targets that are indicated to be outside of $\pm 10,000$ ft of own altitude when ACAS X is airborne, or outside of $\pm 3,000$ ft of own altitude when ACAS X is on the ground, **shall** (1344) continue to be monitored using any, some, or all of DF=0, DF=4 or DF=17 transmissions or, in the absence of such transmissions, by interrogating once every 10 seconds.

Ownership **shall** (1345) interrogate targets from which it does not receive altitude information but does continue to receive error-free squitters.

The ACAS X squitter processing function **shall** (1346) have memory capacity for at least 150 ICAO 24-bit Aircraft Addresses.

Note: Although the required surveillance range of ACAS X is 14 NM, Mode S squitters will typically be received from targets out to the nominal detection range of 30 NM. In a density of 0.3 Mode S aircraft per sq. NM, there will be about 141 aircraft within 30 NM of ACAS X.

Appendix A, Section A.8 describes one acceptable method of processing squitters and altitude replies to reduce unnecessary interrogations.

2.2.4.6.4.2.2 Acquisition

A valid Mode S target **shall** (1368) be acquired, through the exclusive use of extended squitters, provided that the following criteria are met:

- Ownship data meets the Extended Hybrid Surveillance quality requirements (§2.2.4.6.4.2.2.1.1.).
- Its signal strength is \leq Extended Hybrid Surveillance MTL (§2.2.4.6.4.2.2.1.3.) or ownship is operating on the surface (per §2.2.4.6.3).
- The Extended Hybrid Surveillance quality requirements are met (§2.2.4.6.4.2.2.1.1).

Aircraft that do not qualify for acquisition through the exclusive use of extended squitters **shall** (2112) be acquired through the use of active interrogations (§2.2.4.6.4.2.2.2).

2.2.4.6.4.2.2.1 Acquisition Using Passive Position Reports

The system **shall** (1367) perform Extended Hybrid Surveillance track acquisition and establishment in such a way that no interrogations are performed for a track when the conditions in §2.2.4.6.4.2.2 are satisfied.

Notes:

1. *To accomplish this, the system could process airborne extended squitters received below the Standard TCAS MTL in order to ensure that a passive Extended Hybrid Surveillance track is established before an active track would be acquired.*
2. *Use of DF=17 below the standard TCAS MTL is permitted for use by Extended Hybrid Surveillance as it does result in additional interrogations. RTCA/DO-260B was written to prohibit providing DF=17 extended squitters to TCAS below the standard TCAS MTL in order to prevent TCAS interrogations. This prohibition does not apply to Extended Hybrid Surveillance which does not use TCAS interrogations, but does apply to active and Hybrid Surveillance which do perform active interrogations. Additionally, interrogations to intruders whose signal strength is below the standard TCAS MTL are prohibited.*

There are three acceptable methods of acquiring the required ADS-B message elements:

1. ADS-B reports (not ADS-R or TIS-B) from an RTCA/DO-260B (Ref. G) compliant receiver.
2. ADS-B reports (not ADS-R or TIS-B) associated with RTCA/DO-317B (or later versions) (Ref. T) AIRB qualified track (but meeting additional requirements of §2.2.4.6.4.2.2.2).
3. Decoded directly per §2.2.3.8.3.2.8.

Note: Use of ADS-R and TIS-B data is not permitted. The received signal level of ADS-R and TIS-B data is not related to the relative range of the corresponding aircraft, and so cannot be used to alert TCAS when that aircraft may be closer than its position report indicates. A portion of UAT equipped ADS-B OUT aircraft will be equipped with ATCRBS transponders.

2.2.4.6.4.2.2.1.1 Extended Hybrid Surveillance Quality Requirements

Ownship and Traffic data quality requirements **shall** (1369) both be met for a target to qualify for Extended Hybrid Surveillance.

A Mode S intruder **shall** (1370) qualify for Extended Hybrid Surveillance when all of the following conditions are true:

- a) The ADS-B Version Number ≥ 2
- b) The reported NIC ≥ 6 (<0.6 NM)
- c) The reported NACp ≥ 7 (<0.1 NM)
- d) The reported SIL = 3
- e) The reported SDA = 2 or 3
- f) The barometric altitude is valid

Ownship position sources **shall** (1371) meet the following data quality standards:

- a) Ownship horizontal position uncertainty (95%) is < 0.1 NM
- b) Ownship horizontal position integrity bound is < 0.6 NM with integrity of $1e^{-7}$

2.2.4.6.4.2.2.1.2 Establishing an Extended Hybrid Surveillance Track

An Extended Hybrid track **shall** (1372) be established when it meets the following conditions:

- a) Two valid airborne position messages have been received within 5 surveillance update intervals
- b) The altitude in the two airborne position messages are within 500 ft of each other or are within a window large enough to accommodate a 10,000 fpm altitude rate – whichever is greater
- c) The Q-bit values in the two airborne position messages are identical
- d) The ICAO 24-bit Aircraft Address is the same in both airborne position messages and the address is valid

Once the track is considered established periodic surveillance updates using airborne position messages may be used to update Extended Hybrid Surveillance tracks.

2.2.4.6.4.2.2.1.3 Extended Hybrid Surveillance MTL

The Extended Hybrid Surveillance MTL **shall** (1373) be set to the maximum of -68dBm ± 2 dB or the MTL established by the interference limiting algorithms.

Note: Extended Hybrid Surveillance MTL should always be set at least as high as the interference limiting MTL in order to prevent the condition where an intruder's estimated signal strength is above the Extended Hybrid Surveillance MTL, but below the interference limiting MTL – this would result in the undesirable dropping of a target.

2.2.4.6.4.2.2.1.4 Determination of Estimated Signal Strength

The signal strength for an intruder **shall** (1374) be estimated using DF=11 and DF=17 squitters. The signal strength **shall** (1375) be estimated at least once every surveillance processing cycle if DF=11 or DF=17 squitters are available. The estimated signal strength **shall** (1376) be set to the maximum signal strength of the DF=11 and DF=17 squitters received since the last estimated signal strength was determined.

Notes:

1. *The purpose of these requirements is to ensure a timely update rate of an intruder's estimated signal strength. Using the maximum received signal strength provides a more conservative estimate of the received signal strength.*
2. *Filtering of signal strength to avoid frequent changes between Hybrid Surveillance and Extended Hybrid Surveillance provides no benefit since the requirements of §2.2.4.6.4.2.3.1.7 preclude unnecessary revalidation interrogations due to Extended Hybrid to Hybrid Surveillance transitions.*
3. *If DF=17 messages are used to estimate the signal strength, then it is acceptable to just use position messages.*

2.2.4.6.4.2.2 Acquisition Using Interrogations

The equipment **shall** (1347) transmit an acquisition interrogation (UF=0, AQ=1) to determine the range of each valid active acquisition target as defined in §2.2.4.6.4.2 Squitter Processing, or from which inadequate altitude information has been received. Error correction decoding (§2.2.4.4.2.2d) **shall** (1348) be applied to the received replies. If two or more valid replies are received in response to a single Mode S acquisition interrogation, only the reply that is closest in range **shall** (1349) be retained.

If an acquisition interrogation fails to elicit a valid reply, additional interrogations **shall** (1350) be transmitted. The total number of acquisition interrogations addressed to a single target **shall** (1351) not exceed three within a single surveillance update cycle and a total of nine within the first six surveillance update cycles. The first acquisition interrogation **shall** (1352) be transmitted using the top antenna or the antenna most likely to elicit a reply. If two acquisition interrogations to a target fail to elicit valid replies, the next two acquisition interrogations to that target **shall** (1353) be transmitted using the opposite antenna than used for the first acquisition interrogations. If acquisition interrogations fail to elicit replies within 6 surveillance update intervals, the acquisition process **shall** (1354) cease until enough additional squitters/fruit are received to indicate that successful acquisition is likely enough to warrant a subsequent acquisition attempt.

One acceptable means of accomplishing this is described in Appendix A, Section A.8.

If additional attempts are made to acquire the target, they **shall** (1355) conform to the pattern described in the requirement above for the first attempt except that:

- a. On the second and third attempt, only one interrogation is made during each single surveillance update interval; and in the absence of valid replies, six interrogations are transmitted during the first six surveillance update intervals.
- b. Any further attempts consist of a single interrogation during the entire six surveillance update intervals.

When a valid acquisition reply is received, the VS field in the reply **shall** (1356) be used to determine whether the aircraft is on the ground as described in §2.2.4.6.2.2. Mode S aircraft that are determined to be on the ground **shall** (1357) be monitored passively with replies (DF=0 or 4) or squitters (CA field of DF=11 or 17). In the absence of passive monitoring, an active interrogation once every five surveillance update intervals **shall** (1358) be used to monitor the air/ground status as long as the aircraft remains on the ground and continues to transmit squitters.

Following successful receipt of a valid acquisition reply from an airborne aircraft, one or more additional interrogations **shall** (1361) be transmitted to the target in order to confirm the reliability of the altitude data and the altitude quantization bit and to determine whether to establish track.

When two replies have been received (which may include the acquisition reply) with altitudes that are within 500 ft of each other, have identical Q-bit values and have the correct ICAO 24-bit Aircraft Address, the track is considered established and periodic surveillance interrogations **shall** (1362) be initiated for that target according to §2.2.4.6.4.2.3.

Acquisition interrogations **shall** (1366) have RL=0.

Note: The requirement that RL=0 is meant to prohibit attempts to use the TCAS crosslink to validate an intruder's Airborne Position Message at the same time as it is being acquired. Transponders that transmit DF=17 extended squitters are not necessarily required to support the TCAS crosslink capability and transponders without crosslink capability will not reply to interrogations with RL=1 per Ref. B. Attempts to combine passive surveillance validation using the TCAS crosslink with acquisition would lead to failure to acquire those transponders.

2.2.4.6.4.2.3 Maintenance of Established Tracks

Following its initial acquisition, ACAS X will use one of three surveillance methods:

- 1) Extended Hybrid Surveillance, as specified in §2.2.4.6.4.2.3.1.1.
- 2) Hybrid Surveillance, as specified in §2.2.4.6.4.2.3.1.4.
- 3) Active Surveillance, as specified in §2.2.4.6.4.2.3.2.

Figure 2-11 illustrates the state transitions between each surveillance region. The three blue boxes represent the potential surveillance regions or techniques – Extended Hybrid, Hybrid, and Active Surveillance. The text which labels the arrows represents what must be true to transition from one state to another. The grey boxes summarize all the transition criteria for a particular state – i.e. what must be true to be in that particular state. The figure does not illustrate the two different active surveillance techniques: normal (1 Hz) or reduced (0.2 Hz). Section 1.3 also provides a general overview of the three surveillance regions.

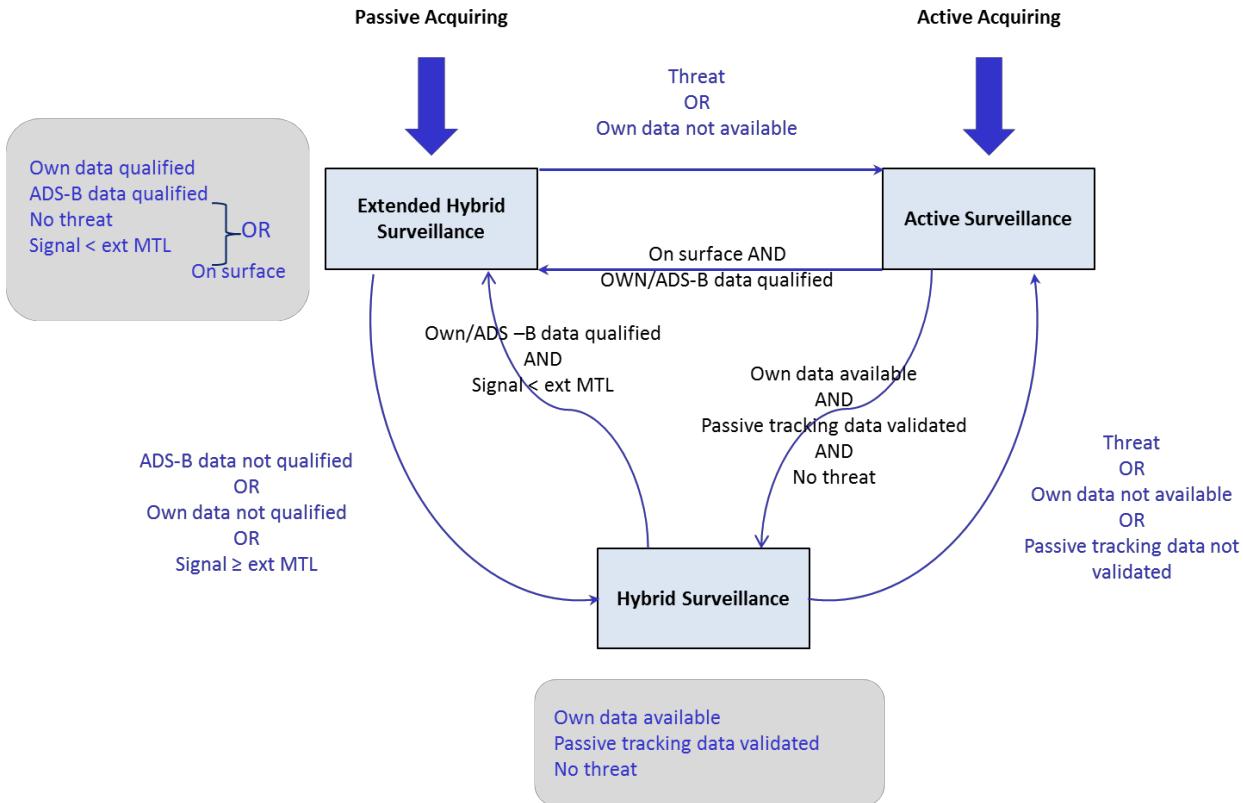


Figure 2-11: Surveillance Region State Transition Diagram

Note: Untracked passive surveillance data received from airborne ADS-B/ADS-R 1090 Extended Squitters (1090ES) will be provided directly to the STM at all times, as specified in §2.2.4.8. Untracked Mode S Reply (DF=0) data will be provided to the STM as processed by Hybrid Surveillance. The STM will perform additional validation of the passive surveillance data and perform the passive and active surveillance tracking. The STM maintains both a passive and an active track for a target if the appropriate data is available. The Hybrid Surveillance passive and active surveillance states are interpreted as follows: 1. Extended Hybrid Surveillance performs no active interrogations and provides no DF=0 data to the STM, 2. Hybrid Surveillance performs validation of the passive surveillance data with active interrogations and provides DF=0 data to the STM with the surv_mode set to Hybrid Surveillance (HS), and 3. Active Surveillance performs active interrogations and provides DF=0 data to the STM with the surv_mode set to Reduced (Red) or Normal (Norm).

2.2.4.6.4.2.3.1 Maintenance of Tracks Using Passive Surveillance

This section addresses tracking an intruder aircraft with ADS-B using a passive surveillance mode. All requirements within this section refer only to intruders under or entering a passive surveillance mode unless otherwise stated.

2.2.4.6.4.2.3.1.1 Conditions for Extended Hybrid Surveillance

An established passive track will enter the Extended Hybrid Surveillance state if any of the following are true:

- a) It is initially acquired and established using passive surveillance, as specified in §2.2.4.6.4.2.2.1.
- b) It is in the active surveillance state and the transition criteria in §2.2.4.6.4.2.3.1.3 are satisfied.
- c) It is in the Hybrid Surveillance state and the transition criteria in §2.2.4.6.4.2.3.1.3 are satisfied.

2.2.4.6.4.2.3.1.2 Persistence of Extended Hybrid Surveillance

An established track that is under Extended Hybrid Surveillance **shall** (1377) be maintained under Extended Hybrid Surveillance if it qualifies for Extended Hybrid Surveillance (per §2.2.4.6.4.2.2.1) and it does not satisfy any conditions requiring active surveillance as specified in §2.2.4.6.4.2.3.2.1.

Note: A track will also leave the Extended Hybrid Surveillance state if it is dropped according to any of the requirements of this MOPS.

2.2.4.6.4.2.3.1.3 Transitions to Extended Hybrid Surveillance

Active To Extended Hybrid Surveillance Transition

A track under active surveillance **shall** (1378) transition directly to Extended Hybrid Surveillance if own is operating on the surface (per §2.2.4.6.3) and it qualifies for Extended Hybrid Surveillance as specified in §2.2.4.6.4.2.2.1.

Note: This requirement does not preclude passive track initiation and establishment as described in §2.2.4.6.4.2.2.1. However, once an intruder is under active surveillance, this requirement ensures that the track will not transition directly to Extended Hybrid Surveillance unless own is operating on the airport surface. If own is airborne, then it will first transition to Hybrid Surveillance before transitioning to Extended Hybrid Surveillance (§2.2.4.6.4.2.3.1.6).

Hybrid to Extended Hybrid Surveillance Transition

A track under Hybrid Surveillance **shall** (1379) transition to Extended Hybrid Surveillance if it qualifies for Extended Hybrid Surveillance as specified in §2.2.4.6.4.2.2.1.

2.2.4.6.4.2.3.1.4 Conditions for Hybrid Surveillance

An established passive track will enter the Hybrid Surveillance state if:

- a) It is in active surveillance state, satisfies the modified hybrid threat criteria specified in §2.2.4.6.4.2.3.1.6, and passes validation tests as specified in §2.2.4.6.4.2.3.1.7.
- b) It is in Extended Hybrid Surveillance state, but the conditions for Extended Hybrid Surveillance in §2.2.4.6.4.2.2.1 are no longer satisfied, as specified in §2.2.4.6.4.2.3.1.7.

2.2.4.6.4.2.3.1.5 Persistence of Hybrid Surveillance

An established track that is under Hybrid Surveillance **shall** (1380) be maintained under Hybrid Surveillance unless:

1. It is required to transition to active surveillance as specified in §2.2.4.6.4.2.3.2.1.
2. It is required to transition to Extended Hybrid Surveillance as specified in §2.2.4.6.4.2.3.1.3.

Note: A track will also leave the Hybrid Surveillance state if it is dropped according to any of the requirements of this MOPS.

2.2.4.6.4.2.3.1.6 Hybrid Surveillance Region

A track under active surveillance **shall** (1381) transition to Hybrid Surveillance if it does not satisfy the conditions requiring active surveillance specified in §2.2.4.6.4.2.3.2.1 and any of the following conditions are true:

- 1) $-(s - 4900\text{ft})/\min(-1\text{ft/sec}, \dot{s}) \geq 65 \text{ sec}$ (which implies $s \geq 4900\text{ft}$)
OR
- 2) $-(r - 3.2\text{NM})/\min(-6\text{kt}/3600, \dot{r}) \geq 65 \text{ sec}$ (which implies $r \geq 3.2\text{NM}$)
OR
- 3) Ownership is operating on the surface as defined in §2.2.4.6.3 and the track does not qualify for Extended Hybrid Surveillance as specified in §2.2.4.6.4.2.2.1.

where:

$s = |\text{own altitude} - \text{track altitude}|$ = altitude separation, in ft

$\dot{s} = (\text{own altitude rate} - \text{track altitude rate}) \text{ sign } (\text{own altitude} - \text{track altitude})$;

= rate of change of s , in ft/s, with negative values indicating decreasing separation;

$r = \text{track slant range}$, in NM;

$\dot{r} = \text{rate of change of } r \text{ in NM/s}$, with negative values indicating decreasing range;

$\text{sign}(x) = 1 \text{ if } x \geq 0; -1 \text{ if } x < 0$.

Conditions 1) and 2) are referred to as the modified hybrid threat criteria.

The range rate used in the computation above **shall** (1382) be based on active range measurements only. If a range rate estimate is not available -1200 kt **shall** (1383) be assumed.

Notes:

1. §2.2.4.6.4.2.3.2.3 defines the criteria for the active surveillance region to be slightly smaller than the Hybrid Surveillance region. The difference in the hybrid and active surveillance regions creates a Hybrid Surveillance transition window (hysteresis) of a 5 sec tau, 400 ft alt and 0.2 NM range between hybrid and active tracking.
2. The transition criteria specified above ensure that a track does not transition to Hybrid Surveillance if it qualifies for an RA. See Appendix C.

2.2.4.6.4.2.3.1.7 Transitions to Hybrid Surveillance

Extended Hybrid to Hybrid Surveillance Transition

A track that is under Extended Hybrid Surveillance **shall** (1384) transition to Hybrid Surveillance if it does not satisfy the requirement for active surveillance stated in

§2.2.4.6.4.2.3.2.3 and no longer qualifies for Extended Hybrid Surveillance as specified in §2.2.4.6.4.2.2.1.

The first time a track transitions from Extended Hybrid to Hybrid Surveillance the track's Airborne Position Message data **shall** (1385) be validated per the requirements of §2.2.4.6.4.2.3.1.8 prior to transitioning to Hybrid Surveillance. Failure of validation when a reply is received **shall** (1386) cause the track to transition to active surveillance. Failure of validation when no reply is received **shall** (1387) cause the track to be deleted according to the requirements of §2.2.4.6.4.2.3.2.5. Measured range and bearing from the validation replies **shall** (1388) not be used to update the track. If the Extended Hybrid Surveillance track had been previously validated it **shall** (1389) transition to Hybrid Surveillance and be revalidated per §2.2.4.6.4.2.3.1.10.

Notes:

1. *The measured range and bearing are not used to update the track because of the undesirability of mixing measured and calculated range data in updating the track, as noted in §2.2.4.6.4.2.3.2.4.*
2. *The purpose of the last requirement is to ensure that if a validation interrogation was previously performed on a track, another interrogation may not be required solely based on the track transitioning to Hybrid Surveillance.*

Active to Hybrid Surveillance Transition

An active surveillance track **shall** (1390) transition to Hybrid Surveillance if the conditions of §2.2.4.6.4.2.3.1.6 are met and it passes the validation test specified in §2.2.4.6.4.2.3.2.7. However, if there have been one or more attempts to validate or revalidate the track, and the last prior such attempt was unsuccessful, transition to Hybrid Surveillance **shall** (1391) occur only if the conditions of §2.2.4.6.4.2.3.1.6 are met and the validation test in §2.2.4.6.4.2.3.2.7 is passed on two consecutive validation attempts. The rate of validation attempts is specified in §2.2.4.6.4.2.3.1.10.

Note: *An equipment manufacturer may limit the number of validation attempts for any given track as a method of managing processing time and/or resources.*

2.2.4.6.4.2.3.1.8 Track Updates Using Airborne Position Messages

Established tracks under passive surveillance **shall** (1392) be updated when all of the following apply:

- a. A valid Airborne Position Message is received for that track's ICAO 24-bit Aircraft Address. §2.2.3.8.3.2.8 defines how valid latitude, longitude and altitude are determined.
- b. That Airborne Position Message contains both valid position (latitude / longitude) and valid altitude data.
- c. The calculated slant range derived by combining the position and altitude data from the Airborne Position Message with ownship's position and altitude falls within a range window centered on a range predicted from the previous track update history.
- d. The Airborne Position Message altitude falls within an altitude window centered on an altitude predicted from the previous track update history.

In addition to the above criteria, established tracks under Extended Hybrid Surveillance **shall** (1393) be updated only if the quality parameters of the track and the airborne position message qualify per §2.2.4.6.4.2.2.1.

Surveillance **shall** (1394) be performed in such a way that prevents re-acquisition of a track during transitions between surveillance modes (Active, Hybrid and Extended Hybrid).

Note: The purpose of this requirement is to ensure that ACAS X does not perform acquisition interrogations during transitions between passive and active surveillance, and to maintain display continuity. For example, the window used to correlate a passive and active track in range should be at least as large as the range validation window to prevent the track from coasting out and being re-acquired.

2.2.4.6.4.2.3.1.9 Tracking in the Absence of Airborne Position Messages

If an established track that is being updated using Airborne Position Messages is not updated (because of a missed Airborne Position Message or an Airborne Position Message with invalid altitude or position as defined in §2.2.3.8.3.2.8) during a surveillance update interval then the track **shall** (1397) follow the requirements in §2.2.4.6.4.2.3.2.5 for missed replies.

Note: The idea is that Airborne Position Messages can be treated as “pseudo replies”. Therefore if an Airborne Position Message is not received when the active track normally would have been updated then the track should be coasted.

If the passive track is not updated with Airborne Position Message data at the nominal active interrogation rate (0.2 Hz or 1 Hz) required by §2.2.4.6.4.2.3.2.5 due to messages with missing or invalid position data then the track **shall** (1398) transition to an active track.

If the passive track is coasted due to lack of Airborne Position Message data for more time than is allowed per §2.2.4.6.4.2.3.2.5 then the track **shall** (1399) be dropped.

Notes:

1. *This requirement ensures that a track will not be deleted based solely on invalid Airborne Position Messages.*
2. *In order for an Airborne Position Message to update a track it is required to contain a valid altitude and position. If the transponder stops reporting altitude, updates will cease and the requirements in this section will cause the track to transition to active surveillance or be dropped. The altitude reported in the Airborne Position Message is important to the safety of passive surveillance, and if it is not available for an intruder then passive surveillance should be discontinued for that target.*
3. *In summary, the first paragraph of this section requires coasting if no squitter is received or an invalid squitter is received during any given second as is required of active surveillance in §2.2.3.8.3.2.8. The second paragraph requires transition to active surveillance if the minimum update interval that would normally be required by active surveillance in §2.2.3.8.3.2.8 is not achieved with valid Airborne Position data. The third paragraph requires a passive track to be coasted and dropped if no Airborne Position data is received.*

-
4. If a valid Airborne Position Message is received for a track, but the track is not updated because the range calculated from that message does not fall within the correlation window required in §2.2.4.6.4.2.3.1.8(c) or the altitude in that message does not fall within the altitude correlation window required in §2.2.4.6.4.2.3.1.8(d), then it should be treated as required by §2.2.3.8.3.2.8 for such correlation failures.

2.2.4.6.4.2.3.1.10 Revalidation

An established track that is under Hybrid Surveillance (per §2.2.4.6.4.2.3.1.4) **shall** (1400) be subject to revalidation. If a track under Hybrid Surveillance does not satisfy the first (altitude) condition of §2.2.4.6.4.2.3.2.3, it **shall** (1401) be subject to revalidation every 60th surveillance update interval; if it satisfies the first and second (altitude and range) conditions of §2.2.4.6.4.2.3.2.3 but not the third (airborne) condition, it **shall** (1402) be subject to revalidation every 10th surveillance update interval; if it satisfies the first condition of §2.2.4.6.4.2.3.2.3 but not the second (range) condition, it **shall** (1403) be subject to revalidation at intervals calculated according to the following procedure. The revalidation interval t **shall** (1404) be calculated at the time of the initial successful validation and at the time of each successful revalidation. It **shall** (1405) be used as the number of surveillance update intervals until the next revalidation attempt.

1. If $v_0 \geq +300$ kt, then set the revalidation interval, t , to 60 seconds.
2. Otherwise, the revalidation interval, t , is determined from the equation below.

$$t = \max\left(10, \min\left(60, \text{trunc}\left(\frac{-(v_0 + at_{thr}) - \sqrt{(v_0 + at_{thr})^2 - 2a(r_0 + v_0 t_{thr} - s_{mod})}}{a}\right)\right)\right)$$

The revalidation interval, t , is the time in seconds that it would take to satisfy the range condition for active interrogation under the assumed acceleration, starting from the time of the current successful revalidation, truncated to the next smaller integer, and constrained to values between 10 and 60 seconds. r_0 is the estimated range in feet of the intruder determined from passive surveillance. v_0 is the estimated range rate in ft/s of the intruder, with positive range rates indicating divergence in range, also determined from passive surveillance. Since t is only calculated at the time of a successful validation or revalidation, the r_0 and v_0 values are those at the time of that validation or revalidation. a is the assumed range acceleration of -11 ft/s²; the negative value indicating acceleration toward ownship. s_{mod} is a range offset of 18228 ft (3 NM) that appears in the range condition for transitioning from passive to active surveillance. t_{thr} is the range tau threshold of 60 seconds for transition from passive to active surveillance. The function ‘trunc’ is the truncation function that converts a real number to the next smaller integer.

Alternatively, for closure rates of magnitude less than or equal to 1200 knots, instead of using the procedure specified above, the revalidation interval may be determined according to Table 2-19 below, where the range and range rate have been quantized as shown and the value of t determined by the above procedure has been further quantized to the values 10, 20, 30, 40 and 60 seconds by selecting the largest of those values that is equal to or smaller than the calculated value.

Active interrogations **shall** (1406) be transmitted and the replies **shall** (1407) be used to revalidate the Airborne Position Message data using the procedure specified in §2.2.4.6.4.2.3.2.7 with a maximum slant range difference of 340 m. When the required revalidation rate changes, the first revalidation at the new rate **shall** (1408) be timed from

the last revalidation at the old rate. If a valid reply is not received during the current ACAS X Processing Cycle then attempts to elicit a valid reply **shall** (1409) be performed during subsequent ACAS X Processing Cycles. The revalidation interrogations **shall** (1410) count as tracking interrogations with respect to the interrogation limits in §2.2.4.6.4.2.3.2.5 and those interrogation limits **shall** (1411) be observed.

Failure of revalidation when a reply is received **shall** (1412) cause the track to transition to active surveillance.

Failure of revalidation when no reply is received **shall** (1413) cause the track to be deleted according to the requirements of §2.2.4.6.4.2.3.2.5.

Note: In the event of a failure to reply to revalidation interrogations, intruder ADS-B report data should still be provided to STM according to the requirements of §2.2.5.5.9, §2.2.5.5.10, §2.2.5.5.11, etc. The STM will maintain the intruder as an ADS-B Only target available for display and optional ADS-B Only TA-Only alerting.

Measured range and bearing from the revalidation replies **shall** (1414) not be used to update the track.

Notes:

1. *The measured range and bearing are not used to update the track because of the undesirability of mixing measured and calculated range data in updating the track, as noted in §2.2.4.6.4.2.3.2.4.*
2. *Validation and revalidations do not apply to established tracks under Extended Hybrid Surveillance.*

Table 2-19: Interval Between Revalidations

Range Rate (kt)	Range (NM)																											
	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	≥30
≥300	60	60	60	60	60	60	60	60	60	60	60	60	60	60	60	60	60	60	60	60	60	60	60	60	60	60	60	60
200	30	40	40	40	50	60	60	60	60	60	60	60	60	60	60	60	60	60	60	60	60	60	60	60	60	60	60	60
100	10	20	30	30	40	40	40	60	60	60	60	60	60	60	60	60	60	60	60	60	60	60	60	60	60	60	60	60
0	10	10	10	20	20	30	40	40	40	60	60	60	60	60	60	60	60	60	60	60	60	60	60	60	60	60	60	60
-100	10	10	10	10	10	20	20	30	30	40	40	40	40	40	60	60	60	60	60	60	60	60	60	60	60	60	60	60
-200	A	10	10	10	10	10	10	20	20	30	30	40	40	40	40	40	40	60	60	60	60	60	60	60	60	60	60	60
-300	A	A	A	10	10	10	10	10	10	20	20	30	30	30	40	40	40	40	40	60	60	60	60	60	60	60	60	60
-400	A	A	A	A	A	10	10	10	10	10	10	20	20	20	30	30	40	40	40	40	40	40	40	40	40	60	60	60
-500	A	A	A	A	A	A	10	10	10	10	10	10	10	20	20	20	30	30	30	40	40	40	40	40	40	40	40	40
-600	A	A	A	A	A	A	A	10	10	10	10	10	10	10	10	20	20	20	20	30	30	30	40	40	40	40	40	40
-700	A	A	A	A	A	A	A	A	A	10	10	10	10	10	10	10	10	20	20	20	20	30	30	30	30	30	40	
-800	A	A	A	A	A	A	A	A	A	A	10	10	10	10	10	10	10	10	20	20	20	20	20	20	20	30	30	
-900	A	A	A	A	A	A	A	A	A	A	A	10	10	10	10	10	10	10	10	10	10	10	10	20	20	20	20	
-1000	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	10	10	10	10	10	10	10	10	10	20		

-1100	A	A	A	A	A	A	A	A	A	A	A	A	A	A	10	10	10	10	10	10	10	10	20
-1200	A	A	A	A	A	A	A	A	A	A	A	A	A	A	10	10	10	10	10	10	10	10	20

Notes: The table entries are the safe time interval in seconds until the next revalidation, or equivalently the number of surveillance update intervals until the next revalidation.

Range = track slant range, in NM. A given column applies to a track if the track's range is greater than or equal to the range in the column header but less than the range of the next column to the right.

Range Rate = rate of change of Range in kt (NM/hour), with negative values indicating decreasing range. A given row applies to a track if the track's range rate is greater than or equal to the range rate in the row header but less than the range rate in the row header of the row above.

A = All of the range and range rate combinations in that cell satisfy the conditions for transitioning to active interrogations, and therefore that cell should never be accessed to determine the safe interval until the next revalidation.

2.2.4.6.4.2.3.1.11 Error Budget Allocated to ACAS X for Slant Range Validation

To get the benefit of Hybrid Surveillance, the errors under the control of Hybrid Surveillance should be minimized as much as practical to ensure that aircraft that could be validated as hybrid are validated, and that slant range is computed sufficiently accurately for the purpose of ACAS X. To ensure timely transition from passive to active surveillance, the range and range rate estimates developed from tracking passive position reports must be sufficiently accurate. To develop a reasonable error budget for the hybrid data processing errors, the size of the error sources for the system as a whole was estimated and compared with the validation threshold. Based on this analysis, the hybrid data processing errors **shall** (1415) add no more than 145 m of error, 95% of the time, at the reference time for which that slant range is calculated, and for ownship speeds up to 600 kt, ownship altitude rates within the range $\pm 10,000$ ft/min, and ownship horizontal and vertical accelerations less than 0.5 g (16 ft/sec 2).

Appendix D provides information on different methods of computing slant range and bearing from latitude, longitude, and altitude. Details on the error analysis are provided in Appendix E. The requirements for developing a sufficiently accurate range rate estimate based on tracking successive range reports are developed in RTCA/DO-300A Appendix D, §D.3.4.2.4. Note that Appendix E and RTCA/DO-300A Appendix D, §D.3.4.2.4 make assumptions about the maximum uncompensated latency of ownship position reports at the time of their delivery to the ACAS X unit and also about the maximum variation in that uncompensated latency from its average value. Both of those parameters are outside the control of the ACAS X system and so outside the scope of this MOPS. If the uncompensated latency or the variation in that latency in an installed system under realistic operational loading will be substantially larger than the assumptions in Appendices C and E, those analyses should be revisited to ensure both beneficial and safe operation. It might be possible to reduce the budget allowed for errors in Hybrid Surveillance data processing to compensate for a larger uncompensated latency in ownship's position reports, for example. The assumed variations in the uncompensated latency do not have a major effect in comparison to the other sources of range variation, but much larger values might affect the tracking parameters used to track passive position reports, increase the time required to obtain sufficient range rate accuracy, and increase the required size of the range correlation windows.

2.2.4.6.4.2.3.2 Maintenance of Tracks Using Active Surveillance

This section addresses tracking an intruder aircraft with Mode S interrogation/reply using an active surveillance mode. All requirements within this section refer only to intruders under or entering an active surveillance mode unless otherwise stated.

2.2.4.6.4.2.3.2.1 Conditions for Active Surveillance of a Track

An established Mode S track enters the active surveillance state if any of the following are true:

- a) It is acquired and established by means of active surveillance as specified in §2.2.4.6.4.2.2.2.
- b) It fails the validation test during a transition from Extended Hybrid Surveillance to Hybrid Surveillance as specified in §2.2.4.6.4.2.3.1.7.
- c) It fails the revalidation test while under Hybrid Surveillance, as specified in §2.2.4.6.4.2.3.1.10.
- d) It is under passive surveillance (either Hybrid or Extended Hybrid) and satisfies the hybrid threat criteria for transition to active surveillance as specified in §2.2.4.6.4.2.3.2.3.
- e) It is under passive surveillance and its Airborne Position Messages continue to be received, but have missing or invalid position data as specified in §2.2.4.6.4.2.3.1.9
- f) It is under passive surveillance and the source of ownship latitude and longitude data is declared invalid or unavailable as specified in §2.2.7.2.9.

2.2.4.6.4.2.3.2.2 Persistence of Active Surveillance

An established track that is under active surveillance **shall** (1416) be maintained under active surveillance as specified in §2.2.4.6.4.2.3.2.5 unless it:

- a) Transitions to Hybrid Surveillance as specified in §2.2.4.6.4.2.3.1.7, or
- b) Transitions to Extended Hybrid Surveillance as specified in §2.2.4.6.4.2.3.1.3.

Note: This requirement addresses only surveillance state transitions while the track remains established. A track will also leave the active surveillance state if it is dropped according to any of the requirements of this MOPS.

2.2.4.6.4.2.3.2.3 Active Surveillance Region

A track under passive surveillance **shall** (1417) transition to active surveillance if the following conditions are all true:

- 1) $-(s - 4500\text{ft})/\min(-1\text{ft/sec}, \dot{s}) \leq 60 \text{ sec}$
- 2) $-(r - 3\text{NM})/\min(-6\text{kt}/3600, \dot{r}) \leq 60 \text{ sec}$
AND
- 3) Ownship is taking off or airborne per §2.2.4.6.3.

The terms in the above equations are defined in §2.2.4.6.4.2.3.1.6.

Conditions 1) and 2) are referred to as the hybrid threat criteria.

The range and range rate used in the computation above **shall** (1418) be based on passive range measurements only. When a track initially transitions to passive surveillance it is possible that sufficient passive reports are not available to determine a range rate. This period will be no longer than 5 surveillance update intervals (§2.2.4.6.4.2.3.1.8). While a passive surveillance range rate is not available the 2nd condition above **shall** (1419) be considered false.

Note: §2.2.4.6.4.2.3.1.6 *Hybrid Surveillance Region defines the criteria for the passive surveillance region to be slightly larger than the active surveillance region. The difference in the active and passive surveillance regions creates a Hybrid Surveillance transition window (hysteresis) of a 5 sec tau, 400 ft alt, and 0.2 NM range between active and passive tracking.*

If no altitude rate has been determined for a track, then the value $\pm 10,000$ ft/min, with the sign chosen to be sign (own altitude – track altitude), **shall** (1420) be used in place of [dot] in equation 1 above.

Note: $\pm 10,000$ ft/min (± 167 ft/sec) is the maximum intruder altitude rate for which ACAS X was designed. Using the worst case vertical closure rate when no data is available ensures that the system will not miss a potential threat and delay transition from passive to active surveillance.

When a track is in the active surveillance region all interrogations of the track **shall** (1421) have RL=0.

Note: When it is determined that a transition to active surveillance is required the target is interrogated. The altitude, slant range, and bearing of the reply will be used to update the track based on the requirements in §2.2.4.6.4.2.3.2.4.

2.2.4.6.4.2.3.2.4 Passive to Active Surveillance Transition

Surveillance must be performed such that a track is not dropped when transitioning from passive to active surveillance. A nominal transition is defined as one in which qualification, validation, or revalidation tests passed successfully. Therefore, the range correlation window used during this transition **shall** (1424) be at least as large as the revalidation window.

A nominal transition from passive to active surveillance **shall** (1425) be done in such a way that passive measurements are continuously received and provided to the STM as described in §2.2.4.6.4.2.3.1.

Note: The purpose of this requirement is to ensure that ACAS X does not drop and restart tracks or change track numbers on the same aircraft during normal transitions between passive and active surveillance.

2.2.4.6.4.2.3.2.5 Track Updates Using Active Surveillance

Tracks under active surveillance **shall** (1426) be updated using replies that meet the following criteria as a minimum:

- The ICAO 24-bit Aircraft Address, following the error correction decoding process of §2.2.4.4.2.2(d), corresponds to an established track.
- The reply range occurs within a range window centered on a range predicted from previous reply history.

- c. The reply altitude occurs within an altitude window centered on the altitude predicted from previous reply history.

Note: The altitude reporting status of a track may change from non-altitude reporting to altitude reporting and from altitude reporting to non-altitude reporting. Item c applies only if the established track is an altitude reporting track and the reply contains altitude data.

If ownship is on the ground, Mode S intruders with established tracks **shall** (1427) continue to be tracked as long as the intruder is airborne according to §2.2.4.6.2.2.

The range and altitude correlation window process described in Appendix A, Section 6 for Mode C targets is acceptable for Mode S targets.

In the event that two or more surveillance tracks are associated with the same ICAO 24-bit Aircraft Address, only the track that is closest in range **shall** (1428) be retained.

The range of the target **shall** (1429) be used with its estimated range rate relative to ACAS X to determine its potential threat to ACAS X and whether it can be interrogated less frequently than the nominal 1-second rate. Each scan, the potential threat level (*TAU*) of the target **shall** (1430) be calculated as follows:

$$TAU = -\frac{r - \frac{SMOD^2}{r}}{\min(-6 \text{ kt}, rdot)}$$

where *r* is the tracked range, *rdot* is the estimated relative range rate and *SMOD* is a surveillance distance modifier which for this purpose **shall** (1431) be equivalent to 3 NM. This value of *SMOD* ensures that ACAS X will always use the nominal 1-second update rate in situations where *TAU* can change rapidly, such as in a parallel approach. The denominator ensures use of the reduced rate for diverging intruders beyond *SMOD*.

All actively tracked intruders **shall** (1432) be interrogated at least once every five surveillance update intervals. An intruder with a *TAU* value of equal to or less than 60 seconds **shall** (1433) be interrogated at the nominal surveillance update rate of once every surveillance update interval. An intruder with a *TAU* value greater than 60 seconds **shall** (1434) be interrogated at a rate of no more than once every five surveillance update intervals if:

- a. The tracked barometric altitude of ownship is less than 18,000 ft, and
- b. The tracked altitude of the intruder aircraft is less than 18,000 ft.

A hysteresis of ± 500 ft **shall** (1435) be applied to the above 18,000-ft altitude boundary in order to prevent rapid oscillations between two different interrogation rates.

Note: The above statement is not intended to prohibit re-interrogation(s). If a tracking interrogation fails to elicit a valid reply from a target being updated at a 5-second rate, additional interrogations are transmitted as specified below, i.e., surveillance should not wait five seconds to re-interrogate.

Each scan surveillance makes available to the STM the data specified in §2.2.5.5.

Appendix A, Section A.11 describes one acceptable algorithm for tracking of aircraft with a 5-second interrogation rate.

If, during active surveillance, a tracking interrogation fails to elicit a valid reply, additional interrogations **shall** (1436) be transmitted. The total number of tracking interrogations addressed to a single target **shall** (1437) not exceed five during a single surveillance update

interval or sixteen distributed over six successive surveillance update intervals. The first tracking interrogation **shall** (1438) be transmitted using the antenna that was used in the last successful interrogation of that target. If two successive tracking interrogations fail to elicit valid replies from a target, the next two interrogations to that target **shall** (1439) be transmitted using the other antenna.

Note: The surveillance update interval in which the last correlating reply is received will lead to an advance of the track estimate, not a coast. The first coast will occur in the following surveillance update interval. An example timeline is illustrated in Figure 2-12 below:

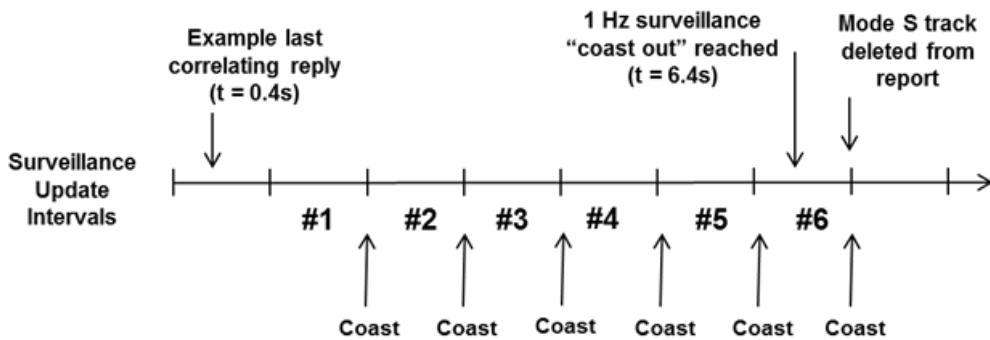


Figure 2-12: Example Coast Timeline

The ACAS X equipment **shall** (1440) delete the established track on a Mode S transponder-equipped aircraft after the sixth surveillance update interval (5 coasts) following receipt of the last valid correlating reply if the track was maintained with interrogations transmitted once every surveillance update interval, or after the tenth surveillance update interval (9 coasts) following receipt of the last valid correlating reply if the track was maintained with interrogations transmitted once every five surveillance update intervals. The ACAS X equipment **shall** (1441) also delete the established track on a Mode S transponder-equipped aircraft following five consecutive replies in which the VS field indicated the aircraft to be on the ground. Any subsequent squitters and replies from that aircraft are subject to the squitter processing requirements of §2.2.4.6.4.2.1. The ICAO 24-bit Aircraft Address associated with the deleted surveillance track **shall** (1442) be retained an additional four surveillance update intervals so as to enable immediate reacquisition if a squitter is received within that period.

If a track under active surveillance satisfies the modified Hybrid Surveillance criteria of §2.2.4.6.4.2.3.1.6, but has not been given a validation test as specified in §2.2.4.6.4.2.3.1.7, then each active interrogation **shall** (1443) also be used as a validation interrogation as specified in §2.2.4.6.4.2.3.2.7, and any reply **shall** (1444) be subjected to the validation test. If the validation test succeeds, the track transitions to Hybrid Surveillance.

2.2.4.6.4.2.3.2.6 Power Programming

The transmit power level of Mode S tracking interrogations to targets (but not air-to-air coordination interrogations) **shall** (1445) be automatically reduced as a function of range for targets within 10 NM as follows:

$$PT = P_{max} + 20 \log \frac{R}{10}$$

where PT is the adjusted power level. P_{max} is the nominal power level, which is transmitted to targets at ranges of 10 NM or more, and R is the predicted range of the target in NM.

The actual transmitted power is the lesser of PT and the limit imposed by the interference limiting procedures of §2.2.3.6.

Air-to-air coordination interrogations **shall** (1446) always be transmitted at full power.

If an initial power programmed tracking interrogation does not elicit a reply the system **shall** (2130) use at least one maximum power retry interrogation (in accordance with interference limiting).

Note: The maximum power interrogation is used to cope with some types of encounters (e.g. intruder is just below or just above, intruder transponder antenna or RF computer fails or degraded) for which the signal can be lost with power programming because the RF link is not nominal.

2.2.4.6.4.2.3.2.7 Validation of Airborne Position Message Data

The Airborne Position Message data from a Mode S transponder **shall** (1448) be validated by comparing Airborne Position Message data to measured range, bearing, and the reported altitude using standard ACAS X surveillance interrogations with a $UF=0$, $RL=0$. If no reply is received, additional validation interrogations **shall** (1449) be transmitted. However, all validation interrogations **shall** (1450) count as tracking interrogations with respect to the interrogation limits in §2.2.4.6.4.2.3.2.5, and those interrogation limits **shall** (1451) be observed.

Note: The purpose of requiring interrogations with $RL=0$ is to prevent the use of the ACAS X crosslink for validation of Airborne Position Message data thus reducing the overall usage of the 1090 MHz downlink.

If a reply is received, the position and altitude data from an Airborne Position Message or multiple Airborne Position Messages **shall** (1452) be combined with position and altitude data of ownship at a common reference time, in order to determine a calculated slant range and bearing based on these reported positions. The overall error in the calculated slant range **shall** (1453) meet the requirements of §2.2.4.6.4.2.3.1.11. The calculated slant range and bearing and the altitude reported in the Airborne Position Message **shall** (1454) be compared at a common reference time (e.g., the time of the active reply) to values determined from measured range and bearing data and the reported altitude from standard ACAS X surveillance interrogations with $UF=0$, $RL=0$.

Notes:

1. §2.2.3.8.3.2.8 defines the requirements for determining valid latitude, longitude, and altitude.
2. Specific equations for the transformation from reported positions to calculated slant range are provided as guidance material in Appendix D.

The Airborne Position Message data from a Mode S transponder **shall** (1455) pass the validation test if:

$|$ slant range difference $| \leq 290$ meters; and

$|$ bearing difference $| \leq 45$ degrees; and

$|$ altitude difference $| \leq 100$ feet.

If bearing data is available then the active and passive bearing **shall** (1456) meet the criteria above. However, if bearing is not available then the bearing comparison **shall** (1457) not be required to meet the validation requirements.

Note: One reason that bearing may not be available is the case of an aircraft that is tracked only with an omni-directional lower antenna.

2.2.4.6.4.2.4 NTA3/NTA6 Range Determination for Active CAS On-Ground Intruders

When ownship is airborne and at or below 2000 ft AGL, the range of Mode S aircraft that are determined to be both on the ground according to their extended squitter CA field, and Active CAS-equipped, according to the TCAS Broadcast Interrogation **shall** (1363) have their range determined for the purposes of NTA3 and NTA6 through:

- a) Surface Position Messages with DF=17 Format Type Code 5, 6, 7 or 8 received within the last 25 seconds.
- b) On-Ground Ref. T tracks which qualify for AIRB or SURF application active interrogations once every 5 seconds when Surface Position Messages are not available.

Note: For simplicity, the use of Format Type Code 8 surface squitters without examination of NIC supplement is permitted. The potential error associated with the use of Format Type Code 8 surface squitters will only impact the NTA3 and NTA6 estimates, but will not impact on the ACAS X collision avoidance function. It is permissible for an equipment manufacturer to limit the use of Format Type Code 8 squitters through examination of the NIC supplement or NACP as specified in Ref. T or §2.2.4.6.4.2.2.2.

2.2.4.6.5 Non-Altitude-Reporting Aircraft

Replies from non-altitude-reporting aircraft **shall** (1458) be tracked using range and bearing information.

Appendix A, Section A.9 describes one method that has been used successfully to track altitude unknown targets.

2.2.4.6.6 Bearing Estimation

The ACAS X equipment **shall** (1459) provide a bearing estimation function with the following requirements.

Note: The bearing estimation described in this section supports surveillance and traffic display functions separate from STM tracking functions. An ACAS X implementation may utilize STM tracking algorithms to provide bearing estimates for surveillance and/or target display, or may implement separate bearing estimation functions. In all cases, the bearing estimates must adhere to the requirements of this section.

2.2.4.6.6.1 General Requirements

The bearing estimation of active surveillance targets **shall** (1460) be realized using a top-mounted direction-finding antenna. The bottom antenna can be an omnidirectional (not direction finding) antenna.

Bearing information for the display of targets under ADS-B surveillance **shall** (2108) be calculated based on the present ownship WGS84 horizontal position.

Targets meeting the criteria for a TA or an RA **shall** (1461) be displayed even if there is no bearing estimate available.

Note: The bearing estimate is used by the flight crew of ownship to visually locate intruder aircraft. ACAS X bearing accuracy is not adequate to support horizontal resolution maneuvers.

The system utilizes this bearing estimation in several capacities.

1. Mode C surveillance uses this bearing estimation capability to support the identification and rejection of replies received from outside the interrogated azimuth sector (§2.2.4.6.4.1.2).
2. The bearing estimate can assist the Mode S surveillance function in the acquisition of Mode S intruders when associated with a Mode S interrogation capability (§2.2.4.7.3).
3. The bearing estimate can assist in the acquisition of Mode C surveillance tracks and the association of Mode C replies to existing tracks (§2.2.4.6.4.1.2 and §2.2.4.6.4.1.3).

2.2.4.6.6.2 Active Surveillance Bearing Accuracy With Standard Ground Plane

2.2.4.6.6.2.1 Accuracy, -10 Deg. To +10 Deg. Elevation

Active surveillance bearing error **shall** (1462) not exceed nine degrees RMS or 27 degrees peak over all azimuth angles and over elevation angles from -10 degrees to +10 degrees in the absence of interference and multipath, using Mode S replies or Mode C replies having five or more code pulses (in addition to the framing pulses). The bearing accuracy performance requirements assume the antenna is installed at the center of a minimum 1.2 m (4 ft) diameter circular ground plane that can be either flat or cylindrical.

2.2.4.6.6.2.2 Accuracy, greater than 10 Deg. To +20 Deg. Elevation

For elevation angles greater than 10 degrees to +20 degrees the active surveillance bearing error **shall** (1463) not exceed 15 degrees RMS or 45 degrees peak over all azimuth angles under the conditions of §2.2.4.6.6.2.1.

Note: The bearing estimation system may include a built-in automatic bearing bias correction capability if the bearing bias can become inconsistent with these requirements due to drift or component change. If provided, the automatic calibration should be accomplished immediately after every power turn-on event and should be capable of maintaining bearing bias correction during any flight leg.

2.2.4.6.6.3 Bearing Accuracy in the Presence of Interference

The requirements in the following subparagraphs apply under the same conditions as defined in §2.2.4.6.4.1 except as expressly defined herein.

2.2.4.6.6.3.1 Mode C Interleaved Replies

The RMS bearing error for a Mode C reply containing five code pulses **shall** (1464) not increase by more than one degree when that reply is interleaved with another reply of the same amplitude containing five code pulses.

Note: This requirement ensures that pulse bearing measurements are correctly associated with the correct reply in an interleaved garble situation.

2.2.4.6.6.3.2 Mode C Overlapped Replies

The RMS bearing error for a Mode C reply containing seven code pulses **shall** (1465) not increase by more than one degree when that reply is interfered with by another reply of the same amplitude such that any two of the code pulses in the first reply are garbled.

Note: This requirement ensures that garbled pulse bearing measurements are not used in developing a reply bearing estimate.

2.2.4.6.6.3.3 Mode S Overlapped Replies

The RMS bearing error for Mode S reply **shall** (1466) not increase by more than three degrees when interfered with by a Mode A or Mode C reply whose power is 7 dB below the Mode S reply power.

Note: This requirement ensures that an interfering Mode A or Mode C fruit reply does not significantly degrade the Mode S reply bearing measurement when the interference power is -7 dB relative to the Mode S reply power.

2.2.4.6.6.4 Bearing Filter Performance

The bearing measurements for each intruder aircraft **shall** (1468) be filtered to reduce angular fluctuations.

Algorithms PredictCartesianTracker and UpdateCartesianTracker of the ACAS X ADD describe Cartesian Kalman filter estimation algorithms used to develop smoothed target bearing estimates of both passive and active surveillance targets for threat resolution. The outputs of these filters (StmDisplayStruct described in §2.2.5.6.1.1) may be used to successfully provide target bearing estimates for intruder display.

The bearing filter **shall** (1470) produce new estimates of target bearing once each surveillance update interval.

2.2.4.6.6.4.1 Bearing Filter

The bearing filter **shall** (1471) produce continuous, smoothed target bearing estimates for ADS-B targets, Mode S targets or Mode C targets containing one or more ungarbled pulses. The estimates **shall** (1472) have an RMS error of less than five degrees and a peak error of less than ± 15 degrees from five seconds after valid bearing measurements begin until valid measurements terminate, when interference, multipath and systematic antenna errors are not present and when the actual target bearing rate does not exceed one degree per second. Valid bearing measurements for this requirement are derived from Mode C replies using up to seven ungarbled pulses, Mode S replies for which the address is correctly decoded, or ADS-B position reports meeting minimum quality standards as stated in §2.2.4.6.4.2.2.1.1.

The bearing filter **shall** (1473) produce target bearing estimates that lag valid reply bearing measurements by no more than five seconds for steady-state bearing rates of three degrees per second.

Note: The maximum constant bearing rate of practical significance is three degrees per second, due to ownship heading changes during standard turns. Higher short-term rates can occur during the lateral passage of an aircraft. When an aircraft flies by at close range, its bearing changes by nearly 180 degrees within a few seconds. The resulting angular acceleration is difficult to track continuously with a bearing tracker alone. For this reason, it is recommended that the algorithm for predicting bearing be based on Cartesian position estimates rather than polar position estimates.

Algorithm PredictCartesianTracker of the ACAS X ADD describes a Cartesian Kalman filter estimation algorithm that has been used successfully to develop smoothed target bearing estimates.

2.2.4.6.6.5 Dynamic Range and Reply Frequency

The ACAS X bearing estimation function **shall** (1474) meet the requirements of §2.2.4.6.6.2.1 and §2.2.4.6.6.2.2 for input signal levels between MTL and -21 dBm and input signal frequencies between 1087 MHz and 1093 MHz.

2.2.4.7 Antenna System

The equipment **shall** (1475) transmit interrogations and receive replies from top mounted and bottom mounted antennas.

Note: An ACAS X unit and Mode S transponder may share a single pair of antennas.

2.2.4.7.1 Polarization

The antennas **shall** (1476) be vertically polarized.

2.2.4.7.2 Radiation Pattern

2.2.4.7.2.1 Transmit Radiation Pattern

The ACAS X equipment **shall** (1477) be capable of providing directional interrogations from top-mounted antennas for surveillance of Mode C targets in densities up to 0.3 aircraft per sq. NM. An omnidirectional bottom transmit antenna is sufficient for ACAS X since most of the interrogations are transmitted from the top antenna in order to reduce susceptibility to multipath interference from the ground. The use of a bottom directional transmit antenna and the use of directional interrogations for surveillance of Mode S targets and for transmission of TCAS Broadcast interrogations is optional.

The directional interrogation antenna **shall** (1478) generate an azimuth beam that is sequentially positioned to provide adequate surveillance coverage over 360 degrees azimuth. To ensure adequate coverage, the directional antenna 3 dB beamwidth in azimuth **shall** (1479) not be less than the separation between adjacent azimuth beam positions. Beamwidth control is further provided by the P2 suppression pulse as defined in §2.2.4.5.4.2.1. Also, the shapes of the antenna azimuth patterns **shall** (1480) be controlled such that a minimum-suppression transponder, defined as one that replies when the received ratio of P1 to P2 exceeds 0 dB, will reply to interrogations from no more than two

adjacent directional beams. The requirements of this paragraph **shall** (1482) apply for each elevation angle between +20 degrees and -15 degrees.

The shape of the elevation pattern at the azimuth peak-of-beam of each directional beam should match the shape of the elevation pattern of a matched quarter-wave stub within ± 1 dB over 90% of the region from -15 degrees to +20 degrees in elevation when installed at the center of a 1.2 m (4 ft) diameter (or larger) circular ground plane that can be either flat or cylindrical.

The gain of an omni-directional transmit pattern (if employed) should not be less than the gain of a matched quarter-wave stub minus one dB over 90% of a coverage volume from 0 to 360 degrees in azimuth and -15 to +20 degrees in elevation when installed at the center of a 1.2 m (4 ft) diameter (or larger) circular ground plane that can be either flat or cylindrical.

If the antenna gain is not as specified above, the transmitted power **shall** (1483) be adjusted to satisfy the power requirements specified in §2.2.3.1 and §2.2.4.5.4.2.2.

2.2.4.7.2.2 Receive Radiation Pattern

An ACAS X that employs an omni-directional receive pattern **shall** (1484) meet all of the gain requirements specified in §2.2.4.7.2.1 for an omni-directional transmit pattern.

An ACAS X that employs a directional receive pattern consisting of four simultaneous directional beams **shall** (1485) meet the following pattern requirements when installed at the center of a 1.2 m (4 ft) diameter (or larger) circular ground plane that can be either flat or cylindrical. The shape of the elevation pattern at the azimuth peak-of-beam of each directional beam **shall** (1486) match the shape of the elevation pattern of a matched quarter wave stub within ± 1 dB over 90% of the region from -15 degrees to +20 degrees in elevation. The gain at the crossover points between adjacent directional beams **shall** (1487) not be less than the adjacent peak-of-beam gain minus 4 dB from -15 degrees to +20 degrees in elevation.

An ACAS X that employs a receive pattern consisting of more than four simultaneous directional beams **shall** (1488) meet the elevation pattern requirement stated above for the four-beam antenna. The gain, relative to peak-of-beam, at the crossover points between adjacent directional beams of this antenna **shall** (1489) match, within 1 dB, the gain required to account for the maximum closing speed that occurs in the direction of the crossover point.

If the antenna gain is not as specified above, the nominal receiver MTL **shall** (1490) be adjusted to account for the antenna gain.

Note: For example, for a relative peak-of-beam antenna gain of -3 dB, the MTL value associated with a reply received via this beam would be lowered by 3 dB.

2.2.4.7.3

Use of a Directional Antenna for Mode S Interrogations

Availability of bearing information will allow Mode S interrogations to be made using a directional antenna. If a directional antenna is used for transmission of Mode S interrogations, the transmit radiation pattern **shall** (1491) be as specified in §2.2.4.7.2.1 with the exception that side-lobe suppression is not used.

2.2.4.7.4 Antenna Selection

2.2.4.7.4.1 Squitter Listening

The equipment **shall** (1492) monitor squitters via top and bottom antennas that are capable of simultaneous reception over 360 degrees of azimuth. If reception is switched, the switching times **shall** (1493) be controlled to avoid undesirable synchronism with the squitters transmitted by Mode S diversity transponders.

Note: This can take the form of simultaneous reception using two receivers and two decoders or switched reception using a single receiver. Mode S transponders alternate the antennas used for squitter transmissions at nominal 1-second intervals. It is acceptable for ACAS X to switch antennas for squitter monitoring after two successive surveillance update periods.

2.2.4.7.4.2 Interrogations and Replies

The equipment **shall** (1494) transmit each Mode C or Mode S interrogation via one or the other of two antennas. Interrogations **shall** (1495) not be transmitted simultaneously via both antennas. Replies **shall** (1496) be received from the same antenna that was used to transmit the interrogation.

2.2.4.8 Relationship Between Front-End Surveillance and the STM

This section provides an overview of the relationship between Front End Surveillance and the STM.

Front End Surveillance includes those portions of surveillance that satisfy the non-ADD surveillance requirements of these MOPS. Examples of this include:

- Hybrid Surveillance
- Mode S Interrogation Rate determination
- Mode C whisper-shout sequence determination.
- DF-11 squitter monitoring and Mode S acquisition
- ADS-B report generation

Front End provides active and passive “observations” to the STM. The STM processes these observations into tracked data which is provided to the TRM. Front End Surveillance requires track information to meet its objectives. For example, range and range rate are required to determine the Mode S interrogation interval (Normal, Reduced, or Hybrid). An equipment manufacturer may decide to implement an independent (from STM) Front End Surveillance tracking system providing range and range rate for its own use. Alternatively, the tracked range and range rate information contained within the STM for a particular Mode S track could be used. The STM design presented in these MOPS does not define an interface for internal STM data to be transferred to Front End Surveillance. This does not preclude an equipment manufacturer from implementing such an interface.

Front End Surveillance requirements are provided as textual requirements so that an equipment manufacturer is afforded the flexibility to design a system that takes into account a specific system attributes. For example, a system may or may not provide a bearing estimate for monitored DF-11 short acquisition squitters. The presence or absence of DF-11 short acquisition squitter bearing estimate has a significant impact on a system’s

Mode S acquisition design. Therefore, active Mode S track acquisition was retained as high level textual requirements instead of lower level prescriptive algorithms in the ADD.

2.2.5

Surveillance and Tracking Module / Threat Resolution Module

This section includes:

- An overview of the STM and TRM
- General integration requirements related to order of execution
- Integration requirements related to spatial correlation requirements associated with suggested algorithms are provided.
- Combined STM/TRM input interface requirements
- Combined STM/TRM output interface requirements

The STM and TRM algorithms are described in Volume II.

The equipment **shall** (1514) include collision avoidance algorithms that meet the requirements specified in the ADD (Volume II).

2.2.5.1

STM – Surveillance and Tracking Module Overview

This section provides a high level description of the STM. For a detailed description refer to Chapter 2 of the ADD (Volume II).

The STM processes all surveillance and ownship data and provides it in the form required by the TRM. Specifically the STM generates the following outputs:

- Own and intruder data in the format required by the TRM including RA coordination data (Vertical Resolution Complement – VRC).
- Intruder information required for the display (but not including whether ownship has issued an RA or TA against the intruder).
- Mode information used to communicate the system's Sensitivity Level (SL), RA capability (RI), and other fields required to populate the Data Link Capability Report (DLCR).

2.2.5.1.1

STM Tracking

The STM makes use of active surveillance data, ADS-B data, and ADS-R data. The STM maintains independent tracks on each source. These tracks are correlated and associated to targets. Correlation requirements are discussed in §2.2.5.4. The STM selects one track source (active, ADS-B, or ADS-R) for use by the TRM.

The STM may reject inputs that would normally be associated with a track through an outlier rejection process.

The STM also marks tracks as Proximate or Other (Non-Threat).

2.2.5.1.2

STM Track Source Selection

The STM marks each track with information allowing the TRM to determine if the track can be used for generating RAs and TAs or TAs only.

ADS-B data is marked as qualifying for RA generation if it passes an STM active validation which is different and in addition to the front end hybrid surveillance validation. In general, once STM active validation fails on an ADS-B track, that track will no longer be provided to the TRM for its duration.

Table 2-20 indicates the track type and the protection capability passed to the TRM and Display as a function of the type active surveillance, passive surveillance, and the STM active validation state.

Table 2-20: Summary of STM Outputs for Intruder Surveillance Inputs

ID	Active Surveillance	Passive Surveillance	Active Validation State	TRM Protection	Track Passed to TRM & Display
1	Mode S NORMAL	None or ADS-R	N/A	TA/RA	Active
2	Mode S REDUCED	None or ADS-R	N/A	None	Active
3	Mode S NORMAL	High Quality ADS-B	Passes	TA/RA	Passive
4	Mode S NORMAL	High Quality ADS-B	Fails	TA/RA	Active
5	Mode S NORMAL	Low Quality ADS-B	Passes or Fails	TA/RA	Active
6	Mode S REDUCED	High Quality ADS-B	Passes	None	Passive
7	Mode S REDUCED	High Quality ADS-B	Fails	None	Active
8	Mode S REDUCED	Low Quality ADS-B	Passes or Fails	None	Active
9	Mode C	None, ADS-B or ADS-R	N/A	TA/RA	Active
10	Mode S HS Validation	High or Low Quality ADS-B	N/A	None	Passive
11	None	High Quality ADS-B or ADS-R	N/A	None (see Note 1)	Passive
12	None	Low Quality ADS-B or ADS-R	N/A	None	None

Note 1: TA Only protection will be provided if ADS-B only TA option is enabled per the ReceiveDiscretes optional parameter as described in §2.2.5.5.5.

When ownship heading is degraded, traffic bearing data provided to the display will be sourced from the traffic's active track, if available. Otherwise no bearing will be provided for the traffic.

Note that a passively tracked intruder correlated with an active track that is in the extended hybrid or hybrid surveillance region should not qualify for TA protection. However, when the optional ADS-B only TA only capability is implemented, there is nothing in the ADD

code to specifically prohibit TAs on ADS-B only tracks in these regions. But analysis shows that TAs are not expected in those regions, regardless (see (Appendix C, §C.3.4)).

In this table, high quality ADS-B or ADS-R is determined by meeting the minimum values required for use of ADS-B for RA protection (NIC \geq 6, NACp \geq 7, NACv=1, SIL=3, version=2).

2.2.5.1.3 STM Belief State Outputs for the TRM

The STM provides information on intruders and ownship with a set of weighted samples which represents the uncertainty of the intruder and ownship position data. This set of samples is referred to as “belief states”. Each intruder is described with a weighted set of horizontal and vertical belief states. Ownship is defined with a set of vertical belief states.

The Intruder vertical and Own vertical belief states include an estimate of altitude and altitude rate. Five “weighted” samples are provided. One “middle” sample ($[z, dz]$) with probability or weight of 1/3 and four samples with weight of 1/6 ($[z, dz+\sigma dz], [z, dz-\sigma dz], [z+\sigma z, dz], [z-\sigma z, dz]$) are provided. These five vertical sigma samples (belief states) represent the vertical or altitude distribution.

The intruder horizontal belief state includes relative x and y and rates of x and y, (dx, dy). Nine “weighted” samples are provided. One middle sample ($[x, y, dx, dy]$) with weight of 1/3 and eight additional samples on either side of the middle sample with weight of 1/12 are provided.

2.2.5.2 TRM - Threat Resolution Module Overview

The Threat Resolution Module (TRM) uses the track provided by the STM to determine if the traffic is a threat. If so, the TRM will determine the TA and RA information which should be presented to the flight crew, the RA coordination data which should be provided to other collision avoidance systems on intruder aircraft, and the RA for transmission to the ground or other aircraft (via the transponder or directly broadcast by the system).

The main blocks of the TRM are shown in Figure 2-13 and Figure 2-14. This section provides a high level description of the TRM. For detailed understanding of the TRM, refer to Chapter 3 of the ADD (Volume II).

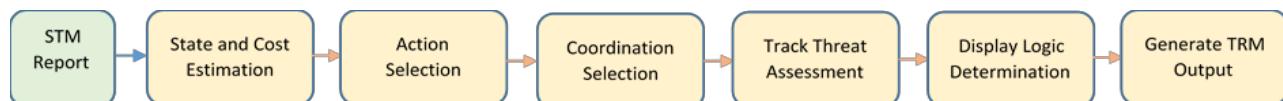


Figure 2-13: The Main Blocks of TRM

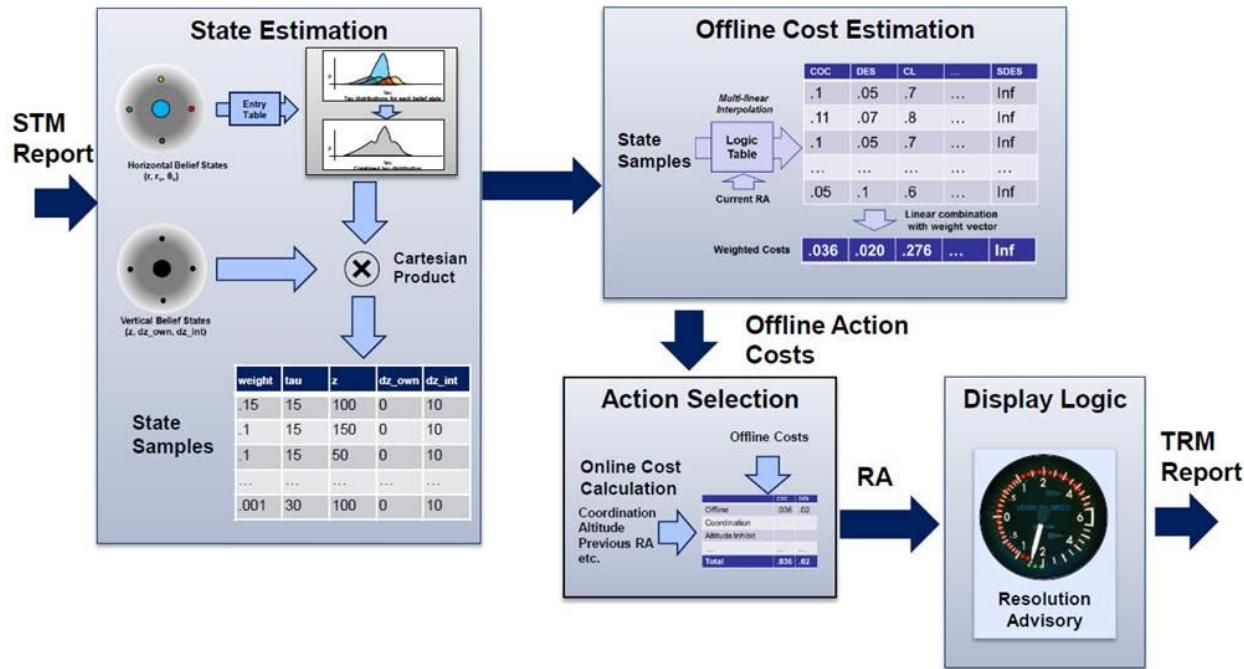


Figure 2-14: State Estimation

The TRM assigns costs to actions, where actions represent RAs (e.g. Climb, Descend, etc.), Clear of Conflict, or no advisory. The lowest “cost” action is selected. These costs are also used to determine if a TA should be generated.

Ownership related inputs used by the TRM include: radio altitude, heading, ICAO 24-bit Aircraft Address, and equipage states. The estimated vertical state of ownership is represented by a collection of own vertical beliefs.

Intruder related inputs used by the TRM include: reported values for ICAO 24-bit Aircraft Address, altitude quantization, TCAS equipage, and the received Vertical Resolution Advisory Complement (VRC). Designation information input to the TRM includes designated mode, protection mode, designated no alerts (DNA) flag, Xo validity flags, and designation status. Surveillance information for the intruder is provided by the surveillance source and flag settings for surveillance state. The estimated vertical state of the intruder is provided as a collection of intruder vertical beliefs. Likewise, the estimated relative horizontal state of the intruder is provided as a collection of intruder horizontal beliefs.

The STM display output is provided to the TRM for inclusion of RA threat identity information used in RA reports of non-ICAO intruders. The STM display output contains the intruder state data that is provided to the display and is part of the STM Report. It includes the vertical rate arrow for the display, indication that the intruder is reporting altitude, the intruder ID, relative altitude, tracked ground range, bearing relative to own airframe, and the intruder ICAO 24-bit Aircraft Address.

2.2.5.2.1 State Estimation

State estimation produces an estimate of the time to closest approach (τ) and combines the vertical information from intruder and ownship (combined samples). τ is output as a discrete probability distribution between 0 and 40 (values greater than 40 are set to 40). Depending on geometric considerations, the logic can use a horizontal τ distribution, a vertical τ distribution or a combination of both in cost estimation. Generally, the horizontal τ distribution alone is selected except in cases with a slow horizontal closure rate and a high vertical convergence rate.

The horizontal belief states are converted to the three-dimensional system based on ground range (r), ground range rate (s) and angle between intruder bearing and the relative velocity vector (ϕ) as illustrated in Figure 2-15. This compact representation exploits the rotational symmetry of the problem. A horizontal entry table indexed in an r , s , and ϕ coordinate system transforms horizontal state to a distribution of τ (time to loss of separation). This table contains pre-calculated horizontal τ distributions (probability that τ is 1 to 40s or more) for all relevant combinations of r , s , and ϕ .

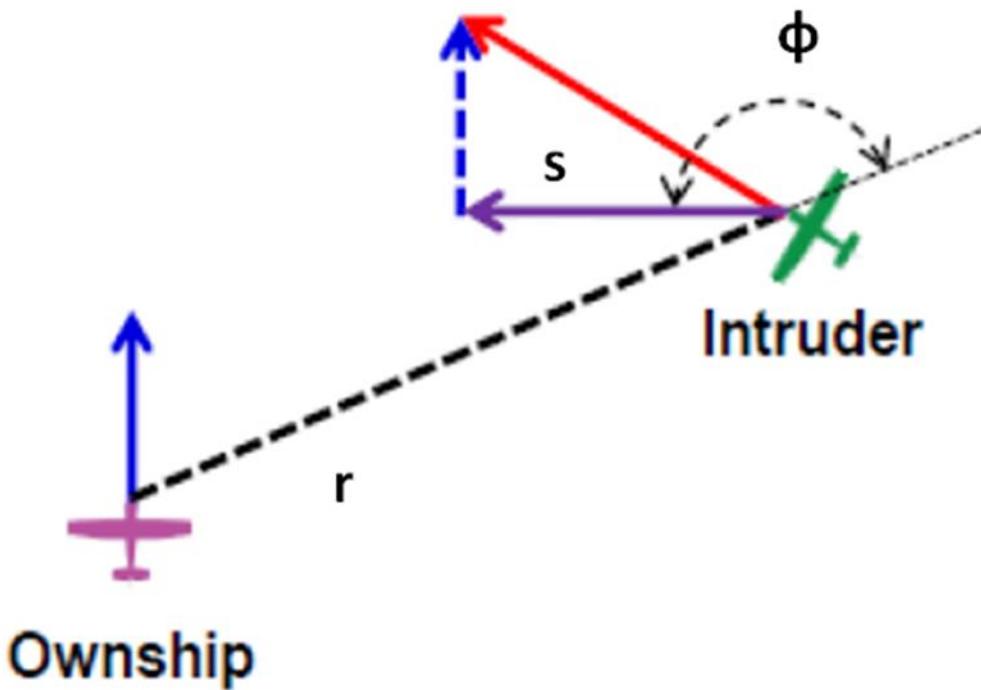


Figure 2-15: Transformation of Inputs

The STM provides nine horizontal beliefs corresponding to the mean and uncertainty for each of the four dimensions. These beliefs are weighted according to likelihood. The individual τ distributions for each horizontal belief are combined according to the belief weight to form a single τ distribution for each intruder, as shown in Figure 2-16. These distributions are discrete, with integer values for τ .

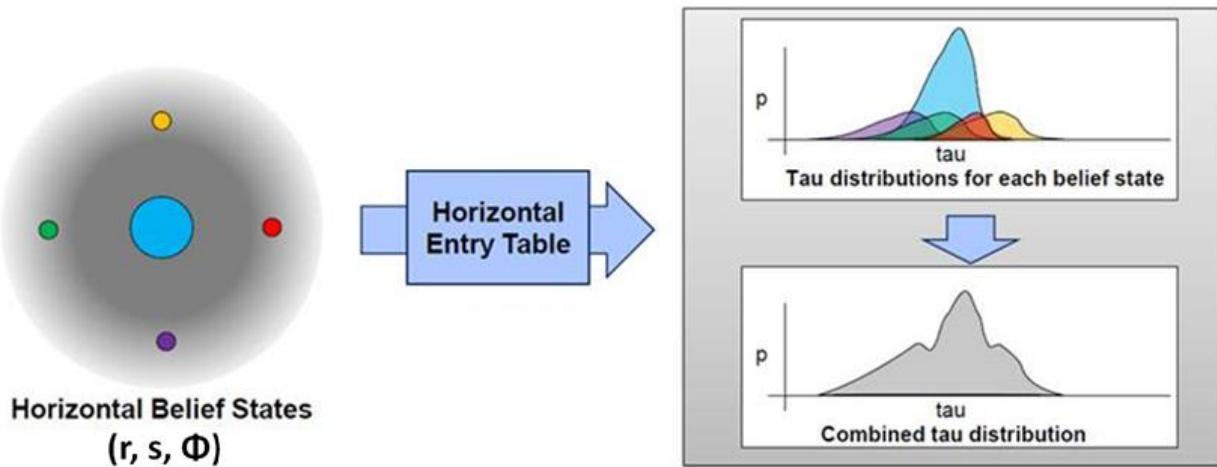


Figure 2-16: Estimation of Tau Distribution – Horizontal Entry Table

The pre-calculated vertical tau distribution (probability that tau is 1 to 40s or more) is stored in the vertical entry table. The vertical entry table is a 3-dimensional table indexed by relative altitude (dh), own altitude rate (dz_own), and intruder altitude rate (dz_int) and contains tau distributions for all relevant combinations of these states. As vertical belief states are provided in this format, no additional coordinate transformation is required. The STM provides five vertical beliefs for the intruder corresponding to the mean and uncertainty for each of the two vertical dimensions. The STM provides a single belief for the ownership corresponding to the mean. When necessary, the individual vertical beliefs are looked up in the vertical entry table to determine the tau distributions and then combined based on the weights to a single tau distribution.

The combined samples are obtained using the vertical information from ownership and intruder. While the intruder has five samples corresponding to the mean and uncertainty of the two dimensions, the ownership states use a single sample corresponding to the mean. Five combined samples containing the relative altitude, own vertical rate, and intruder vertical rate are produced for use in cost estimation.

2.2.5.2.2 Cost Estimation and Update of Intruder Coordination Data

Cost estimation is performed in two parts: Off-line cost estimation and On-line cost estimation. The outputs of both are fused.

Received intruder coordination data (VRC) is updated right before Online Cost estimation. This ensures that the most recently received coordination data is utilized in the Online Cost estimation. This is the latest time that the VRC for each intruder retrieved from the STM can be received by the TRM for processing. Subsequent threat processing requires intruder VRC updates for individual response estimations, online cost estimation which takes into account the coordination information, and action selection in multiple conflict scenarios.

2.2.5.2.2.1 Offline Cost Estimation

Offline Cost outputs a vector containing offline costs for all possible actions for each intruder. These costs are derived using lookup tables that were generated using an offline optimizaton process. These tables are indexed by tau, relative altitude, ownship vertical rate, intruder vertical rate, and the current RA. All combined vertical belief and tau sample pairs are used to index into the table and provide a cost vector, as shown in Figure 2-17. A weight for each pair is computed as the product of the tau weight and combined sample weight.

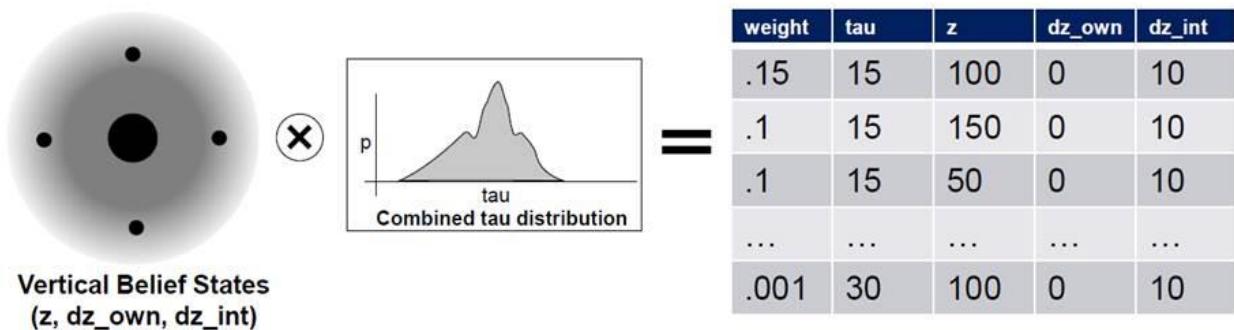


Figure 2-17: Sample Combining

The cost estimation process used by the TRM is illustrated by Figure 2-18 below. The offline cost for each action is formed by determining costs for each state sample using multi-linear interpolation (as in the tau estimation process). Total costs are formed by linear combination with all weighted cost vectors.

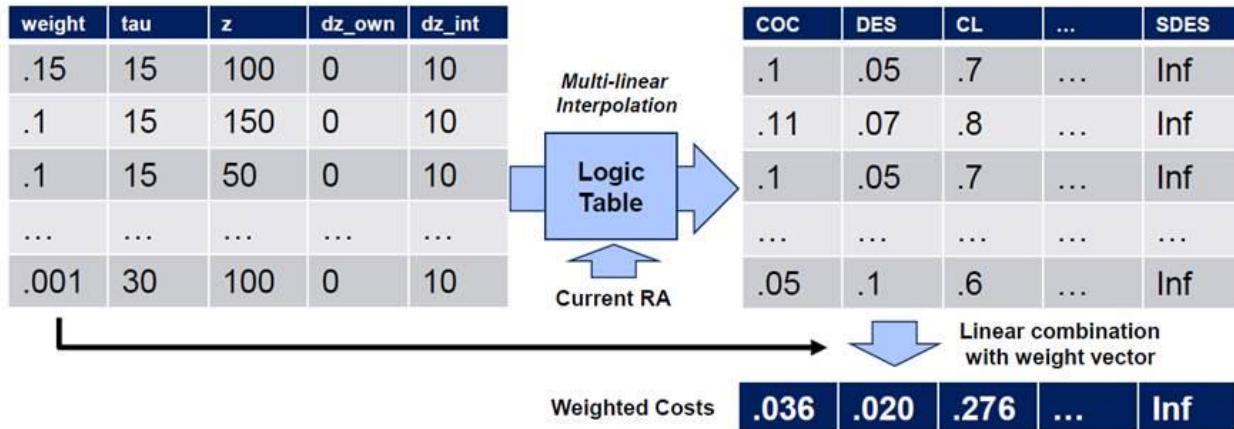


Figure 2-18: Offline Cost Estimation – Cost Table Look-up

2.2.5.2.2.2 Online Cost Estimation

Online costs capture characteristics of the current state, such as altitudes, that cannot be suitably reflected in the offline cost tables. There are two types of online costs – one is independent and the other is dependent on the RA coordination state of ownship and the intruder aircraft.

The outputs are two vectors, each containing the sum of online costs associated with each action for an individual intruder. One vector is used for determining the online costs for RAs, the other for TAs.

Examples of coordination independent online costs include: altitude inhibit cost, advisory restart cost, and prevent early COC (clear of conflict) cost. Coordination dependent online costs include: crossing no alert cost, max reversal cost, prevent early weakening cost, and compatibility cost.

2.2.5.2.2.3 Individual Cost Estimation

The last action in the *State and Cost Estimation* block is fusing the offline and online costs into final TA (to determine TAs) and RA (to select RA) costs.

2.2.5.2.3 Action Selection

Action selection accepts a set of action costs from each of the intruders. A global action for ownership is determined along with individual actions for each intruder considered independently. The global action is converted into final climb or descend rates and the advisory that the system issues. Action selection is performed separately for single threat intruder and multiple threat encounters. A list of possible actions is provided in Table F-2 in Volume II.

2.2.5.2.4 Coordination Selection

Coordination selection block returns the coordination message to be sent to the Active CAS-equipped intruder. This includes a coordination message to cancel the previously sent VRC for dropped intruders.

2.2.5.2.5 Track Threat Assessment

Track threat assessment block assigns each intruder with a code indicating Non Threat, Proximity Advisory (PA), Traffic Advisory (TA), or Resolution Advisory (RA). A Track Display Score to prioritize intruders for the display is generated.

The TA logic serves to help pilots visually acquire intruding traffic and to prepare pilots to respond to a potential RA. The TA logic utilizes the offline cost table to determine if a target is a potential threat. The TA logic triggers an alert based on optimized thresholds that assess the cost of *No Alert* (i.e. the cost of COC) and the difference between the cost of *No Alert* and whichever other RA has the closest cost. In addition to utilizing the offline cost table, the TA logic also applies certain online costs to increase or decrease alerting sensitivity. Using TA-specific parameters, the applicable online costs provide the TA logic with knowledge of the potential issuance of an RA.

2.2.5.2.6 Display Logic Determination

Display Logic Determination block converts the action and rates for the current advisory into parameters that drive the RA display, including the visual indicators and aural alarms. It also indicates crossing type RAs.

2.2.5.2.7 Generate TRM Output

The last block, *Generate TRM Output*, processes the individual intruders with respect to the operational mode (Global TA-Only or Global TA/RA), Xo designation and degraded surveillance quality. It also processes dropped tracks and invalid intruders. The RA information for broadcast and downlink (air-ground and air-air transmissions) is generated.

2.2.5.2.8 ACAS X TRM Processing Modes

The ACAS X TRM has several modes for processing individual intruder inputs. The TRM intruder processing mode is selected for an individual intruder based on ACAS X operational mode, ACAS Xo mode and degraded surveillance.

Table 2-21 provides an overview of how the TRM intruder processing is modified for each type of TRM output based on the ACAS X operational mode, ACAS Xo mode, and surveillance quality.

Three broad categories for intruder processing are available: TA/RA, TA-Only, and Degraded Surveillance. TA/RA and TA-Only are ACAS X operational modes, selectable by the flight crew (“Flight Crew”) and also selectable by the system based on other conditions such as ownship altitude (“Internal”). Degraded Surveillance is selectable only by the system, and is used when the surveillance quality of individual traffic is inadequate to produce RAs on that traffic.

Each of these intruder processing categories has subcategories for ACAS Xa and the available ACAS Xo modes. There is no CSPO-3000 subcategory for TA-only processing because CSPO-3000 is not available when ACAS X cannot provide RAs.

The “TRM Outputs” provide information on how each intruder is processed by the TRM for each of its outputs:

“Individual RA” indicates the logic mode used to determine RAs for the individual intruder. Xo DNA mode uses the Xa logic, but since DNA suppresses alerts, the RA state for the individual intruder is reset each processing cycle. TA-only and Degraded Surveillance modes also suppress RAs, so the RA state for the individual intruder is also reset each processing cycle.

“Global RA” indicates adjustments to the processing used to produce the RA presented to the flight crew. In order to establish whether there are multiple threats, an intruder designated to DNA is used when determining the global RA. If the only threat is designated to DNA or ownship is in TA-only mode, no RAs are presented to the flight crew. In those cases, the global RA state is reset as well as the RA state for the individual intruder. When the intruder has Degraded Surveillance it is not used to produce the RA presented to the flight crew (i.e., it is “excluded from Global RA processing”). Because the intruder with Degraded Surveillance was not used to produce the Global RA, the global RA state does not need to be reset.

“Coordination” indicates adjustments to the coordination messages sent to the individual intruder for each processing category. Coordination is delayed when the threat is designated to DNA and is a TCAS-equipped slave to allow the designated aircraft to choose the RA sense. Coordination is also delayed following transition of the ACAS X operational mode from TA-Only to TA/RA. Otherwise, coordination is not delayed. Coordination messages are cancelled when switching from TA/RA to TA-Only operational mode and for Degraded Surveillance. Coordination messages are not sent in most cases when no RA will be displayed to ownship’s flight crew.

“Individual Display” indicates the traffic display symbol. When the intruder is designated to DNA, no alerts are provided to the flight crew, but the traffic is shown as designated. In

TA-Only and Degraded Surveillance modes (RAs are suppressed), RAs are displayed as TAs.

“Global Display” indicates the alert displayed to the flight crew. It has two subcategories. The first is for an individual threat, and the second for multi-threat situations. In both cases, the designated intruder is a threat. During multi-threat situations, advisories for an intruder designated to DNA are not suppressed. In TA-Only and Degraded Surveillance processing modes, RAs are displayed as TAs, even when the intruder is designated to DNA.

“ARA” indicates modifications to the Active RA subfield used for broadcasting RA information. It has two subcategories matching the Global Display (above), and broadcasts RA information reflecting the Global Display.

Table 2-21: ACAS X TRM Processing Modes

TRM Output	Category for Processing Intruder						
	TA/RA (Flight Crew or Internal)			Ownship TA-Only (Flight Crew or Internal)		Degraded Surveillance ¹ (Internal)	
	Xa	Xo CSPO-3000	Xo DNA	Xa ²	Xo DNA	Xa or Xo CSPO-3000	Xo DNA (Designation)
Individual RA	Xa processing	Xo CSPO-3000 processing	Xa processing then reset RA state ³	Xa processing then reset RA state ⁴	Xa processing then reset RA state ⁴	Xa processing or Xo CSPO-3000 processing then reset RA state ⁵	Xa processing then reset RA state ⁵
Global RA	Normal	Normal	Normal then reset RA state ³	Normal then reset RA state ⁴	Normal then reset RA state ⁴	Exclude from Global RA processing	Exclude from Global RA processing
Coordination	Normal	Normal	See note 8 No delay following undesignation	None sent for individual except CANCEL Delay after switch to TA/RA ⁴	None sent for individual except CANCEL Delay after switch to TA/RA ⁴	None sent for individual except CANCEL Delay after switch to TA/RA ⁵	None sent for individual except CANCEL Delay after switch to TA/RA ⁵
Individual Display	Normal	Normal	TA and RA → Designated with no alert	RA → TA	TA and RA → Designated with no alert	RA → TA	TA and RA → Designated with no alert
Global Display Only this threat	Normal	Normal	No TA or RA	RA → TA	No TA or RA	RA → TA	No TA or RA
Global Display Other threats	Normal	Normal	Normal	RA → TA	RA → TA	Normal ⁶	Normal ⁶
ARA Only this threat	Normal	Normal	RAT ⁷ or no RA	RAT ⁷ or no RA	RAT ⁷ or no RA	RAT ⁷ or no RA	RAT ⁷ or no RA
ARA Other threats	Normal	Normal	Normal	RAT ⁷ or no RA	RAT ⁷ or no RA	Normal ⁶	Normal ⁶

Notes:

1. *If intruder surveillance is sufficiently degraded that it is flagged as invalid for TRM processing:*
 - *There is no TRM processing.*
 - *The RA state is fully reset (initialization delay is re-activated).*
 - *Any ongoing coordination is terminated by sending a CANCEL message.*
 - *A pre-existing TA is maintained until timeout.*
 - *Designation is maintained until timeout.*
 - *If this intruder is the only intruder, the global display is cleared and the ARA output is “RAT or no RA”.*

2. *Xo CSPO-3000 is not valid for designation when ownship is in TA-Only operational mode.*
3. *Neither coordination delay cost counter nor initialization cost counter are reset.*
4. *The coordination delay cost counter is reset when the ACAS X operational mode transitions from TA-Only to TA/RA; otherwise, it is not reset.*

The initialization cost counter is not reset (no added initialization delay).

5. *The coordination delay cost counter is reset when the ACAS X operational mode transitions from TA-Only to TA/RA; otherwise, it is not reset.*

The initialization cost counter is reset when surveillance returns to nominal and is no longer degraded, otherwise, it is not reset.

6. *The intruder with degraded surveillance is excluded from the global RA which is presented to the flight crew.*
7. *RAT is “RA Terminated”. As used here, it means RAT=1 and the information about the RA from the previous cycle is output in the ARA fields.*
8. *Coordination is dependent on the equipage of the intruder and whether the intruder is a master or slave to ownship as follows:*

- *If the DNA-designated intruder is known to be ACAS X-equipped:*
No coordination; a CANCEL message is sent to terminate any coordination messages previously sent by ownship.
- *If the DNA-designated intruder is TCAS-equipped and is a master to ownship:*
No coordination; a CANCEL message is sent to terminate any coordination messages previously sent by ownship.
- *If the DNA-designated intruder is TCAS-equipped and is a slave to ownship:*
If a VRC is received from the intruder and there is an RA for that intruder a VRC will be sent to the slave.

Otherwise, no coordination; a CANCEL message is sent to terminate any coordination messages previously sent by ownship.

2.2.5.3 General Integration Requirements

The equipment **shall** (2159) invoke the following ADD functions every second in the order given below:

- GenerateSTMReport (STM)
- VerticalTRMUpdate (TRM)
- STMHousekeeping

Prior to execution of GenerateSTMReport, the system provides the required inputs as defined in §2.2.5.5. GenerateSTMReport **shall** (2607) complete execution without invocation of STM receive functions which create tracks. Furthermore, the system extracts all the outputs defined in §2.2.5.6 while satisfying the timing requirement specified in §2.2.4.2 “System Delay”.

In the case of a system failure, as determined by the monitor (i.e. “opflg” ReceiveDiscretes input set to FALSE), ownership inputs **shall** (2161) be provided to the STM; however, intruder inputs **shall** (2162) only be provided when the system is operational (“opflg” discrete input set to TRUE).

An equipment manufacturer may implement an interface to the STM which prevents a track from going to the TRM in cases where the STM continues to maintain a track but the track should not go to the TRM for CAS processing (e.g. when an intruder transitions to the ground).

2.2.5.3.1 GenerateSTMReport

ACAS X **shall** (1993) generate an STM Report after all inputs have been provided to the STM for a given surveillance update interval. This requirement implies that the STM Report will be generated at a frequency of 1 Hz since the ACAS X surveillance update interval is one second in length.

The ReportTime provided to the STM **shall** (2160) be greater than or equal to the toa of the most recent observation provided through the interfaces of §2.2.5.5 and less than the end of the current surveillance interval.

ACAS X **shall** (1994) generate an STM Report every surveillance update interval, even in the absence of any STM inputs or during a failed or standby condition during that surveillance update interval. This is required so that the SL and RI produced by the STM are updated appropriately.

2.2.5.3.2 VerticalTRMUpdate

ACAS X **shall** (1996) generate a TRM Report every surveillance update interval by invoking VerticalTRMUpdate, even in the absence of any STM inputs during that surveillance update interval.

2.2.5.4 STM / Surveillance Correlation Requirements

The purpose of this section is to provide requirements for correlation processing. The ADD CorrelationProcessing algorithm calls three algorithms that provide guidance with respect to implementation of these requirements. These algorithms together with the algorithms associated with ReceiveModeCReplies have been shown to meet the correlation requirements with one set of manufacturer supplied data. These algorithms are suggestive algorithms (See Volume II, §2.2.1.2 for further discussion).

The equipment manufacturer must ensure the implemented design (whether using the suggestive algorithms or other design) satisfies the requirements of this section with collected data.

CorrelationProcessing assumes that all ICAO 24-bit Aircraft Addressed STM track files have been associated to the correct target. The main purpose of CorrelationProcessing is to perform spatial correlation / decorrelation of all STM track files prior to execution of GenerateSTMReport so that all targets represent unique aircraft.

ICAO 24-bit Aircraft Addressed correlation is prescriptively specified by the ADD, but outside of the algorithm CorrelationProcessing.

After execution of CorrelationProcessing the STM target data base (target_db) must be configured so that the intruders included in the STMReport meet the spatial correlation requirements of this section.

Consideration should be given when merging two STM targets to properly handle data associated with the overall target (not any particular target track, but fields in the target header). Guidance and minimum requirements are provided below.

- The target data base entry of the oldest target or the target that would qualify for an RA (targets with ModeC or ModeS tracks) **shall** (2600) be kept. This maintains continuity of the target id. Maintaining the same target id during an encounter is desirable to prevent alerts from being reset, as the TRM uses target id to determine if a track is new.
- For targets with a ModeS track the av_history, av_state fields, Coord_data, and Bad_UF16UDS30 fields associated with the ModeS track **shall** (2601) be retained.
- For targets with ADS-B or ADS-R tracks the adsb_qual_history, adsb_qual_overide, passive_only_adsb_qual_history, daa, and ca_operational fields **shall** (2602) be maintained.

When decorrelating a spatially correlated track, one of the tracks is removed from the existing target and a new target entry is created. The track that is decorrelated should normally be the track that would not qualify to generate an RA.

Correlation Pairs Which Impact RA Eligible Tracks

The false track and tracking reliability test requirements in §3.4.4.1 and §3.4.4.2, along with the surveillance objectives in §2.2.2.2.1.1 and §2.2.2.2.1.2, define the correlation performance of the following track type pairs:

- ATCRBS / ATCRBS
- ATCRBS / Mode S

Note: It has been shown that Mode C tracks may be formed using replies generated by Mode S transponders.

- Mode S / Mode S
- ADS-B / Mode S

Spatial Correlation Pairs Which Do Not Impact RA Eligible Tracks

Spatial correlation is required for the track pair types specified below:

- ADS-R ICAO / ATCRBS (Real or Image)
- ADS-R Non-ICAO / ATCRBS (Real or Image)
- ADS-R Non-ICAO / MODE S
- ADS-R Non-ICAO / ADS-B
- ADS-B / ATCRBS (Real or Image)

Miscorrelation of the track type pairs above will not impact RA performance as the Active CAS track (ATCRBS or Mode S) will always be the selected track. ADS-R tracks cannot be used for RA determination. These restrictions may change in future versions of ACAS X.

For the Spatial Correlation Pairs which do not impact RA eligible tracks, the correlation performance **shall** (2163) meet the associated correlation performance of RTCA/DO-317B (Ref. T) §2.2.3.2.3 or alternatively the objectives below in the area of interest as defined by the Mode S surveillance flight test requirements:

- Greater than 95% tracking reliability
- Less than 2% false track rate

As a goal, and to support future versions of ACAS X, spatial correlation should achieve less than 1.2% false track rate.

2.2.5.5

Combined STM/TRM Input Interfaces

This section specifies the input interface requirements to the combined STM and TRM logic.

ACAS X provides surveillance data to the prescriptively defined algorithms of the STM through multiple entry points (see §2.1 of the ADD [Volume II]). Each entry point corresponds to a unique sensor measurement, discrete ownship input or intruder designation request. The following subsections detail the requirements associated with the various STM entry points.

Each surveillance update interval, surveillance **shall** (1497) provide available ownship and intruder inputs to the STM through the following interfaces:

1. Ownship Barometric Altitude (§2.2.5.5.1)
2. Ownship Radio Altitude (§2.2.5.5.2)
3. Ownship Heading (§2.2.5.5.3)
4. Ownship WGS84 State (§2.2.5.5.4)
5. Ownship Discrete Data (§2.2.5.5.5)
6. Intruder Mode S Reply Data (§2.2.5.5.6)
7. Intruder Mode C Reply Data (§2.2.5.5.7, 2.2.5.5.8)
8. Intruder ADS-B / ADS-R Position Data (§2.2.5.5.9)

-
9. Intruder ADS-B / ADS-R Velocity Data (§2.2.5.5.10)
 10. Intruder ADS-B / ADS-R Mode Status Data (§2.2.5.5.11)
 11. Intruder Designation Data (§2.2.5.5.12)
 12. Intruder Coordination Data (§2.2.5.5.13)
 13. Intruder Capability Information (§2.2.5.5.14)

Surveillance **shall** (1498) provide the STM with ownship data in the chronological time order.

Surveillance **shall** (1499) provide the STM with intruder data (Mode S reply, Mode C reply, coordination data, and ADS-B data) in chronological time order on a per intruder basis.

ACAS X **shall** (1500) provide data to the STM only when a new measurement is available, unless otherwise specified in the subsections that follow. For example, if surveillance fails to elicit a Mode S reply from an intruder during a surveillance cycle, no Mode S reply information for that intruder will be provided to the STM in that surveillance cycle.

2.2.5.5.1

ReceiveBaroAltObservation

Surveillance **shall** (1501) provide the STM with the ownship barometric altitude data defined in Table 2-22 (identical to Table 2-10 of the ADD [Volume II]).

Table 2-22: Barometric Altitude Observation

Data Item	Description
h	own barometric altitude
toa	time of applicability

Additional input requirements include the following:

1. Surveillance **shall** (1502) provide untracked ownship barometric altitude data to the STM.
2. Surveillance **shall** (1503) provide ownship barometric altitude data to the STM at least once in every surveillance update period during which Surveillance has received an ownship barometric altitude input. This requirement ensures a minimum update rate of 1 Hz whenever updated ownship barometric altitude data is available to ACAS X.
3. Surveillance **shall** (1504) declare an invalid ownship barometric altitude source to the Monitor when an updated barometric altitude input is not received by ACAS X within a surveillance update interval.
4. The time of applicability error (from time of receipt by the equipment) for ownship barometric altitude **shall** (1505) be less than 150 msec.
5. The precision of the ownship barometric altitude provided to this function **shall** (1506) be 1 ft or less.

Note: If Quant in ReceiveDiscretes indicates 100 ft altitude resolution then the values of own altitude passed into this routine should be rounded to the nearest 100 ft.

6. If the barometric altitude is not valid or not credible a value of NaN **shall** (2580) be provided for barometric altitude.

Notes:

1. *The STM ownship barometric altitude tracker does not reset automatically after sequential outliers. Any outlier inputs (as determined by comparison to the current STM estimate) are indicated in the STM report as non-credible and may lead the Monitor to declare an ACAS X failure (see §2.2.7.2.5.1).*
2. *Surveillance may manually command the STM to reset its ownship barometric altitude estimate in cases where Surveillance can recover from an anomaly without leading the Monitor to declare an ACAS X failure. This reset command is indicated to STM by providing an ownship barometric altitude input consisting of NaN values for both time of applicability and own barometric altitude fields. The STM ownship barometric altitude estimate will then be reinitialized on the next ownship barometric altitude input with non-NaN values for both time of applicability and own barometric altitude fields. Any intruder surveillance inputs provided to the STM between the ownship barometric altitude estimate reset command (input with all NaN fields) and the reinitialization (input with non-NaN fields) will not be used to update STM intruder tracks. The reset command and subsequent reinitialization input can be provided within the same surveillance cycle (“back-to-back”) to minimize or eliminate intervening intruder surveillance inputs which would be ignored by STM. Local variables within some STM algorithms may assume NaN values if intruder surveillance input is provided to STM between the ownship barometric altitude estimate reset command and the reinitialization input. The Julia-language reference implementations of the STM algorithms handle the potential NaN values of local variables in a safe manner.*

2.2.5.5.2 **ReceiveRadAltObservation**

Surveillance **shall** (1507) provide the STM with the ownship radio altitude data defined in Table 2-23 (identical to Table 2-11 of the ADD [Volume II]).

Table 2-23: Radio Altitude Observation

Data Item	Description
h	own radio altitude

Additional input requirements include the following:

1. The minimum update rate for ownship radio altitude **shall** (1508) be 1 Hz.
2. Surveillance **shall** (1509) provide the STM with ownship radio altitude only when an updated input is available. This requirement prevents Surveillance from providing coasted ownship radio altitude data to the STM.
3. Surveillance **shall** (1510) represent ownship radio altitude as *NaN* when the measured value is greater than 2300 ft or the Monitor has declared it non-credible.
4. The precision of the ownship radio altitude provided to this function **shall** (1512) be 1 ft or less.

The prescriptive algorithms of the STM assume that all quality checks for ownship radio altitude are performed by the Monitor. The requirements for radio altitude credibility testing are defined in §2.2.7.2.5.2.

2.2.5.5.3 ReceiveHeadingObservation

Surveillance **shall** (1513) provide the STM with the ownship true heading data defined in Table 2-24 (identical to Table 2-12 of the ADD [Volume II]).

Table 2-24: Heading Observation

Data Item	Description
Psi	own heading, clockwise from 0 to 2pi (north = 0, true east = pi/2)
toa	time of applicability
heading_degraded	if false, indicates that input is valid true heading. if true, indicates that input is other than true heading (ex: track angle).

Additional input requirements include the following:

1. Surveillance **shall** (1920) provide untracked ownship true heading data to the STM and set heading_degraded to false.

Note: True heading is a requirement and must nominally be available in any installation.

2. Surveillance **shall** (1921) provide ownship heading data to the STM at least once in every surveillance update period. This requirement ensures a nominal update rate of 1 Hz.
3. If ownship true heading is not available in a surveillance update period, Surveillance **shall** (1922) declare a failed ownship heading source to the STM by providing NaN as input, or may optionally provide track angle as input.

Note: Track angle provides some improvement compared to declaring failed input.

4. Surveillance **shall** (2603) set the “heading_degraded” flag to True if and only if accompanied by track angle as input.
5. The time of applicability error (from time of receipt by the equipment) for ownship heading **shall** (1923) be less than 250 msec.
6. The precision of the ownship heading provided to this function **shall** (1924) be 0.1 degrees or less.

2.2.5.5.4 ReceiveWgs84Observation

Surveillance **shall** (1925) provide the STM with the ownship WGS84 state data defined in Table 2-25 (identical to Table 2-13 of the ADD [Volume II]).

Table 2-25: WGS84 Observation

Data Item	Description
Lat	own latitude
Lon	own longitude
vel ew	ownship east-west velocity
vel ns	ownship north-south velocity
toa	time of applicability

Additional input requirements include the following:

1. Surveillance **shall** (1926) provide untracked WGS84 state data to the STM.
2. Surveillance **shall** (1927) provide ownship WGS84 state data to the STM at least once in every surveillance update period during which Surveillance has received an ownship WGS84 state input. This requirement ensures a minimum update rate of 1 Hz whenever updated ownship WGS84 state data is available to ACAS X.
3. The time of applicability error (from time of receipt by the equipment) for ownship WGS84 state **shall** (1928) be less than 50 msec.
4. The precision of the measured ownship latitude and longitude **shall** (1929) be within (10^{-10}) degrees.

Note: Systems that have utilized IEEE 754 64 bit compatible types have been successful in passing the established Test Suite).

5. The resolution of ownship velocity provided to this function **shall** (1930) be 1 knot or less.
6. When either ownship latitude or longitude are unavailable or invalid for the duration of a surveillance update period, toa **shall** (2231) be marked as NaN. This has the effect of disabling passive surveillance within the STM while a valid ownship WGS84 state is unavailable.

2.2.5.5.5

ReceiveDiscretes

Surveillance **shall** (1931) provide the STM with the ownship discrete data defined in Table 2-26 (identical to Table 2-8 of the ADD [Volume II]).

Table 2-26: Ownship Discretes

Data Item	Description
address	24-bit own address
modeACode	own Mode A address
opflg	system operational flag
manualSL	externally set sensitivity level
own_ground_display_mode_on	display on ground
on_surface	flag to indicate whether ownship operating on surface (true) or taking off / airborne (false)
aoto_on	ADS-B only TA-Only flag
is_coarsely_quant	flag to indicate if ownship altitude is 100 foot quantized

Additional input requirements include the following:

1. Ownship discrete data **shall** (1932) be updated at a minimum when the value of any input argument changes. Therefore this input to the STM can be considered event driven. Ownship discrete data may be delivered to the STM at a higher update rate.
2. **own_ground_display_mode_on**: This input supports a system configuration parameter to control whether the system sensitivity level enters TA-Only mode or Standby mode when ownship is determined to be on the ground based on radar altitude. If **true** the system will be placed into TA-Only mode when radar altitude indicates ownship is on the ground, otherwise it will be placed into Standby mode when radar altitude indicates ownship is on the ground.

Note: Regardless of the own_ground_display_mode_on setting, all TAs are disabled when radar altitude and groundspeed indicate ownship is operating on the surface as per the requirements of §2.2.4.6.3. The setting of own_ground_display_mode_on affects operation of ACAS X active surveillance of non-ADS-B intruders while ownship is on the ground.

3. **on_surface**: This input **shall** (2234) be set to **true** whenever the ownship aircraft is considered to be Operating on the Surface as per the requirements of §2.2.4.6.3. The input **shall** (2235) be set to **false** whenever the ownship is considered to be Taking Off / Airborne.
4. **opflg**: This input **shall** (2116) be set to **true** to indicate that the system health, based on validity of required inputs and system health of the equipment, supports operation of ACAS X. If the system is not operational it **shall** (2117) be set to **false**. For example, if the system did not support interrogation transmissions because of a hardware problem, this input flag would be set to **false**.

5. **manualSL**: This represents the crew or other input controls which select one of three values: Standby (1), TA-Only (2) , or Automatic (0). §2.2.7.3.4 specifies optional inputs of 3,4,5,6,7. If these input values for SL are provided each value **shall** (2582) be mapped to the manualSL value Automatic (0).

Note: In standby mode, 1030 MHz interrogations are not transmitted, ACAS X tracks are not updated, and advisories are not issued.

6. **is_coarsely_quant**: This input **shall** (2120) be set to the resolution of the own altitude provided to ReceiveBaroAltObservation. The value **shall** (2122) be

set to false if the resolution is 10 ft or less. Otherwise the value **shall** (2121) be set to true (100 ft).

Note: This does not define the LSB resolution of the own altitude provided to the ADD algorithm ReceiveBaroAltObservation – just the granularity or resolution of the data. For example, if is_coarsely_quant is set to true, then the own altitude changes in 100 ft increments.

7. aoto_on: Flag to indicate if ADS-B Only TA-Only is allowed. The flag **shall** (2232) be set to true if the system is to be configured to generate TAs against aircraft that are only tracked with ADS-B data.

Note: Setting the system to true requires a display that differentiates that the traffic is only tracked with ADS-B and would never have an RA declared against it. See §1.3.1, §2.2.3.12.9 and §2.2.6.1.2.1.3 for further information.

2.2.5.5.6 ReceiveDF0

Surveillance **shall** (1933) provide to the STM the Mode S Reply (DF=0) data fields defined in Table 2-27 (identical to Table 2-2 of the ADD [Volume II]).

Table 2-27: DF0 Message

Data Item	Description
r_slant	raw slant range to intruder
Chi_rel	raw decoded intruder bearing (nose = 0, right = pi/2)
z_baro	raw intruder barometric altitude
mode_s	intruder ICAO 24-bit Aircraft Address
q_int	intruder alt quantization 0 (NAR); 25, 100
ri	intruder reply info (i.e. equipage)
surv_mode	type of surveillance (0/Norm; 1/Red; 2/HS)
toa	time of applicability

Additional input requirements include the following:

1. Upon track establishment surveillance **shall** (1934) provide the STM with Mode S reply (DF=0) data associated with front end established tracks.
2. While the track is established surveillance **shall** (2107) provide the STM with Mode S reply (DF=0) data used to maintain the surveillance track.
3. Surveillance **shall** (1935) update the surv_mode during changes in state of this value even if no reply is received. When no reply is received the r_slant, Chi_rel, and z_baro inputs **shall** (2165) be set to *NaN*.
4. The time of applicability for a Mode S reply **shall** (1936) be equivalent to the time of message reception +/- 20 msec.

5. An invalid bearing measurement **shall** (1937) be represented to the STM as *Nan*.
6. Altitude from a non-altitude reporting aircraft or invalid reported altitude from the intruder **shall** (1938) be represented to the STM as *Nan*.
7. The measured slant range **shall** (1939) be greater than or equal to zero.

Note: Any reception that does not result in a valid range measurement is not passed to the STM.

8. The precision of the Mode S reply data fields **shall** (1940) be retained when delivered to the STM.
9. Surveillance provides the STM with only Mode S Reply messages that have an ICAO 24-bit Aircraft Address not equal to own, not all equal to all ONES, and not equal to all ZEROS.
10. Surveillance **shall** (1942) provide Mode S reply messages only from intruder aircraft that are determined to be airborne (per §2.2.4.6.2.2).
11. Surveillance **shall** (1945) provide untracked slant range, bearing and reported altitude data to the STM.

Note: Barometric altitude values quantized at 100 ft are encoded in a Mode S reply as described in §2.2.3.8.3.1.2(c). All barometric altitude values must be decoded as described in §2.2.3.8.3.1.2 prior to input into the STM. Barometric altitude values input to the STM are accompanied by a q_int bit which denotes the quantization of the input data.

2.2.5.5.7 ReceiveModeCReply

The ReceiveModeCReply algorithm or the ReceiveModeCReplies algorithm of the ADD can be used to provide reply data from aircraft equipped with ATCRBS transponders. This section defines the interface requirements for systems which use the ReceiveModeCReply interface.

Surveillance **shall** (2166) provide to the STM the Mode C Reply data fields defined in the ADD for the ModeCReply type identified in Table 2-28 (identical to Table 2-3 of the ADD [Volume II]) below.

Table 2-28: Receive Mode C Reply Message

Data Item	Description
external_ID	track number assigned by the front end tracker
coded_alt	reported Gilham altitude code
conf	altitude confidence bits
r_slant	raw slant range to intruder
Chi_rel	raw decoded intruder bearing (nose = 0, right = pi/2)
toa	time of applicability

Additional input requirements include the following:

1. Upon track establishment surveillance **shall** (2167) provide the STM with Mode C reply data associated with front end established tracks including the 4 replies used to establish the track per §2.2.4.6.4.1.2.

Note: *ReceiveModeCReply must not be utilized until all the replies that satisfy the establishment are available to prevent a non-established track from being presented to the display or the TRM.*

2. The conf field is not used by the ReceiveModeCReply interface and may be omitted by the implementation or set to a constant value.
3. While the track is established surveillance **shall** (2168) provide the STM with Mode C reply data received this surveillance update interval.
4. The time of applicability for a Mode C reply **shall** (2169) be equivalent to the time of message reception +/- 20 msec.
5. An invalid bearing measurement **shall** (2170) be represented to the STM as *NaN*.
6. The coded_alt field in the ModeCReply data structure **shall** (2171) contain a bit vector representation of the reported altitude in order of pulse reception but without the framing pulses or the X bit. This bit vector representation can be summarized as follows: C1 A1 C2 A2 C4 A4 B1 D1 B2 D2 B4 D4.
7. The measured slant range **shall** (2172) be greater than or equal to zero.

Note: *Any reception that does not result in a valid range measurement is not passed to the STM.*

8. Surveillance **shall** (2173) provide Mode C reply messages only from aircraft that are determined to be airborne.
9. Surveillance **shall** (2174) provide untracked slant range, bearing and reported altitude data to the STM.
10. During track initiation when only an initial altitude estimate is available (due to garbling) surveillance **shall** (2175) provide the initial altitude estimate for the replies used to form the estimate (§2.2.4.6.4.1.2).
11. external_ID **shall** (2588) be set to the same value for the duration of the established front-end track.
12. external_ID assigned to a track **shall** (2589) be a number that has not been used for a different track in the past 6 surveillance intervals.

2.2.5.5.8

ReceiveModeCReplies

The RecieveModeCReply algorithm or the ReceiveModeCReplies algorithm of the ADD can be used to provide reply data from aircraft equipped with ATCRBS transponders. One of the algorithms must be used by a system.

This section defines the interface requirements for systems which use the ReceiveModeCReplies interface. The algorithms associated with ReceiveModeCReplies invoke suggested algorithms in the ADD.

ReceiveModeCReplies expects an input array of ADD type ModeCReply as identified in Table 2-28.

Additional input requirements include the following:

1. ReceiveModeCReplies **shall** (2176) be invoked only once every surveillance update interval whether or not any Mode C replies are available.
2. Every surveillance update interval surveillance **shall** (2177) provide the STM with the reply data for all in-beam and non-phantom Mode C replies received in a given surveillance update interval (i.e. replies from all whisper-shout interrogations, all beams, both antennas) from a whisper-shout sequence (not a monitor interrogation).
 - a. The requirements for in-beam Mode C replies are given in §2.2.4.6.4.1.2 and §2.2.4.6.4.1.3.
 - b. The requirements for phantom rejection are given in §2.2.4.4.2.1(d).
3. The time of applicability for a Mode C reply **shall** (2178) be equivalent to the time of message reception +/- 20 msec.
4. An invalid bearing measurement (Chi_rel) **shall** (2179) be represented to the STM as *Nan*.
5. The coded_alt field in the ModeCReply data structure **shall** (1948) contain a bit vector representation of the reported altitude in order of pulse reception but without the framing pulses or the X bit. This bit vector representation can be summarized as follows: C1 A1 C2 A2 C4 A4 B1 D1 B2 D2 B4 D4.
6. The conf field has the same bit definition as the coded_alt field except that a 1 bit indicates that the corresponding altitude bit was potentially garbled or declared to have low-confidence.
7. The measured slant range **shall** (2180) be greater than or equal to zero.

Note: Any reception that does not result in a valid range measurement is not passed to the STM.

8. The precision of the Mode C reply data fields **shall** (2181) be retained when delivered to the STM.
9. Surveillance **shall** (2182) provide untracked slant range, bearing and reported altitude data to the STM.

2.2.5.5.9 **ReceiveStateVectorPosition**

Surveillance **shall** (1949) provide the STM with the ADS-B State Vector Position Report data fields defined in Table 2-29 (identical to Table 2-5 of the ADD [Volume II]). This requirement is intended to apply regardless of the surveillance region (Extended Hybrid Surveillance, Hybrid Surveillance or Active Surveillance) the intruder is under.

Table 2-29: State Vector Position Report

Data Item	Description
lat	intruder latitude
lon	intruder longitude
alt	intruder barometric alt
mode_s	intruder ICAO 24-bit Aircraft Address
nic	intruder NIC
q_int	intruder alt quant
rebroadcast	flag to indicate message received from ADS-R
non_icao	flag to indicate that the address is non-standard ICAO 24-bit Aircraft Address (anonymous address)
toa	time of applicability

Additional input requirements include the following:

1. The time of applicability for the ADS-B State Vector Report **shall** (1950) be consistent with the requirements specified in §2.2.8.1.4.1 of RTCA/DO-260B (Ref. G).
2. The time of applicability for the ADS-B State Vector Report **shall** (1951) reflect the reception of an ADS-B Airborne Position Message.
3. Surveillance **shall** (1952) provide at a minimum the most recent ADS-B State Vector Report to the STM in a given surveillance update interval. This implies a minimum nominal update rate of 1 Hz. Surveillance may provide the STM with all ADS-B State Reports, which in the nominal case would be an update rate of 0.5 Hz.
4. The precision of the ADS-B State Vector Report data fields **shall** (1953) be retained when delivered to the STM.
5. Surveillance **shall** (1954) provide the STM only State Vector Reports that have an ICAO 24-bit Aircraft Address not equal to ownship, and not equal to all ONES and not equal to all ZEROS.
6. Surveillance **shall** (1955) provide ADS-B State Vector Reports only from aircraft that are determined to be airborne.
7. Surveillance **shall** (1957) provide raw, decoded, untracked latitude, longitude and reported altitude data to the STM.
8. No ADS-B State Vector Report data **shall** (1958) be provided to the STM if the reported latitude or longitude associated with the report is invalid.
9. Surveillance **shall** (1959) provide the STM with ADS-B or ADS-R reports that are NOT identified as duplicate tracks according to the Report Validity Checks in §2.2.3.1.3.1 of RTCA/DO-317B (Ref. T) or are NOT marked as duplicates using the requirements of RTCA/DO-260B (Ref. G).
10. ADS-B messages **shall** (2131) be provided to the STM regardless of ADS-B link version number.

Notes:

1. In the case of ADS-B link version 0 messages, NIC values may be derived according to RTCA/DO-260B Table 2-200, but do not affect the performance of the STM.
2. RA protection based on ADS-B surveillance is provided only for ADS-B link versions 2 and above.

2.2.5.5.10 ReceiveStateVectorVelocity

Surveillance **shall** (1961) provide the STM with the ADS-B State Vector Velocity Report data fields defined in Table 2-30 (identical to Table 2-6 of the ADD [Volume II]). This requirement is intended to apply regardless of the surveillance region (Extended Hybrid Surveillance, Hybrid Surveillance or Active Surveillance) the intruder is under.

Table 2-30: State Vector Velocity Report

Data Item	Description
vel_ew	intruder east-west velocity
vel_ns	intruder north-south velocity
mode_s	intruder ICAO 24-bit Aircraft Address
nic	intruder NIC
rebroadcast	flag to indicate message received from ADS-R
non_icao	flag to indicate that the address is non-standard ICAO 24-bit Aircraft Address (anonymous address)
toa	time of applicability

Additional input requirements include the following:

1. The time of applicability for the ADS-B State Vector Velocity Report **shall** (1962) be consistent with the requirements specified in §2.2.8.1.4.1 of RTCA/DO-260B (Ref. G).
2. The time of applicability for the ADS-B State Vector Velocity Report **shall** (1963) reflect the reception of an ADS-B Airborne Velocity Message.
3. Surveillance **shall** (1964) provide at a minimum the most recent ADS-B State Vector Velocity Report to the STM in a given surveillance update interval. This implies a minimum nominal update rate of 1 Hz. Surveillance may provide the STM with all ADS-B State Reports, which in the nominal case would be an update rate of 2 Hz.
4. The precision of the ADS-B State Vector Velocity Report data fields **shall** (1965) be retained when delivered to the STM.
5. Surveillance **shall** (1966) provide the STM only State Vector Velocity Reports that have an ICAO 24-bit Aircraft Address not equal to ownship, and not equal to all ONES and not equal to all ZEROS.

6. Surveillance **shall** (1967) provide ADS-B State Vector Velocity Reports only from aircraft that are determined to be airborne.
7. Surveillance **shall** (1969) provide raw, decoded, untracked velocity data to the STM.
8. No ADS-B State Vector Velocity Report data **shall** (1970) be provided to the STM if the reported east/west or north/south velocities associated with the report are invalid.
9. Surveillance **shall** (1971) provide the STM with ADS-B or ADS-R reports which are NOT identified as a duplicate tracks. The system **shall** (1972) meet the duplicate address detection requirements specified in RTCA/DO-317B (Ref. T), §2.2.3.1.3.4.
10. ADS-B messages **shall** (2132) be provided to the STM regardless of ADS-B link version number.

Notes:

1. *In the case of ADS-B link version 0 messages, NIC values may be derived according to ED-102A / RTCA/DO-260B Table 2-200, but do not affect the performance of the STM.*
2. *RA protection based on ADS-B surveillance is provided only for ADS-B link versions 2 and above.*
3. *The first ADS-B State Vector Velocity Report need not be provided until after the first position report has been provided for that track. This is because the STM does not use velocity reports until a position report for that track has been received.*

Airborne ADS-B or ADS-R reports from duplicate address traffic are not provided to the STM for display. The system **shall** (1975) detect airborne ADS-B duplicate address traffic between ADS-B sources and detect airborne ADS-R duplicate address traffic between ADS-R sources (i.e. not between ADS-B and ADS-R traffic).

2.2.5.5.11 ReceiveModeStatus

Surveillance **shall** (1976) provide the STM with the ADS-B Mode Status Report data fields defined in Table 2-31 (identical to Table 2-7 of the ADD [Volume II]).

Table 2-31: Mode Status Report

Data Item	Description
adsb_version	intruder ADS-B version
nacp	Nav. Accuracy Position (0-11)
nacv	Nav. Accuracy Velocity (0-4)
sil	Source Integrity Level (0-3)
sda	System Design Assurance (0-3)
mode_s	intruder ICAO 24-bit Aircraft Address
rebroadcast	flag to indicate message received from ADS-R
non_icao	flag to indicate that the address is a non-standard ICAO 24-bit Aircraft Address (anonymous address)

Additional input requirements include the following:

1. Surveillance **shall** (1978) provide the STM with ADS-B Mode Status Report data upon the reception of an ADS-B Operational Status Message or Target State and Status Message.
2. Surveillance **shall** (1983) provide the STM with ADS-B or ADS-R reports which are NOT identified as a duplicate tracks. The system **shall** (1984) meet the duplicate address detection requirements specified in ED-194A / RTCA/DO-317B (Ref. T), §2.2.3.1.3.4.
3. ADS-B messages **shall** (2133) be provided to the STM regardless of ADS-B link version number.

Notes:

1. *In the case of ADS-B link version 0 messages, NACp, NACv, and SIL values may be derived according to RTCA/DO-260B Appendix N.2.3.7 (NACp), N.2.3.8 (NACv), N.2.3.9 (SIL) and Table 2-200 (NACp, SIL), but do not affect the performance of the STM.*
2. *RA protection based on ADS-B surveillance is provided only for ADS-B link versions 2 and above.*
3. *The first ADS-B Mode Status Report need not be provided until after the first position report has been provided for that track. This is because the STM does not use Mode Status Reports until a position report for that track has been received.*

2.2.5.5.12 ReceiveTargetDesignation

Surveillance **shall** (1985) provide to the STM the target designation data fields defined in Table 2-32 (identical to Table 2-9 of the ADD [Volume II]).

Table 2-32: Target Designation

Data Item	Description
mode_s	intruder ICAO 24-bit Aircraft Address
designation	selected designation (see Table 2-33)

Additional input requirements include the following:

1. Surveillance **shall** (1986) provide the STM with intruder designation data when a request is issued by the flight crew identifying an intruder with an ICAO 24-bit Aircraft Address. There is no minimum update rate for target designation data as it is an event driven input to the STM.
2. Intruder designation data **shall** (1988) be provided to the STM within one surveillance update cycle the command is issued by the flight crew.
3. The designation field **shall** (2183) be one of the values listed in Table 2-33 below. It is acceptable to just utilize Designation_None in lieu of Designation_Undesignate_No_Alerts or Designation_Undesignate_Protection_Mode.

Table 2-33: Designation Field Values

Designation value	Meaning
DESIGNATION_NONE	No Designation. This value undesignates an intruder.
DESIGNATION_DESIGNATED_NO_ALERTS	Indicates when designated to DNA by the crew.
DESIGNATION_Xo_CSPO3k	Indicates when designated to CSPO-3000 by the crew
DESIGNATION_UNDESIGNATE_NO_ALERTS	Indicates when an intruder previously designated to No_Alerts is undesignated.
DESIGNATION_UNDESIGNATE_PROTECTION_MODE	Indicates when an intruder previously Designated to CSPO-3000 is undesignated.

2.2.5.5.13 ReceiveUF16UDS30

Upon the receipt of a valid coordination interrogation message (UF=16, UDS=30) passed to ACAS X from own Mode S transponder, surveillance **shall** (1989) provide to the STM the data fields defined in Table 2-34 (identical to Table 2-14 of the ADD [Volume II]).

Table 2-34: UF16UDS30 Message

Data Item	Description
mode_s	intruder ICAO 24-bit Aircraft Address
cvc	cancel vertical RA complement
vrc	vertical RAcomplement
vsb	vertical sense bits
toa	time of applicability

Note: CHC, HRC, and HSB are not mapped.

Additional input requirements include the following:

1. The time of applicability for a coordination interrogation **shall** (1990) be equivalent to the time of message reception +/- 20 msec.
2. The timing requirements associated with this ADD algorithm are specified as described in §2.2.3.9.3.4 of this document.

2.2.5.5.14 **ReceiveCapabilityReport**

Upon the receipt of an ADS-B or ADS-R Aircraft Operational Status Messages subtype 0 (airborne) with ADS-B version greater than or equal to 3, surveillance **shall** (2184) provide these inputs defined in Table 2-35 (Table 2-15 of the ADD [Volume II]) below. The operational status message is defined in §2.2.3.8.3.2.9.2.

Table 2-35: Coordination Capability Report

Data Item	Description
adsb_version	ADS-B version indicator
ca_operational	indicates that a collision avoidance system is operational
sense	vertical and/or horizontal sense bit(s)
type_capability	CAS and type capability
priority	priority flag
daa	detect and avoid subfield that indicates the coordination type to be provided
mode_s	reported ICAO 24-bit Aircraft Address

Additional input requirements include the following:

1. The Aircraft Operational Status Message may be delivered to the STM at a higher update rate.
2. Sense, type_capability, and priority are extracted from the CCCB bits defined in §2.2.3.8.3.2.9.2.

3. ca_operational and daa are extracted from the Airborne Capability Class (CC) subfield defined in §2.2.3.8.3.2.9.2.
4. Aircraft Operational Status message data from aircraft that are determined to be on the ground **shall** (2186) not be provided to the STM.
5. Surveillance **shall** (2187) provide the STM with ADS-B reports which are NOT identified as a duplicate tracks. The system **shall** (2188) meet the duplicate address detection requirements specified in Ref. T (ED-194A / RTCA/DO-317B), §2.2.3.1.3.4.

2.2.5.6 Combined STM/TRM Output Interfaces

This section specifies the output interface requirements of the combined STM and TRM logic. The section is organized functionally:

- Data intended for the crew
- Data transmitted by ACAS X on 1030 MHz
- Data intended for ownship transponder.

Table 2-36 lists the outputs of the STMReport and TRMReport with a brief summary of each output and which subsection provides additional interface information.

Table 2-36: STM /TRM Output Data

STM / TRM Output Data	Content	Interface Section
STMReport.trm_input	Intruder and Own Belief states and other related information	2.2.5.6.4
STMReport.transponder	Information for setting SL, RI, VI, and the Data Link Capability Report	2.2.5.6.3.1
STMReport.display	Information for Traffic Display	2.2.5.6.1.1.1
TRMReport.display (cc, vc, ua, da, target_rate, turn_off_aurals, crossing, alarm)	Information for the RA Display, RA aural, and enabling and disabling voice commands	2.2.5.6.1.2
TRMReport.display.intruder	Information for Traffic Display of RA	2.2.5.6.1.1.2
TRMReport.coordination	Information for generating UF 16, UDS=48 Coordination Intent Messages	2.2.5.6.2.1
TRMReport.designation	Designation Information	2.2.5.6.1.3
TRMReport.broadcast	Information For Generating 1030 RA Broadcast, RA Report to Transponder	2.2.5.6.2.2
TRMReport.ground_msg	Information For Generating, RA Report to Transponder in support of coordination replies and RA Reports to the ground	2.2.5.6.3.3

The TRMInput structure (reference ACAS X MOPS Volume II Appendix E §E.3) is consumed by the TRM and is not defined further in this document. The transponder and display structures are defined in the following subsections.

2.2.5.6.1 Crew Outputs

The STM/TRM outputs support the follow crew outputs:

- Traffic Display
- RA Display and Annunciation
- Target Designation Display interactions

2.2.5.6.1.1 Traffic Display Outputs

ACAS X **shall** (1997) combine the relevant data fields from the STM Report and TRM Report for use by the display.

The TRMReport.display.intruder and STMReport.display structures contain the data for displaying ACAS X traffic data. The STMReport.display provides all the data with the exception of the intruder advisory type (RA, TA, Proximate, Other (Non-Threat)) and order prioritization which is provided by TRMReport.display.intruder.

Please note that the requirements of §2.2.3.12.9 limit whether ASD-B only or ADS-R only traffic can be provided to the display.

2.2.5.6.1.1.1 STMReport.display

The display substructure of the StmReport is an array of type StmDisplayStruct defined in the ADD. Table 2-37 below identifies its contents.

Table 2-37: StmDisplayStruct

Data Item	Description
arrow	display vertical rate arrow
alt_reporting	indicates that the intruder is altitude reporting
bearing_valid	indicates that a bearing is available for the intruder
id	intruder ID inside the target data base
z_rel	relative altitude
r_ground	tracked ground (horizontal) range
Chi_rel	bearing relative to own frame
mode_s	intruder 24-bit aircraft address
is_icao	indicates whether provided 24-bit address is ICAO compliant

arrow: The system uses this value to indicate to the display if the intruder is climbing (+1), level (0), or descending (-1).

alt_reporting: A ‘true’ value indicates that that z_rel represents the relative altitude of the intruder to ownship. A ‘false’ value indicates that the intruder’s altitude is not known.

id: This unique id is the same id used in the TRMReport.display.intruder structure to identify an intruder.

z_rel: The relative altitude of the intruder where values greater than zero indicate that the intruder is below ownship.

Chi_rel: The bearing of the intruder relative to own frame.

mode_s: The 24-bit aircraft address for TCAS Mode S, ADS-B, or ADS-R tracks. Mode C tracks will have a 24-bit aircraft address of 0. To determine that the track is non-ICAO, the TRM Input structure must be utilized. This structure is defined in the ADD (Volume II).

2.2.5.6.1.1.2 TRMReport.display.intruder

The display substructure of the TrmReport is an array of type TrmIntruderDisplayData defined in the ADD. Table 2-38 below identifies its contents.

Table 2-38: TrmIntruderDisplayData

Data Item	Description
id	intruder ID inside the target database
mode_s	intruder 24-bit aircraft address
is_icao	indicates whether provided 24-bit address is ICAO compliant
tds	track display score; used for display prioritization
code	track display advisory code (see TACODE_constants)

id: This unique ID is the same ID used in the STM output structure to identify an intruder.

mode_s: The 24 bit address (ICAO or non-ICAO) for TCAS Mode S, ADS-B, or ADS-R tracks. The STMReport.intruder is_icao flag can be used to distinguish whether this is an ICAO 24-bit Aircraft Address or non-ICAO address.

tds: Traffic Display Score. The higher the number the greater the threat. ACAS X **shall** (2189) transmit the intruder data to the displays in order of priority (highest to lowest) as determined by the TRM traffic display score (tds). When necessary, the TDS score can be used as a tiebreaker in meeting the requirements for selecting which OCM is transmitted per §2.2.3.12.2.2.5.1.

code: This value identifies the threat level of the target: RA, TA, Proximate, or Other (Non-Threat). The specific codes are enumerated in the ADD starting with the prefix TACODE_.

2.2.5.6.1.2 RA Output (TRMReport.display)

The TRMReport.display structure is of type TrmDisplayData and is defined in the ADD. Table 2-39 below identifies its contents with the exception of the Intruder field which is associated with traffic display and is defined in the section above.

Table 2-39: TrmDisplayData

Data Item	Description
cc	combined control
vc	vertical control
ua	up advisory
da	down advisory
target_rate	vertical rate to target (f/s)
turn_off_aurals	below low altitude cutoff for aurals
crossing	crossing alert flag
alarm	annunciate RA flag

alarm: When this field is “true” it indicates that an RA has been issued and a voice annunciation is performed per §2.2.6.3.5.

Table 2-49 provides the mapping and definition of the fields cc, vc, ua, da, crossing, and target_rate and the RA phrase to be spoken and the industry standard ARINC label 270 RA word. target_rate is unbounded, though the rate provided at the display may be any value from 0 to 6300 feet per minute.

turn off aurals: When true it indicates that aural advisories are to be inhibited due to proximity to the ground and may be used by the system to satisfy the requirements of §2.1.12.

2.2.5.6.1.3 ACAS Xo Traffic Designation Information (TRMReport.designation)

A Class 2 ACAS X uses TRMReport.DesignationData to communicate all information associated with the supported Xo modes and designation of an intruder to an Xo mode. The supported Xo modes are DNA and CSPO-3000.

Table 2-40: TRMDesignationData

Data Item	Description
availability	Xo mode availability for each Xo mode

availability: A fixed-size array or vector containing a flag for each Xo mode which indicates whether that mode is available. When an Xo mode is unavailable the flag is false and no intruders can be designated to that mode.

2.2.5.6.1.3.1 TRMReport.Designation.Intruder

The intruder substructure of the TrmReport.Designation is an array or vector of type TRMIintruderDesignationData defined in the ADD. Table 2-41 below identifies its contents. TRMReport.Designation data of ADD defined type TRMIintruderDesignationData is used to communicate to the displays the designation status of each intruder and also communicates whether an intruder is eligible for DNA and/or CSPO-3000 designation.

Table 2-41: TRMIintruderDesignationData

Data Item	Description
id	intruder ID inside the target data base
address	24-bit aircraft address of intruder
is_icao	indicates whether provided 24-bit address is ICAO compliant
designated_mode	Xo designation, requested at the flight deck (see DESIGNATION constants in ADD)
logic_mode	TRMLogicModeData. The actual Xo mode in force
active_ra	this intruder has an active RA (may be suppressed) used by STMHousekeeping
suppressed_ra	indicates whether an RA has been suppressed by Xo DNA mode. Used by STMHousekeeping
multithreat	there are multiple threats. If active_ra is true, includes this intruder
valid	Xo mode validity for each Xo mode
status	designation status (see Table 2-67)

id: This unique ID is the same ID used in the TRMReport.display.intruder structure to identify an intruder. However, there may also be unique IDs that do not correlate to an ID in the TRMReport.display.intruder when the intruder has been dropped but the designation is maintained for a period of time.

address: The ICAO 24-bit Aircraft Address for the intruder. Intruders with a non-ICAO address cannot be designated but TRMIintruderDesignationData may still be produced for an intruder with a non-ICAO address.

is_icao: Flag to indicate whether the address is an ICAO address or not.

designated_mode: Indicates the Xo mode, if any, requested at the flight deck. This designation may not be in force.

logic_mode: The logic_mode contains three subfields:

processing: Coded value to indicate the type of TRM processing (TA-Only, TA/RA, Degraded Surveillance, None, Dropped) applied for this intruder. “None” indicates the intruder was marked by the STM as invalid for TRM processing due to inadequate surveillance. “Dropped” indicates either surveillance was lost on the intruder, or, if there are more than 30 intruders, the intruder was filtered by the STM from its output

to the TRM. This subfield can be used to help determine the reason the other subfields in the logic_mode differ from the designated mode.

dna: Boolean flag to indicate whether Designated No Alerts was applied for this intruder.

protection_mode: Coded value to indicate which protection mode, Xa or Xo CSPO-3000, was applied for this intruder.

active_ra: Boolean flag to indicate whether an RA was produced. Information needed by the STM for bookkeeping.

suppressed_ra: Boolean to indicate whether an RA has been suppressed by the DNA Xo mode. Information needed by the STM for book keeping.

multithreat: Boolean flag to indicate whether the RA is a multithreat RA. Information needed by the STM for bookkeeping. May also provides additional information to the display when an RA is produced while an intruder is designated to DNA mode.

valid: Fixed-size array or vector containing a boolean flag for each Xo mode. Needed by the display to determine whether an intruder is valid for designation to a given Xo mode. ReceiveTargetDesignation disallows designation of an additional intruder if the limit on designations has been reached. The setting of the valid flags do not reflect whether or not the designation limit has been reached. The flags reflect Xo mode validity for a given intruder based on geometric and surveillance quality criteria.

status: Coded value providing information about the designation status of a specific intruder when the requested designation is not in force.

2.2.5.6.2

RA Information Transmitted by ACAS X

The STM/TRM outputs provide data to support the transmissions of coordination data and RA Broadcast message on 1030 MHz.

2.2.5.6.2.1

Data for TCAS Resolution Message (TRMReport.coordination)

The TRMReport.coordination element is an array of TRMCoordinationInterrogationData type defined in the ADD. Table 2-42 below identifies the contents of this structure.

Table 2-42: TRMCoordinationInterrogationData

Data Item	Description
id	intruder ID inside the target data base
cvc	Cancel Vertical RA Complement
vrc	Vertical RA Complement
vsb	Vertical Sense Bits (parity)
chc	Cancel Horizontal RA Complement
hrc	Horizontal RA Complement
hsb	Horizontal Sense Bits (parity)
mtb	multiple threats for this RA
mid	own ICAO 24-bit Aircraft Address (sender)
taa	intruder ICAO 24-bit Aircraft Address (intended receiver)
coordination_msg	coordination mechanism (see COORDINATION_constants)

This data is used by the system to assemble the TCAS Resolution Message as defined in §2.2.3.8.3.2.4.1 to perform coordination interrogations to Active CAS systems as required in §2.2.3.9.3.2.

This data is also used by the system to assemble the Operational Coordination Message as defined in §2.2.3.8.3.2.9.1 to provide coordination data to ownship transponder as required in §2.2.3.9.3.6.2.1 for transmission.

The array includes entries for each intruder in the STMReport.display and TRMReport.intruder and additional array elements for intruders dropped this surveillance update interval that need an intent canceled.

id: This is the same ID used in the STMReport.display.intruder output structure. Intruders which were dropped from the display in this cycle will persist as an ID one additional second in order to have a CVC sent to them.

cvc, vrc, vsb, chc, hrc, hsb, mtb, mid, taa are used to encode the OCM or the TCAS Resolution message.

coordination_msg: ACAS X **shall** (2190) communicate the coordination data using the protocol as specified in Table 2-43 below if the VRC and CVC are not both zero. The

ADD logic determines this coordination protocol in accordance with the requirements of §2.2.3.9.3.1.

Table 2-43: Coordination Protocol

Coordination_Msg	Action
COORDINATION_NONE	No action required.
COORDINATION_TCAS	Transmit the coordination data using a UF=16, UDS=48 interrogation.
COORDINATION_OCM	Provide the coordination data to own transponder for broadcast as an Operational Coordination Message using 1090 ES.

2.2.5.6.2.2 Data for the RA Broadcast Interrogation Message (TRMReport.broadcast)

The TRMReport.broadcast element is of TRMRABroadcastData type defined in the ADD. Table 2-44 below identifies the contents of this structure. This data is used by the system to assemble the RA Broadcast Interrogation Message as defined in §2.2.3.8.3.2.4.3 to satisfy the requirements of §2.2.3.9.6.1.

Table 2-44: TRMRABroadcastData

Data Item	Description
ra_data	TRMRACoordinationData Contains ARA, (LDI), RMF, RAC, RAT, and MTE fields (20 bits)
spi	RA suppression indicator (1 bit)
aid	own Mode A identifier code (13 bits)
cac	own Mode C altitude code (13 bits)

The ra_data field is of type TRMRACoordinationData as defined in the ADD. Its contents are identified in Table 2-45 below:

Table 2-45: TRMRACoordinationData

Data Item	Description
avra_single_intent	single intent / conflicting intents (1 bit)
avra_crossing	crossing / not crossing (1 bit)
avra_down	up/down (1 bit)
avra_strength	strength bits as integer value (4 bits)
ldi	low-level descend inhibits, for Xa (2 bits)
rmf	RA Message Format (2 bits)
rac	RA Complement (4 bits)
rat	when 1, this field indicates when the RA is terminated. (1 bit)
mte	there are multiple threats for this RA (1 bit)

2.2.5.6.3 Data To Be Provided to Own Transponder**2.2.5.6.3.1 SL/RI and DCLR (STMReport.transponder)**

The STMReport.transponder substructure is of type TransponderData defined in the ADD. Table 2-46 below identifies its contents.

Table 2-46: TransponderData

Data Item	Description
ri	air-to-air reply information
sl	sensitivity level
vi	version indicator
bit48	Data Link subfield bit 48 in MB
bit69	Data Link subfield bit 69 in MB
bit70	Data Link subfield bit 70 in MB
bit71	Data Link subfield bit 71 in MB
bit72	Data Link subfield bit 72 in MB

ri: Collision avoidance capability of the system. Provided to ownship transponder to satisfy requirements of §2.2.3.12.2.2.1. This field may also be required by some display interface characteristics.

sl: TCAS II sensitivity level. Provided to ownship transponder to satisfy requirements of §2.2.3.12.2.2.1. This field may also be required by some display interface characteristics.

vi: Version Indicator. Provided to ownship transponder to satisfy requirements of 2.2.3.12.2.2.4. These correspond to bits 43 to 46 of the Data Link Capability Report (DCLR) provided to ownship transponder indicating the version of ACAS X.

Bits 48, 69, 70, 71, and 72 make up the bits of the Data Link Capability Report which are provided to ownship transponder. These outputs **shall** (2191) be used to satisfy the requirements of §2.2.3.9.5.2 and the coding specified in §2.2.3.8.3.2.3.2.1.

The value of Bit 69 output by the TRM will always be zero. The system sets bit 69 per the requirements of §2.2.3.8.3.2.3.2.1.

2.2.5.6.3.2 Coordination Data Provided to the Transponder (TRMReport.coordination)

Coordination data is contained in the TRMReport.coordination element which is an array of TRMCordinationInterrogationData type defined in the ADD. This data is used by the system to communicate the complement of ownship's intent via Ownship ACAS X UF16 UDS=48 Interrogations or via the Operational Coordination Message transmitted by ownship transponder in a 1090 Extended Squitter. See §2.2.5.6.2.1 for interface requirements.

2.2.5.6.3.3 Data for the RA Report (TRMReport.ground_msg)

The data in the TRMReport.ground_msg is used to satisfy the requirements related to the coordination reply of §2.2.3.12.2.2.3, and RA report to the ground and in the 1090 ES RA broadcast message of §2.2.3.12.2.2.4.

Table 2-47: TRMGroundMsgData

Data Item	Description
ra_data	TRMRACoordinationData Contains ARA, (LDI), RMF, RAC, RAT, and MTE fields (20 bits) See §2.2.5.6.2.2 for more information.
tti	threat type indicator for most recent threat (1 bit)
tid	threat identifier for most recent threat (24 bits)
dsi	designation indicator (1 bit)
spi	RA suppression indicator (1 bit)

2.2.5.6.4 STMReport.trm_input

The field own_surveillance indicates when DEGRADED OWN BAROALT COAST is used to satisfy the requirements of §2.2.7.2.5.1 and §2.2.7.3.3 related to STM declaration of altitude credibility.

2.2.6 Displays and Controls

Displays and controls for ACAS X consist of several functions:

- **Traffic Displays** provide a graphical display of traffic position, generally relative to ownship. The traffic display provides situation awareness to the flight crew and helps the flight crew locate both traffic causing alerts and other traffic of interest when maneuvering. Traffic symbology provides additional information about the traffic, and identifies traffic with alerts.
- **RA Displays** provide specific maneuvering guidance for RAs.
- **Aural Alerts** provide voice information to alert the flight crew of TAs and RAs. The alerts indicate that attention should be given to the traffic display and/or RA display.
- **ACAS X controls** provide some control over the ACAS X system operation.

2.2.6.1 Traffic Displays

2.2.6.1.1 Functions

The traffic display provides the following functions:

- a. Aid in the visual acquisition of traffic and differentiation of intruder threat levels.
- b. Provide traffic situation awareness.
- c. Instill confidence in the displayed RAs.

2.2.6.1.2 Characteristics

2.2.6.1.2.1 General Characteristics

2.2.6.1.2.1.1 Ownship Symbol

The traffic display **shall** (1515) contain a symbol representing the location of the ownship. The color of the symbol **shall** (1516) be either white or cyan. Ownship symbol **shall** (2594) be differentiated from proximate and other traffic symbols (e.g. via color, shape, or other means) (see §2.2.6.1.2.1.3).

Notes:

1. *Cyan is the preferred color of the ownship symbol.*
2. *If ownship directionality is valid, the ownship symbol should be directional (e.g., not a circle or square).*

2.2.6.1.2.1.2 Range Rings

A means of indicating range (e.g. rings or markings) **shall** (1517) be placed at specified radii from the ownship symbol. The range indications should be the same color as the ownship symbol and of a size and shape that will not clutter the display. If ACAS X information is shown on a shared display (see §2.2.6.1.2.7) which does not provide range information, range indications **shall** (1521) be provided when the ACAS X information is selected.

2.2.6.1.2.1.3 Traffic Symbology

Note: The use of the RTCA/DO-317B symbol set, which expands upon this symbol set to provide directional symbology, is recommended for traffic meeting RTCA/DO-317B directionality requirements.

- a. The symbol for an RA **shall** (1522) be a red filled square.
- b. The symbol for a TA **shall** (1523) be an amber or yellow filled circle. If the AOTO option is enabled then the TA **shall** (2583) indicate whether the traffic can proceed to an RA.
- c. The symbol for Proximate Traffic **shall** (1524) be a white or cyan filled diamond. The Proximate Traffic symbol should be differentiated from that used for the ownship symbol (e.g. via color, shape, or other means) (see §2.2.6.1.2.1.1) to ensure the symbol is readable.

Note: White is the preferred color for this symbol.

- d. The symbol for Other Traffic **shall** (1525) be a white or cyan diamond, outline only. The Other Traffic symbol should be differentiated from that used for the ownship

symbol (e.g. via color, shape, or other means) (see §2.2.6.1.2.1.1) to ensure the symbol is readable.

Note: White is the preferred color for this symbol.

2.2.6.1.2.1.4 Off-Scale Symbology

RA or TA traffic which is beyond the selected display range of the traffic display **shall** (1526) be indicated. One means could be by placing a portion of the symbol at the edge of the display area at the measured bearing of the traffic. Data tags and appropriate vertical trend arrows **shall** (1527) remain fixed in the position relative to the traffic symbol defined in §2.2.6.1.2.1.5 and in §2.2.6.1.2.1.6, even if a portion of the information is masked by the edge of the active display area.

On shared displays (see §2.2.6.1.2.7), the off scale symbology described in this paragraph **shall** (1528) be used unless the symbology is inconsistent with that used for other functions of the shared display. In these cases, a message of either “TRAFFIC OFF-SCALE”, “OFFSCALE”, or “OFF-SCALE” **shall** (1529) be used. These messages **shall** (1530) be written in the colors specified in §2.2.6.1.2.1.3 which correspond to the traffic's threat level.

2.2.6.1.2.1.5 Altitude Data Tag

A data tag **shall** (1531) indicate the relative altitude, if available, of the intruder aircraft and **shall** (1532) consist of two digits indicating the altitude difference in hundreds of feet. For an intruder above ownship, the tag **shall** (1533) be placed above the traffic symbol and preceded by a “+” sign; for one below ownship, the tag **shall** (1534) be placed below the traffic symbol and preceded by a “-” sign.

Note: It is recommended that the “+” or “-” character be emphasized by using a slightly larger character set than that used for the digits.

The tag for co-altitude traffic **shall** (1535) be displayed as the digits “00”. The “00” characters should be placed above the symbol if the intruder aircraft closed from above; below the symbol if the intruder aircraft closed from below. If no trend information is available, the co-altitude “00” symbol should be placed below the traffic symbol.

The color of the data tag **shall** (1536) be the same as the symbol.

The display **shall** (1537) be capable of displaying relative altitudes of up to a maximum of ±9900 feet.

As an option, the data tag may indicate the intruder's reported altitude instead of the relative altitude between the two aircraft. If this option is implemented, a switch **shall** (1538) be provided to permit a pilot to select this type of altitude data tag. If the actual altitude is to be shown on a continuous basis after selection by the flight crew, it **shall** (1539) be corrected for the local barometric pressure using the same correction used by the flight crew. If the actual altitude is not corrected for the local barometric pressure, the display of this type of data tag **shall** (1540) be limited to a maximum of 30 seconds if the ownship is below 18,000 feet (FL180) before it automatically reverts back to the display of relative altitude. When the display of the actual altitude is selected, it **shall** (1541) be clearly annunciated on the traffic display. If implemented, the actual altitude **shall** (1542) be displayed as a three digit number representing hundreds of feet, MSL. For example, 007 represents 700 feet MSL and 250 represents 25,000 feet MSL (FL250). Actual altitude tags **shall** (1543) be positioned above or below the traffic symbol in a manner consistent with the relative altitude data tags. As with the display of relative altitude, the display **shall** (1544) be capable of displaying tracked traffic up to a maximum of ±9900 feet of the ownship.

2.2.6.1.2.1.6 Intruder Vertical Speed Arrow

A vertical arrow **shall** (1545) be placed to the immediate right of the traffic symbol if the vertical speed of the intruder (as determined by the ACAS X tracker) is equal to or greater than 500 fpm, with the arrow pointing up for climbing traffic and down for descending traffic. The color of the arrow **shall** (1546) be the same as the traffic symbol.

2.2.6.1.2.1.7 Non-Altitude Reporting Intruders

Neither a data tag nor a trend arrow **shall** (1547) be associated with the traffic symbol for an intruder that is not reporting altitude. The colors described in §2.2.6.1.2.1.3 for various threat levels **shall** (1548) be used for the display of non-altitude reporting intruders.

2.2.6.1.2.1.8 Intruder Bearing

The display **shall** (1789) be capable of displaying bearing of intruders. However, if valid bearing information is not received from the TRM for an intruder, that intruder will only be displayed when it has an associated TA or RA as indicated below.

Advisories issued against an intruder for which bearing information is not available (No-Bearing advisories) **shall** (1549) be presented for traffic generating either a TA or an RA. The No-Bearing advisory **shall** (1550) be displayed on the traffic display. The advisory **shall** (1551) be an alpha-numeric string which presents the information in the following order: threat level (TA or RA); range in NM; relative altitude (hundreds of feet); and the intruder vertical speed arrow. For example, “TA 5.2 -06↑” represents an intruder causing a TA at 5.2 NM with a relative altitude of -600 feet, and climbing. The alpha-numerics **shall** (1552) be written in the colors corresponding to the level of the threat, i.e., red for an RA and amber or yellow for a TA. The No-Bearing advisory may also be written using slashes to separate the different information fields and using NM after the range, e.g., “TA/5.2 NM/-06↑”.

The altitude data in a No-Bearing advisory **shall** (1553) be consistent with the selected altitude mode, i.e., relative altitude or actual altitude.

When a No-Bearing TA is issued against an intruder that is not reporting altitude, the altitude field of the message and the intruder vertical trend arrow **shall** (1554) be dropped. For example, a TA issued against a non-altitude reporting intruder at a range of two NM would be annunciated as “TA 2.0”.

The capability **shall** (1555) exist to display at least two No-Bearing advisories simultaneously.

The recommended location of the No-Bearing advisories is dependent on the implementation of the traffic display and is defined in Table 2-48.

If an intruder for which bearing information is available is being displayed in the same area as the No-Bearing advisory, the intruder with the higher priority, as defined by CAS, **shall** (1556) remain readable.

Table 2-48: Recommended Location of No-Bearing Advisories

Traffic Display Implementation	No-Bearing Advisory Location
TA/RA/VSI	Centered horizontally and below the ownship symbol.
Shared Weather Radar Display – WX/ACAS X Mode	Centered horizontally and below the ownship symbol.
Shared Weather Radar Display – WX-and-Traffic Mode	In any of the four corners of the display area.
Shared Weather Radar Display – Traffic-Only Mode	Centered horizontally and below the ownship symbol.
Navigation Display – MAP or North-up Modes	In an area which does not obscure navigation messages or data. It is recommended the information be centered vertically on the left side of the display.
Navigation Display – Rose Mode	Centered horizontally and below the ownship symbol.
Navigation Display – Arc, Expanded, or Approach Modes	In any of the four corners of the display area which does not result in the loss of navigation data or messages.
Engine Indication and Crew Alerting System (EICAS)/SYSTEMS Displays	Centered horizontally and below the ownship symbol.

2.2.6.1.2.1.9 Display of Traffic

Whenever an RA or a TA is displayed, all intruders causing an RA or TA and all Proximate Traffic within the selected display range, subject to any limitations to the maximum number of intruders that can be shown on the display, **shall** (1557) be displayed. It is recommended that Other Traffic within the selected display range also be displayed whenever an RA or TA is displayed to maximize the probability of the pilot visually acquiring the intruder causing the RA or TA.

The traffic display **shall** (1558) have the capability to display a minimum of eight intruder aircraft. All intruders being tracked by ACAS X are ranked by the CAS logic and the intruder information is sent to the display in a prioritized order. The display **shall** (1559) display the intruders in the order received from ACAS X to ensure the intruders most relevant to collision avoidance are displayed (as per Traffic Display Score requirement in §2.2.5.6.1.1.2, TRMReport.Display.Intruder). The maximum number of intruders that can be displayed at any one time **shall** (1560) be fixed and not variable. If the traffic display is not a TCAS Only Traffic Display and not an RTCA/DO-317 compliant display, then it **shall** (2587) depict intruders which are ADS-B Only or ADS-R such that they are distinguishable from intruders that have been validated by active surveillance. See §2.2.3.12.9 for additional information.

2.2.6.1.2.1.10 Altitude Band for the Display

The normal altitude band for the display of traffic **shall** (1561) be ±2700 feet from the ownship. If an intruder causing a TA or RA is outside this altitude band, it **shall** (1562) be

displayed with the appropriate relative or reported altitude displayed. Proximate and Other traffic outside the normal altitude band may also be displayed while a TA or RA is displayed.

As an option, a pilot selectable mode may be provided to permit the expansion of the normal altitude band. If this option is implemented, two additional modes, “Above” and “Below”, **shall** (1563) be provided. In the “Above” mode, tracked traffic **shall** (1564) be displayed if it is within the altitude band from 2700 feet below the ownship up to a maximum that **shall** (1565) not exceed 9900 feet above the ownship. In the “Below” mode, tracked traffic **shall** (1566) be displayed if it is within the altitude band from 2700 feet above the ownship down to a maximum that **shall** (1567) not exceed 9900 feet below the ownship. The upper and lower bounds of the “Above” and “Below” mode **shall** (1568) be fixed and not variable.

As a further option, a pilot selectable mode may be provided to permit the simultaneous selection of the “Above” and “Below” mode.

2.2.6.1.2.1.11 ACAS X Operating Mode and Selected Display Range Annunciation

The selected operating mode of ACAS X **shall** (1569) be annunciated on the traffic display. The ACAS X operating mode (§2.2.3.12.3) may be manually selected by the pilot using the ACAS X Control Panel (§2.2.6.5) or automatically by the CAS logic. In addition, the selected display range **shall** (1570) be annunciated on the traffic display.

Note: The operating mode selected by the pilot through the control panel (TA/RA, TA ONLY, STANDBY) is an "upper limit" to the actual operating mode of the ACAS X system. For example, if the pilot selects TA ONLY, the actual operating mode of the system cannot be TA/RA, but it can be STANDBY based on other inputs into the ACAS X system. This is why it is necessary to have an independent display of the actual mode that the ACAS X system is in.

2.2.6.1.2.1.12 RA Annunciations

With the exception of the red traffic symbology described in §2.2.6.1.2.1.3 and §2.2.6.1.2.1.8, no RA maneuver information **shall** (1571) be displayed on the Traffic Display. This requirement does not apply to the TA/RA/VSI display defined in §2.2.6.1.2.7.4 or to displays in which the TA and RA information are shown in separate sections of a display.

2.2.6.1.2.1.13 Lighting Control

The lighting intensity of the traffic display **shall** (1572) either be automatic or controllable by the pilot. For night operations, the brightness **shall** (1573) be controllable to a dim enough setting such that outside vision is not impaired while maintaining an acceptable presentation. For day operations, the display **shall** (1574) be readable in all sunlight conditions described in Paragraph 16.a.(3) of reference H.

2.2.6.1.2.1.14 Status and Failure Annunciations

Whenever status and failure annunciations are written in text on the traffic display, the annunciations **shall** (1575) be consistent with any existing annunciations shown on the display and **shall** (1576) have a single meaning for all available display modes.

2.2.6.1.2.2 Fixed Range Displays

2.2.6.1.2.2.1 Display Range

Note: It is recommended that the scale of the display area be between five and seven NM to the front and at least 2.5 NM to the rear of the ownship symbol.

2.2.6.1.2.2.2 Range Ring

A two mile range ring **shall** (1577) be provided.

2.2.6.1.2.3 Variable Range Displays

2.2.6.1.2.3.1 Display Range

A variable range display **shall** (1578) include a display range selection whose full scale display range is approximately five NM to the front and at least 2.5 NM to the rear.

Note: The requirement of 2.5 NM to the rear is only applicable to dedicated displays.

2.2.6.1.2.3.2 Range Selection

A range selector **shall** (1579) provide for the setting of different full range scales.

2.2.6.1.2.3.3 Range Rings

A range reference (ring or markings) **shall** (1580) be provided at either two or three NM for scales of 12 NM or less. For display ranges greater than 12 NM, at least one range reference (ring or markings) **shall** (1581) be provided. If the variable range display is also a shared display (see §2.2.6.1.2.7) which already provides range rings (or markings), the ACAS X range references specified by this subparagraph are not required.

2.2.6.1.2.4 Part Time (Pop-up) Displays

2.2.6.1.2.4.1 Activation

A part time display **shall** (1582) either activate (pop-up) the appropriate traffic symbology or display the written text “TRAFFIC”, “TFC”, or “TCAS” (e.g., on a navigation display, see §2.2.6.1.2.7.2) whenever a TA or an RA occurs. The words “TRAFFIC”, “TFC”, or “TCAS” **shall** (1583) be written in amber or yellow for a TA and in red for an RA. If ACAS X is issuing both a TA and an RA simultaneously, the word **shall** (1584) be written in red. For display implementations that incorporate the written text indication of TAs and RAs, the display **shall** (1585) include a crew selection for display of the traffic symbology.

The traffic information or the written text **shall** (1586) be displayed within one-half second of receipt of the information from the ACAS X processor.

2.2.6.1.2.4.2 Information Displayed

Once activated, the display **shall** (1587) remain active until the TA or RA is completed unless the selected display mode is changed by the pilot. The display **shall** (1588) meet the requirements of §2.2.6.1.2.1 when activated.

2.2.6.1.2.4.3 Display Format

A part time display may be a dedicated display (see §2.2.6.1.2.6) or a shared display (see §2.2.6.1.2.7).

A part time display may be either a fixed range display (see §2.2.6.1.2.2) or a variable range display (see §2.2.6.1.2.3).

2.2.6.1.2.4.4 Pilot Selection

If a display implementation provides the pilot the capability to manually switch between a part-time and a full-time display of traffic, the control setting used to make this selection **shall** (1589) be clearly labeled and obvious.

2.2.6.1.2.5 Full Time Displays**2.2.6.1.2.5.1 Display Format**

A full time display may be a dedicated display (see §2.2.6.1.2.6) or a shared display (see §2.2.6.1.2.7).

A full time display may be either a fixed range display (see §2.2.6.1.2.2) or a variable range display (see §2.2.6.1.2.3).

2.2.6.1.2.6 Dedicated Displays

Note: A dedicated display is a display or a display mode in which the only information shown on the display or on a designated section of the display is ACAS X traffic information.

2.2.6.1.2.6.1 Display Format

A dedicated display may be either a part-time display (see §2.2.6.1.2.4) or a full-time display (see §2.2.6.1.2.5). A dedicated display **shall** (1590) meet the requirements of §2.2.6.1.2.1.

2.2.6.1.2.6.2 Ownership Symbol and Location

The ownership symbol **shall** (1591) be centered horizontally on the display and approximately 1/3 of the height from the bottom of the display.

2.2.6.1.2.7 Shared Displays

Note: A shared display is a display or a display mode in which ACAS X information is shown simultaneously with any other information. It is possible for a display to provide multiple display modes, some of which have the characteristics of a dedicated display and others which have the characteristics of a shared display.

2.2.6.1.2.7.1 Shared Weather (WX) Radar Displays**2.2.6.1.2.7.1.1 Display Modes**

A shared weather radar display **shall** (1592) provide at least one of the following display modes. The other mode may be provided. When traffic information is shown on a shared display, the information **shall** (1593) comply with the requirements of §2.2.6.1.2.1.

- a. WX/ACAS X Mode. This is a part time display (see §2.2.6.1.2.4) in which either a TA or an RA **shall** (1594) cause the display to show ACAS X information. The ACAS X information may be shown with either a fixed range or a variable display range. When a TA or an RA is displayed, all weather information **shall** (1595) be removed from the display. If the ACAS X information uses a variable range display, the ACAS X information **shall** (1596) be displayed at the selected range of the weather radar display. When the encounter is over, the display **shall** (1597) revert back to the display of weather information at the selected radar range. If the ACAS X information is implemented on a variable range basis and the display range is changed while traffic information is displayed, the newly selected range **shall** (1598) be retained when the display reverts back to the display of weather information.
- b. WX-and-Traffic Mode. A TA or RA condition **shall** (1599) cause the traffic information to be displayed and **shall** (1600) preempt any other shared services (e.g., checklists or ACARS information). It is permissible for navigation information to remain displayed. If traffic is displayed at the same location as weather information, the traffic information **shall** (1601) be readily discernible and readable. The traffic information **shall** (1602) be displayed at the range selected for the weather radar.

Note: It is recommended that the range selection for both ACAS X and the weather information be combined into a single switch, or series of push buttons.

If a TA or RA is issued for an aircraft that is between the three o'clock and the nine o'clock positions (in a clockwise direction), a written message of “TA BEHIND” or “RA BEHIND” **shall** (1603) be annunciated on the display to alert the pilot a change in display configuration is required to display the traffic. The message “TA BEHIND” **shall** (1604) be written in amber or yellow; the message “RA BEHIND” **shall** (1605) be written in red with the RA given priority for simultaneous advisories. As an option, the message “TRAFFIC” or “TFC” may be used and **shall** (1606) be written in amber or yellow when a TA is displayed and in red when an RA is displayed.

This mode may be implemented on either a part time (see §2.2.6.1.2.4) or full time (see §2.2.6.1.2.5) basis. This mode requires the implementation of a variable range traffic display.

In addition to the modes described above, optional display modes may be provided, as follows. If these modes are provided, they **shall** (1607) adhere to the following requirements.

- a. WX-Only Mode. In this mode, no ACAS X information **shall** (1608) be displayed without a crew action. The TA and RA aural annunciations (see §2.2.6.3) will alert the crew that traffic information is available for display. The crew **shall** (1610) be able to display the traffic at a pre-set or defaulted range scale with a single action. When the encounter is completed, the display **shall** (1611) revert back to WX-Only mode at the previously selected range.
- b. Traffic-Only Mode. In this mode, traffic within the selected range and altitude band (see §2.2.6.1.2.1.10) **shall** (1612) be displayed on a full time basis. This mode may be implemented using either a fixed range display (see §2.2.6.1.2.2) or a variable range display (see §2.2.6.1.2.3).

2.2.6.1.2.7.1.2 Ownership Symbol and Location

The ownership symbol **shall** (1613) be located in the following position for each of the required and optional display modes.

- a. In the WX/ACAS X mode, the ownership symbol **shall** (1614) be centered horizontally on the display and approximately 1/3 of the height from the bottom of the display.

- b. In the WX-and-Traffic mode, a unique ownship symbol **shall** (1615) not be displayed. The ownship reference of the weather radar **shall** (1616) be used as the ownship symbol.
- c. In the Traffic-Only mode, the ownship symbol **shall** (1617) be centered horizontally on the display and approximately 1/3 of the height from the bottom of the display.

2.2.6.1.2.7.2 Shared Navigation Displays (ND) or Electronic Horizontal Situation Indicator (EHSI)

2.2.6.1.2.7.2.1 Display Characteristic

The display of ACAS X information may be provided on either a part-time (see §2.2.6.1.2.4) or a full time (see §2.2.6.1.2.5) basis.

If desired, the pilot can be provided with a means of selecting whether the display of traffic is shown on a part time or a full time basis.

2.2.6.1.2.7.2.2 Failure Annunciations

ACAS X failures **shall** (1618) be annunciated in accordance with §2.2.6.6 on the ND or EHSI if the display of traffic has been selected.

2.2.6.1.2.7.2.3 Traffic Display

ACAS X traffic information **shall** (1619) be displayed in the MAP mode. ACAS X traffic information may be displayed in the EXPANDED, North-up, and horizontal situation indicator (HSI) modes.

Note: Sample depictions of the MAP and HSI modes are shown in Figure 2-19. The exact nomenclature of the display modes implemented differs from manufacturer to manufacturer. Thus, all references made to specific display implementations in this subparagraph also apply to display implementations having similar characteristics, but different names.

When a mode is selected which does not display traffic, a written message of either “TRAFFIC”, “TFC”, or “TCAS” **shall** (1620) be displayed to indicate a TA or an RA is in progress. The message **shall** (1621) be written in amber or yellow for a TA and in red for an RA with the RA given priority for simultaneous advisories. Provisions **shall** (1622) be provided which can be used by the pilot to select a display mode capable of displaying the traffic symbology in implementations using the words “TRAFFIC”, “TFC”, or “TCAS”.

2.2.6.1.2.7.2.4 ACAS X and Weather Information

When a mode is selected which displays both ACAS X traffic and weather information, the traffic information **shall** (1623) be readily discernible.

2.2.6.1.2.7.2.5 ACAS X and Navigation Information

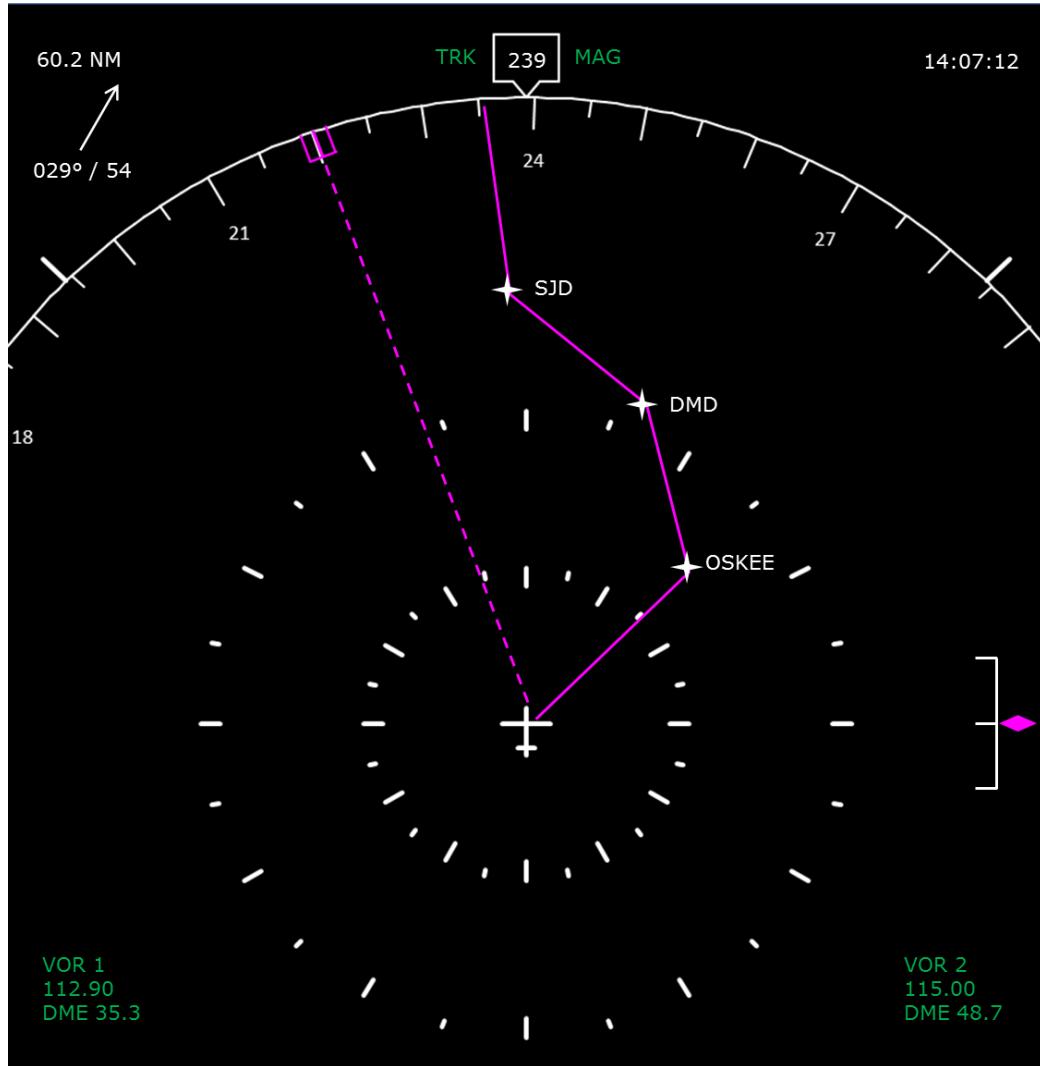
Whenever ACAS X information is displayed at the same location as navigation data, it **shall** (1624) be readily discernible and readable.

2.2.6.1.2.7.2.6 HSI Display Mode

2.2.6.1.2.7.2.6.1 Display Characteristics

If traffic information is displayed while the HSI mode is selected, it may be implemented on either a part time (see §2.2.6.1.2.4) or full time (see §2.2.6.1.2.5) basis.

Note: The HSI mode requirements apply to both the display of ILS and VOR navigation data.



Map Display Mode

Figure continues on next page.

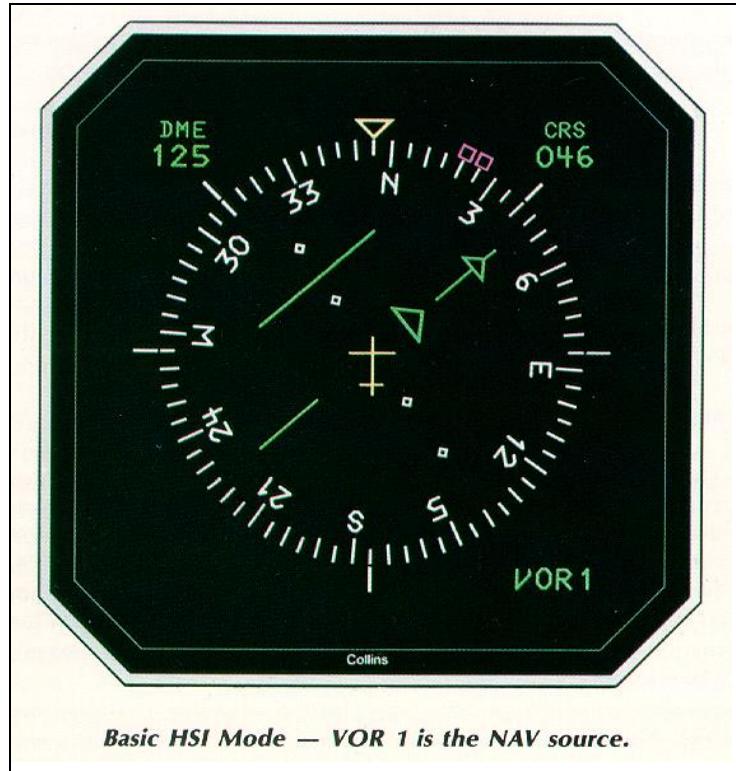


Figure 2-19: Sample Navigation Display Modes

2.2.6.1.2.7.2.6.2 Ownship Symbol and Location

A unique ownship symbol **shall** (1625) not be displayed for ACAS X. The ownship symbol of the HSI, located at the center of the HSI, **shall** (1626) be used for ACAS X.

2.2.6.1.2.7.2.7 EXPANDED Mode

2.2.6.1.2.7.2.7.1 Display Characteristics

If traffic information is displayed when the EXPANDED mode is selected, it may be implemented on either a part time (see §2.2.6.1.2.4) or full time (see §2.2.6.1.2.5) basis. If traffic information is displayed in this mode, it **shall** (1627) not be with a fixed range.

Note: The EXPANDED mode requirements apply to both the display of ILS and VOR navigation data.

2.2.6.1.2.7.2.7.2 Display Range and Range Ring

The range of the ACAS X display **shall** (1628) be consistent with the navigation information.

Unless range rings (or markings) are displayed when the EXPANDED mode is selected, a two or three mile range arc **shall** (1629) be provided in a manner which does not interfere with the displayed navigation information.

2.2.6.1.2.7.2.7.3 Ownership Symbol and Location

A unique ownship symbol **shall** (1630) not be displayed. The ownship reference of the selected display mode **shall** (1631) be used as the ownship symbol.

2.2.6.1.2.7.2.8 MAP Display Mode

2.2.6.1.2.7.2.8.1 Display Characteristics

Traffic information may be displayed on either a part time (see §2.2.6.1.2.4) or a full time (see §2.2.6.1.2.5) basis. Traffic information **shall** (1632) not be displayed with a fixed range.

2.2.6.1.2.7.2.8.2 Display Range and Range Rings

The range **shall** (1633) be selected via the mode control panel for the MAP display. If the MAP display provides range markings for the navigation information, dedicated range rings or arcs for the traffic information are not required.

Automatic range reversion to a lower display range when a TA or RA is issued **shall** (1634) not be implemented.

2.2.6.1.2.7.2.8.3 Display Orientation

ACAS X traffic information **shall** (1635) be available in a heading up orientation when a TA or RA is displayed.

2.2.6.1.2.7.2.8.4 Ownership Symbol and Location

A unique ownship symbol **shall** (1636) not be displayed. The ownship reference on the MAP **shall** (1637) be used as the ownship symbol.

2.2.6.1.2.7.2.9 North-up Mode

2.2.6.1.2.7.2.9.1 Display Characteristics

If traffic information is displayed when a North-up mode is selected, it may be implemented on either a part time (see §2.2.6.1.2.4) or full time (see §2.2.6.1.2.5) basis. If traffic information is displayed in this mode, it **shall** (1638) not be with a fixed range.

2.2.6.1.2.7.2.9.2 Display Range and Range Rings

The range of the ACAS X display **shall** (1639) be consistent with the navigation information.

If the North-up provides range markings for the navigation information, dedicated range rings or arcs for the traffic information are not required.

2.2.6.1.2.7.2.9.3 Ownership Symbol

If traffic information is displayed when a North-up mode is selected, an ownship symbol **shall** (1640) be displayed. The ownship symbol **shall** (1641) be shown with the front of the symbol oriented in the direction of the ownship's heading.

If an ownship symbol is not depicted in a North-up mode, e.g., a waypoint provided by a navigation system is centered on the display, traffic information **shall** (1642) not be displayed.

2.2.6.1.2.7.2.9.4 Display Orientation

If traffic information is displayed when a North-up mode is selected, the traffic **shall** (1643) be displayed referenced to the front of the ownship symbol.

2.2.6.1.2.7.2.9.5 TA or RA Annunciation

When a TA or RA is issued while a display is in a North-up mode, the written messages defined in §2.2.6.1.2.7.2.3 **shall** (1644) be displayed.

2.2.6.1.2.7.2.9.6 Selection of Heading Up Display

Provisions **shall** (1645) be provided so that a pilot can switch from a North-up mode to a mode showing traffic in a heading up orientation with a single action when a TA or RA is issued.

2.2.6.1.2.7.3 Shared EICAS/SYSTEMS Displays

Note: The primary functions of these displays are to provide engine instrumentation; graphical representation of aircraft system configurations; and information to assist the crew in resolving anomalous conditions with on board systems. The exact nomenclature of these displays differs from airframe manufacturer to manufacturer. Thus, all references made to specific display implementations known as EICAS/SYSTEMS in this subparagraph also apply to display providing the same types of information, but referred to by a different name.

2.2.6.1.2.7.3.1 Display Characteristics

Traffic information may be displayed on either a part time (see §2.2.6.1.2.4) or full time (see §2.2.6.1.2.5) basis.

Traffic information may be displayed with either a fixed range (see §2.2.6.1.2.2) or with a variable range (see §2.2.6.1.2.3).

2.2.6.1.2.7.3.2 Ownship Symbol and Location

The ownship symbol **shall** (1646) be centered horizontally on the display or display area and approximately 1/3 of the height from the bottom of the display or display area.

2.2.6.1.2.7.3.3 Traffic Display Inhibit

The pilot **shall** (1647) be provided with the capability to override the display of ACAS X information at anytime and return the EICAS/SYSTEMS display to its normal operation.

The ACAS X traffic display **shall** (1648) be inhibited if one or more EICAS/SYSTEMS displays is inoperative.

2.2.6.1.2.7.3.4 Restrictions

The display of traffic **shall** (1649) not be shared with the primary engine display.

2.2.6.1.2.7.4 TA/RA/VSI Displays

2.2.6.1.2.7.4.1 General Requirements

2.2.6.1.2.7.4.1.1 Format

This display combines the functions of a vertical speed indicator (VSI) with the RA display described in §2.2.6.2 and the traffic display described in §2.2.6.1.2 through §2.2.6.1.2.6.

This type of display **shall** (1650) be implemented only on round dial presentations of the VSI.

2.2.6.1.2.7.4.1.2 VSI Requirements

The length and width of the VSI needle **shall** (1651) be adequate to support quick and accurate interpretation of the actual vertical rate and vertical rate trends by peripheral vision only. The stem of the VSI needle **shall** (1652) overlap traffic information, but still allow traffic information to be readable.

2.2.6.1.2.7.4.2 Traffic Display Function

2.2.6.1.2.7.4.2.1 Display Characteristics

Traffic information may be displayed on either a part time (see §2.2.6.1.2.4) or full time (see §2.2.6.1.2.5) basis.

Traffic information may be displayed with either a fixed range (see §2.2.6.1.2.2) or with a variable range (see §2.2.6.1.2.3).

2.2.6.1.2.7.4.2.2 Ownership Symbol and Location

The ownership symbol **shall** (1653) be centered horizontally on the display and approximately 1/3 of the height from the bottom of the available display area.

2.2.6.1.2.7.4.2.3 Interference with Vertical Speed Scale

Traffic information (symbols, data blocks, vertical trend arrows, annunciations, or No-Bearing advisories) **shall** (1654) not overwrite any portion of the vertical speed scale.

2.2.6.2 RA Displays

2.2.6.2.1 General

Note: The RA display provides guidance on the vertical speed or pitch angle to be flown, and the range of vertical speeds or pitch angles to be avoided, to attain or maintain the desired vertical miss distance from an aircraft causing an RA.

2.2.6.2.2 RA/VSI (Round dial VSI)

This implementation **shall** (1655) indicate the vertical speeds to be flown and avoided using a series of red, green, and black arcs displayed around the periphery of the VSI.

Note: The term "black arcs" refers to the area of the VSI scale, usually the background of the scale, that is not illuminated by the lighted red and green arcs.

The scale of the VSI **shall** (1656) have sufficient range to display the required red and green arcs for all RAs which can be generated by the collision avoidance logic. This will require a range of ± 6000 fpm.

2.2.6.2.2.1 RA Display Characteristics

2.2.6.2.2.1.1 Red Arcs

The red arcs on the RA/VSI indicate the vertical speed range that must be avoided to maintain or attain the ACAS X-desired vertical miss distance from one or more intruders. The red arcs **shall** (1659) have the capability of displaying a resolution no larger than 500 fpm for maintain rate RAs issued by the CAS logic. If the display is capable of displaying a finer resolution, it **shall** (1660) be used. The red arcs **shall** (1661) accurately depict vertical speed restrictions for all RAs shown in Table 2-49. The red arcs **shall** (1662) be readily discernible and distinguishable. The length of the red arc **shall** (1663) be adjusted as appropriate when the RA is strengthened or weakened by the CAS logic.

2.2.6.2.2.1.2 Green Arcs

A green “fly-to” arc **shall** (1664) be used to provide a target vertical speed whenever a change in the existing vertical speed is desired or when an existing vertical speed (not less than 1500 fpm) must be maintained.

The nominal size of the green arc **shall** (1666) be approximately that defined by the distance between the 1500 and 2000 fpm marks on the VSI scale. The size of the green arc **shall** (1667) remain constant no matter where the arc is placed on the display, with the exception of the multi-aircraft encounter described in §2.2.6.2.2.1.4.

The green arcs **shall** (1668) be readily discernible and distinguishable. In addition, the green arc **shall** (1669) either be wider than the red arc or offset from the red arc to assist in visually differentiating between the red and green arcs.

The green arc **shall** (1670) remain displayed for the entire duration of the RA. Its position **shall** (1671) move to the appropriate position when an RA is strengthened, weakened or reversed by the CAS logic.

Note: For Maintain Rate RAs the following formulae could be used to determine the extent of the green arc:

*vertical rate in the range $zdot = 1,500$ fpm to $3,200$ fpm – green arc extends to $((20 * zdot) + 4000)/17$;*

*vertical rate in the range $zdot = 3,200$ fpm to $4,400$ fpm – green arc extends to $((0.0004085 * zdot^2) - (1.438 * zdot) + 4419)$.*

Round results to the nearest integer.

2.2.6.2.2.1.3 Black Arcs

The portions of the VSI scale not covered by either a red or green arc **shall** (1672) remain black.

2.2.6.2.2.1.4 Multi-aircraft Encounters

For the special situation where a multi-aircraft encounter results in an RA where neither a climb nor descent is permitted, a green arc **shall** (1673) be displayed from approximately -250 fpm to +250 fpm. The remainder of the VSI scale **shall** (1674) be illuminated with red arcs.

2.2.6.2.2.2 VSI Characteristics

Note: It is recommended that inertial quickening of the vertical speed function be provided.

2.2.6.2.2.3 Lighting Control

The lighting intensity of the red and green arcs **shall** (1675) either be automatic or connected to an adjustable lighting control for other similar alerting indicators. For night operations, the brightness should be controllable to a dim enough setting such that outside vision is not impaired while maintaining an acceptable presentation. For day operations, the display **shall** (1676) be readable in all sunlight conditions described in Paragraph 16.a.(3) of reference H. The lighting controls **shall** (1677) meet the requirements of reference I, §5.5.

2.2.6.2.3 RA/VSI (Integrated Tape VSI on a Primary Flight Display [PFD])

This implementation **shall** (1678) indicate the vertical speeds to be flown and avoided using a series of red, green, and black zones displayed within the vertical speed tape portion of the PFD.

Notes:

1. *The requirements of this subparagraph apply only when the sole means of displaying RA guidance information is shown on an Integrated Tape VSI on the PFD.*
2. *The term “black zones” refers to the area of the VSI scale, usually the background of the scale, that is not covered by either a red or green zone.*

2.2.6.2.3.1 RA Display Characteristics

2.2.6.2.3.1.1 Red Zone

The red zone on the RA/VSI **shall** (1679) indicate the vertical speed range that must be avoided to maintain or attain the ACAS X-desired vertical miss distance from one or more intruders. The red zone **shall** (1680) be readily discernible and distinguishable. The length (height) of the red zone **shall** (1681) be adjusted as appropriate when the RA is strengthened or weakened by the CAS logic.

2.2.6.2.3.1.2 Green Zone

A green “fly-to” zone **shall** (1682) be used to provide a target vertical speed whenever a change in the existing vertical speed is desired and when an existing vertical speed (not less than 1500 fpm) must be maintained.

The nominal size of the green zone **shall** (1684) be approximately that defined by the distance between the 1500 and 2000 fpm marks on the VSI scale. The size of the green zone **shall** (1685) remain constant no matter where the zone is placed on the display, with the exception of the multi-aircraft encounter described in §2.2.6.2.2.1.4.

The green zone **shall** (1686) be readily discernible and distinguishable. In addition, the green zone **shall** (1687) be wider than the red zone to assist in visually differentiating between the red and green zone.

The green zone **shall** (1688) remain displayed for the entire duration of the RA. Its position **shall** (1689) move to the appropriate position when an RA is strengthened, weakened or reversed by the CAS logic.

2.2.6.2.3.1.3 Black Zones

The portions of the VSI scale not covered by either a red or green zone **shall** (1690) remain black.

2.2.6.2.3.1.4 Multi-aircraft Encounters

For the special situation where a multi-aircraft encounter results in an RA where neither a climb nor descent is permitted, a green zone **shall** (1691) be displayed from approximately -250 fpm to +250 fpm. The remainder of the VSI scale **shall** (1692) be illuminated with red zones.

Note: When the RA is corrective, it is also acceptable to allow a nominal length green arc beginning at 0 fpm in the corrective sense.

2.2.6.2.3.2 VSI Characteristics

Note: It is recommended that inertial quickening of the vertical speed function be provided.

2.2.6.2.3.3 VSI Scale

The scale of the VSI tape **shall** (1693) have sufficient range to display the required red and green zones for all RAs which can be generated by the collision avoidance logic. This will require a range of ± 6000 fpm.

2.2.6.2.3.4 Lighting Control

The lighting intensity of the red and green zones **shall** (1695) either be automatic or connected to an adjustable lighting control for other similar alerting indicators. For night operations, the brightness should be controllable to a dim enough setting such that outside vision is not impaired while maintaining an acceptable presentation. For day operations, the display **shall** (1696) be readable in all sunlight conditions described in Paragraph 16.a.(3) of reference H.

2.2.6.2.4 Pitch Cues on the PFD

This implementation **shall** (1697) indicate the pitch angles to be flown and avoided while responding to an RA using the symbology as follows. A representation of a typical pitch cue implementation on the PFD is shown in Figure 2-20.

Notes:

1. *The requirements of this subparagraph are applicable whenever pitch cues are used to provide RA guidance information. This subparagraph does not require the implementation of pitch cues for RA guidance on all PFDs.*
2. *In this document, the term “PFD” refers to the display used to show the aircraft’s relationship to the horizon. The requirements for a PFD are also applicable to similar types of displays that may be referred to by different nomenclature.*

3. Similar implementations have been used where a PFD is using a Flight Path Vector (FPV) or Path Cues. The requirements for when the path cues are used match the requirements for the pitch cues.

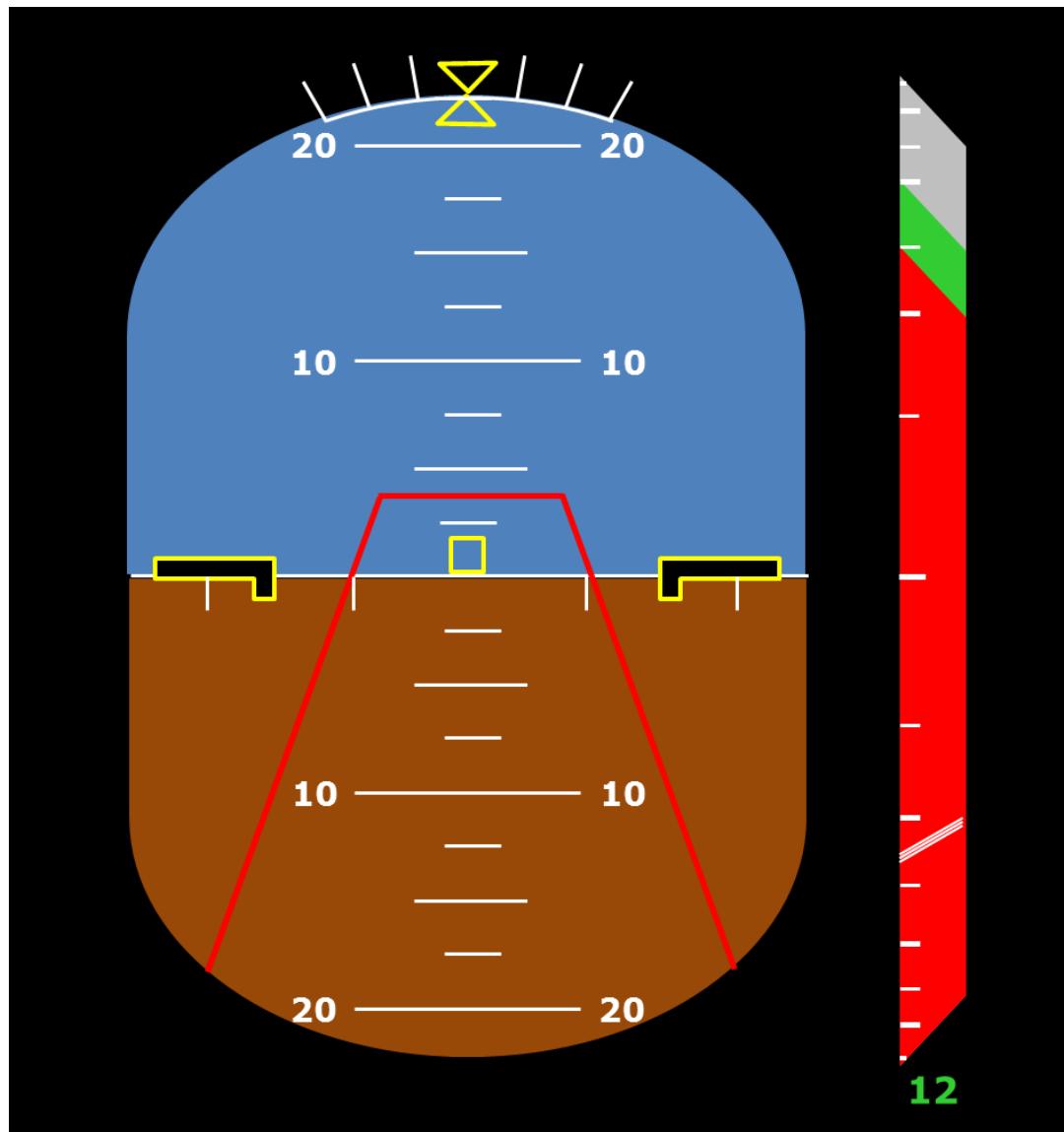


Figure 2-20: RA Pitch Cues on a PFD

2.2.6.2.4.1 No-Fly Pitch Angles

A red trapezoid overlaying the other information on the PFD **shall** (1698) indicate the range of pitch angles that must be avoided to maintain or attain the ACAS X-desired vertical miss distance from one or more intruders. When the trapezoid is displayed, the other information shown on the PFD **shall** (1699) remain readily discernible and readable. The trapezoid **shall** (1700) begin at the bottom of the PFD and extend upward to the desired pitch angle for up sense RAs; for down sense RAs, the trapezoid **shall** (1701) begin at the top of the PFD and extend downwards to the desired pitch angle. The closed end of the trapezoid **shall** (1702) correspond to the pitch angle which will provide the vertical speed desired by ACAS X.

The use of a trapezoid in this subparagraph is not intended to preclude the use of other geometric shapes, e.g., a rectangle, to depict the pitch angles to be avoided to attain or maintain the ACAS X-desired vertical miss distance. If other geometric shapes are used to display ACAS X RA guidance, the requirements shown here for trapezoids **shall** (1703) be met.

The red trapezoid **shall** (1704) remain displayed for the entire duration of the RA. It **shall** (1705) move as appropriate when an RA is strengthened, weakened or reversed by the CAS logic.

The symbology used in the ACAS X trapezoid **shall** (1706) be readily discernible and distinguishable from any other information displayed on the PFD.

The use of a green “fly-to” target is permitted to provide a target pitch angle whenever a change in the existing vertical speed is desired. If implemented, a green band or box **shall** (1707) be displayed at the top (up sense RA) or bottom (down sense RA) of the red trapezoid for all RAs except initial preventive RAs. If implemented, the green pitch target **shall** (1708) be readily discernible and distinguishable, and **shall** (1709) remain displayed for the entire duration of the RA. It **shall** (1710) move to the appropriate position when an RA is strengthened or weakened by the CAS logic.

When the combination of the Up Advisory bits (Climb_RA) and the Down Advisory bits (Descend_RA) indicate Do Not Descend (VSL 0) and Do Not Climb (VSL 0) respectively (a dual VSL 0 RA) the display **shall** (1755) leave sufficient room between the two red trapezoids for the ownship reference symbol. If a green “fly-to” target is used, it **shall** (1754) be displayed for this case even though the RA is classified as preventive.

2.2.6.2.4.2

Mode Annunciation

When an RA is displayed, a written annunciation of “TCAS”, “TFC”, or “TRAFFIC”, written in red, **shall** (1711) be displayed in the primary field of view of each pilot. The exact implementation of the annunciation **shall** (1712) be compatible with the implementation of other mode annunciations.

2.2.6.2.4.3

Multi-aircraft Encounters

For the special situation where a multi-aircraft encounter results in an RA where neither a climb nor descent is permitted, two red trapezoids **shall** (1713) be simultaneously displayed. One **shall** (1714) begin at the top of the PFD and extend downwards to the pitch angle that will result in level flight, while the other **shall** (1715) begin at the bottom of the PFD and extend upwards to the pitch angle that will result in level flight. There **shall** (1716) be sufficient room left between the two trapezoids to permit the ownship reference on the PFD to fit between the two trapezoids.

2.2.6.2.5

Flight Director Guidance

Refer to §3.2.13 Integration with Aircraft Flight Controls

2.2.6.2.6

Heads-up Display

This implementation **shall** (1719) indicate the vertical flight path to be flown and avoided using a unique display symbology on the HUD.

2.2.6.2.6.1 No-Fly Zone

A trapezoid overlaying the other information on the HUD shall (1720) indicate the flight path that must be flown to maintain or attain the ACAS X-desired vertical miss distance from one or more intruders. When the trapezoid is displayed, the other information shown on the HUD shall (1721) remain readily discernible and readable. The trapezoid shall (1722) begin at the bottom of the display and extend upward to the desired flight path angle for up sense RAs; for down sense RAs, the trapezoid shall (1723) begin at the top of the display and extend downward to the desired flight path angle. The closed end of the trapezoid shall (1724) correspond to the flight path which will provide the vertical speed desired by ACAS X.

The ACAS X trapezoid shall (1725) be readily discernible and distinguishable from any other information displayed on the HUD.

The trapezoid shall (1726) remain displayed for the entire duration of the RA. It shall (1727) move as appropriate when an RA is strengthened or weakened by the CAS logic.

2.2.6.2.6.2 Mode Annunciation

When an RA is displayed, the HUD shall (1728) display a written annunciation of “TCAS” or “TRAFFIC”. The exact implementation of the annunciation shall (1729) be compatible with the implementation of other mode annunciations on the HUD.

2.2.6.2.6.3 Flight Path Target

A flight path target shall (1730) be provided whenever a change in the existing vertical speed is desired. A box consisting of lines twice the width of the lines comprising the trapezoid shall (1731) be displayed at the top (up sense RA) or bottom (down sense RA) of the trapezoid for all RAs except initial preventive RAs.

The flight path target shall (1732) be readily discernible and distinguishable.

The flight path target shall (1733) remain displayed for the entire duration of the RA. It shall (1734) move to the appropriate position when an RA is strengthened, weakened or reversed by the CAS logic.

2.2.6.2.6.4 Multi-aircraft Encounters

For the special situation where a multi-aircraft encounter results in an RA where neither a climb nor descent is permitted, two trapezoids shall (1735) be simultaneously displayed. One shall (1736) begin at the top of the display and extend downwards to the flight path angle that will result in level flight, while the other shall (1737) begin at the bottom of the display and extend upwards to the flight path angle that will result in level flight. There shall (1738) be sufficient room left between the two trapezoids to display the flight path target box. This distance shall (1739) nominally permit the ownship reference on the HUD to fit between the two trapezoids.

2.2.6.2.6.5 Display Decluttering

When an RA is displayed, the display may be automatically decluttered by removing certain data and symbols. Items such as ground speed, mach, wind speed and direction, digital heading, digital selected course, and digital selected heading may be removed if this does not interfere with the pilot's ability to comply with the RA or operate the aircraft in compliance with the appropriate regulations and requirements. The navigation data, as well as the guidance cue, shall (1740) remain displayed at all times.

2.2.6.2.6.6 RA Guidance Availability

RAs **shall** (1741) be displayed in all display modes.

2.2.6.3 Aural Annunciations

2.2.6.3.1 General

2.2.6.3.1.1 Announcement Methodology

Aural alerts **shall** (1742) be presented by voice announcements only.

2.2.6.3.1.2 Aural Annunciation Generation

An aural annunciation **shall** (1743) be generated when the first RA of an encounter is displayed and each time a subsequent change in the advisory is displayed (strengthened, weakened or reversed). An aural annunciation **shall** (1744) also be provided to indicate that ownship is clear of conflict with all threatening aircraft.

Note: The CAS logic determines when any RA will be displayed and will provide the information required to initiate the aural annunciation.

2.2.6.3.1.3 Aural Annunciation Description and Interruption

The aural annunciations used **shall** (1745) be the aural annunciations as described in Table 2-49. The CAS logic may set a flag to indicate that a new aural annunciation condition, e.g., "Increase Climb, Increase Climb", has occurred. If this occurs during an aural annunciation, the old annunciation should be interrupted before it is completed, and the new aural annunciation declared in conformance with Table 2-49.

2.2.6.3.2 Quality

Aural alerts **shall** (1747) be announced in a high fidelity, distinguishable voice that is audible in all expected flight deck ambient noise conditions. Automatic volume adjustment for ambient conditions is highly recommended. If automatic volume adjustment is implemented, the volume level of the automatic adjustment **shall** (1748) provide a range of 0 to 4 watts RMS at 1000 Hz into 8 ohms for a speaker output and **shall** (1749) provide a range of 0 to 40 milliwatts at 1000 Hz into a 600 ohm audio distribution system.

2.2.6.3.3 Aural Annunciation Inhibit

Note: Aural annunciations are inhibited by annunciations issued by higher priority systems (e.g. wind shear systems and ground proximity warning systems) as indicated in §2.2.7.2.8, and below an altitude determined by the CAS logic.

2.2.6.3.4 TAs

When a TA is initially issued, the aural annunciation "TRAFFIC-TRAFFIC" **shall** (1750) be spoken once. For a subsequent TA (when an initial TA already exists and has been annunciated with "Traffic Traffic") it is acceptable to announce it with either "Traffic Traffic" or just a single "Traffic". No aural annunciation **shall** (1751) be issued when an RA against an intruder reverts to a TA at the end of an encounter. Additionally, no aural annunciation **shall** (2230) be made for a TA issued when an RA against another intruder is active.

The “TRAFFIC-TRAFFIC” annunciation **shall** (1752) be pre-empted by any annunciation associated with an RA.

2.2.6.3.5 RAs

The annunciations shown in Table 2-49 are spoken whenever the indicated RA is issued by ACAS X.

The annunciations for an RA reversal and for an increase rate RA indicate the previously announced RA has reversed or been increased in strength, respectively. These aural annunciations **shall** (1753) be spoken with a sense of urgency.

Table 2-49 provides the mapping between the aural annunciations, the type of RA, CRS Variables, and the contents of ARINC Word Label 270.

2.2.6.4 Visual Alerts

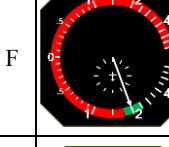
2.2.6.4.1 TAs

Note: An independent amber or yellow visual alert, i.e., a dedicated glare shield or instrument panel light, for TAs is optional.

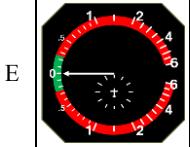
Table 2-49: Mapping Between RA, Aural Annunciation, and ARINC Word 270 Contents

	Advisory	Green Arc / Target ²	Aural Annunciation ³	Crossing Flag ⁴	IVSI Display Example ⁵	ARINC WORD LABEL 270 CONTENTS ¹												MTIE ⁷	ARA SUBFIELD								Reserved	Low-level Descend Inhibit (LDI) ¹⁷	RA Message Format (RMF) ¹⁸						
						Rate to Maintain ⁶ (Bits 11-17) (fpm)			Combined Control		Vertical Control		Up Advisory		Down Advisory				Single Intent ⁸			Crossing ⁴		Down/Up ¹⁵		Strength ⁹									
						18	19	20	21	22	23	24	25	26	27	28	29		60	41	42	43	44	45	46	47									
1	Climb	Y	Climb	F		+1500	0	0	1	0	0	0	1	0	0	0	0	E	E	0	0	0	1	0	0	0	0	EE	01						
2	Descend	Y	Descend	F		-1500	1	0	1	0	0	0	0	0	0	1	0	0	E	E	0	1	0	1	0	0	0	EE	01						
3	Altitude Crossing Climb	Y	Climb, Crossing Climb	T		+1500	0	0	1	1	0	0	1	0	0	0	0	E	E	1	0	0	1	0	0	0	0	EE	01						
4	Altitude Crossing Descend	Y	Descend, Crossing Descend	T		-1500	1	0	1	1	0	0	0	0	0	1	0	E	E	1	1	0	1	0	0	0	0	EE	01						
5	Do Not Descend (Issued while descending) ¹⁰	Y	Level-off	E		0	0	0	1	0	0	0	0	1	0	0	E	E	E	0	0	1	1	0	0	1	0	EE	01						

	Advisory	Green Arc / Target ²	Aural Annunciation ³	Crossing Flag ⁴	IVSI Display Example ⁵	ARINC WORD LABEL 270 CONTENTS ¹										MTE ⁷	ARA SUBFIELD								Low-level Descend Inhibit (LDI) ¹⁷	RA Message Format (RMF) ¹⁸			
						Rate to Maintain ⁶ (Bits 11-17) (fpm)		Combined Control			Vertical Control			Up Advisory			Down Advisory			60	41	42	43	44	45	46	47		
18	19	20	21	22	23	24	25	26	27	28	29	60	41	42	43	44	45	46	47	48-50	51-52	53-54							
6	Do Not Climb (Issued while climbing) ¹⁰	Y	Level-off	E		0	1	0	1	0	0	0	0	0	0	0	1	0	E	E	E	1	0	0	1	1	0	EE	01
7	RA Reversal (Descend to Climb)	Y	Climb, Climb NOW	E		+1500	0	0	1	0	1	0	1	0	0	0	0	0	E	E	E	0	0	1	0	1	0	EE	01
8	RA Reversal (Climb to Descend)	Y	Descend, Descend NOW	E		-1500	1	0	1	0	1	0	0	0	0	1	0	0	E	E	E	1	0	1	0	1	0	EE	01
9	RA Reversal (Descend to Maintain Climb Rate) ¹¹	Y	Climb, Climb NOW	E		Existing V/S	0	0	1	0	1	0	1	0	0	0	0	E	E	E	0	1	0	0	0	0	EE	01	
10	RA Reversal (Climb to Maintain Descent Rate) ¹¹	Y	Descend, Descend NOW	E		Existing V/S	1	0	1	0	1	0	0	0	0	1	0	0	E	E	E	1	1	0	0	0	0	EE	01
11	Increase Climb	Y	Increase Climb	E		+2500	0	0	1	1	1	0	1	0	0	0	0	E	E	E	0	0	1	1	0	0	EE	01	

	Advisory	Green Arc / Target ²	Aural Annunciation ³	Crossing Flag ⁴	IVSI Display Example ⁵	ARINC WORD LABEL 270 CONTENTS ¹										MTE ⁷	ARA SUBFIELD								Low-level Descend Inhibit (LDI) ¹⁷	RA Message Format (RMF) ¹⁸				
						Rate to Maintain ⁶ (Bits 11-17) (fpm)		Combined Control			Vertical Control			Up Advisory			Down Advisory			Single Intent ⁸		Crossing ⁴		Down/Up ¹⁵		Strength ⁹				
18	19	20	21	22	23	24	25	26	27	28	29	60	41	42	43	44	45	46	47	48-50	51-52	53-54								
12	Increase Descent	Y	Increase Descent	E		-2500	1	0	1	1	1	0	0	0	0	1	0	0	E	E	E	1	0	1	1	0	0	EE	01	
							5		3			0			1							6								
13	Maintain Climb Rate ¹¹	Y	Climb	F		Existing V/S	0	0	1	0	0	1	1	0	0	0	0	0	0	E	E	0	0	0	1	1	1	0	EE	01
14	Maintain Descent Rate ¹¹	Y	Descend	F		Existing V/S	1	0	1	0	0	1	0	0	0	1	0	0	E	E	0	1	0	1	1	1	0	EE	01	
15	Altitude Crossing Maintain Climb Rate ¹¹	Y	Climb, Crossing Climb	T		Existing V/S	0	0	1	0	0	1	1	0	0	0	0	0	E	E	1	0	0	1	1	1	1	0	EE	01
16	Altitude Crossing Maintain Descent Rate ¹¹	Y	Descend, Crossing Descend	T		Existing V/S	1	0	1	0	0	1	0	0	0	1	0	0	E	E	1	1	0	0	1	1	1	0	EE	01
17	Weakening of Positive RA ¹² (After Climb)	Y	Level-off	E		0	0	1	0	0	0	0	1	0	0	0	0	E	E	E	0	0	0	1	0	0	EE	01		
							4		0			2			0							2								

	Advisory	Green Arc / Target ²	Aural Annunciation ³	Crossing Flag ⁴	IVSI Display Example ⁵	ARINC WORD LABEL 270 CONTENTS ¹										MTE ⁷	ARA SUBFIELD								Low-level Descend Inhibit (LDI) ¹⁷	RA Message Format (RMF) ¹⁸	
						Rate to Maintain ⁶ (Bits 11-17) (fpm)		Combined Control			Vertical Control			Up Advisory			Down Advisory			Single Intent ⁸	Crossing ⁴	Down/Up ¹⁵	Strength ⁹				
18	Weakening of Positive RA ¹² (After Maintain Climb Rate)	N	Monitor Vertical Speed	E		Not Displayed	0	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	EE	01
19	Weakening of Positive RA ¹² (After Descend)	Y	Level-off	E		0	1	0	1	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	EE	01
20	Weakening of Positive RA ¹² (After Maintain Descent Rate)	N	Monitor Vertical Speed	E		Not Displayed	0	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	EE	01	
21	Weakening of Descend RA ¹³ (Low Altitude Inhibit)	N	Monitor Vertical Speed	E		Not Displayed	0	1	1	0	0	0	0	0	0	0	1	0	1	0	1	0	0	10	~or~	11	01
22	Do Not Descend	N	Monitor Vertical Speed	E		Not Displayed	0	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	EE	01
23	Do Not Climb	N	Monitor Vertical Speed	E		Not Displayed	0	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	EE	01	

	Advisory	Green Arc / Target ²	Aural Annunciation ³	Crossing Flag ⁴	IVSI Display Example ⁵	ARINC WORD LABEL 270 CONTENTS ¹										MTE ⁷	ARA SUBFIELD								Low-level Descend Inhibit (LDI) ¹⁷	RA Message Format (RMF) ¹⁸		
						Rate to Maintain ⁶ (Bits 11-17) (fpm)		Combined Control			Vertical Control			Up Advisory			Down Advisory			Single Intent ⁸		Crossing ⁴		Down/Up ¹⁵		Strength ⁹		
18	19	20	21	22	23	24	25	26	27	28	29	60	41	42	43	44	45	46	47	48-50	51-52	53-54						
24	Multi-threat Level-off ^{14,15} (Issued while level)	Y	Level-off	E		0	0	1	1	0	0	1	0	1	0	0	1	0	1	1	1	0	0	EE	01			
25	Multi-threat Level-off ¹⁴ (Issued while descending)	Y	Level-off	E			0	0	0	1	0	0	0	0	1	0	0	1	0	1	1	1	1	0	EE	01		
26	Multi-threat Level-off ¹⁴ (Issued while climbing)	Y	Level-off	E		0	1	0	1	0	0	0	0	1	0	0	1	0	1	1	1	1	1	0	EE	01		
27	RA Reversal (Descend to Do Not Descend) ¹⁶ (Issued while descending)	Y	Level-off	E			0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	1	0	EE	01		
28	RA Reversal (Climb to Do Not Climb) ¹⁶ (Issued while climbing)	Y	Level-off	E		0	1	0	1	0	0	0	0	0	0	0	1	0	1	0	0	1	0	0	EE	01		
29	RA Reversal (Climb to Do Not Climb) ¹⁶ (Issued while level or descending)	N	Monitor Vertical Speed	E			Not Displayed	0	1	1	0	0	0	0	0	0	0	1	0	1	0	1	1	0	EE	01		

	Advisory	Green Arc / Target ²	Aural Annunciation ³	Crossing Flag ⁴	IVSI Display Example ⁵	Rate to Maintain ⁶ (Bits 11-17) (fpm)	ARINC WORD LABEL 270 CONTENTS ¹										MTE ⁷	ARA SUBFIELD								Low-level Descend Inhibit (LDI) ¹⁷	RA Message Format (RMF) ¹⁸		
							Combined Control			Vertical Control			Up Advisory			Down Advisory			Single Intent ⁸		Crossing ⁴		Down/Up ¹⁵		Strength ⁹				
						18	19	20	21	22	23	24	25	26	27	28	29	60	41	42	43	44	45	46	47	48-50	51-52	53-54	
30	RA Reversal (Descend to Do Not Descend) ¹⁶ (Issued while level or climbing)	N	Monitor Vertical Speed	E		Not Displayed	0	1	1	0	0	0	0	1	0	0	0	0	E	E	E	0	1	0	1	1	0	E E	0 1
							6			0			2			0			11		0		0		0		0		0 1
31	Clear of Conflict	N	Clear of Conflict	F	N/A	N/A	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	E E	0 1
						1			0 ¹⁹			0 ¹⁹			0 ¹⁹			0		0		0		0		0		0 1	

Notes for Table 2-49

1. The mapping/meaning of the ARINC Label 270 values are shown below. For the fields CC (Bits 18-20), VC (Bits 21-23), Up (Bits 24-26), and Down (Bits 27-29), the ordering of the bits are reversed from the usual ordering used in these MOPS. The reason for this reversed ordering is to comply with ARINC practice. (Bit values are read right to left.)

Combined Control Label 270 Bit Mapping

18	19	20	Octal	<u>Meaning</u>
0	0	0	= 0	No Advisory
1	0	0	= 1	Clear of Conflict
0	1	0	= 2	Spare
1	1	0	= 3	Spare
0	0	1	= 4	Up Sense Advisory Green Arc displayed
1	0	1	= 5	Down Sense Advisory Green Arc displayed
0	1	1	= 6	Monitor Vertical Speed Advisory, No Green Arc displayed, except for Multi-threat Level-off issued when level
1	1	1	= 7	Not assigned

Vertical Control Label 270 Bit Mapping

21	22	23	Octal	<u>Meaning</u>
0	0	0	= 0	Advisory is not one of the following
1	0	0	= 1	Altitude Crossing RA
0	1	0	= 2	RA Reversal
1	1	0	= 3	Increase Rate RA
0	0	1	= 4	Maintain Rate RA
1	0	1	= 5	Not assigned
0	1	1	= 6	Not assigned
1	1	1	= 7	Not assigned

Up Advisory Label 270 Bit Mapping

24	25	26	Octal	<u>Meaning</u>
0	0	0	= 0	No Up Sense RA
1	0	0	= 1	Climb RA
0	1	0	= 2	Do Not Descend RA
1	1	0	= 3	Not assigned
0	0	1	= 4	Not assigned
1	0	1	= 5	Not assigned
0	1	1	= 6	Not assigned
1	1	1	= 7	Not assigned

Down Advisory Label 270 Bit Mapping

27	28	29	Octal	<u>Meaning</u>
0	0	0	= 0	No Down Sense RA
1	0	0	= 1	Descend RA
0	1	0	= 2	Do Not Climb RA
1	1	0	= 3	Not assigned
0	0	1	= 4	Not assigned
1	0	1	= 5	Not assigned
0	1	1	= 6	Not assigned
1	1	1	= 7	Not assigned

2. A “Yes” in the “Green Arc / Target” column indicates that a green arc or fly-to target is displayed to the pilot to indicate the vertical speed or pitch guidance appropriate to the advisory. A “No” in this column indicates that no green arc / target is displayed.
3. Aural Annunciations are each sounded twice, EXCEPT for “Monitor Vertical Speed” and “Clear of Conflict” which are sounded only once
4. The “Crossing Flag” is set by the CAS logic to “True” for any altitude crossing RA and indicates that there is a crossing, regardless of whether “Crossing” is included in the aural annunciation. If the value in the column is “Either” then the Crossing Flag may have a value of either “True” or “False” but there will be NO AURAL ANNUNCIATION OF “CROSSING” regardless of the value set by the logic.

In a multi-aircraft encounter, the “Crossing Flag” may be set to either “True” or “False” depending on the encounter geometry. If complying with the RA is expected to result in ownship and one or more intruder crossing altitude, the “Crossing Flag” will be set to “True.” If complying with the RA is not expected to result in ownship and intruder(s) crossing altitude, the “Crossing Flag” will be set to “False.”

Bit 42 will be set to match the “Crossing Flag” value set by the logic, i.e., it will be set to “I” if the “Crossing Flag” is set to “True” and “0”, if the “Crossing Flag” is set to “False.” When there are multiple intruders, a “1” in this field indicates the RA issued by the logic is a crossing associated with one or more of the intruders.

5. The “IVSI Display Example” column gives an example of the vertical guidance displayed for an IVSI-type display. However, in these examples, the vertical rate needle shows a rate consistent with the RA issued. It does not necessarily indicate the current vertical rate of the aircraft. The actual vertical guidance displayed to the flight crew will vary by display type and implementation, e.g., round IVSI-type display, vertical tape-type VSI, RA pitch guidance on a Primary Flight Display, etc.
6. The “Rate to Maintain” column records the target vertical rate, specifically the ADD variable “target_rate.”
 - A value of “Existing V/S” in this column means that the current vertical rate of the aircraft (as opposed to either the vertical rate of the aircraft at the time the first RA was generated or the rate required by a previous RA) satisfies the constraints of the new RA. While the vertical rate provided by the TRM is unbounded, the vertical rate provided at the display may be any value from 0 to 6300 feet per minute with a 100 foot per minute resolution.
 - A value “Not Displayed” in this column means the appropriate vertical rate is output but the value is not shown on the onboard pilot display.
7. A value of “E” for the MTE bit indicates that the value can be either “0” or “1.” This bit will be set to “1” if there are multiple intruders or “0” if there is a single intruder.

8. A value of “E” for the Single Intent bit indicates that the value can be either “0” or “1.” This bit will be set to “1” if there is a single intruder or multiple intruders AND the RA Intents are the same (e.g., ownership is above or below all intruders), or “0” if there are multiple intruders AND the RA Intents are CONFLICTING (e.g., a vertical sandwich geometry).

9. The mapping/meaning of the ARA Subfield Strength values are shown below.

ARA Subfield Strength Bit Mapping

44	45	46	47	Value	Meaning
0	0	0	0	= 0	No Advisory / Clear of Conflict
0	0	0	1	= 1	Monitor Vertical Speed [DND (Bit 43=0) or DNC (Bit 43=1)]
0	0	1	0	= 2	Level Off Weaken DND [DND (Bit 43=0) or DNC (Bit 43=1)]
0	0	1	1	= 3	Level Off [DND (Bit 43=0) or DNC (Bit 43=1)]
0	1	0	0	= 4	Climb (Bit 43=0) or Descend (Bit 43=1)
0	1	0	1	= 5	Reversal to Climb (Bit 43=0) or Descend (Bit 43=1)
0	1	1	0	= 6	Increase Climb (Bit 43=0) or Increase Descend (Bit 43=1)
0	1	1	1	= 7	Maintain Climb Rate (Bit 43=0) or Maintain Descend Rate (Bit 43=1)
1	0	0	0	= 8	Reversal to Maintain Climb Rate (Bit 43=0) or Maintain Descend Rate (Bit 43=1)
1	0	0	1	= 9	Reversal to DND (Bit 43=0) [Issued while descending] or to DNC (Bit 43=1) [Issued while climbing]
1	0	1	0	= 10	Monitor Vertical Speed [DNC]; due to Descend Inhibit
1	0	1	1	= 11	Reversal to DND (Bit 43=0) [Issued while level or climbing] or to DNC (Bit 43=1) [Issued while level or descending]
1	1	0	0	= 12	Not assigned
1	1	0	1	= 13	Not assigned
1	1	1	0	= 14	Multi-threat Level-off (MTLO)[Issued while level]
1	1	1	1	= 15	Multi-threat Level-off (MTLO)[Issued while climbing or descending]

10. It is expected that this RA will most often be an initial RA, but rarely could occur as a subsequent RA provided that the current and previous RAs had the same vertical sense, contrasted with the case described on lines 27 and 28 for “RA Reversal (“Climb” to “Do Not Climb” or “Descend” to “Do Not Descend”).”

11. The annunciation of “Maintain Rate” RAs is changed from prior versions of TCAS, to directly annunciate the sense of the action including annunciation of “Reversal” or “Crossing”, e.g., “Climb, Climb” or “Climb, Climb NOW” or “Climb, Crossing Climb”, instead of the ambiguous “Maintain Vertical Speed.” The annunciation and Label 270 Contents are the same as for their standard rate equivalents, except for the “Rate to Maintain” given in Bits 11-17.

12. Rows 17, 18, 19, and 20 apply when the logic determines that a positive RA can safely “Weaken” to a negative RA:

- If the positive RA was a Climb (Row 17), the new RA is displayed with a green arc from 0 – 300 fpm and annunciated “Level-off.” If the positive RA was a Descend (Row 19), the new RA is displayed with a green arc from -300 – 0 fpm and annunciated as “Level-off” EXCEPT for the special case described in row 21 and Note 13. Use of the green arc helps to reduce altitude deviation.

The advisories described in Rows 17 and/or 19 are issued when the aircraft is climbing or descending. If the aircraft is already level, the advisories described in Rows 18 and/or 20 are issued.

- If the positive RA was a Maintain Climb Rate (Row 18), the new RA is displayed only by a red arc for all rates less than zero fpm, i.e. do not descend. If the positive RA was a Maintain Descent Rate (Row 20), the new RA is displayed only by a red arc for all rates greater than zero fpm, i.e., do not climb. For both cases, the new RA is annunciated as “Monitor Vertical Speed.”
 - “Increase Climb” RA does not “weaken” to “Climb” and “Increase Descent” does not weaken to “Descend.”
13. When an aircraft with an active “Descend” RA reaches the low altitude inhibit, the RA will weaken to “Do Not Climb”, indicated by a red arc for all rates greater than zero fpm, and be annunciated as “Monitor Vertical Speed.” A green arc / target is not displayed.
14. The combination of the Up Advisory bits and the Down Advisory bits in the ARINC Label 270 contents indicate “Do Not Descend” and “Do Not Climb”, respectively (a dual VSL 0 RA), annunciated as “Level-off.” Per §2.2.6.2, a green arc / target is displayed between ± 250 fpm when a VSI or VSU tape display is used. When pitch guidance is used for the display of RAs, it is recommended that a green “fly-to” target be displayed between the two RA “avoidance” trapezoids, leaving sufficient room between the two RA “avoidance” trapezoids for the ownship reference symbol.
15. The value for the “Down/Up” bit indicates the vertical sense of the advisory. A “1” indicates that the advisory has a “Down” sense and a “0” indicates that the advisory has an “Up” sense.
- A Multi-threat Level-off issued while ownship is already level indicates that the action is to maintain level flight. It is annunciated “Level-off” and a Green Arc is displayed. This advisory has neither “Down” nor “Up” sense. By convention the “Down/Up” bit will be set to “0” in this case.
16. In general, RA reversals are from a positive RA to the opposite sense positive RA. However, reversals from “Climb” to “Do Not Climb” or from “Descend” to “Do Not Descend” can occur during multi-threat encounters. The reversals from “Climb” to “Do Not Climb” can also occur at low altitudes when descend RAs are inhibited.
- The advisories described in row 27 and row 28 involve a reversal that requires a level-off because the new RA is “Do Not Climb” and ownship is climbing (row 28), or the new RA is “Do Not Descend” and ownship is descending (row 27), probably in compliance with the previous RA. These RAs are annunciated as “Level-off” and a green arc / target is displayed.
 - The advisories described in row 29 and row 30 are issued when the new RA is “Do Not Climb” and ownship is level or descending (row 29), or the new RA is “Do Not Descend” and ownship is level or climbing (row 30). In this case, ownship was probably failing to comply with the previous RA. These RAs are annunciated as “Monitor Vertical Speed” and a green arc / target is not displayed.
17. Bits 51 and 52 indicate whether low level descend inhibit costs are being applied when selecting RAs. These inhibits depend on radar altitude as described in §2.2.3.8.3.2.3.1. The coding is as follows, but note that LDI = 1 does not inhibit the continuation of increased rate descend RAs for aircraft that are descending.

LDI Bit Mapping

51	52	Value	Meaning
0	0	= 0	No RAs are inhibited
0	1	= 1	Increased rate descend RAs are inhibited
1	0	= 2	All positive descend RAs are inhibited
1	1	= 3	All RAs are inhibited

18. The "RA Message Format" is set to indicate the format used to encode and decode RA messages as shown below.

RA Message Format Bit Mapping

53	54	Value	Meaning
0	0	= 0	TCAS
0	1	= 1	ACAS Xa
1	0	= 2	ACAS Xu
1	1	= 3	Not assigned

19. For the Clear of Conflict advisory the values of vertical control, up advisory and down advisory fields retain the values associated with the previous RA.

2.2.6.4.2 RAs

A red visual alert **shall** (1756) be provided in the primary field of view for each pilot. The red arcs on an RA/VSI display will fulfill this requirement. Likewise, conspicuous illumination of the red zones on the vertical speed tape, appropriate pitch guidance, or suitable written message (termed a Visual Alert) on a PFD will fulfill this requirement.

2.2.6.4.3 Message on PFD

If a written message is shown on the PFD, it **shall** (1757) flash or otherwise be highlighted using the guidance provided by Paragraph 31.f.(4) of reference H. If a written message is shown on the PFD, it **shall** (1758) be the message shown in Table 2-49. If necessary, the message may be written on two lines. Abbreviations may be used if they are unambiguous.

If the written message "CLEAR OF CONFLICT" is shown on the PFD, it **shall** (1759) not be shown in red nor amber nor yellow.

2.2.6.4.4 Visual Alert Inhibit

Note: All RA visual alerts shown in Table 2-49 are automatically inhibited below the RA inhibit altitude as determined by the CAS logic.

2.2.6.5 Controls

2.2.6.5.1 ACAS X/Mode S Controls

Means **shall** (1760) be provided to select the following modes of operation.

- a. Operation of Mode S transponder only.

Note: The selection of this mode places ACAS X into Standby.

- b. Operation of the Mode S transponder and ACAS X in the TA/RA mode.

- c. Operation of the Mode S transponder and ACAS X in the TA-Only mode.

Note: In this mode, all RAs are inhibited by the CAS logic.

- d. ACAS X Self Test.

It is recommended that the controls for the Mode S transponder and ACAS X be located on a single control panel. The exact implementation of the ACAS X controls **shall** (1761) be compatible with the flight deck design philosophy.

The position of all control switches **shall** (1762) be readily discernible by inspection.

2.2.6.5.2 Traffic Display Controls

The traffic display controls will depend on the type of display used and the features available on the display. The following types of controls may be used as appropriate. If these controls are provided, they **shall** (1763) comply with the requirements shown in the following subparagraphs. The position of all control switches **shall** (1764) be readily discernible by inspection.

- a. Altitude Range Selector. If the traffic display provides an option for displaying proximate and other traffic with relative altitudes between ± 2700 and a maximum of ± 9900 feet (see §2.2.6.1.2.1.10), a switch **shall** (1765) be provided to permit a pilot to control the vertical range of the display. The control panel markings for this selector **shall** (1766) use the annotations of “Above”, “Normal”, and “Below”, or a suitable abbreviation of these words.
- b. Range Selector. On variable range displays (see §2.2.6.1.2.3), a switch **shall** (1768) be provided to permit the desired display range to be selected.
- c. Actual Altitude. If the traffic display provides an option for displaying the actual altitude of an intruder (see §2.2.6.1.2.1.5), a switch **shall** (1769) be provided to select between the display of relative and actual altitude in the intruder's altitude data block.
- d. Traffic Override. On shared displays (see §2.2.6.1.2.7), a switch **shall** (1770) be provided to override the traffic display and the traffic that has popped-up and is no longer required for visual acquisition, e.g., return an EICAS/SYSTEMS display to its normal function.
- e. Display Mode Selector. On displays which provide both a full-time and a part time display mode, a means **shall** (1771) be provided to permit a pilot to select the desired display mode.

2.2.6.5.3 Traffic Display Controls on Shared Weather Radar

When ACAS X traffic information is implemented on a shared weather radar display (see §2.2.6.1.2.7.1), at least one of two display modes are required. An additional two modes may be implemented.

2.2.6.5.3.1 WX/ACAS X Mode

If this mode is implemented, a means **shall** (1772) be provided to select the WX/ACAS X mode of operation (see §2.2.6.1.2.7.1.1). The position of the selection mechanism **shall** (1773) be readily discernible by inspection.

2.2.6.5.3.2 WX-and-Traffic Mode

If this mode is implemented, a means **shall** (1774) be provided to select the WX-and-Traffic mode of operation (see §2.2.6.1.2.7.1.1). The position of the selection mechanism **shall** (1775) be readily discernible by inspection.

2.2.6.5.3.3 Optional Mode Selection

If implemented on the display, a switch **shall** (1776) be provided to select the WX-Only and the ACAS X-Only mode of operation. The position of the switch **shall** (1777) be readily discernible by inspection.

2.2.6.5.4 ACAS Xo Controls

For Class 2 systems, ACAS Xo operations allow alternative CAS logic to be applied to suitable traffic. This requires both designation of the traffic and selection of the Xo mode to be applied. Undesignating traffic from ACAS Xo returns the traffic to normal CAS logic. See §2.2.8.2.1.

2.2.6.6 Status and Failure Annunciations

Note: Additional information related to status and failure annunciations, as determined by the Monitor, is shown in §2.2.7.2.4.

2.2.6.6.1 General

Visual annunciations **shall** (1781) be provided to indicate the normal operating and the failure modes of ACAS X. Additional or more specific annunciations may be provided as long as they do not present confusing or distracting information to the flight crew. The color of the annunciations **shall** (1782) conform to reference J information (level 0) or advisory (level 1) annunciation, as appropriate. The color of the level 1 annunciation **shall** (1783) be amber or yellow. All ACAS X status and failure annunciations should be consistent with any other annunciations provided by the display on which they are shown and **shall** (1784) have only one meaning for all available display modes.

2.2.6.6.2 Traffic Display Annunciations

The ACAS X Traffic Display **shall** (1785) be capable of annunciating the following operating modes and failure conditions:

- a. ACAS in Standby or turned off. (Level 0 annunciation)
- b. ACAS operating in the TA-Only mode. (Level 0 annunciation)
- c. ACAS has failed. (Level 1 annunciation)
- d. The Traffic Display is unable to display traffic. (Level 1 annunciation)
- e. A pilot initiated Self Test is in progress. (Level 0 annunciation)

2.2.6.6.3 RA Display

The RA display **shall** (1786) provide the capability to annunciate the following operating modes and failure conditions:

- a. ACAS in Standby or turned off. (Level 0 annunciation)
- b. ACAS operating in the TA-Only mode. (Level 0 annunciation)
- c. The RA display has failed. (Level 1 annunciation)
- d. ACAS has failed. (Level 1 annunciation)
- e. A pilot initiated Self Test is in progress. (Level 0 annunciation)

Notes:

1. *The annunciations shown in §2.2.6.6.2 and §2.2.6.6.3 do not have to be individually annunciated, or displayed on both the traffic and RA display, if that is inconsistent with the aircraft's warning and caution display implementation. The status of the overall ACAS X system and each display should be readily discernible to the crew.*

-
2. *Augmentation of the TA-Only annunciation shown in §2.2.6.6.2 and §2.2.6.6.3 is permissible when a TA is issued.*

2.2.6.7 Display and Control Interfaces

2.2.6.7.1 Display and Control Outputs

The ACAS X displays and controls **shall** (1787) provide the information shown in Table 2-50, Table 2-51 and Table 2-52 to the other components and functions of ACAS X.

2.2.6.7.2 Display and Control Inputs

To function in accordance with the requirements of §2.2.6, the information and data shown in Table 2-53, Table 2-54 and Table 2-55 **shall** (1788) be provided to the ACAS X displays and controls

Table 2-50: ACAS X Display and Controls Output (Control Panel)

Signal	Source/User	Range	Units	Resolution	Static/ Dynamic	Optional/ Required	Comments
Pilot Selected Sensitivity Level	Control Panel/Mode S Transponder to CAS	Standby TA-Only TA/RA		1	Dynamic	Required	This parameter is not used in ACAS X to make maneuver selection, but is carried forward to maintain interoperability with legacy TCAS avionics in operation.
4096 Code	Control Panel/Mode S	0000-7777			Dynamic	Required	This is not an ACAS X interface if separate control panels are used for ACAS X and the transponder. The 4096 code must be the same that the Mode S transponder is reporting.
Altitude Reporting	Control Panel/Performance Monitor	Altitude Reporting/Non-altitude Reporting		1	Dynamic	Required	Switch enables altitude reporting by the transponder. If transponder is not reporting altitude, ACAS X should be non-operational.
Intruder Altitude	Control Panel/Traffic Display	Relative/Actual		1	Dynamic	Optional	Selects relative or actual altitude display for the displayed intruders.
Altitude Select	Control Panel/Display	Above Normal Below		1	Dynamic	Optional	Selects the range of relative altitudes for which an intruder is displayed.
Traffic Display	Configuration Input/Traffic Display	Part Time Full Time		1	Static	Required	Defines whether traffic is displayed on a full time or pop-up basis.
ACAS X Range	Display or Control Panel/Traffic Display	0.5-127	NM	0.5	Dynamic	Optional	Selects the range displayed by the traffic display.
Self Test	Control Panel/Monitor				Dynamic	Required	Indicates pilot initiated Self Test.

Table 2-51: ACAS X Display and Controls Output (Displays)

Signal	Destination	Range	Units	Resolution	Static/ Dynamic	Optional/ Required	Comments
Traffic Display Status	Monitor	Operational/ Not Operational	Discrete	1	Dynamic	Optional	Indicates status of the traffic display.
RA Display Status	Monitor	Operational/ Not Operational	Discrete	1	Dynamic	Required	See Note.

Note: Both RA displays provide a status indication to the Monitor. If at least one RA display indicates not valid, the Monitor should indicate the appropriate failure as defined by the Monitor.

Table 2-52: ACAS X Display and Controls Outputs (Aural Annunciations)

Signal	Source	Range	Units	Resolution	Static/ Dynamic	Optional/ Required	Comments
Voice Output Data	CAS				Dynamic	Required	Check active RA field to trigger aural annunciation.

Table 2-53: ACAS X Display and Controls Inputs (RA Displays)

ADD Signal	Source	Range	Units	Resolution	Static/ Dynamic	Optional/ Required	Comments
TRMDisplayData.target_rate	CAS	±6300	fpm	±100	Dynamic	Required	The TRM outputs display rate to maintain which is unbounded; however, the displays are specified to handle at least +/- 6000 fpm, and ACAS X unit to display interfaces may only support 6300 fpm. Therefore ACAS X equipment manufacturers should be careful to limit the output display rate to account for limitations of the interface characteristic.
TRMDisplayData.cc	CAS	No_Advisory Clear_of_Conflict Corrective_Climb Corrective_Descend Preventive			Dynamic	Required	See Note 1 for meaning of expected values of signals.
TRMDisplayData.vc	CAS	Other Crossing Reversal Increase Maintain			Dynamic	Required	See Note 2 for meaning of expected values of signals.
TRMDisplayData.ua	CAS	No_Climb_RA Positive Level-Off			Dynamic	Required	See Note 3 for meaning of expected values of signals.
TRMDisplayData.da	CAS	No_Descend_RA Positive Level-Off			Dynamic	Required	See Note 4 for meaning of expected values of signals.
TRMDisplayData.crossing	CAS				Dynamic	Required	See Note 5 for meaning of expected values of signals.

ACAS X Status Annunciations	CAS and Monitor			Dynamic	Required	Indicates the status or operating mode of the ACAS X system including Standby, TA-Only, TA/RA, or Fail condition.
-----------------------------	-----------------	--	--	---------	----------	---

Notes for Table 2-53:

1. <u><i>ADD Variable and Expected Values</i></u> <i>Combined_Control_Out</i>	<u><i>Meaning</i></u> <i>Combined Control Advisory Type</i>
<i>No_Advisory</i> <i>Clear_of_Conflict</i> <i>Corrective_Climb</i> <i>Corrective_Descend</i> <i>Preventive</i>	<i>No Advisory</i> <i>Clear of Conflict</i> <i>Corrective Up Sense Advisory</i> <i>Corrective Down Sense Advisory</i> <i>Preventive Advisory</i>
2. <u><i>ADD Variable and Expected Values</i></u> <i>Vertical_Control_Out</i>	<u><i>Meaning</i></u> <i>Vertical Control</i>
<i>Other</i> <i>Crossing</i> <i>Reversal</i> <i>Increase</i> <i>Maintain</i>	<i>Advisory is not one of the following types</i> <i>Altitude Crossing RA</i> <i>RA Reversal</i> <i>Increase Rate RA</i> <i>Maintain Rate RA</i>
3. <u><i>ADD Variable and Expected Values</i></u> <i>Climb_RA</i>	<u><i>Meaning</i></u> <i>Up Sense Advisory</i>
<i>No_Climb_RA</i> <i>Positive</i> <i>Level-Off</i>	<i>No Up Sense RA</i> <i>Climb RA</i> <i>Do Not Descend RA</i>
4. <u><i>ADD Variable and Expected Values</i></u> <i>Descend_RA</i>	<u><i>Meaning</i></u> <i>Down Sense Advisory</i>
<i>No_Descend_RA</i> <i>Positive</i> <i>Level-Off</i>	<i>No Down Sense RA</i> <i>Descend RA</i> <i>Do Not Climb RA</i>

5.	<i>ADD Variable and Expected Values</i>	<i>Meaning</i>
	<i>Crossing_Out</i>	<i>Crossing Indication Flag</i>
	<i>True</i>	<i>Crossing</i>
	<i>False</i>	<i>Not Crossing</i>

Table 2-54: ACAS X Display and Controls Inputs (Aural Annunciations)

ADD Signal	Source	Range	Units	Resolution	Static/ Dynamic	Optional/ Required	Comments
TRMDisplayData.alarm	CAS	True, False	Discrete		Dynamic	Required	Set true when a new RA or modified RA is displayed to signal an aural annunciation should be spoken.
TRMDisplayData.turn_off_aurals	CAS or Monitor	True, False	Discrete		Dynamic	Required	Set by CAS below 500 feet AGL or by the Monitor when aural annunciations are inhibited due to a higher priority annunciation (e.g. wind shear, ground proximity warning system, stall warning).
Audio Level	Configuration Input				Static	Required	Controls the volume of the aural annunciations. This is not a signal that comes from the ADD.

Table 2-55: ACAS X Display and Controls Inputs (Traffic Display)

Signal	Source	Range	Units	Resolution	Static/ Dynamic	Optional/ Required	Comments
Pilot Selected Operational Mode (Sensitivity Level)	CAS	Standby(1), TA-Only(2), TA/RA(0-automatic)	Discrete	1	Dynamic	Required	Both pilot and automatic selection are required.
Reported (Actual) Sensitivity Level	CAS	Standby(1), TA-Only(2), TA/RA(3)	Discrete	1	Dynamic	Required	Automatic selection of these levels is required.
Intruder Range	CAS	0-128	NM	1/16 NM	Dynamic	Required	Compute and transmit horizontal range to the display instead of slant range when intruder altitude information is available. Sent for each intruder in the track file.
Intruder Altitude	CAS	±12,700	ft	100 ft	Dynamic	Required	Intruder's relative altitude. Sent for each intruder in the track file.
Intruder Bearing	CAS	±180	degrees	0.2 degs	Dynamic	Optional	<ol style="list-style-type: none"> When available, intruder bearing is displayed. However, if valid bearing information is not received from CAS, only intruders causing a TA or an RA will be displayed. The 0.2 degree resolution shown is the resolution available on the interface protocol. This is not intended to imply a requirement on the resolution of the traffic display or the accuracy of the surveillance function.
Intruder Valid Bearing Flag	CAS	True, False	Discrete	1	Dynamic	Required	Set true when surveillance and CAS have a valid bearing estimate for the intruder aircraft.

Signal	Source	Range	Units	Resolution	Static/ Dynamic	Optional/ Required	Comments
Intruder Altitude Reporting Flag	CAS	True, False	Discrete	1	Dynamic	Required	Set true when CAS has a valid altitude for the intruder.
Intruder Vertical Sense	CAS	0-2	Discrete	1	Dynamic	Required	Data used to define when to display a vertical trend arrow for the intruder. A value of 0 indicates intruder is level and no arrow is displayed. A value of 1 indicates the intruder is climbing > 500 fpm and an up arrow is displayed. A value of 2 indicates the intruder is descending > 500 fpm and a down arrow is displayed.
Intruder Threat Level	CAS	0-4	Discrete	1	Dynamic	Required	Defines the threat level of each intruder. Display of each threat level is defined in §2.2.6.1.2.1.3. These values are interpreted as follows (which match the TCAS TA Codes in Appendix D of the ADD): 0 = Clear / Other Traffic 1 = Proximate Traffic 2 = Degraded Surveillance TA 3 = Nominal Surveillance TA 4 = RA
Ownship Altitude	Transponder	±65,536	ft	1 ft	Dynamic	Required	Ownship pressure altitude.
Corrected Ownship Altitude	Transponder	±65,536	ft	1 ft	Dynamic	Optional	Ownship barometric altitude. Contains the local barometric correction.
Ownship Magnetic Heading	Aircraft Source (Gyro, IRS, etc.)	0-360	degrees		Dynamic	Optional	Used for display stabilization.

Signal	Source	Range	Units	Resolution	Static/ Dynamic	Optional/ Required	Comments
Control Panel Settings	Control Panel				Dynamic	Required	See the Control Panel Output definition. Information is passed unchanged from the transponder through the ACAS X processor.
Altitude Select Limit	Control Panel	0 to ±12,300	ft	100 ft	Static	Optional	Controls the altitude band in which traffic will be displayed. Normal range is ±2700 feet. CAS logic will not send values greater than ±9900 feet.
Display Enable Discrete	Monitor	0-1	Discrete	1	Dynamic	Optional	Enables the traffic display.
Display Mode	Monitor	0-1	Discrete	1	Static	Required	An input is provided to define to the displays whether traffic (including RA, TA, Proximate, and Other) is: 1) always displayed, or 2) displayed only when a TA or RA is present.
Display Limit	Monitor	1-31		1	Static	Optional	Defines the maximum number of intruders to be displayed. The input value is sent to the traffic display in the Request to Send word.

2.2.7 Monitor Requirements

An automatic performance monitoring system, the Monitor, **shall** (1790) be implemented.

Note: The purpose of performance monitoring is to detect malfunctions that degrade or preclude ACAS X protection. Particular attention should be given to monitoring functions whose failure could result in a disruption in the ability of the STM to adequately track targets or ownship and maintain those tracks or a failure to generate the appropriate RA by the TRM .

2.2.7.1 General Requirements

2.2.7.1.1 Failure Response

When a failure is detected, i.e., when the Monitor declares an ACAS X failure, the Monitor **shall** (1791):

- a. Indicate to the flight crew that an abnormal condition exists.
- b. Cause any Mode S transmissions that report ownship status to show that ownship has no on-board resolution capability (§2.2.3.8.3.1.14 and §2.2.3.8.3.2.3.2).
- c. Prevent interrogations by ownship ACAS X.
- d. Deactivate the normal ACAS X display functions.

The Monitor responses **shall** (1792) be accomplished by positive means such that the response functions show an abnormal condition within one second of failure detection.

2.2.7.1.2 Noninterference with Normal Operation

The Monitor **shall** (1793) not interfere with the normal operation of the ACAS X equipment or the Mode S transponder. Any RF test signals used by the Monitor **shall** (1794) be restricted by the compatibility requirements of §2.2.3.10.

2.2.7.1.3 Self-Test

The Monitor **shall** (1795) include a self-test function, capable of being initiated by the pilot. As a minimum, self-test **shall** (1796) test the (aural) alarm and activate each display element (including discrete outputs) in a pre-determined temporal pattern to allow visual verification that display outputs issued by the digital processor can be correctly interpreted by the pilot. The self-test function **shall** (1797) not interfere with the normal operation of the equipment.

Note: Flight-crew-initiated operation of display indications for test purposes is not considered interference with normal equipment operation.

2.2.7.2 Monitoring of ACAS X Components

2.2.7.2.1 Computer Monitoring

The Monitor **shall** (1798) include provisions for computer performance monitoring. As a minimum these tests **shall** (1799) include random access memory (RAM) pattern tests, central processing unit (CPU) instruction tests, program memory tests, CPU input/output functions tests, and CPU timing tests. The Monitor **shall** (1800) be capable of detecting a

failure in the computer performance monitoring and upon detection **shall** (1801) declare an ACAS X failure.

Additional computer performance testing may be required under normal bench test conditions and under environmental extremes. These computer performance tests may be included in the Monitor or may be a separate test program for use during bench and environmental testing.

2.2.7.2.2

Coordination Monitoring

The Monitor **shall** (1804) be capable of detecting an equipment or software failure that results in the loss of coordinaton data (§2.2.3.12.2). Detection of such a failure for three consecutive seconds **shall** (1805) cause the Monitor to declare an ACAS X failure.

2.2.7.2.3

ACAS X/Transponder Interface Monitoring

The Monitor **shall** (1806) be capable of detecting loss of data integrity as described in §2.2.3.12.2.1.5 and upon detection **shall** (1807) declare an ACAS X failure.

2.2.7.2.4

Display Monitoring

The Monitor **shall** (1808) monitor the status of the installed traffic display(s) and RA displays. When the status indicates that a display or display function is “Not Operational”, the Monitor **shall** (1809) determine the appropriate ACAS X system level operating mode (i.e., TA-Only, TA/RA, or TCAS Fail). The Monitor **shall** (1810) not allow RAs to be issued if all of the RA displays have failed. In addition, if all the traffic displays and all the RA displays have failed, the Monitor **shall** (1811) fail the ACAS X system. Also, for those display implementations requiring it, the Monitor **shall** (1812) provide information to the display so that an appropriate status or failure annunciation can be shown on the display. The requirements for the failure and status annunciations are contained in §2.2.6.6.

Notes:

1. *It is strongly recommended that each traffic display or traffic display function and each RA display or RA display function always provide status information to the Monitor, even when the display or display function is capable of detecting and annunciating its own failures, to ensure that ACAS X does not issue RAs when no means are available to display RA guidance.*
2. *The recommended display and Monitor actions associated with possible display failure conditions are shown in Table 2-56.*

Definitions:

- RA Capability:* *The ability of an RA display to provide resolution advisory guidance. For RA/VSI implementations the display must have both the RA function and the VSI function operational on a given display to provide RA capability. For pitch guidance implementations, both the ACAS X pitch cue function and the aircraft pitch indicator must be operating to provide the complete RA capability.*
- RA Function:* *The red and green arcs or zones provided for vertical speed guidance on an RA/VSI display or a VSI tape.*
- VSI Function:* *The vertical speed indicator portion of an RA/VSI display.*
- Pitch Function:* *The aircraft pitch indicator on a PFD.*
- Pitch Cue Function:* *An RA display implementation where pitch angles to be flown and avoided are displayed on a PFD. This function replaces the RA/VSI function of RA/VSI implementations.*

Table 2-56: Recommended Display Actions and Monitor Response to “Not Operational” Status Received from ACAS X Displays

Failure Condition	Display Response	ACAS X Processor Monitor Response
<i>Traffic display failure(s). RA capabilities still available on at least one display.</i>	<ul style="list-style-type: none"> • Detect that the traffic display has failed. • Annunciate that the traffic display has failed. • If a dedicated traffic display, provide a “not operational” indication on the TA status output. • If a shared display or an integrated TA/RA display, the display should provide a TA status output specifically for the traffic display function indicating that it is “not operational”. 	<ul style="list-style-type: none"> • If in TA-Only Mode, the Monitor should allow the ACAS X processor to continue to issue TAs. • If in TA/RA Mode, the Monitor should allow the ACAS X processor to continue to issue TAs and RAs.
<i>Loss of RA capability on one display only.</i>	<ul style="list-style-type: none"> • The affected display should detect the specific failure condition. • The affected display should annunciate the appropriate failure indication. • The affected display should set its RA status output to a “not operational” status. 	<ul style="list-style-type: none"> • The Monitor should allow the ACAS X processor to issue TAs and RAs.
<i>Loss of RA capability on all displays. Traffic display capability still available on at least one display.</i>	<ul style="list-style-type: none"> • Each RA display should detect its specific failure condition. • Each RA display should annunciate the appropriate failure indication. • Each RA display should set its RA status output to a “not operational” status. • The operational traffic display(s) should indicate “operational” on its TA status output. 	<ul style="list-style-type: none"> • The Monitor should permit the system to operate in TA-Only mode. • The Monitor prohibits the processor from issuing RAs (in accordance with the requirements of §2.2.7.2.4).
<i>Loss of RA capability on all displays and loss of all traffic display capability.</i>	<ul style="list-style-type: none"> • Each display should detect its specific failure condition. • Each RA display should set its RA status output to a “not operational” status. • Each traffic display should set its TA status output to a “not operational” status. • Each display should annunciate an ACAS X System failure as the result of the ACAS X Monitor declaring the system failed. 	<ul style="list-style-type: none"> • The Monitor fails the ACAS X System (in accordance with the requirements of §2.2.7.2.4).

2.2.7.2.5 Altitude Input Monitoring

2.2.7.2.5.1 Barometric Altitude Monitoring

Note: The STM logic determines credibility of the barometric altitude source. The determined credibility will be either "credible" or "not credible".

Each ACAS X processing cycle, the STM logic will provide the Monitor with the status of ownship's barometric altitude source credibility. The Monitor **shall** (1813) declare an ACAS X failure when this status has been "not credible" for five consecutive ACAS X processing cycles. Unless this status has been "not credible" for five consecutive ACAS X processing cycles, the Monitor **shall** (1814) not declare a failure.

2.2.7.2.5.2 Radio Altitude Monitoring - General

The Monitor **shall** (1815) determine the status, "failed" or "not failed", of the radio altitude source. If the status of the radio altitude source is determined to be "failed", as described in the following, the Monitor **shall** (1816) declare an ACAS X failure. Also as described in the following, the Monitor **shall** (1818) determine the credibility of the radio altitude value and **shall** (1819) provide a radio altitude value and credibility to the STM.

Note: Accurate radio altitude is required for inhibition of descend advisories at low altitudes. Consequently, undetected erroneous radio altitudes may cause an unsafe condition. It is recommended that manufacturers rigorously develop and verify the implementation of the Monitor which deals with radio altitude. It is also recommended that radio altimeters with integral failure monitors, such as those conforming to the requirements of ARINC CHARACTERISTIC 707 (Ref. E), be used with ACAS X.

2.2.7.2.5.2.1 Radio Altimeter Input to the Monitor

An input of height above ground from a radio altitude source **shall** (1820) be provided to the Monitor at least once each ACAS X processing cycle for processing. The Monitor provides processed input to the STM. The Monitor will provide height above ground from the radio altitude source to the STM at least once per ACAS X processing cycle. In addition, if the radio altimeter has its own failure monitor, e.g., as required by Ref. E., the outputs from such a failure monitor **shall** (1821) be supplied to the Monitor at least once per ACAS X processing cycle.

2.2.7.2.5.2.2 Monitor Inputs to the STM and STM Inputs to the TRM

The data supplied to the STM by the Monitor **shall** (1822) consist of an indication that the radio altitude value is "credible" or "not credible" (§2.2.7.2.5.2.4). The data supplied to the STM by the Monitor **shall** (1823) consist of a tracked or coasted radio altitude value, if either is available, and an indication that the radio altitude value is "credible" or "not credible". In the event that neither a tracked nor coasted radio altitude value is available, the Monitor **shall** (1824) supply the STM with an indication that the radio altitude is not credible. This data **shall** (1825) be supplied to the TRM once per ACAS X processing cycle.

2.2.7.2.5.2.3 Monitor Verification of Radio Altitude Source Prior to Issuance of TAs or RAs

In the event of a power off/on system reset, the Monitor **shall** (1827) verify that a radio altitude source has provided data for five consecutive ACAS X processing cycles without a failure indication (see also §2.2.7.2.5.2.5) prior to allowing ACAS X to issue either TAs

or RAs. It is required that such a verification be performed even when ACAS X is in the Standby mode of operation. If such verification cannot be achieved within the ten ACAS X processing cycles immediately following a power off/on system reset, the Monitor **shall** (1828) declare an ACAS X failure.

2.2.7.2.5.2.4 Radio Altitude Credibility

Subsequent to radio altitude source verification, the Monitor **shall** (1829) assess the credibility of the sequence of radio altitude values being supplied to it in accordance with the following. Only in the event that both Test #1 and Test #2 result in "credible" **shall** (1831) the Monitor supply the STM with "credible". Otherwise, the Monitor **shall** (1832) supply the STM with "not credible".

2.2.7.2.5.2.4.1 Radio Altitude Credibility Determination Test #1

The Test #1 result **shall** (1833) be not credible if either of the following conditions exist for each of the three most recent successive ACAS X processing cycles:

- a. Any condition other than "NORMAL" is indicated in the Sign Status Matrix (SSM) of an altimeter in conformance with Ref. E. (or equivalent interface), or
- b. the radio altitude value is greater than the reliable operational range of the radio altitude.

If the altitude source does not provide a credibility indication, then the radio altitude **shall** (2137) be considered non-credible at radio altitudes greater than 2350 ft.

Note: In order to determine the reliable operational range of the radio altitude source, the characteristics of the altimeter must be known.

If the above conditions are not met, the Test #1 result **shall** (1834) be credible.

When the Test #1 result is not credible, any existing radio altitude track (§2.2.7.2.5.2.4.2) **shall** (1835) also be discontinued.

As long as the Test #1 result is credible, Test #2 **shall** (1837) be conducted. If the Test #1 result is not credible, there is no need to conduct Test #2.

Notes:

1. *In normal operation the a. and b. conditions above will routinely occur as the ACAS X-equipped aircraft climbs after take-off and will no longer exist at some point as the ACAS X-equipped aircraft descends. Test #1, therefore, should continually be conducted to allow for timely altitude track reinitialization as the aircraft descends.*
2. *Two radio altitude sources are also provided for safety concerns. If two radio altitude sources are provided, the two radio altitude sources should be used to determine if an RA inhibition has to be removed. Indeed improper reporting from one radio altitude could wrongly indicate that it is out of its functional range while the aircraft is flying at low altitude. Subsequently, if one of the two Radio Altitude source indicates a NORMAL condition with altitude < 1500 ft while the other one indicates that it is out of its functional range, then the one indicating a NORMAL condition should be used to determine if an RA (and TA) have to be inhibited or not.*

2.2.7.2.5.2.4.2 Radio Altitude Credibility Determination Test #2

The Monitor incorporates tracking algorithms to further test the credibility of the sequence of altitude inputs from the radio altimeter. These algorithms **shall** (1839) condition the radio altitude inputs from the radio altimeter to preclude the input of extremely erratic or spurious transient altitudes to the TRM.

Note: As a minimum, the tracking algorithms should consider: (1) the ACAS X-equipped aircraft's estimated vertical rate and acceleration; and (2) the requirements necessary to reliably coast the radio altitude track when transient input values, such as those caused by terrain fluctuations, to the Monitor are not credible.

When a radio altitude value is received by the Monitor prior to radio altitude tracker initialization, the Test #2 result **shall** (1840) be not credible.

When a radio altitude value is received by the Monitor subsequent to radio altitude tracker initialization, the Test #2 result **shall** (1841) be credible if the tracking algorithms so indicate. Otherwise, the Test #2 result **shall** (1842) be not credible.

2.2.7.2.5.2.4.2.1 Radio Altitude Track Initiation

The radio altitude tracker **shall** (1843) be initialized after:

- a. radio altimeter source verification, or
- b. the track has been coasted for three consecutive ACAS X processing cycles.

During the period of tracker initialization, the Monitor **shall** (1844) supply the STM with an indication that the radio altitude is not credible.

2.2.7.2.5.2.4.2.2 Operation Subsequent to Radio Altitude Track Initiation

Following successful track initialization, the Monitor **shall** (1845) supply the STM the most recent tracked or coasted radio altitude value received from the Monitor for as long as the track exists.

Under the following conditions, the Monitor supplies the STM logic with the stated data for each condition:

- a. If the received radio altitude value is determined credible by the Monitor, the Monitor **shall** (1847) supply to the STM the tracked altitude value.
- b. If the received radio altitude value is deemed not credible by the Monitor, the radio altitude track **shall** (1848) be coasted, and the Monitor **shall** (1849) supply to the STM the coasted altitude value and an indication that it is credible.
- c. If the received radio altitude values have been coasted for three consecutive ACAS X processing cycles, the Monitor **shall** (1850) attempt to reestablish the track.

2.2.7.2.5.2.5 Radio Altitude Source Failure

The Monitor **shall** (1851) declare an ACAS X failure if any of the following conditions are met:

- a. The radio altitude source has been inoperative for three consecutive ACAS X processing cycles. During this period, the Monitor **shall** (1852) indicate to the STM that the radio altitude value is not credible, according to the requirements of §2.2.5.5.2.
- b. The Radio Altimeter Monitor (Note 1), e.g., such as for an altimeter conforming to Ref. E., indicates a failure for three consecutive ACAS X processing cycles. During

this period, the Monitor **shall** (1853) indicate to the STM that the radio altitude value is not credible, according to the requirements of §2.2.5.5.2.

Notes:

1. *This "Radio Altimeter Monitor" referenced is any monitor integral to the radio altimeter, such as the one required by ARINC CHARACTERISTIC 707 (Ref. E), and is not the ACAS X Monitor.*
2. *Because of the criticality of the Radio Altimeter to proper ACAS X operation, it is recommended that other failure detection mechanisms be implemented wherever practicable. These may include cross comparisons of ownship's vertical rates or vertical displacements as determined by barometric altitude and radio altitude changes, the use of ownship's Mode S transponder's air-ground status to determine if the radio altitude value reported when the aircraft is on the ground is reasonable, and the use of radio altimeter failure indications from other aircraft systems.*

2.2.7.2.6 ICAO 24-Bit Aircraft Address

The Monitor **shall** (1854) declare an ACAS X failure in the event that ownship ICAO 24-bit Aircraft Address is all 0s or all 1s.

2.2.7.2.7 Monitoring of Transponder Status

Each ACAS X processing cycle, the Monitor **shall** (2196) determine the status of ownship transponder. A functioning Mode S transponder is required for the ACAS X system to operate. Accident experience has shown that the loss of transponder functionality requires flight crew awareness and may require subsequent flight crew response. This response includes checking for proper ACAS X system settings as well as Mode S transponder settings and operation.

Note: The U.S. NTSB has issued a Safety Recommendation (A-07-035) that for all aircraft required to have a traffic alert and collision avoidance system installed, that the airborne loss of collision avoidance functionality, for any reason, provide an enhanced aural and visual warning requiring pilot acknowledgement.

If ownship transponder status is determined to be Standby or Failed when the aircraft is airborne and radio altitude > 1550 ft +/- 100 ft (as indicated when LDI = 0), the ACAS X system **shall** (2197) provide an indication of this status. This indication may be used by another aircraft display system to provide aural and visual annunciations to alert the flight crew that collision avoidance is not available.

The required aural and visual annunciation may be provided through an aircraft master caution and alerting system. In the absence of other suitable means, the ACAS X system **shall** (2201) provide a visual annunciation “TCAS Standby” on the TCAS display and a flight deck aural annunciation “TCAS Standby.”

The annunciations are not required during normal transitions to TCAS Standby, e.g., upon landing or during higher priority flight deck alerts.

2.2.7.2.8 Higher Priority Warning Systems

The Monitor **shall** (1855) recognize warnings from higher priority warning systems (e.g., wind shear, ground proximity warning systems, and stall warning). Upon recognition of such warnings, the Monitor **shall** (1856) cause ACAS X to operate in TA-Only mode and

shall (1857) suppress all ACAS X aural annunciations. The Monitor **shall** (1858) keep ACAS X in TA-Only mode until 10 seconds after the higher priority warning has ceased. During this 10 second suppression period, TA aural annunciations **shall** (1859) not be suppressed.

2.2.7.2.9

Monitoring of Ownship Position Source , Ground Speed and Heading Input

Each ACAS X processing cycle, the Monitor **shall** (1865) determine the status, “failed” or “not failed” of ownship position source (e.g. GNSS). If ownship positionsource is determined to be failed, then maintenance of tracks using passive data **shall** (1860) be disabled and all tracks **shall** (1861) transition to active surveillance if they qualify for active surveillance, while all other tracks will coast out.

Each ACAS X processing cycle, the system **shall** (1862) monitor the ownship ground speed input. If the ground speed source is declared invalid or unavailable then use of ownship ground speed for determination of whether own is operating on the airport surface is not permitted as specified in §2.2.4.6.3.

ACAS X **shall** (1863) continue to operate during a failure in own latitude/longitude input or own ground speed input. ACAS X may continue to operate without true heading.

Each ACAS X processing cycle, the Monitor **shall** (1867) determine the status, “failed” or “not failed” of ownship heading source.

Note: this information is necessary for the front end surveillance informing the STM on Ownship heading status.

The ACAS X system **shall** (1864) provide an output indicating the failure of ownship position source, ownship ground speed input, or heading. This output may be used by aircraft display systems or aircraft maintenance systems to alert the flight crew and/or maintenance crew to the input failures.

2.2.7.2.10

Monitoring of ACAS X Transmitter

The Monitor **shall** (2194) be capable of detecting loss of ACAS X 1030 MHz transmitter functionality and upon detection **shall** (2195) declare an ACAS X failure.

Note: If ownship transmitter fails, it is important for ownship transponder to report ‘no RA capability’ in outgoing surveillance replies and in ADS-B transmissions. Otherwise, other aircraft that are tracking ownship will think they are in a coordinated encounter when they are not; this can limit the performance of the other aircrafts’ collision avoidance systems. (And ownship, with a failed transmitter, will not be providing collision avoidance protection).

2.2.7.3

Monitor Interfaces

2.2.7.3.1

Data Provided by the Monitor to Surveillance

The Monitor **shall** (1869) provide the following data to Surveillance at least once per surveillance update interval:

ACAS X operational status (fully-operational, TA-Only, not operational).

2.2.7.3.2

Data Provided by the Monitor to the STM

The Monitor **shall** (1870) provide the following data to the STM at least once per surveillance update interval:

Tracked radio altitude,
Radio altitude status (credible, not credible) (§2.2.7.2.5.2.4),
ACAS X operational status (operational, not operational),
Monitor-selected SL (standby, TA-Only, TA/RA).

2.2.7.3.3 Data Received by the Monitor from the STM

The Monitor **shall** (1871) be capable of receiving the following data from the STM each surveillance update interval:

Own barometric altitude credibility (§2.2.7.2.5.1)

2.2.7.3.4 Data Received by the Monitor from the Control Panel via the Mode S Transponder

The Monitor **shall** (1872) be capable of receiving the following data from the control panel:

Pilot-selected SL (standby, TA-Only, TA/RA, 3, 4, 5, 6, 7)

Note: Inputs 3, 4, 5, 6, and 7 are not used by ACAS X and will be interpreted as TA/RA mode.

2.2.7.3.5 Data Received by the Monitor from Aircraft Discretes

The Monitor **shall** (1873) be capable of receiving the following data from aircraft discretes (or equivalent):

RA/TA discretes
4 advisory inhibit discretes

2.2.7.3.6 Data Provided by the Monitor to Displays and Controls

Table 2-57: ACAS X Display and Controls Inputs (RA Displays)

Signal	Source	Range	Units	Resolution	Static/ Dynamic	Optional/ Required	Comments
ACAS X Status Annunciations	CAS and Monitor				Dynamic	Required	Indicates the status and operating mode of ACAS X and the RA display.

Table 2-58: ACAS X Display and Controls Inputs (Aural Annunciations)

Signal	Source	Range	Units	Resolution	Static/ Dynamic	Optional/ Required	Comments
Aural Inhibit	CAS or Monitor	True, False	Discrete	1	Dynamic	Required	Set by TRM below 500 feet AGL or by the Monitor when a higher priority advisory (e.g. GPWS, wind shear) is active.

Table 2-59: ACAS X Display and Controls Inputs (Traffic Display)

Signal	Source	Range	Units	Resolution	Static/ Dynamic	Optional/ Required	Comments
Display Enable	Monitor	0-1	Discrete	1	Dynamic	Optional	Enables the traffic display.
Display Mode	Monitor	0-1	Discrete	1	Static	Optional	One configuration input is provided to define to the displays whether traffic (including RA, TA, Proximate, and Other) is: 1) always displayed, or 2) displayed only when a TA or RA is present.
Display Limit	Monitor	1-31		1	Static	Optional	Defines the maximum number of intruders to be displayed. Information is read from the configuration input and sent to the traffic display in the Request to Send word.

2.2.7.3.7 Data Received by the Monitor from Displays and Controls

Table 2-60: ACAS X Display and Controls Output (Control Panel)

Signal	Source/User	Range	Units	Resolution	Static/ Dynamic	Optional/ Required	Comments
Altitude Reporting	Control Panel/ Monitor	Altitude Reporting/ Non-altitude Reporting		1	Dynamic	Optional	Switch enables altitude reporting by the transponder. If transponder is not reporting altitude, ACAS X should be failed.
Self Test	Control Panel/ Monitor				Dynamic	Required	Indicates pilot initiated Self Test.

Table 2-61: ACAS X Display and Controls Output (Displays)

Signal	Destination	Range	Units	Resolution	Static/ Dynamic	Optional/ Required	Comments
Traffic Display Status	Monitor	Operational/ Not Operational	Discrete	1	Dynamic	Optional	Indicates status of the traffic display.
RA Display Status	Monitor	Operational/ Not Operational	Discrete	1	Dynamic	Required	Indicates status of the RA display.

2.2.7.3.8 Data Received by the Monitor from Ownship Position Source

The Monitor **shall** (1874) be capable of receiving ownship position source status.

2.2.7.3.9 Data Received by the Monitor from Ownship Heading Source

The Monitor **shall** (1875) be capable of receiving ownship heading status.

2.2.8 ACAS Xo

ACAS X (Class 2) includes ACAS Xo and ACAS Xa functionality. Class 1 systems do not support ACAS Xo functionality.

2.2.8.1 ACAS Xo Overview

ACAS Xo provides additional collision avoidance logic modes designed to support closely-spaced flight operations. ACAS Xo allows specifically designated traffic to be monitored by an alternative ACAS logic more compatible with the flight operation than the standard ACAS Xa logic. Each alternative logic is referred to as an ACAS Xo mode. All capable traffic that is not specifically designated to an ACAS Xo mode is monitored, as usual, by ACAS Xa logic.

Two ACAS Xo modes are included in this version of the MOPS.

1. Designated No Alerts (DNA) mode normally suppresses all ACAS Xa alerts and guidance on the specifically designated traffic. The flight crew must visually acquire the desired traffic before designating it to DNA, then maintain visual separation from the DNA-designated aircraft. DNA mode prevents alerts on the designated traffic while still allowing full ACAS Xa protection from all other cooperative traffic.

While DNA operates using ACAS Xa logic, it normally suppresses any TA and RA outputs (for the designated intruder) to the flight crew. When a DNA aircraft is part of a multi-threat RA encounter, it will be considered together with all the other threats in the selection of a global RA.

2. Closely Spaced Parallel Operations down to 3,000 ft runway separation (CSPO-3000) mode provides designated traffic with modified CAS logic monitoring that is appropriate for parallel operations. CSPO-3000 may be used in both visual and instrument conditions. The CSPO-3000 mode desensitizes the CAS logic to typical closely-spaced runway operations, but still provides TAs and RAs on blundering traffic. ACAS Xa protection is maintained on all other cooperative traffic.

Note: Studies indicated that use of ACAS Xo CSPO-3000 logic is actually feasible down to 2,500 ft runway separation when using active surveillance information and manual RA responses. Runway separations lower than 2,500 ft would likely require additional surveillance information and automatic RA response.

Additional ACAS Xo modes are expected to be included in future versions of this document. Manufacturers should be aware that future modes may have different validity requirements. Logic requirements to support such additional modes are unknown at this time. Follow-on operational concepts will determine necessary modes and associated logic requirements.

The risks involved in switching between RA-generating logic modes during an RA have not been studied fully and are potentially complex. For this reason, switching between the RA-generating logic modes (i.e., ACAS Xa and CSPO-3000) during an RA (including suppressed RAs) on the designated traffic is not permitted. Mode changes will be delayed until the RA has cleared. Transitions between Xa and DNA during an RA are permitted because DNA runs the Xa TA/RA generation logic but suppresses the display of alerts (see §2.2.8.4).

2.2.8.1.1 Xo Parameters

ACAS Xo uses three parameters as part of the determination of mode availability, traffic validity, automatic undesignation, and dropped traffic designation hold time. The parameters are:

Diverging DNA Traffic Range Limit

Note: This parameter is used for automatic undesignation of diverging DNA traffic, and is based on the proximate traffic range. Too small a value may lead to unwanted TAs and RAs in normal operations. Too large a value may limit automatic undesignation of the traffic.

Maximum CSPO-3000 Altitude

Note: This parameter sets the maximum altitude for use of the CSPO-3000 mode, as well as the maximum altitude for CSPO-3000 traffic. This provides a coarse filter to

stop inappropriate use of the mode, and is based on the maximum altitude of existing parallel runways. A lower value could exclude use of the mode at some or all airports, and a higher value limits the efficacy of the coarse filter.

Designation Timeout Interval

Note: This parameter sets the maximum time that traffic with a poor or lost track may be maintained as designated to an Xo mode. Too small a value will result in quick or immediate undesignation from the Xo mode, likely resulting in unwanted TAs and RAs if surveillance is reestablished. Too large a value could result in resuming the Xo mode on a reestablished track after the flight operation is complete.

The Diverging DNA Traffic Range Limit parameter value is 6 NM.

The Maximum CSPO-3000 Altitude parameter nominal value is 14,000 ft MSL.

Note: Hysteresis is applied to this parameter. The threshold for designation when the traffic is not designated for CSPO-3000 is 14,000 ft and the threshold for undesignation when the traffic is designated is 14,500 ft.

The Designation Timeout Interval parameter value is 30 seconds.

Note: System performance and safety analyses were performed using these values. Manufacturers wishing to use alternative parameter values should be aware that such changes will require new analyses.

2.2.8.2 Xo Controls and Displays

Systems implementing ACAS Xo **shall** (1877) be integrated with an ASA System (or equivalent) compliant with RTCA/DO-317B (Ref. T) providing the display/control system tools necessary for Xo mode designation and associated displays.

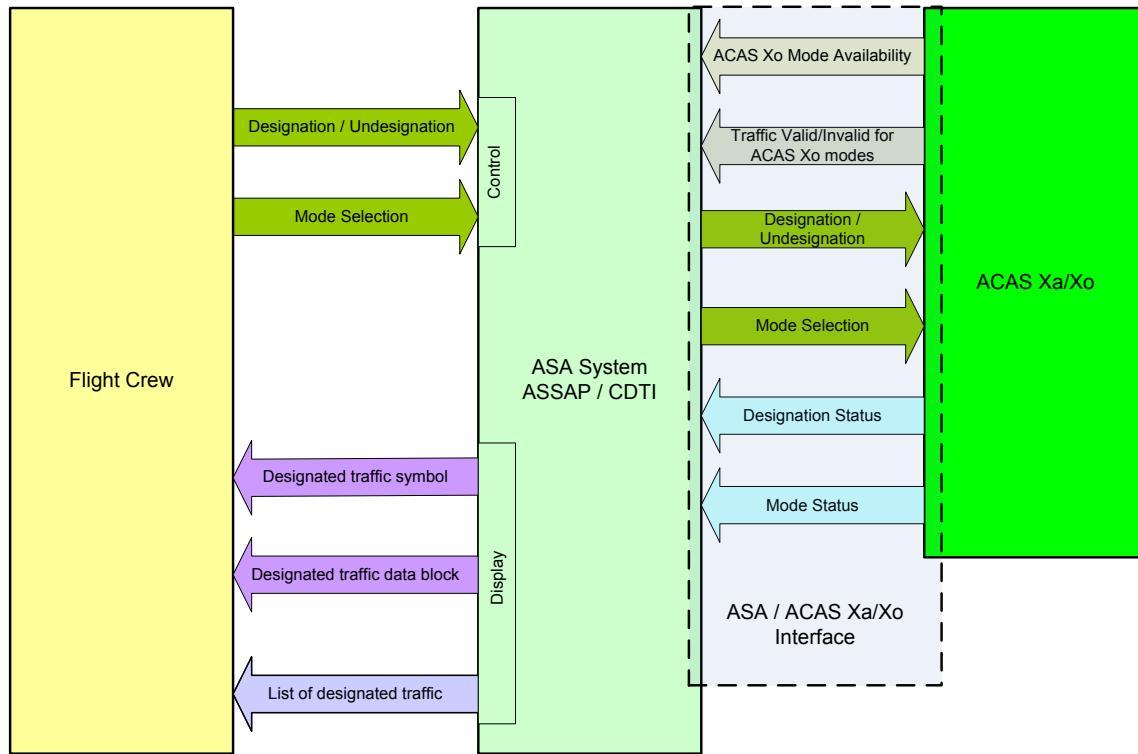


Figure 2-21: ASA / ACAS X System Interface

2.2.8.2.1 Xo Traffic Designation/Undesignation

ACAS Xo control **shall** (1878) be accomplished in compliance with ASA System designation tools [RTCA/DO-317B (Ref. T)].

The display/control device **shall** (1879) allow the flight crew to select the ACAS Xo DNA Mode.

The display/control device **shall** (2585) allow the flight crew to select the ACAS Xo CSPO-3000 Mode.

The display/control device **shall** (1880) allow the flight crew to designate valid traffic for ACAS Xo operations.

Note: Designation for ACAS Xo requires selection of both the traffic and the ACAS Xo mode. This can be accomplished in various ways, such as selecting the mode first, then the traffic, or selecting the traffic and then the mode.

The display/control device **shall** (2579) prevent the flight crew from designating traffic to an Xo mode when other tracked traffic is currently designated to an Xo mode.

Notes:

1. “Tracked traffic” as used here indicates traffic for which the ACAS X System outputs information. Under unusual circumstances, higher-priority tracks may

preempt designated traffic for output. If the track of a designated aircraft is not output, a second aircraft could be designated.

2. *This requirement limits Xo to only one designated tracked traffic and one Xo mode at a time. Future versions of this MOPS may permit or require the capability to designate more than one traffic.*

The display/control device **shall** (1883) be capable of allowing the flight crew to undesignate tracked traffic from ACAS Xo.

Notes:

1. *If the limit on the number of designated aircraft (currently 1) is reached, traffic must be undesignated before designating another.*
2. *Traffic designation is maintained on dropped tracks for up to the Designation Timeout Interval. There is no requirement for the display/control system to provide controls to undesignate these untracked aircraft (which cannot be displayed because there is no track). During this time, the system will allow designation of other traffic, and any such action will automatically undesignate the dropped track immediately.*
3. *The ICAO 24-bit Aircraft Address and ACAS Xo mode are used by ACAS Xo to identify the traffic and return designated traffic to normal ACAS Xa logic.*

2.2.8.2.2 Xo Display

Traffic designated for ACAS Xo **shall** (1886) be displayed in compliance with RTCA/DO-317B (Ref. T).

Note: i.e., as an outlined traffic symbol.

The display for ACAS Xo **shall** (1887) include an indication of the designated traffic and the Designated and Actual (if different) ACAS Xo modes.

Note: Future human factors studies may determine that the display of additional Xo information is needed by flight crews. It is recommended that implementers make provisions for such future functionality. This may include display of:

- a. *Flight ID*
- b. *Target Directionality*
- c. *Expected Trajectory*
- d. *Target Selected Runway*

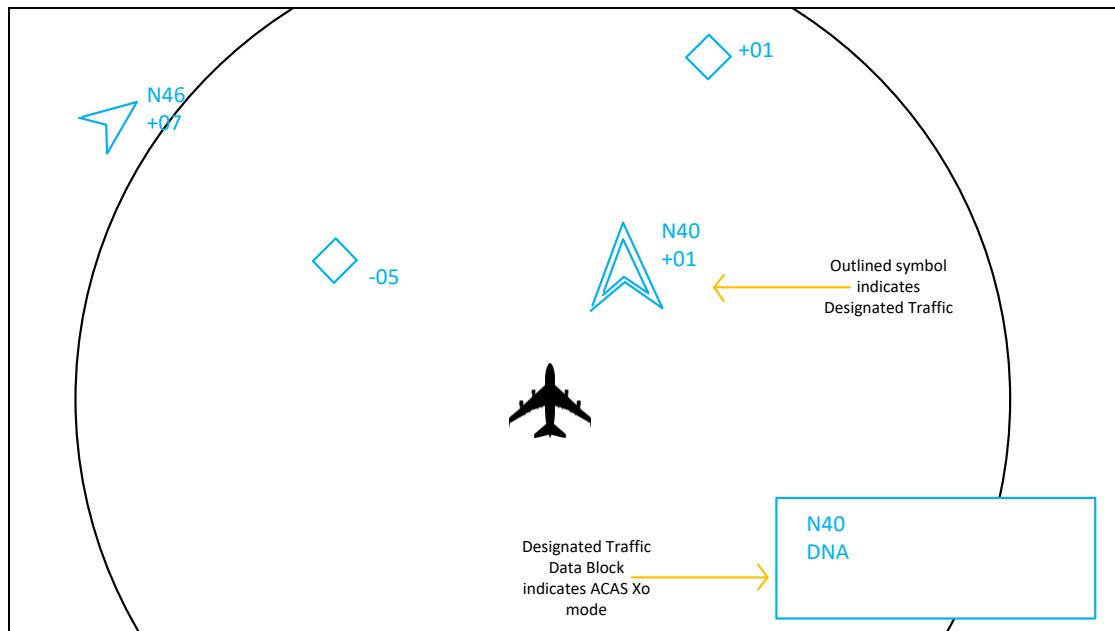


Figure 2-22: Example CDTI display with designated traffic symbology and associated data block indicating ACAS Xo mode

When traffic is undesignedated, the traffic symbol **shall** (1888) revert to the appropriate 'undesignated' traffic symbol, and any associated indication to the flight crew **shall** (1889) be removed.

Note: RTCA/DO-317B requires a visual alert for unexpected loss of designation. See Table 2-67 for conditions requiring this alert.

2.2.8.2.3 ACAS Xo Mode Availability

Xo Mode Availability is similar to the ASA System Application Status, and is provided for each Xo Mode. Each Xo Mode Availability **shall** (2148) be set to one of the following states:

- On: Xo Mode is on/running with designated traffic
- Available to Run: Xo Mode is configured. Required ownship input data is available and meets the performance criteria. (This state represents that no traffic is designated to the Xo Mode, but the mode is available.)
- Unavailable to Run: Required ownship input data is available but does not meet the performance criteria for the Xo mode. See §2.2.8.2.3.1 and §2.2.8.2.3.2.
- Not Configured: Xo Mode is not installed.

The display/control system **shall** (2149) prevent designation of traffic when the Xo mode is Unavailable to Run or Not Configured.

2.2.8.2.3.1 DNA

No system limits are defined for DNA mode availability.

2.2.8.2.3.2 CSPO-3000

CSPO-3000 **shall** (2136) be Unavailable to Run when any of the following are true:

1. Ownship CAS system is in TA-Only mode,
OR
2. Ownship altitude is above the Maximum CSPO-3000 Altitude and no traffic is designated to CSPO-3000,
OR
3. Ownship is above the Maximum CSPO-3000 Altitude with hysteresis, traffic is designated to CSPO-3000 and there is no RA on the designated traffic.

Notes:

1. *This requirement is a coarse filter to prevent use of CSPO-3000 at high altitudes. See also §2.2.8.3.2 and §2.2.8.5.*
2. *To prevent switching logic modes during an RA, CSPO-3000 is maintained (including designation) while an RA is present on the designated traffic regardless of either ownship or the designated traffic exceeding the maximum altitude. See §2.2.8.5.1.*

2.2.8.3 Traffic Xo Mode Validity

ACAS Xo **shall** (1892) identify all ACAS X traffic as valid or invalid for DNA.

ACAS Xo **shall** (2586) identify all ACAS X traffic as valid or invalid for CSPO-3000.

Note: Determination of traffic validity for Xo protection modes may be performed by the ASA System, as long as the validity criteria used are identical to those specified in this MOPS.

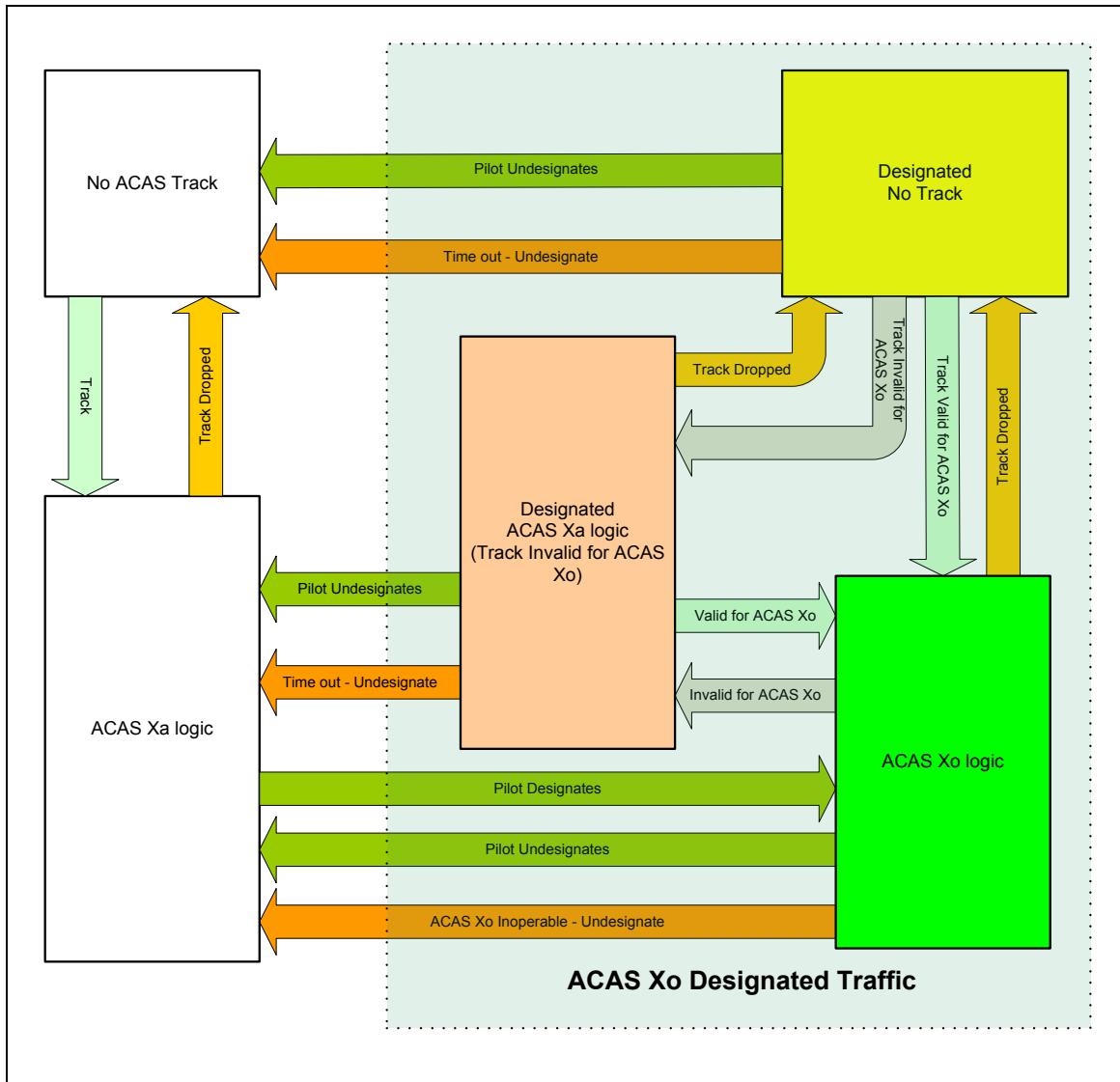


Figure 2-23: Example state chart for one ACAS Xo mode showing CAS logic and validity for a traffic

2.2.8.3.1 Traffic Valid for DNA

ACAS Xa traffic **shall** (1893) be marked valid for DNA when all of the following are true:

1. ACAS Xa is capable of generating a TA on the track.

Note: DNA mode suppresses TAs and RAs. There is no value in designating traffic for DNA if TAs or RAs cannot be generated on the traffic.

2. Traffic track includes an ICAO 24-bit Aircraft Address.

Note: The ICAO 24-bit Aircraft Address is required to associate the designated traffic with the track and maintain consistent tracking.

3. Bearing information is available for the track.

Note: When bearing information is not available for the track, the traffic display cannot display a traffic symbol for the intruder and only No-Bearing advisories

can be generated. Suppression of these advisories would eliminate any display of information about the threatening intruder.

All other traffic **shall** (1894) be marked as invalid for DNA.

2.2.8.3.2

Traffic Valid for CSPO-3000

ACAS Xa traffic **shall** (1897) be marked valid for CSPO-3000 when all of the following are true:

1. ACAS Xa is capable of generating an RA on the track.
2. Traffic track includes an ICAO 24-bit Aircraft Address.

Note: The ICAO 24-bit Aircraft Address is required to associate the designated traffic with the track and maintain consistent tracking.

3. Traffic is at or below the Maximum CSPO-3000 Altitude.

Note: Maximum CSPO-3000 Altitude MSL selected to roughly assure traffic intended for parallel operations. It is used as a coarse filter to prevent unintentional designation of the wrong aircraft. See §2.2.8.2.3.2.

All other traffic **shall** (1898) be marked as invalid for CSPO-3000.

2.2.8.3.3

Traffic Validity Reported to the ASA System

Traffic validity for ACAS Xo is determined by ACAS X and reported to the ASA System (or equivalent). The validity reported to the ASA System may be modified to allow designation to coast for up to the Designation Timeout Interval for dropped or invalid traffic. This is done to prevent the ASA System from immediately undesignating traffic from an ACAS Xo mode during short periods of lost or poor surveillance.

Note: Unlike ASA System applications, which add capability for the designated traffic, the ACAS Xo modes reduce or eliminate alerts as compared with the usual ACAS Xa logic. Undesignating traffic from an Xo mode due to poor surveillance quality would therefore increase the number of operationally inappropriate TAs and RAs.

If traffic designated to an Xo mode is dropped or becomes invalid, ACAS Xo **shall** (1901) continue to report that traffic as designated and valid to the ASA System for the Designation Timeout Interval. If the traffic does not become valid again during the Designation Timeout Interval, ACAS Xo **shall** (1896) report the traffic as invalid for the Xo mode.

In some circumstances, designation can be maintained on invalid traffic that is not dropped. For example, CSPO-3000 traffic maintains designation during an RA even if it would have otherwise been undesignated.

During the 30-second timeout for dropped designated traffic, other valid traffic may be designated to a mode (which would undesignate the dropped traffic).

2.2.8.4

CAS Logic in Use for Designated/Undesignated Traffic

The flight crew may request a switch between ACAS Xa/Xo logic modes at any time. This is accomplished by designating or undesignating the traffic, or switching the logic mode of the designated traffic. However, the system will not switch the mode in use when there is an active RA involving that aircraft (see Figure 2-24). In such situations, the mode change will be delayed until the RA clears.

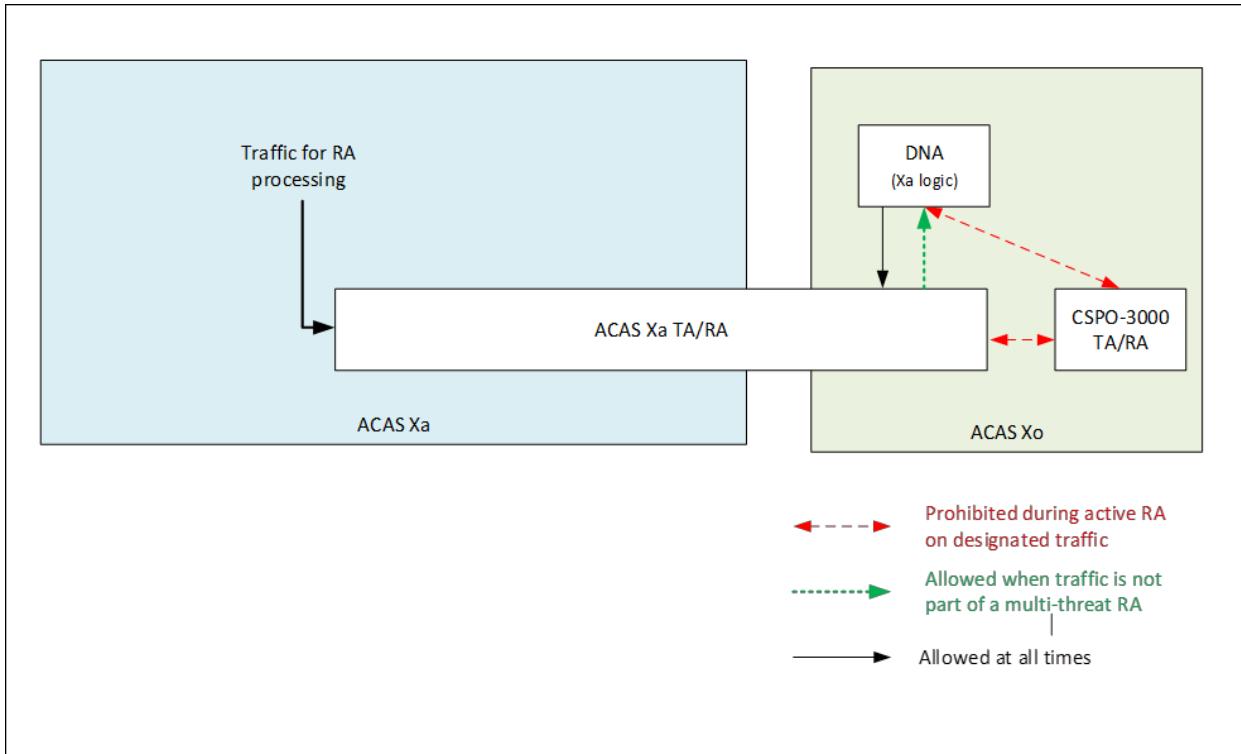


Figure 2-24: Permitted/Prohibited Xo Mode Changes during RA

Designated and Actual (if different) Xo modes are provided on the display. This indicates to the flight crew the Xo mode as designated by the flight crew, and also the actual Xo mode if different.

Note: Differences between the designated and actual Xo modes are temporary. The information is supplied to the flight crew to prevent confusion when the Xo system may not be performing as expected, such as providing an RA on DNA traffic during a multi-threat encounter.

2.2.8.4.1 Switching Logic Modes

Designation involves selecting both the traffic and the Xo mode. The flight crew (or system) may switch between Xo modes without undesignating and redesignating the traffic.

When the flight crew (or system) switches logic modes (i.e., designates, switches or undesignates traffic to or from an Xo mode), the Xo mode or Xa logic **shall** (2157) be applied to the traffic immediately except in these situations:

1. Switching between Xa logic and CSPO-3000 logic is delayed while an RA is present on the designated aircraft.

Note: Xo DNA mode uses the Xa logic.

In this case, the delayed logic mode switching **shall** (2152) be applied when the RA clears.

2. Switching from Xa logic mode to DNA is delayed when the designated aircraft is part of a multi-threat RA.

In this case, the delayed logic mode switch **shall** (2153) be applied when either the RA on the designated traffic clears, or the RAs on the other traffic clear.

When traffic designated for DNA mode is undesignated, TAs and RAs **shall** (1906) be produced and displayed. RA guidance **shall** (1907) be provided if an RA is present.

2.2.8.4.2

Multi-threat Encounters with DNA Traffic

The DNA mode is maintained during RAs on other aircraft unless the designated aircraft is part of a multi-threat encounter. During a multi-threat encounter that includes the DNA-designated traffic, DNA alert suppression on the designated aircraft **shall** (1908) be temporarily suspended until either:

1. the RA on the designated aircraft has been cleared, or
2. the multi-threat encounter has cleared (i.e., RAs on all other traffic have been cleared).

Notes:

1. *During a multi-threat RA, the traffic symbol for the DNA traffic will change to an RA symbol and the aural alert is not suppressed.*
2. *Condition 2) allows a continuing RA on the DNA-designated traffic to be suppressed from view once the threat of all other traffic has been resolved.*

2.2.8.5

Automatic Undesignation

ACAS Xo **shall** (1909) automatically undesignate traffic from DNA mode under any of these conditions:

1. The designated traffic has been invalid for DNA for more than the Designation Timeout Interval.
2. Ownship descends below the on-ground threshold (50 ft AGL on descent).

Note: This requirement automatically undesignates traffic when descending through 50 ft AGL to the “on ground” altitude. Below this altitude, all traffic is regarded as on ground. Designation of traffic when ownship is on-ground is permitted. However, ACAS X surveillance may not support on-ground traffic.

3. Ownship descends below the aural inhibit altitude (400 ft AGL), then climbs above the aural inhibit altitude (600 ft AGL).

Note: This requirement is intended to automatically undesignate traffic in some go-around situations.

4. Traffic range of diverging DNA-designated traffic exceeds the Diverging DNA Traffic Range Limit.

Note: This requirement automatically undesignates diverging traffic beyond the Diverging DNA Traffic Range Limit. Designation of traffic is permitted at any range.

5. Traffic bearing becomes invalid.
6. DNA Mode Availability is “Unavailable to Run”.

ACAS Xo **shall** (1910) automatically undesignate traffic from CSPO-3000 mode under any of these conditions:

Note: If an RA is present on the designated traffic, undesignation is delayed until the RA clears. See §2.2.8.4.1.

1. The designated traffic has been invalid for CSPO-3000 for more than the Designation Timeout Interval.

2. The designated traffic has climbed above the Maximum CSPO-3000 Altitude.

Note: See §2.2.8.2.3.2.

3. CSPO-3000 Mode Availability is “Unavailable to Run”.

Note: This automatically undesignates CSPO-3000 traffic when ACAS X switches into TA-Only, such as when ownship descends through the ACAS X RA-inhibit altitude or the pilot selects TA-Only mode.

2.2.8.5.1 Dropped Traffic

Designated dropped or invalid traffic **shall** (2032) be retained for the Designation Timeout Interval. The designated ACAS Xo mode may resume on this traffic if it becomes valid again within the Designation Timeout Interval.

Note: This allows traffic with spotty surveillance to maintain the Xo designation rather than reverting to ACAS Xa when the track is reestablished.

2.2.8.6 Logic Requirements

2.2.8.6.1 DNA

TAs and RAs generated in response to DNA traffic, except those RAs generated as part of a multi-threat RA, **shall** (1912) not be reported to the pilot. DNA traffic **shall** (1913) be displayed with a normal designated traffic symbol (i.e., not a TA or RA symbol) while an active TA or RA on the traffic is suppressed.

Note: TAs are never displayed or annunciated for traffic designated for DNA.

ACAS Xo **shall** (1914) report RAs that are generated in response to DNA traffic as ‘suppressed RAs’ in output for RA Reports and RA Broadcast Interrogation Messages.

Note: The SPI bit in the RA Report and ADS-B TCAS RA Broadcast message indicates whether an RA is currently being suppressed or not. See §2.2.5.6.3.3 and §2.2.3.9.4.

RAs generated in response to DNA traffic as part of a multi-threat RA **shall** (1915) be reported in output to the display system and in output for RA Reports and RA Broadcast Interrogation Messages. The DNA traffic **shall** (1916) be displayed with an RA symbol while the multi-threat RA is active.

2.2.8.6.1.1 DNA Coordination

ACAS Xo DNA mode performs a slightly modified air-to-air coordination with traffic identified as being equipped with TCAS II when that traffic is also a slave. In this protocol, ownship delays transmission of its initial coordination interrogation until the DNA aircraft (i.e., the traffic designated to DNA) has initiated coordination (i.e., until ownship has received a VRC from the DNA aircraft). The DNA aircraft can then independently choose its own RA. This mitigation is unnecessary when the DNA aircraft is identified as another ACAS X aircraft.

Notes:

1. *The delay in sending a coordination interrogation allows the TCAS II on the DNA aircraft to make the initial RA choice best suited for its collision avoidance logic, and the pilot of the DNA aircraft is able to see and act upon the displayed RAs. Ownship's ACAS Xo coordination with the DNA aircraft allows geometric reversals to occur when necessary, even though the pilot on ownship will not see an RA displayed for the DNA aircraft. That is, a master ACAS Xo aircraft can, if necessary, issue a geometric reversal, which then via coordination causes a slave TCAS II aboard a DNA aircraft to reverse its RA sense accordingly. This ability to preserve geometric reversal capability in encounters where a TCAS II aircraft is designated to DNA is considered necessary for safety. If the slave aircraft designated to DNA is equipped with ACAS X, it does not require coordination to perform a slave geometric reversal and this mitigation is not applied.*
2. *Some older transponders installed on ACAS X aircraft lack the ability to report CA equipage information in the CCCB subfield of ADS-B Operational Status Messages (ref §2.2.3.8.3.2.9.2). Since there is no way to know when this occurs, any aircraft reporting CCCB=0 is treated as having TCAS II equipage, and the modified coordination protocol must be used when the TCAS II traffic is the slave.*

If ownship is not coordinating with the DNA aircraft at the time of designation and the aircraft is a slave to ownship and identified as TCAS II, ACAS Xo **shall** (2154) transmit an initial coordination interrogation to the DNA aircraft only following receipt of a coordination interrogation from the DNA aircraft.

Note: The initial coordination interrogation must be transmitted to the DNA aircraft per §2.2.3.9.3 as soon as it is recognized as a threat by ACAS Xo after receipt of the initial coordination interrogation from that designated aircraft. It must not be transmitted before receipt of the initial coordination from the designated aircraft.

While ownship is coordinating with the TCAS II slave DNA aircraft, ACAS Xo **shall** (2155) continue to transmit coordination interrogations to that traffic per §2.2.3.9.3, even if the RA is suppressed for output to the display.

When traffic is designated to DNA, ACAS Xo **shall** (2156) modify Xa RA processing as described in the ADD (Volume II).

Note: In this context "is coordinating" means that the ACAS X TRM received a VRC for the DNA aircraft on the current logic processing cycle. In the absence of a CVC, the STM will continue to use a previously received VRC until its timeout threshold has been reached.

2.2.8.6.2**CSPO-3000**

The CSPO-3000 logic provides collision avoidance protection appropriate for closely-spaced parallel operations down to 3,000 ft runway spacing.

Note: The palette of possible CSPO-3000 mode RA announcements is the same as ACAS Xa. Required pilot response is the same.

2.2.8.7**State Tables**

The following tables may be helpful for understanding the ASA System and ACAS X system states associated with ACAS Xo.

Note: These tables are fairly high-level and include the nominal transitions.

The first table includes the traffic state information, for each traffic, sent by ACAS X to the ASA System.

Table 2-62: ASA System Traffic state information (for each traffic) for display for ACAS Xo

	Default state	Other possible states		
DNA validity	Invalid	Valid		
CSPO-3000 validity	Invalid	Valid		
Alert¹	None	TA	RA	
Traffic Designation²	Undesignated	Designated		
Designated Logic Mode³	ACAS Xa ⁵	DNA	CSPO-3000	
Actual Logic Mode⁴	ACAS Xa	DNA	CSPO-3000	multi-threat ⁶

Notes:

1. *Alert information is supplied for ACAS Xa. No additional field is necessary for Xo.*
2. *Default is Undesignated (ACAS Xa).*
3. *Default is ACAS Xa.*
4. *Nominally, the Actual Logic Mode will be the same as the Designated Logic Mode. Different logic modes can occur during mode transitions during RAs, and during multi-threat RAs while in DNA mode.*
5. *ACAS Xa is not actually a designated mode, but is invoked by undesignation from either Xo mode.*
6. *Multi-threat logic mode is only displayed for DNA traffic. It explains the presence of an RA on DNA traffic.*

The next two tables provide nominal ASA System states for each mode. The Data Block Status Info is provided by the Designated and Actual Logic Modes from ACAS X.

Table 2-63: Nominal ASA System DNA States (for each traffic)

State number	State name	Traffic Display Information		Transition out of state		
		Traffic Symbol	Indication to Flight Crew	Input prompting state change (source of change)	Resulting New state	Output triggered by transition
1	Invalid for DNA	Not designated		Valid (from ACAS Xo)	2	
2	Valid for DNA	Not designated		Invalid (from ACAS Xo)	1	
				Designate to DNA (from flight crew)	3	Mode (DNA) and ICAO 24-bit Aircraft Address sent to ACAS Xo
3	Designated as DNA	Designated	DNA	Invalid (from ACAS Xo)	1	Loss of Designation alert
				Undesignated (from flight crew)	2	Undesignation from Mode (DNA) and ICAO 24-bit Aircraft Address sent to ACAS Xo
				Undesignated (from ACAS Xo)	2	
				Multi-threat RA (from ACAS Xo)	4	
4	Designated as DNA but ACAS Xa active due to multi-threat	Designated RA	DNA multi-threat	Multi-threat RA clears (from ACAS Xo)	3	
				Invalid (from ACAS Xo)	1	Loss of Designation alert
				Undesignate (from flight crew)	2	Undesignation from Mode (DNA) and ICAO 24-bit Aircraft Address sent to ACAS Xo
				Undesignated (from ACAS Xo)	2	
5	Designated as DNA but CSPO-3000 active due to RA	Designated RA	DNA CSPO-3000	RA clears (from ACAS Xo)	3	
				Invalid (from ACAS Xo)	1	Loss of Designation alert
				DNA undesignated (from flight crew)	2	Undesignation from Mode (DNA) and ICAO 24-bit Aircraft Address sent to ACAS Xo

				DNA undesignated (from ACAS Xo)	2	
--	--	--	--	------------------------------------	---	--

Table 2-64: Nominal ASA System CSPO-3000 States (for each traffic)

State number	State name	Traffic Display Information		Transition out of state		
		Traffic Symbol	Indication to Flight Crew	Input prompting state change	Resulting New state	Output triggered by transition
1	Invalid for CSPO-3000	Not designated		Valid (from ACAS Xo)	2	
2	Valid for CSPO-3000	Not designated		Invalid (from ACAS Xo)	1	
				Designate to CSPO-3000 (from flight crew)	3	Mode (CSPO-3000) and ICAO 24-bit Aircraft Address sent to ACAS Xo
3	Designated as CSPO-3000	Designated, Designated TA, Designated RA	CSPO-3000	Invalid (from ACAS Xo)	1	Loss of Designation alert
				Undesignated (from flight crew)	2	Undesignation from Mode (CSPO-3000) and ICAO 24-bit Aircraft Address sent to ACAS Xo
				Undesignated (from ACAS Xo)	2	
4	Designated as CSPO-3000 but Xa active due to active RA	Designated RA	CSPO-3000 ACAS Xa	RA clears	3	
				Invalid (from ACAS Xo)	1	Loss of Designation alert
				CSPO undesignated (from flight crew)	2	Undesignation from Mode (CSPO-3000) and ICAO 24-bit Aircraft Address sent to ACAS Xo
				CSPO undesignated (from ACAS Xo)	2	
5	Designated as CSPO-3000 but DNA active due to active RA	Designated RA	CSPO-3000 DNA	RA clears	3	
				Invalid (from ACAS Xo)	1	Loss of Designation alert
				CSPO undesignated (from flight crew)	2	Undesignation from Mode (CSPO-3000) and ICAO 24-bit Aircraft Address sent to ACAS Xo
				CSPO undesignated (from ACAS Xo)	2	

The next two tables provide nominal ACAS Xo states for each mode. Additional states exist for mode transitions during RAs.

Table 2-65: Nominal ACAS Xo System DNA States (for each traffic)

State number	State name	Status reported to ASA System		Transition out of state		
		Mode Status	Traffic Validity for DNA	Input prompting state change	Resulting New state	Output triggered by transition
1	Invalid for DNA		Invalid	Meets all valid criteria	2	
2	Valid for DNA		Valid	Does not meet all valid criteria	1	
				Designate to DNA (from flight crew)	3	
3	Designated as DNA	DNA	Valid	Does not meet all valid criteria – can support ACAS Xa	4	
				Does not meet all valid criteria – cannot support ACAS Xa	6	
				Undesignated (by flight crew via ASA System)	2	
				Undesignated (by ACAS Xo) – descend below 50 ft	2	
				Undesignated (by ACAS Xo) – descend below 400 then climb above 600 ft	2	
				Undesignated (by ACAS Xo) – diverging traffic range exceeds Diverging DNA Traffic Range Limit	2	
				Multi-threat alert involving designated traffic	5	
4	Designated as DNA but in Invalid time-out – can still support ACAS Xa	DNA ACAS Xa	Valid	Undesignated (by ACAS Xo) – Designation Timeout Interval runs out	1	Loss of Designation alert
				Undesignated (by ACAS Xo) – descend below 50 ft	1	
				Undesignated (by ACAS Xo) – descend below 400 then climb above 600 ft	1	
				Undesignated (by ACAS Xo) – diverging traffic range exceeds Diverging DNA Traffic Range Limit	1	
				Undesignated (by flight crew)	1	
				Meets all valid criteria	3	

5	Designated as DNA but with Multi-threat RA	DNA Multi-threat	Valid	DNA traffic RA clears	3	
				Other traffic RA clears	3	
				Undesignated (by flight crew via ASA System)	2	
				Undesignated (by ACAS Xo) – diverging traffic range exceeds Diverging DNA Traffic Range Limit	2	
				Does not meet all valid criteria – can support ACAS Xa	4	
				Does not meet all valid criteria – cannot support ACAS Xa	6	
				Designation Timeout Interval runs out	Dropped from designation list	Loss of Designation alert
6	Maintain in designation list until Designation Timeout Interval time-out	None	Valid	Track reacquired – meets all valid criteria	3	
				Track reacquired – does not meet all valid criteria	4	
				Undesignated due to flight crew designating another aircraft	Dropped from designation list	
				RA clears	3	
7	Designated as DNA but in CSPO-3000 due to RA	DNA CSPO-3000	Valid			

Table 2-66: Nominal ACAS Xo System CSPO-3000 States (for each traffic)

State number	State name	Status reported to ASA System		Transition out of state		
		Mode Status	Traffic Validity for CSPO-3000	Input prompting state change	Resulting New state	Output triggered by transition
1	Invalid for CSPO-3000			Invalid	Meets all valid criteria	2
2	Valid for CSPO-3000		Valid	Does not meet all valid criteria	1	
				Designate to CSPO-3000 (from flight crew)	3	
3	Designated as CSPO-3000	CSPO-3000	Valid	Does not meet all valid criteria – can support ACAS Xa	4	
				Does not meet all valid criteria – cannot support ACAS Xa	5	
				Undesignated (by flight crew via ASA System)	2	
				CSPO-3000 is Unavailable to Run	2	Loss of Designation alert
4	Designated as CSPO-3000 but in Invalid time-out – can still support ACAS Xa	CSPO-3000 ACAS Xa	Valid	Undesignated (from ACAS Xo) – Designation Timeout Interval runs out	1	Loss of Designation alert
				CSPO-3000 is Unavailable to Run	1	Loss of Designation alert
				Undesignated (by flight crew via ASA System)	1	
				Undesignated (by ACAS Xo)	1	
				Meets all valid criteria	3	
5	Maintain in designation list until time-out	None	Valid	Time out	Dropped from designation list	Loss of Designation alert
				Track reacquired – meets all valid criteria	3	
				Track reacquired – does not meet all valid criteria	4	
				Undesignated due to flight crew designating other traffic	Dropped from designation list	
6	Designated as CSPO-3000 while RA is active in Xa	CSPO-3000 ACAS Xa	Valid	RA clears	3	
				CSPO-3000 undesignated (from flight crew or ACAS Xo)	2	

7	Designated as CSPO-3000 while RA is suppressed in DNA	CSPO-3000 DNA	Valid	RA clears	3 2	
				CSPO-3000 undesignated (from flight crew or ACAS Xo)		

2.2.8.8 Xo / ASA System Interface

ACAS Xo control and display functions are accomplished via an integrated ASA System or equivalent. This section describes the interface requirements necessary to support ACAS Xo.

2.2.8.8.1 Xo Mode Availability

Xo Mode Availability is similar to the ASA System Application Status, and provided for each installed Xo Mode.

The ACAS X system **shall** (2141) send the Xo Mode Availability for all installed Xo modes.

2.2.8.8.2 Send Traffic Xo Mode Validity

For all traffic output from the ACAS X system, the system **shall** (2143) send the individual Traffic Xo Mode Validity, for all installed Xo modes, to the ASA System.

2.2.8.8.3 Send Traffic Designation Information

For all traffic output from the ACAS X system, the system **shall** (2145) send the Designated Xo Mode to the ASA System.

For all traffic output from the ACAS X system, the system **shall** (2146) send the Actual ACAS X logic mode to the ASA System.

For all traffic output from the ACAS X system, the system **shall** (2144) send the Designation Status to the ASA System. See Table 2-67 for possible values.

Note: The Designated ACAS Xo mode is indicated to the flight crew to convey that the designation mode is set. The actual ACAS X logic mode is included to inform the flight crew when ACAS X is temporarily operating in a different mode than designated.

Table 2-67: Designation Status

Code	Meaning	Alert	Xo Mode
0x00	Nothing abnormal to report	No	DNA or CSPO-3000
Designation Processing Delayed			
0x11	Due to active RA	No	DNA or CSPO-3000
Designation Processing Suspended			

0x21	Due to multi-threat RA involving designated traffic Processing will be restored after condition no longer holds	No	DNA
0x22	Traffic is invalid for TA and RA processing Designation is being coasted	No	DNA or CSPO-3000
0x23	Traffic was dropped Designation is being coasted	No	DNA or CSPO-3000
0x24	Designation is being coasted	No	DNA or CSPO-3000
Automatically Undesignated			
0x31	Due to Xo mode becoming Unavailable to Run	Yes	CSPO-3000
0x32	Due to satisfying geometric constraints for undesignation	No	DNA or CSPO-3000
0x33	Dropped traffic was undesignated	No	DNA or CSPO-3000
0x34	Due to timeout of coasting designation	Yes	DNA or CSPO-3000
0x35	Due to loss of bearing	Yes	DNA

2.2.8.8.4 Receive Xo Traffic Designation/Undesignation Request

In order to designate or undesignate traffic for ACAS Xo, Xo receives target identification and the Xo mode designation/undesignation request from the ASA System. Designation and undesignation may be performed by the flight crew or automatically by an ASA System application.

The ACAS X system **shall** (2138) receive a reference to uniquely identify the traffic to designate or undesignate from the ASA System.

The ACAS X system **shall** (2139) receive the requested Xo mode designation/undesignation from the ASA System.

Note: The information associated with the traffic reference must include the traffic's ICAO 24-bit aircraft address. As the STM is defined, the ICAO 24-bit Aircraft Address and ACAS Xo mode are used by ACAS Xo to identify the traffic and the Xo logic to be applied to the traffic. If the ICAO 24-bit address is not available, additional analysis must be performed to ensure that the system will work as specified.

2.3

Equipment Performance - Environmental Conditions

The environmental tests and performance requirements described in this subsection are intended to provide a laboratory means of determining the overall performance characteristics of the equipment under conditions representative of those which may be encountered in actual operations.

Some of the environmental tests contained in this subsection do not have to be performed unless the manufacturer wishes to qualify the equipment for that particular environmental condition. These tests are identified by the phrase "When Required." If the manufacturer wishes to qualify the equipment to these additional environmental conditions, then these "When Required" tests **shall** (1918) be performed.

Unless otherwise specified, the test procedures applicable to a determination of equipment performance under environmental test conditions are set forth in Ref. D.

Some of the performance requirements in §2.1 and §2.2 are not required to be tested to all of the conditions contained in Ref. D. Judgment and experience have indicated that these particular performance parameters are not susceptible to certain environmental conditions, and that the level of performance specified in §2.1 and §2.2 will not be measurably degraded by exposure to these particular environmental conditions.

2.3.1 Environmental Test Conditions

The equipment shall be subjected to the environmental conditions and tests defined in the following tables. Table 2-68 Environmental Test Groupings, places the operational test procedures used during environmental testing into specific test groups. These test groups are then identified in Table 2-69: Applicable Test Groups per Environmental Condition, which indicates which test group(s) are performed for each of the environmental conditions specified in Ref. D. For each environmental condition all of the operational test procedures pertinent to the indicated groups shall be performed.

Table 2-68: Environmental Test Groupings

Test Procedure Paragraph	Description	Environmental Test Group						Comments
		1	2	3	4	5	6	
2.4.2.1.2.1	In Band Acceptance	x	x	x	x			
2.4.2.1.2.2	Out of Band Rejection	x	x	x	x			
2.4.2.1.3.1								
2.4.2.1.3.2	Reply Link Interference	x	x	x	x			
2.4.2.1.1.1	Transmit Frequency	x	x	x	x	x		
2.4.2.1.1.2	Radiated Output Power	x	x	x	x	x		
2.4.2.1.1.6	Unwanted Output Power	x	x		x			
2.4.2.1.1.4	Transmit Pulse Characteristics	x	x	x	x	x		
2.4.2.1.4.1								
2.4.2.1.4.2	Mode C Reply Reception	x	x	x				
2.4.2.1.4.7	TACAN and DME Rejection	x	x	x				Note 1
2.4.2.1.5.1	Mode S Squitter and Reply Reception	x	x	x				
2.4.2.1.1.3.1	Control of Synchronous Garble by Whisper-Shout	x	x	x	x	x		Note 2
2.4.2.1.6.4	Surv Target Capacity (Mode C)	x	x	x				
2.4.2.1.7.5	Surv Target Capacity and Overload (Mode S)	x	x	x				
2.4.2.1.6.1	Mode C Surv Initiation	x	x	x				
2.4.2.1.7.1	Mode S Surv Initiation	x	x	x				
2.4.2.9	Monitor Requirements	x	x	x	x	x		
2.4.2.1.9.2.1	Mode C Interleaved Replies	x	x	x				
2.4.2.1.9.2.3	Mode S Overlapped Replies	x	x	x				
2.4.3	Computer Performance Verification	x						
2.4.2.1.10.1	Antenna System Radiation Pattern						x	
2.4.2.1.9.1	Bearing Accuracy						x	
Ref. G - 2.3.2.3	Receivers Shared with a TCAS unit	x	x	x	x			Note 3
Ref. G - 2.3.2.3.1	TCAS Compatibility	x	x	x	x			Note 3
Ref. G - 2.3.2.3.2	Re-Triggerable Reply Processor	x	x	x	x			Note 3
Ref. G - 2.3.2.3.3	Verification of Enhanced Squitter Reception Techniques							No Test
Ref. G - 2.3.2.3.3.1	Combined Preamble and Data Block Tests with A/C FRUIT	x	x	x	x			Note 3
Ref. G - 2.3.2.3.3.2	Data Block Tests with Mode S FRUIT	x	x	x	x			Note 3
Ref. G - 2.3.2.3.3.3	Re-Triggering Performance	x	x	x	x			Note 3
Ref. G - 2.3.2.4	Receivers Not Shared with TCAS							No Test
Ref. G - 2.3.2.4.1	In-Band Acceptance	x	x	x	x			Note 3
Ref. G - 2.3.2.4.2	Dynamic Range	x	x	x	x			Note 3
Ref. G - 2.3.2.4.3	Re-Triggerable Capability	x	x	x	x			Note 3
Ref. G - 2.3.2.4.4	Out-Of-Band Rejection	x	x	x	x			Note 3

Ref. G - 2.3.2.4.5	Dynamic Minimum Trigger Level	x	x	x	x		Note 3
Ref. G - 2.3.2.4.6	Criteria for ADS-B Message Transmission Pulse Detection	x	x	x	x		Note 3
Ref. G - 2.3.2.4.7	Criteria for Data Block Acceptance in ADS-B Message Signals	x	x	x	x		Note 3

Notes:

1. *Perform test per §2.4.2.1.4.7 except only one run is required with N=10.*
2. *It is only necessary to perform this test in whisper shout test mode 1 since whisper shout test mode 2 is a subset.*
3. *Ref. G tests are applicable only if the ACAS X unit operates with a receiver that is more sensitive than the MTL requirements specified in this document.*

Table 2-69: Applicable Test Groups per Environmental Condition

Environmental Condition	Ref. D Test Paragraph	Applicable Operational Test Groups	Comments
Low Temperature	4.5.1, 4.5.2	1	
High Temperature	4.5.3, 4.5.4, 4.5.5	1	
Altitude	4.6.1	5	
Decompression (when required)	4.6.2	5	
Overpressure (when required)	4.6.3	5	
Temperature Variation	5.3	2	
Humidity	6.3	2 & 6	
Operational Shock	7.2	4 & 2	
Crash Safety Shocks	7.3	NONE	Note 1
Vibration	8.0	4 & 2	
Explosion (when required)	9.0	NONE	
Waterproofness - Drip (when required)	10.3.2	5 & 6	
Waterproofness - Spray (when required)	10.3.3	NONE	
Fluid Susceptibility - Spray (when required)	11.4.1	5 & 6	
Fluid Susceptibility - Immersion (when required)	11.4.2	NONE	
Sand and Dust (when required)	12.4, 12.5	5 & 6	
Fungus Resistance (when required)	13.5	5 & 6	
Salt Fog (when required)	14.3.6	5 & 6	
Magnetic Effect (when required)	15.3	NONE	
Power Input - Normal	16.5.1, 16.6.1	3	
Power Input - Abnormal	16.5.2, 16.6.2	3	Note 2
Voltage Spike Conducted Category A (if applicable)	17.4	3	
Voltage Spike Conducted Category B (if applicable)	17.4	3	
Audio Frequency Conducted Susceptibility	18.3	3	
Induced Signal Susceptibility	19.3	3	
Radio Frequency Susceptibility	20.0	3	
Emission of Radio Frequency Energy	21.0	NONE	
Lightning Induced Transient Susceptibility	22.0	4	

Notes:

1. *The application of the Crash Safety Shock tests may result in damage to the equipment under test. Therefore, this test may be conducted after the other tests have been completed.*
2. *The application of the Low Voltage Conditions (DC) (Category B Equipment) test may result in damage to the equipment under test. Therefore, this test may be conducted after the other tests have been completed.*

2.4

Equipment Test Procedures

The test procedures described in this section are intended to provide a laboratory means of determining the overall performance characteristics of the equipment.

The surveillance bench tests in this section are designed to verify the ACAS X surveillance performance in a deterministic manner in the presence of individual and well defined environmental phenomena. For initial equipment certification, flight tests are required (§3.4.4) in order to verify ACAS X surveillance performance in an environment that includes the effects of simultaneous combinations of environmental phenomena such as multipath, high density fruit and ground sensor interference. These flight tests need to be made in areas that are known to include simultaneous combinations of such phenomena and should be of a duration sufficient to collect adequate data for a satisfactory and detailed post-flight evaluation.

2.4.1

Definitions of Terms and Conditions of Test

The following definitions of terms and conditions of tests are applicable to the equipment tests specified herein:

a. **Power Input Voltage**

All tests **shall** (2237) be conducted with the power input voltage adjusted to design voltage ± 2 percent. The input voltage **shall** (2238) be measured at the input terminals of the equipment under test.

b. **Power Input Frequency**

1. For equipment designed to operate from a constant frequency (e.g., 400 Hz) AC source, the input frequency **shall** (2239) be adjusted to within ± 5 percent of design frequency.
2. In the case of equipment designed for operation from an AC source of variable frequency (e.g., 300 to 400 Hz), tests **shall** (2240) be conducted with the input frequency adjusted to within five percent of a selected frequency and within the range for which the equipment is designed.

c. **Adjustment of Equipment**

The circuits of the equipment under test **shall** (2241) be properly aligned and otherwise adjusted in accordance with the manufacturer's recommended practices prior to application of the specified tests. No adjustments **shall** (2242) be made once the test procedures have started.

d. **Test Instrument Precautions**

Precautions **shall** (2243) be taken during conduct of the tests to prevent the introduction of errors resulting from the connection of voltmeters, oscilloscopes and

other test instruments across the input and output terminals of the equipment under test.

e. Ambient Conditions

All tests **shall** (2244) be conducted under conditions of ambient room temperature, pressure and humidity. However, the room temperature **shall** (2245) not be lower than +10 degrees C.

f. Connected Loads

All tests **shall** (2246) be performed with the equipment connected to loads having the impedance values for which it is designed.

g. Signal Levels

In the test procedures all signal levels are specified at the antenna end of the antenna-to-ACAS X transmission line. This point is referred to as "the RF Reference Point."

h. Collocated Mode S Transponder

The test procedures in this document rely on the availability of a Mode S transponder that is fully integrated into or interfaced with the ACAS X equipment under test and is assumed to be a part of the ACAS X equipment.

i. ACAS X Track File Information

Some of the tests in this section specify that surveillance track file information be provided to evaluate the success or failure of a test. Track file information **shall** (2247) be provided for each surveillance update interval and **shall** (2248) consist of the following data for each target in the scenario:

1. Time reference.
2. Range.
3. Altitude.
4. Target type (i.e., Mode C, Mode S or TCAS).
5. Mode S track state (i.e., squitter, acquisition, 1-second track update rate, or 5-second track update rate).
6. Track update (i.e., valid reply or coasted).

The track file information **shall** (2249) be provided in a format that will enable a convenient comparison and time correlation with the test scenario events and with the ACAS X interrogation signals and, when applicable, RAs normally recorded by the test equipment.

j. ACAS X Mode C Altitude Encoding

Some of the Mode C-related tests in this section specify an altitude and the related Mode C altitude code as encoding of that altitude.

The Mode C altitude encoding uses Gillham code. The altitude code values listed in this section are octal numbers, each of those four digits representing three bits of the altitude code. The bit order of the Gillham code is used as follows:

A1 A2 A4 B1 B2 B4 C1 C2 C4 D1 D2 D4

See Ref. G, §2.4.1.

2.4.1.1 Standard Test Signals

Unless otherwise specified, the Mode C and Mode S test signals applied to the ACAS X equipment **shall** (2250) have signal characteristics corresponding to the reply signals generated by transponders conforming to Ref. B.

Standard Test Signal Characteristics

a. Radio Frequency

The frequency of the signal generator for Mode C and Mode S replies and fruit **shall** (2251) be 1090 ± 0.1 MHz, unless otherwise specified.

b. CW Output

CW output between pulses **shall** (2252) be at least 60 dB below the peak level of the pulse.

c. Pulse Shape and Position

The specified test signal pulse shapes are summarized in Table 2-70 below (all values in microseconds):

Table 2-70: Standard Test Signal Pulse Shapes

Function	Pulse Duration	Tolerance	Rise Time		Decay Time		Pulse Position Tolerance
			Min	Max	Min	Max	
Mode C	0.45	0.05	0.05	0.1	0.05	0.2	0.05
Mode S	0.50	0.05	0.05	0.1	0.05	0.2	0.05
Mode S	1.00	0.05	0.05	0.1	0.05	0.2	0.05

Note: Interval measurements and rise and fall times are measured by a linear detector.

d. Pulse Top Ripple

Between the 90 percent voltage amplitude point on the leading and trailing edges of the pulse, the instantaneous amplitude of the pulses **shall** (2253) not fall more than one dB below the maximum value.

e. Signal Level

The signal level corresponding to each simulated target will be as specified for each test procedure.

f. Mode S Phase Modulation

The short (16.25 microsecond) and long (30.25 microsecond) P6 pulse **shall** (2254) have internal modulation consisting of 180-degree phase reversals of the carrier frequency at the designated times.

g. Mode S Phase Reversal Duration

The duration of the phase reversal **shall** (2255) be less than 0.08 microseconds as measured between the 10 degree and the 170 degree points of the phase transition. The interval between the 80 percent points of the amplitude transient associated with the phase reversal **shall** (2256) be less than 0.08 microseconds.

h. Mode S Phase Relationship

The tolerance on the 0 or 180 degree phase relationship between successive chips within the P6 pulse (including the sync reversal) **shall** (2257) be ± 5 degrees.

i. Jitter

The total jitter corresponding to the variation in the delay time between the interrogation signal and the corresponding simulated target reply signal after accounting for the time delay associated with the programmed movement of the target aircraft in space **shall** (2258) not exceed 0.1 microseconds. The time delay **shall** (2259) be measured in accordance with Table 2-71.

Table 2-71: Standard Test Signal Total Jitter

Function	Start-time Reference	Stop-Time Reference	Total Allowable Jitter
Mode S	P6 Sync Phase Reversal	Leading Edge of First Preamble Pulse	0.1 μ s
Mode C	Leading Edge of P3	Leading Edge of First Reply Pulse	0.1 μ s

2.4.1.2

Test Equipment Capabilities

The test procedures rely on the availability of test equipment that will interface with all external ACAS X input and output ports and that is capable of generating all of the RF test signals specified in each test procedure. These signals include:

- a. Replies produced in response to Mode C-Only All-Call and Mode S interrogations from the ACAS X unit under test.
- b. Mode S squitter signals.
- c. TCAS broadcast interrogation signals.
- d. Mode A/C fruit replies.
- e. Interrogations from simulated TCAS and ground sensor equipment.

The test equipment **shall** (2260) also be capable of receiving and measuring the characteristics of all output RF signals generated by the ACAS X equipment under test.

Figure 2-25 illustrates the signal relationship between the ACAS X equipment, the Mode S transponder under test and the test equipment.

The amplitude, timing and data content of test signals generated by the test equipment **shall** (2261) be controlled by individual test scenarios to simulate specific target conditions depending on the requirements of a particular test. The RF characteristics of all simulated test signals generated by the test equipment **shall** (2262) be in accordance with §2.4.1.1, unless otherwise specified in the individual test procedure.

The test equipment **shall** (2263) be capable of generating and varying over their full range of possible values the following inputs to the ACAS X equipment that are normally provided by external equipments:

- a. ICAO 24-bit Aircraft Address.
- b. Maximum operating airspeed code.
- c. Barometric altitude code.

- d. Pilot manual control.
- e. Radio altimeter input.
- f. Heading.

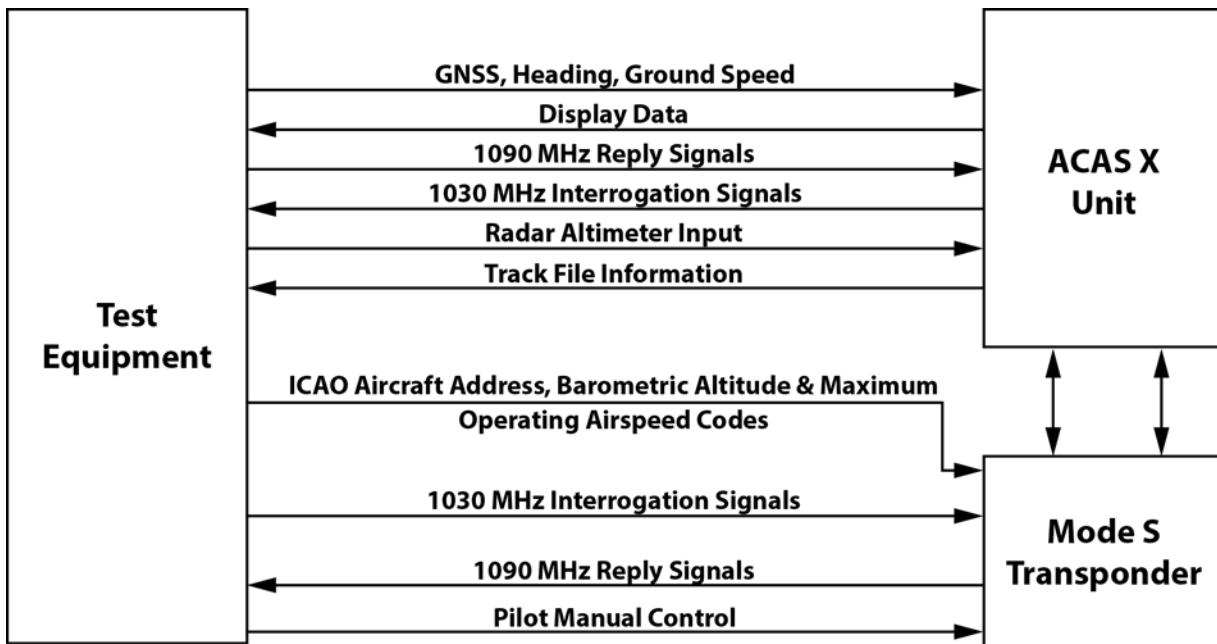


Figure 2-25: Signal Relationship Between ACAS X, Mode S & Test Equipment

The test equipment **shall** (2264) be capable of accepting the ACAS X RA signal that is normally sent to the cockpit display. Recorded data, the ACAS X RA, when generated, the Traffic Display, and RF output signals **shall** (2265) be used as means of verifying the test procedures.

The test equipment **shall** (2266) have the following minimum capabilities:

- a. Means of generating and varying over their full range of possible values the following reply signals in response to each Mode S or Mode C-Only All-Call interrogation from TCAS. The timing of each reply signal **shall** (2267) be controlled by the test equipment to simulate the target range and relative speed specified in each test procedure.

1. A Mode S reply with the characteristics specified in §2.4.1.1 and with the data field, signal level and carrier frequency specified in each test procedure. The data field from a given target **shall** (2268) be able to be changed during a single test at times specified in the test procedures.

A capability **shall** (2269) be provided to generate modified Mode S replies whose data block contains energy in both halves of designated bit positions. The amplitude of the signal in the half that ordinarily contains energy **shall** (2270) be 3 ± 0.5 dB below the amplitude of the signal in the other half of the bit position.

2. A Mode C reply for each simulated target with the characteristics specified in §2.4.1.1 and with the altitude code, signal level and carrier frequency specified in each test procedure. Each Mode C target **shall** (2271) be made to respond only to the whisper-shout step number specified in the test procedure.

Notes:

1. *This capability permits the generation of crossing targets that do not garble.*
 2. *Unless otherwise specified by the test procedure, the scenario should be configured so as to cause ACAS X to generate the necessary whisper-shout sequence to produce the desired results.*
- b. Means of generating garbled Mode C replies in response to a single Mode C interrogation as follows:
 1. As many as three equal-amplitude replies with interleaved or overlapping pulses from a single 1090 MHz source.
Unless otherwise stated the specified range difference between replies **shall** (2272) be maintained to an accuracy of ± 0.05 μ s throughout the scenario.
 2. Two unequal-amplitude replies with interleaved or overlapping pulses from two separate RF sources that have carrier frequencies of 1090 and 1090.3 MHz, respectively. The specified range difference between replies **shall** (2273) be maintained to an accuracy of ± 0.2 μ s throughout the scenario. The range of one reply relative to the other reply **shall** (2274) be variable in steps of 0.05 μ s from 21.3 μ s earlier to 21.3 μ s later.
 - c. Means of generating Mode S squitter signals and Mode S fruit replies for each simulated Mode S target with the characteristics specified in §2.4.1.1 and with the data field, signal level and carrier frequency specified in each test procedure. The relative timing of squitter and fruit signals from each simulated target **shall** (2275) be controlled to provide a minimum interval of 100 μ s between the leading edges of any two squitters or fruit replies arriving at the ACAS X. All squitter and fruit signals **shall** (2276) be generated during the ACAS X squitter listening interval following the last transmission of a whisper-shout sequence.

A capability **shall** (2277) be provided to generate modified squitter and fruit signals whose data block contains energy in both halves of designated bit positions.

For squitters, the amplitude of the pulse in the half that ordinarily contains energy **shall** (2278) be 3 ± 0.5 dB below the amplitude in the other half. For fruit replies, the amplitude in the half that ordinarily contains no energy **shall** (2279) be 3 ± 0.5 dB below the amplitude in the other half.

Unless otherwise specified, the ICAO 24-bit Aircraft Address of each Mode S aircraft within a scenario, including own, **shall** (2280) be unique.

- d. Means of simulating the following combinations of Mode C and Mode S targets that are active simultaneously:
 - 1. A maximum of 24 non-garbling Mode C targets within a range of five NM of ACAS X.
 - 2. A maximum of 24 Mode S targets within a range of five NM of ACAS X and a maximum of 150 Mode S targets within 30 NM of ACAS X.
 - 3. A maximum of 12 non-garbling Mode C targets and 12 Mode S targets within 5 NM of ACAS X and a maximum of 75 Mode S targets within 30 NM of ACAS X.
- e. Means of generating Mode C fruit replies, up to a maximum rate of 30,000 replies per second and with the Mode C signal characteristics specified in §2.4.1.1. The specific reply rate and signal level distribution is specified as required in each test procedure. The timing of individual fruit replies **shall** (2281) be controlled to prevent random interference with other test signals generated by the test equipment. Altitude codes associated with each fruit reply **shall** (2282) be selected randomly from all possible valid codes.
- f. Means of generating Mode C interrogation signals with the RF characteristics specified in Ref. A. and with the interrogation rate and signal level specified in the test procedure.
- g. Means of generating Mode S interrogation signals from each simulated TCAS target and up to two simulated Mode S ground sensors with the RF characteristics specified in Ref. B. The values in the data fields of any TCAS target or ground sensor interrogation **shall** (2283) be as specified in the test procedures and **shall** (2284) be able to be changed during a single test at the times specified. Mode S interrogation signals from specific TCAS targets **shall** (2285) be generated during the test procedure time intervals specified and, unless otherwise specified, **shall** (2286) occur 0.05 second after the receipt of an interrogation from the ACAS X unit under test.
- h. Means of generating TCAS broadcast interrogation signals from simulated TCAS targets at times specified in the test procedure. When specified, the TCAS broadcast interrogation signal from each simulated target **shall** (2287) be generated at random intervals that are uniformly distributed over the range from 9.8 to 10.2 seconds.
 - i. Means of generating barometric and radio altimetry data in the format required by the ACAS X Mode S transponder and with the value specified in each test procedure.
 - j. Means of accepting and decoding all Mode S and Mode C-Only All-Call interrogation signals from the ACAS X under test.
 - k. Means of accepting all ACAS X RA signals and aural alarm signals normally output for annunciation in the cockpit.
 - l. Means of measuring the ACAS X transmitted signal characteristics as specified in the test procedures.

-
- m. Means of processing and recording the data content and arrival time of all interrogation and reply signals and RA signals received from the ACAS X under test as well as the data content and transmission time of all reply and interrogation signals generated by the test equipment. The processed and recorded data should indicate ACAS X performance in terms of the expected output for each test procedure.
 - n. Means of accepting and recording ACAS X track file information specified in §2.4.1(i).
 - o. Means of generating three simultaneous, frequency independent, amplitude and relative time controllable reply signals at RF carrier frequencies of 1065 ± 0.1 MHz, 1087 ± 0.1 MHz, 1090 ± 0.1 MHz, 1090.3 ± 0.1 MHz, 1093 ± 0.1 MHz and 1115 ± 0.1 MHz.
 - p. Means of generating reply signals with amplitude level between -12 and -80 dBm as measured at the RF reference point. The absolute amplitude accuracy **shall** (2288) be within ± 1.0 dB of the level specified in the test procedure for signals between -12 and -68 dBm. The absolute amplitude accuracy **shall** (2289) be ± 0.5 dB of the level specified in the test procedure for signals from -68 to -80 dBm.
 - q. Means of determining that the interrogation signals conform to the requirements of this document.
 - r. Means of measuring the interrogation power levels including multiple interrogation levels that may be employed.
 - s. Means of simulating linear movement of all targets in slant range. Movement is specified independently in the slant range dimension and the altitude dimension. In slant range, the trajectory is expressed in relative coordinates, consisting of initial slant range, minimum slant range and relative speed; however, the relative timing may be offset during a test by a fixed amount. The relative speed remains constant for any given target for the duration of the encounter. The test equipment **shall** (2290) be able to simulate target relative speed from negative 1200 kt (decreasing range) to positive 1200 kt (increasing range). The meaning of these variables is indicated in Figure 2-26.

Page 296 of 300

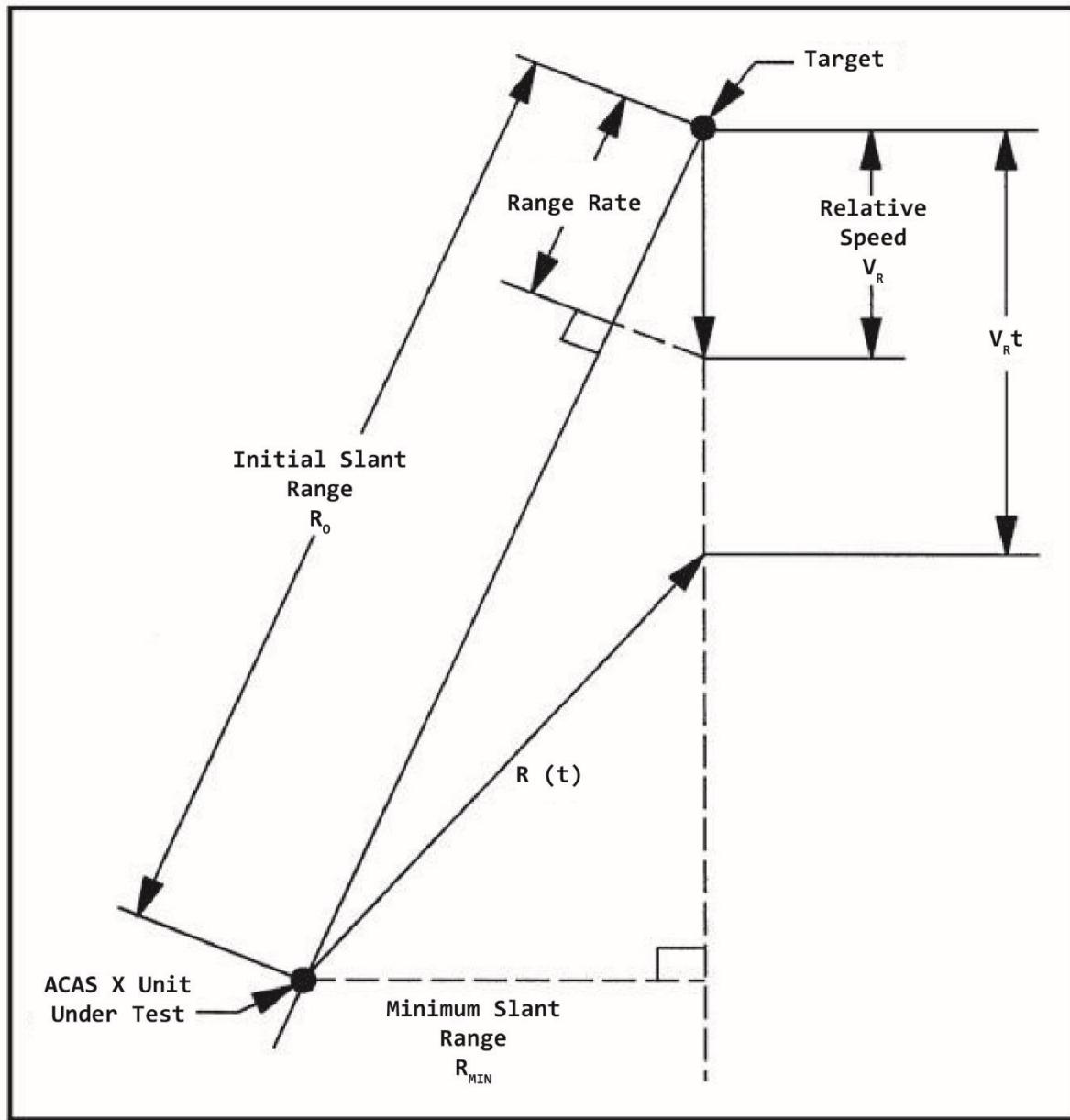


Figure 2-26: Definition Of Variables Associated With The Linear Movement Of Targets In The Range Dimension

Note: Although the actual motion for the target is linear, the slant range as a function of time behaves as a quadratic. The formula for slant range at time, t , in terms of initial slant range R_0 , minimum slant range R_{min} and relative speed V_R , is

$$R(t) = \sqrt{R_0^2 - 2V_R t(R_0^2 - R_{\text{min}}^2)^{1/2} + V_R^2 t^2}$$

Range rate is different from relative speed. Range rate is not constant during an encounter with non-zero R_{min} .

- t. Means of simulating piecewise linear movement of all targets in the altitude dimension. Each target will maintain a constant altitude rate within specified intervals, but may change its rate instantaneously at times specified in each test procedure. The nominal

altitude trajectory for a target will be generated by starting from the initial altitude and applying the initial altitude rate until the first change in altitude rate occurs, then applying that rate until the next change, and so on. The test equipment **shall** (2291) be able to simulate positive or negative altitude rates of 0 to 10,000 feet per minute. To generate a particular altitude report, the nominal altitude of the target at that time will be determined and then rounded to the nearest 100 feet for Mode C-equipped targets and to the nearest 25 feet for Mode S-equipped targets (unless otherwise stated). The altitude reports **shall** (2292) be generated independently of the range reports for a given target.

- u. Means of generating a single Mode C target with nonlinear slant range motion that can simulate multipath replies from a target in linear motion.
- v. Means of generating a single Mode C target with nonlinear slant range motion that can simulate an intruder initiating a turn into ownship from a flight path that is initially parallel to ownship flight path.
- w. Means of adjusting the time of reply to each interrogation to simulate target ranges between 500 feet and 30 NM. The reported altitude and the simulated range for each target at the time of interrogation **shall** (2293) differ from the nominal range and altitude of the specified trajectory by an amount no greater than the displacement of the target in range or altitude in 0.1 second.
- x. Means of inhibiting the Mode C or Mode S replies and Mode S squitters from a given target during time intervals specified in each test procedure.
- y. Means of initiating or terminating Mode C or Mode S targets at times during a single test as specified in each test procedure.
- z. Means of simulating a change in the setting of the pilot manual switch to any position at times as specified in each test procedure.
- aa. Means of generating altitude codes for individual replies for altitudes from -1200 to 126,700 ft MSL.
- bb. Means of adjusting individual reply pulse position, width, and rise and fall time within the tolerance range specified in §2.4.1.1.
- cc. Means of generating reply signals that simulate the actual antenna output signals that would result from each intruder whose bearing relative to ACAS X is specified in the test procedure.
- dd. Means of generating the following nonstandard test signals at the power levels and times specified in the test procedure.
 - 1. Narrow Pulse

Frequency	=	1090 MHz
Pulse Width	=	$0.3 \pm 0.01 \mu s$
Rise Time	=	0.05 to 0.1 μs
Fall Time	=	0.05 to 0.2 μs
 - 2. Wide Pulse

Frequency	=	1090 MHz
Pulse Width	=	$120 \pm 1.0 \mu s$
Rise Time	=	0.05 to 0.1 μs
Fall Time	=	0.05 to 0.2 μs

3. Slow Rise Time

Frequency = 1090 MHz
 Pulse Width = $3.0 \pm 0.2 \mu s$
 Rise Time = $0.6 \pm 0.1 \mu s$
 Fall Time = $0.6 \pm 0.1 \mu s$

4. Mode C Reply with code pulse widths of $0.1 \pm 0.01 \mu s$ and with the remaining characteristics as specified in §2.4.1.1.
5. Mode S reply whose data block conforms to the pulse characteristics specified in §2.4.1.1 and whose four preamble pulses are replaced by a single preamble pulse with the following characteristics:

Frequency = 1090 MHz
 Pulse Width = $5 \pm 0.05 \mu s$
 Rise Time = $0.05 \pm 0.1 \mu s$
 Fall Time = $0.05 \pm 0.2 \mu s$

The leading edge of the single preamble pulse **shall** (2294) occur $8 \pm 0.05 \mu s$ prior to the leading edge of the reply data block.

6. Mode S reply containing only the four preamble pulses.

2.4.1.3

Alternative Mode S Transponder Procedures

Except as specified below, for those tests requiring a Mode S transponder as shown in Figure 2-25, alternative procedures that simulate the functions of the transponder, including all ACAS X/Mode S interfaces, may be used if it is demonstrated that they provide an equivalent level of test coverage. The above statement is not true for the tests in §2.4.2.3, §2.4.2.4, §2.4.2.5 and 2.4.2.6. These subparagraphs contain end-to-end system tests that require use of actual Mode S transponders compatible with RTCA/DO-185A,B or later.

2.4.1.4

Bench Test Coverage of the TCAS Software

The detailed test procedures identified in §2.4.2, including both surveillance and CAS logic bench tests, are intended to serve as functional tests and therefore do not exercise every decision point within the computer algorithms. Manufacturers are advised to design software tests that demonstrate conformance to the appropriate RTCA/DO-178C level to which the ACAS X software is being certified.

2.4.2

Detailed Test Procedures

2.4.2.1

Surveillance

2.4.2.1.1

Transmitter Characteristics

The following procedures verify the proper operation of the ACAS X transmitter.

2.4.2.1.1.1

Transmit Frequency (§2.2.3.5)

Equipment Required:

Spectrum Analyzer (HP 141/8554B, or equivalent).
 Two each SPDT Switch (RLC Model SM-2-N, or equivalent).
 Reference Oscillator (1030 MHz ± 1 kHz, 0 dBm output).

Two each 10 dB Attenuator (Weinschel Model 39-10, or equivalent).
50 dB Attenuator (Weinschel Model 50-50, or equivalent).

Measurement Procedure

Connect equipment as shown in Figure 2-27. Set spectrum analyzer controls as follows:

Resolution Bandwidth	:	1 kHz
Scan Width	:	10 kHz/div
Reference Level	:	0 dBm
Display Mode	:	10 dB/div
Input Attenuation	:	10 dB

Set S1 to the top antenna port. Program ACAS X for Mode S test Mode 3 defined in §2.1.6, top antenna, 50 Hz rate.

Set S2 to reference oscillator. Tune spectrum analyzer center frequency to place calibration signal at center of display. Set S2 to 50 dB attenuator. Adjust reference level until peak of spectrum is within one division of top of display. Switch display mode to two dB/div. Observe frequency of spectrum peak.

Repeat for bottom antenna mode by setting S1 to the bottom antenna port and programming ACAS X for bottom antenna output.

Note: Verify spectrum analyzer stability by remeasuring reference oscillator frequency at the end of the tests.

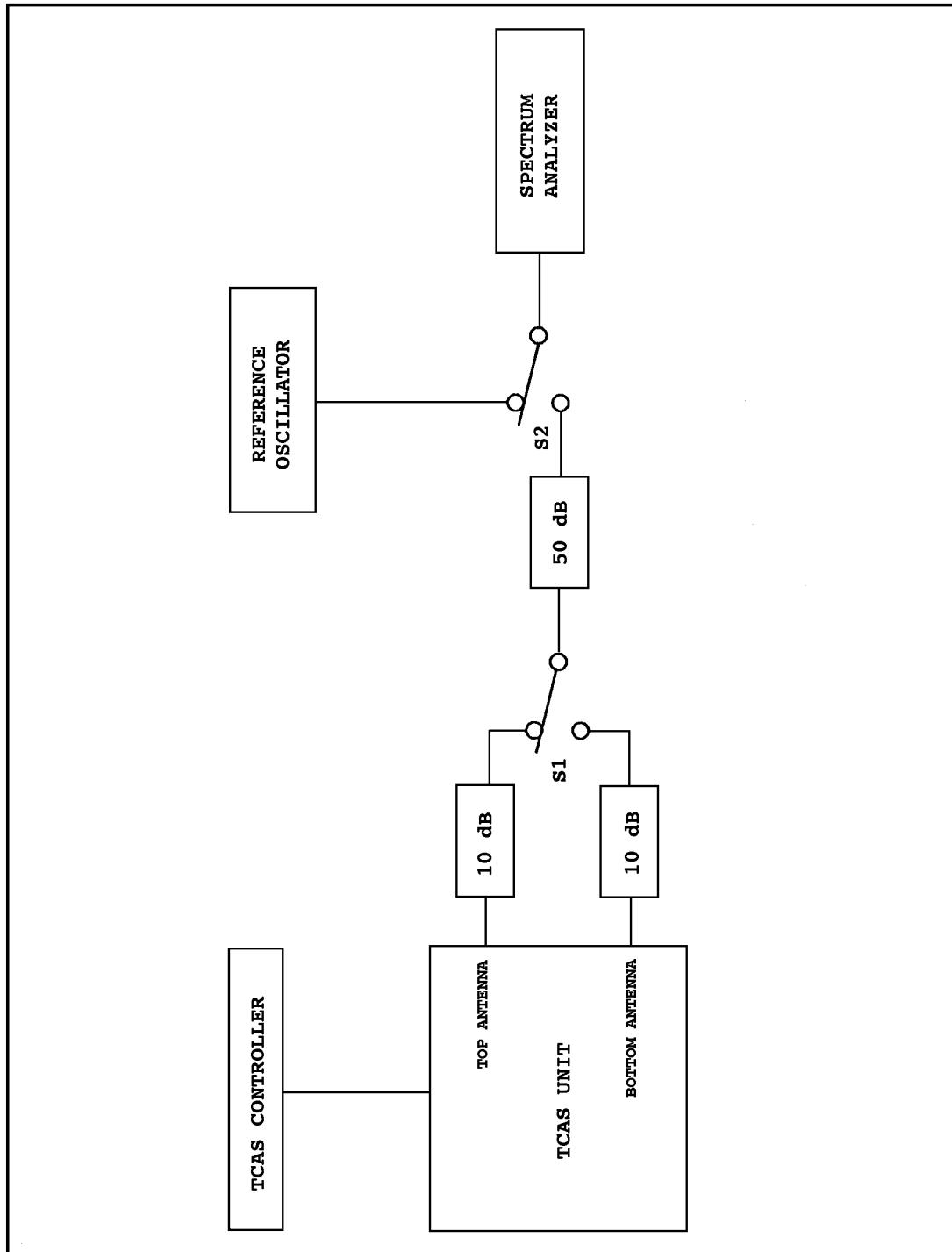


Figure 2-27: Frequency Test Setup

2.4.2.1.1.2 Radiated Output Power (§2.2.3.1)

Equipment Required

Detector Video Amplifier, DVA (Aertech Model 1204, or equivalent).

Power Supply for DVA.

Reference Signal Generator (HP612A, or equivalent).

Power Meter and Sensor (HP436A and HP8481A, or equivalent).

Oscilloscope (Tektronix 465, or equivalent).

Two each 10 dB Attenuator (Weinschel Model 39-10, or equivalent).

50 dB Attenuator (Weinschel Model 50-50, or equivalent).

Two each SPDT switch (RLC Model SM-2-N, or equivalent).

Measurement Procedure

Connect equipment as shown in Figure 2-28. Set S1 and S2 to top antenna output. Program ACAS X for Mode S test mode 1 (§2.1.6), top antenna.

Observe detected waveform on scope. Set S2 to reference signal generator. Adjust signal generator for same level on scope. Maximum and minimum peak output voltage **shall** (2295) correspond to a peak transmitter output power at the RF Reference Point as specified in §2.2.3.1.

Repeat test using Mode C test mode 1 (§2.1.6). Repeat test for bottom antenna output in Mode S and Mode C test modes.

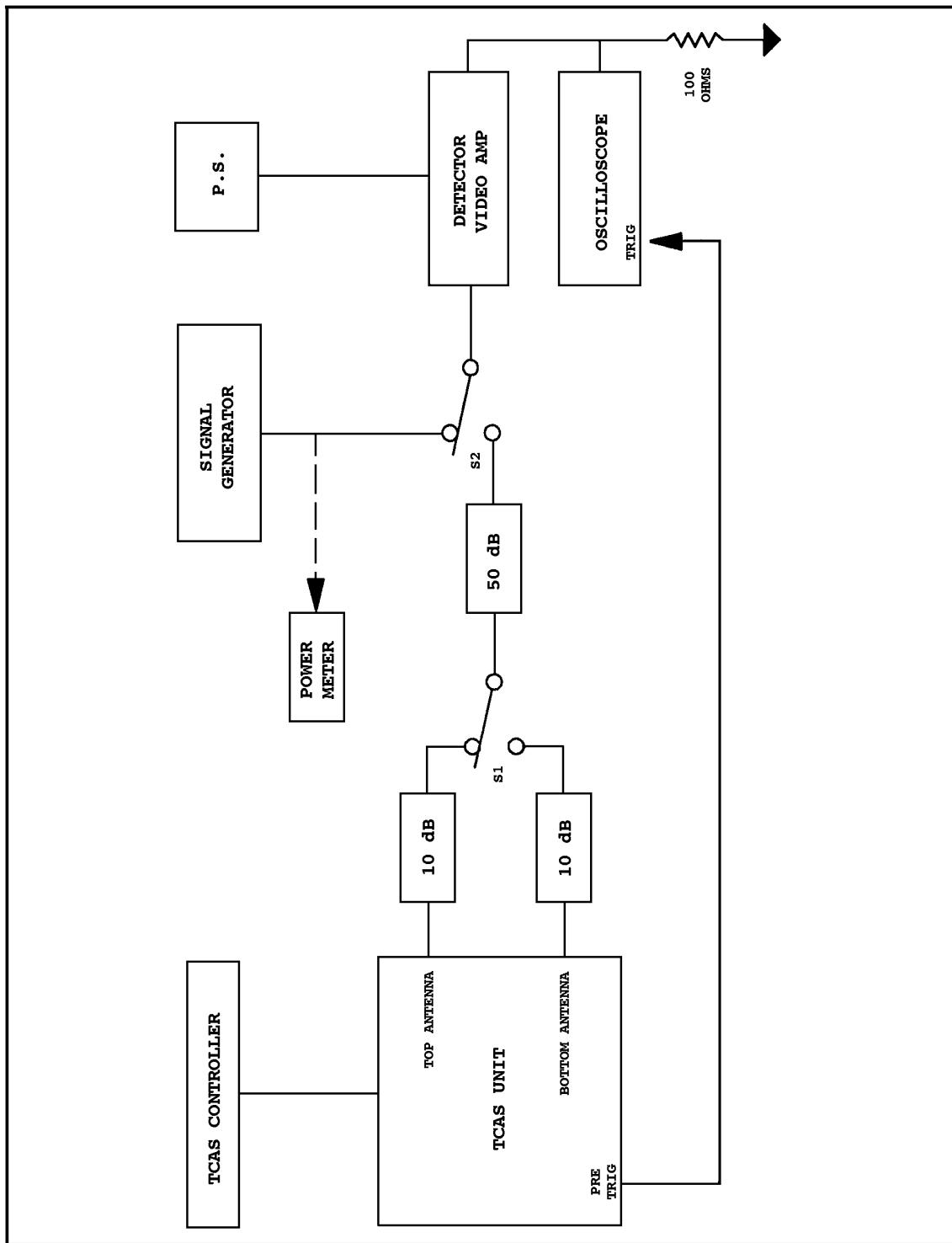


Figure 2-28: Peak Power Test Setup

2.4.2.1.1.3 Control of Synchronous Interference by Transmitter Power (§2.2.4.5.4.1)

2.4.2.1.1.3.1 Control of Synchronous Garble by Whisper-Shout (§2.2.4.5.4, §2.2.4.5.4.1)

This test is designed to verify that the interrogations in the high resolution whisper-shout sequence and the minimum basic whisper-shout sequence associated with each directional interrogation beam and omni beam have the proper relative interrogation pulse timing and the proper amplitudes relative to the highest level interrogation.

Equipment Required

- Detector Video Amplifier, DVA (Aertech Model 1204, or equivalent).
- Power Supply for DVA.
- Reference Signal Generator (HP612A, or equivalent).
- Power Meter and Sensor (HP436A and HP8481A, or equivalent).
- Oscilloscope (Tektronix 7603 Main frame, 7A13 differential amplifier, 7B50A time base, 7D11 Digital Delay, or equivalent).
- Variable Attenuator (Weinschel Model 905, or equivalent).
- 50 dB Attenuator (Weinschel Model 50-50, or equivalent).
- 2 each 10 dB Attenuator (Weinschel Model 39-10, or equivalent).
- 2 each SPDT switch (RLC Model SM-2-N, or equivalent).

Measurement Procedure

Step 1—Relative Power Level of Interrogation/Suppression Sequence

For this test, the ACAS X will be in whisper-shout test mode 1 (§2.1.6) which **shall** (2296) have the following characteristics:

Sequence Rate : 10 sequences/second

Connect equipment as shown in Figure 2-29.

Set oscilloscope as follows:

- Sensitivity : 0.5 V/div
- Sweep Rate : 10 msec/div
- Delay : zero
- Offset : -2 Volts

The measurement procedure **shall** (2297) identify the directional antenna beam and omni beam associated with each of the interrogations in the top and bottom antenna for the minimum basic sequence. If separate antenna beam outputs are not provided (i.e., beam switching occurs in the antenna subsystem) the interrogation beam association for the minimum basic sequence can be accomplished by simultaneously observing the beam switching control signal.

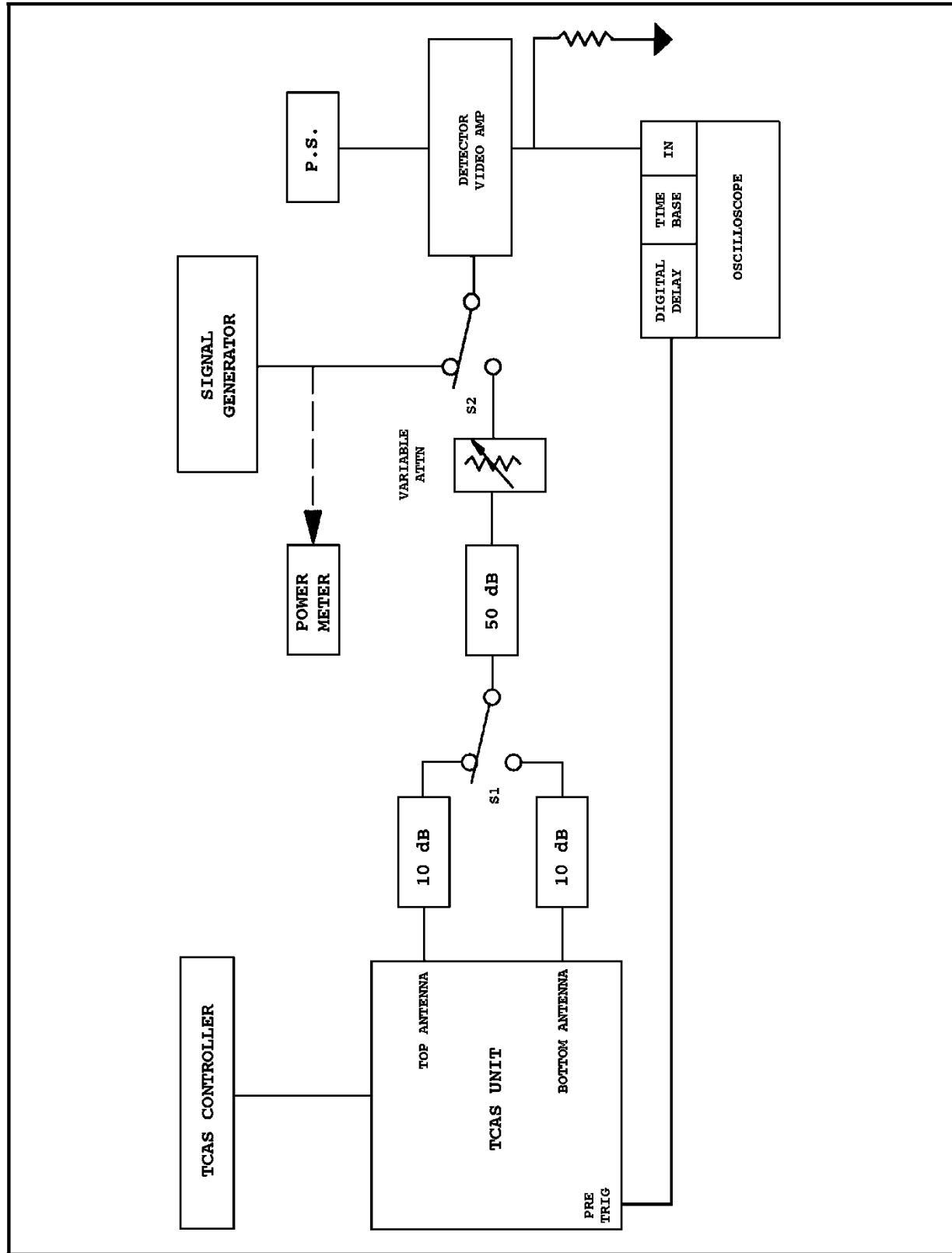


Figure 2-29: Whisper-Shout And Interrogation Rate Jitter Test Setup

Set S1 to top antenna and observe the total whisper-shout sequence with oscilloscope. Adjust variable attenuator to achieve four volts peak (Vp) amplitude for largest pulses of sequence. Adjust digital delay and increase sweep rate to display the largest five pulses of the sequence (two suppression pulses followed by the three interrogation pulses). Fine-tune variable attenuator for four volts peak amplitude for interrogation pulses. Note peak voltage (Vp) for suppression pulses. Set S2 to the reference signal generator and verify the proper relative power level between the suppression pulses and the following interrogation pulses.

Success: The relative timing between interrogations and between interrogation and suppression pulses and the amplitude of each of the interrogation and suppression waveforms in each of the directional and omnidirectional beams relative to the largest interrogation amplitude **shall** (2298) be as specified in §2.2.4.5.4.

Repeat Step 1 using whisper-shout test mode 2.

Repeat Step 1 with S1 set to bottom antenna.

Step 2 — Relative Timing Between Suppression and Interrogation

Use set-up of Step 1 and observe relative timing between suppression and interrogation for each frame of the whisper-shout sequence. Verify proper timing between suppression and interrogation pulses for both the top and bottom antenna ports.

2.4.2.1.1.3.2 Determination of Whisper-Shout Sequence (§2.2.4.5.4, §2.2.4.5.4.1.3, §2.2.4.5.4.1.5)

This test verifies that ACAS X is able to select the proper interrogation sequence for each interrogation beam based on measurements of potential synchronous garble.

Inputs

ACAS X Aircraft

Sensitivity Level Selection	=	Automatic
Altitude	=	8000 ft
Altitude Rate	=	0 FPM

Intruder Aircraft

The following apply to all intruders:

Relative Speed	=	0 kt
Altitude Rate	=	0 FPM
Equipage	=	Mode C
Reply Power	=	-65 dBm

All intruders **shall** (2299) be separated by at least 300 feet from each other in altitude and **shall** (2300) be within the surveillance altitude volume of ±10,000 feet relative to ownship.

Intruder Aircraft 1-4

Bearing	=	Forward Quadrant
Range	=	
Int 1	=	1 NM at T=0 seconds
	=	2.05 NM at T=120 seconds

Int 2	= 3 NM at T=0 seconds
Int 3	= 5 NM at T=0 seconds
Int 4	= 10 NM at T=0 seconds
Reply Code Bit Confidence	
Int 1 & 2	= No low conf bits from T=0 to T=120 seconds
	= At least one low conf bit from T=120 to T=180 seconds
Int 3 & 4	= No low conf bits

Intruder Aircraft 5

Bearing	= Left Quadrant
Range	= Not present T=0 to 120 seconds
	= 5 NM at T=120 seconds
Reply Code Bit Conf	= No low conf bits

Intruder Aircraft 6 & 7

Bearing	= Right Quadrant
Range	
Int 6	= 4 NM at T=0 seconds
Int 7	= 5 NM at T=0 seconds
	= 6 NM at T=120 seconds
Reply Code Bit Confidence	= No low conf bits

Intruder Aircraft 8 & 9

Bearing	= Rear Quadrant
Range	
Int 8	= 1 NM at T=0 seconds
	= Not present T=120 to 180 seconds
Int 9	= 3 NM at T=0 seconds
	= Not present T=120 to 180 seconds
Reply Code Bit Confidence	= No low conf bits

Conditions

ACAS X is initialized and operating at T=0 seconds. Each intruder, when present, **shall** (2301) reply once during each scan and, in the case of a whisper-shout sequence, to different whisper-shout interrogations.

Scenario for Intruder 1-4

Intruders 1 through 4 are within the forward beam of ACAS X. Initially, all four intruders are separated so as not to be within garble range of each other and do not reply with low confidence code bits. At T=120 seconds, Intruders 1 and 2 are within garble range and contain at least one low confidence code bit in their replies.

Success: During the start-up interval from T=0 to T=60 seconds, the forward beam is constrained to the high resolution whisper-shout sequence regardless of the garble situation in that beam. At T=60 seconds, the forward beam is switched on the following scan to the Minimum Basic whisper-shout sequence. At T=120 seconds, the low confidence bits in the

Intruder 1 and 2 reply cause the forward beam to switch on the following scan to the high resolution whisper-shout sequence.

Scenario for Intruder 5

Intruder 5 is not present from T=0 to T=120 seconds. At T=120 seconds, Intruder 5 is within the left beam of ACAS X and at 5 NM range.

Success: During the start up interval from T=0 to T=60 seconds, the left beam is constrained to the high resolution whisper-shout sequence regardless of the garble situation in that beam. At T=61 seconds, the lack of a Mode C intruder track within the left beam causes ACAS X to transmit a single interrogation in that beam. The presence of Intruder 5 at T=120 seconds causes ACAS X to switch to the Minimum Basic whisper-shout sequence in the left beam no later than T=124 seconds.

Scenario for Intruders 6 & 7

Intruders 6 and 7 are within the right beam of ACAS X. At T=0 seconds, Intruders 6 and 7 are within garble range of each other but do not reply with low confidence code bits. At T=120 seconds, Intruders 6 and 7 are no longer within garble range of each other.

Success: During the startup interval from T=0 to T=60 seconds, the right beam is constrained to the high resolution whisper-shout sequence. At T=60 seconds, the right beam continues to transmit the high resolution whisper-shout sequence because Intruders 6 and 7 are within garble range of each other. At T=127 seconds, the right beam switches to the Minimum Basic whisper-shout sequence because Intruders 6 and 7 are no longer within garble range of each other.

Scenario for Intruder 8 & 9

Intruders 8 and 9 are within the rear beam of ACAS X. At T=0 seconds, Intruder 8 is at 1 NM range and Intruder 9 is at 3 NM range. At T=120 seconds, Intruders 8 and 9 are no longer present within the ACAS X surveillance range.

Success: During the startup interval from T=0 to T=60 seconds, the rear beam is constrained to the high resolution sequence. At T=60 seconds, the rear beam continues to transmit the high resolution whisper-shout sequence because the Mode C density is above a threshold value that indicates potential synchronous garble. At T=120 seconds, the lack of a Mode C intruder causes the rear beam to switch to a single interrogation at T=127 seconds following the drop of the Intruder 8 and 9 tracks.

The scenario is terminated at T=180 seconds.

2.4.2.1.1.4

ACAS X Transmit Pulse Characteristics (§2.2.3.7, §2.2.4.5.4.2.1)

ACAS X to Mode S Transmissions (§2.2.3.7.2)

Equipment Required

- Detector Video Amplifier, DVA (Aertech Model 1204, or equivalent).
- Power Supply for DVA.
- Reference Signal Generator (HP612A, or equivalent).
- Power Meter and Sensor (HP436A and HP8481A, or equivalent).
- Oscilloscope (Tektronix 7603 Main frame, 7A13 differential amplifier, 7A15A amplifier, 7B50A time base, or equivalent).
- Variable Attenuator (Weinschel Model 50-50, or equivalent).
- Two each 10 dB Attenuator (Weinschel Model 39-10, or equivalent).
- SPDT switch (RLC Model SM-2-N, or equivalent).

- Power Divider (Narda Model 4321-2, or equivalent).
- Phase detector (Watkins-Johnson Model WJ-M1A-11, or equivalent).
- Phase Shifter (Lorch Devices Model PQ-28, or equivalent).
- 0.5 μ s delay line.

Measurement Procedure

Step 1 — Pulse Amplitude Variation and Droop

Connect equipment as shown in Figure 2-30. Set S1 to top antenna. Program ACAS X for Mode S test mode 1 (§2.1.6) and top antenna. Adjust variable attenuator for four Vp-p DVA output. Use oscilloscope, DVA and reference signal generator to measure pulse amplitude variation and droop. Repeat with Mode S test mode 2 (§2.1.6). Repeat above for bottom antenna output.

Step 2 — Pulse Shape

Use set-up and procedure of Step 1 to measure pulse shape characteristics.

Step 3 — Pulse Spacing Tolerance

Use set-up and procedure of Step 1 to measure pulse spacing tolerance.

Step 4 — DPSK Phase Reversal Timing and Duration

Connect equipment as shown in Figure 2-30. Use same test modes as Step 1. Adjust variable attenuator for four Vp-p DVA output. Observe the DVA output on the oscilloscope. Measure the interval between the 80 percent points of the amplitude transient associated with each phase reversal. The value **shall** (2302) be less than 0.08 μ s. Measure the interval between an 80 percent point on each amplitude transient and the corresponding point on an amplitude transient associated with the next phase reversal. The value **shall** (2303) be 0.25 ± 0.02 μ s.

Measure the interval between an 80 percent point on the amplitude transient associated with the sync phase reversal and the corresponding point on the amplitude transient associated with the next phase reversal. The value **shall** (2304) be 0.5 ± 0.02 μ s.

Step 5 — DPSK Phase Reversal Tolerance

Connect the equipment as shown in Figure 2-31. Program ACAS X for Mode S test mode 3 (§2.1.6) and set S1 to top antenna. Adjust the variable phase shifter for zero volts output during the P6 pulse as observed on the oscilloscope. Adjust the phase shifter ± 5 degrees about its zero-volt-output position and note the corresponding levels on the oscilloscope.

Program ACAS X for Mode S test mode 1 (§2.1.6) and measure the amplitude of each data chip at a point midway between the amplitude transients associated with each phase reversal. The amplitude of each data chip **shall** (2305) be equal to or less than the ± 5 degrees amplitudes noted for the test mode 3 input above.

ACAS X to Mode C Transmissions (§2.2.3.7.1)

Equipment Required

- Detector Video Amplifier, DVA (Aertech Model 1204, or equivalent).
- Power Supply for DVA.
- Reference Signal Generator (HP612A, or equivalent).
- Power Meter and Sensor (HP436A and HP8481A, or equivalent).
- Oscilloscope (Tektronix 7603 Main frame, 7A13 differential amplifier, 7B50A time base, or equivalent).

Variable Attenuator (Weinschel Model 50-50, or equivalent).

Two each 10 dB Attenuator (Weinschel Model 39-10, or equivalent).

SPDT switch (RLC Model SM-2-N, or equivalent).

Measurement Procedure

Step 1 — Pulse Amplitude Variation

Connect equipment as shown in Figure 2-30. Set S1 to top antenna. Program ACAS X for Mode C test mode 1 (§2.1.6). Measure pulse amplitude variation. Repeat above procedure for bottom antenna output.

Step 2 — Pulse Shape

Measure pulse shape characteristics of top antenna output. Repeat for bottom antenna output.

Step 3 — Pulse Spacing Tolerance

Measure pulse spacing of top antenna output. Repeat for bottom antenna output.

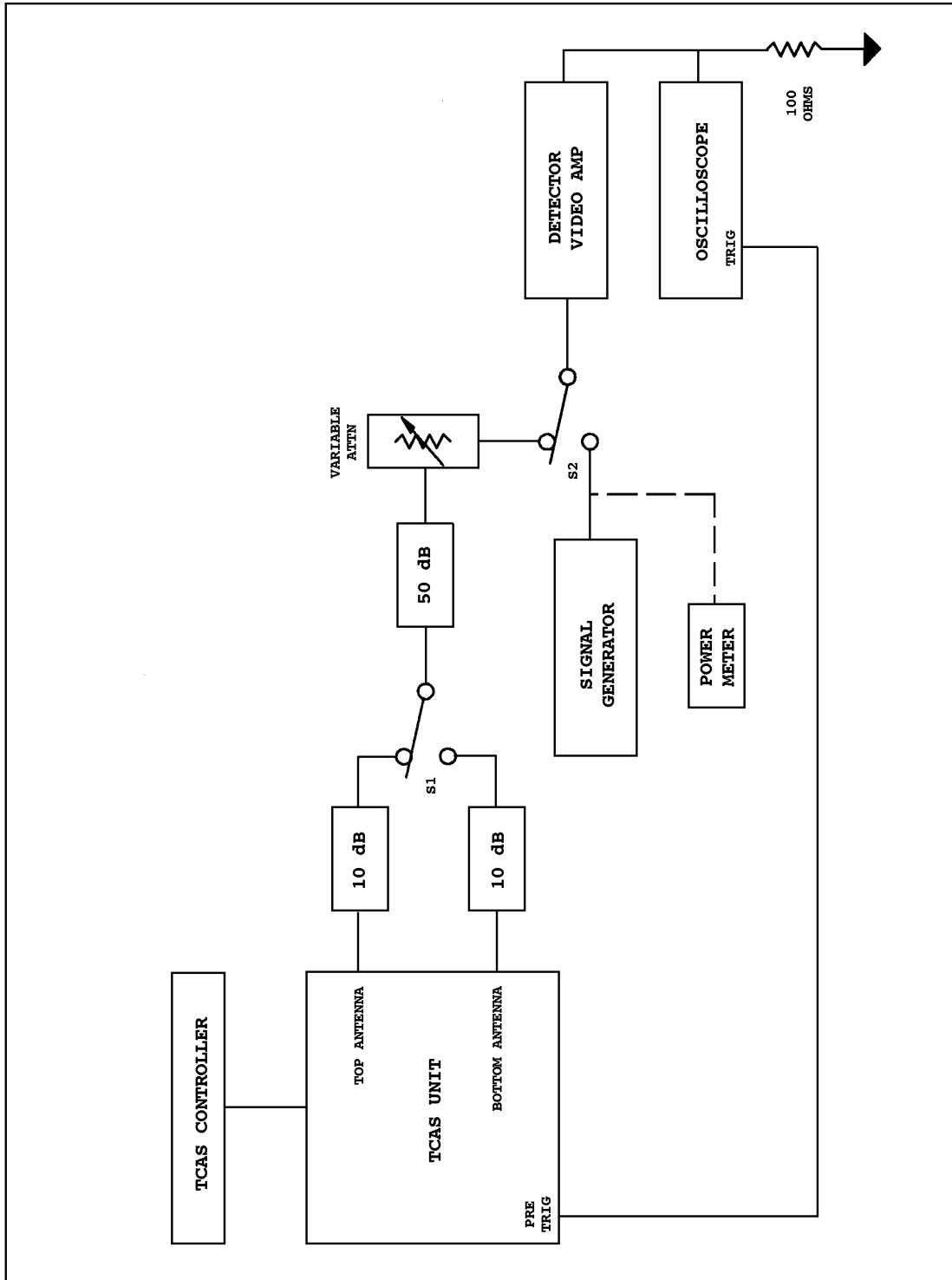


Figure 2-30: Mode S Transmission Test Setup

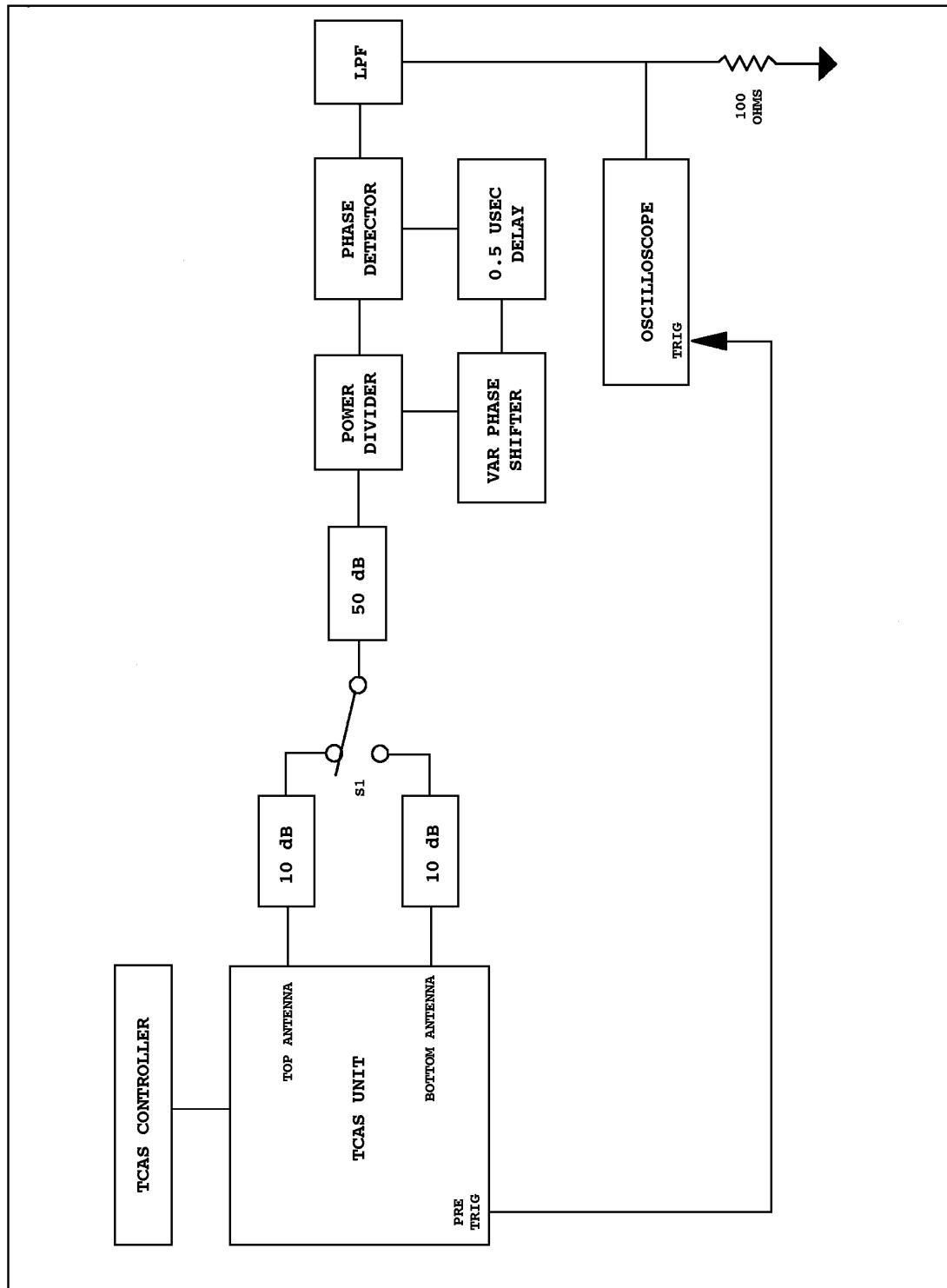


Figure 2-31: DPSK Phase Reversal Tolerance Test Setup

2.4.2.1.1.5 Interrogation Spectrum (§2.2.3.3)

Equipment Required

- Spectrum Analyzer (HP 141/8554B, or equivalent).
- 50 dB Attenuator (Weinschel Model 50-50, or equivalent).
- SPDT switch (RLC Model SM-2-N, or equivalent).
- Two each 10 dB Attenuator (Weinschel Model 39-10, or equivalent).

Measurement Procedure

Connect equipment as shown in Figure 2-32. Set spectrum analyzer as follows:

Scan Width	:	20 MHz/div
Resolution Bandwidth	:	100 kHz
Center Frequency	:	1030 MHz
Reference Level	:	-20 dBm

Set S1 to top antenna. Program ACAS X for Mode S test mode 1 (§2.1.6) and top antenna. Verify proper signal spectrum bounds.

Note: Adjust spectrum analyzer as necessary to obtain sufficient measurement resolution.

Repeat with ACAS X in Mode S test mode 2 and Mode C test mode 1 (§2.1.6).

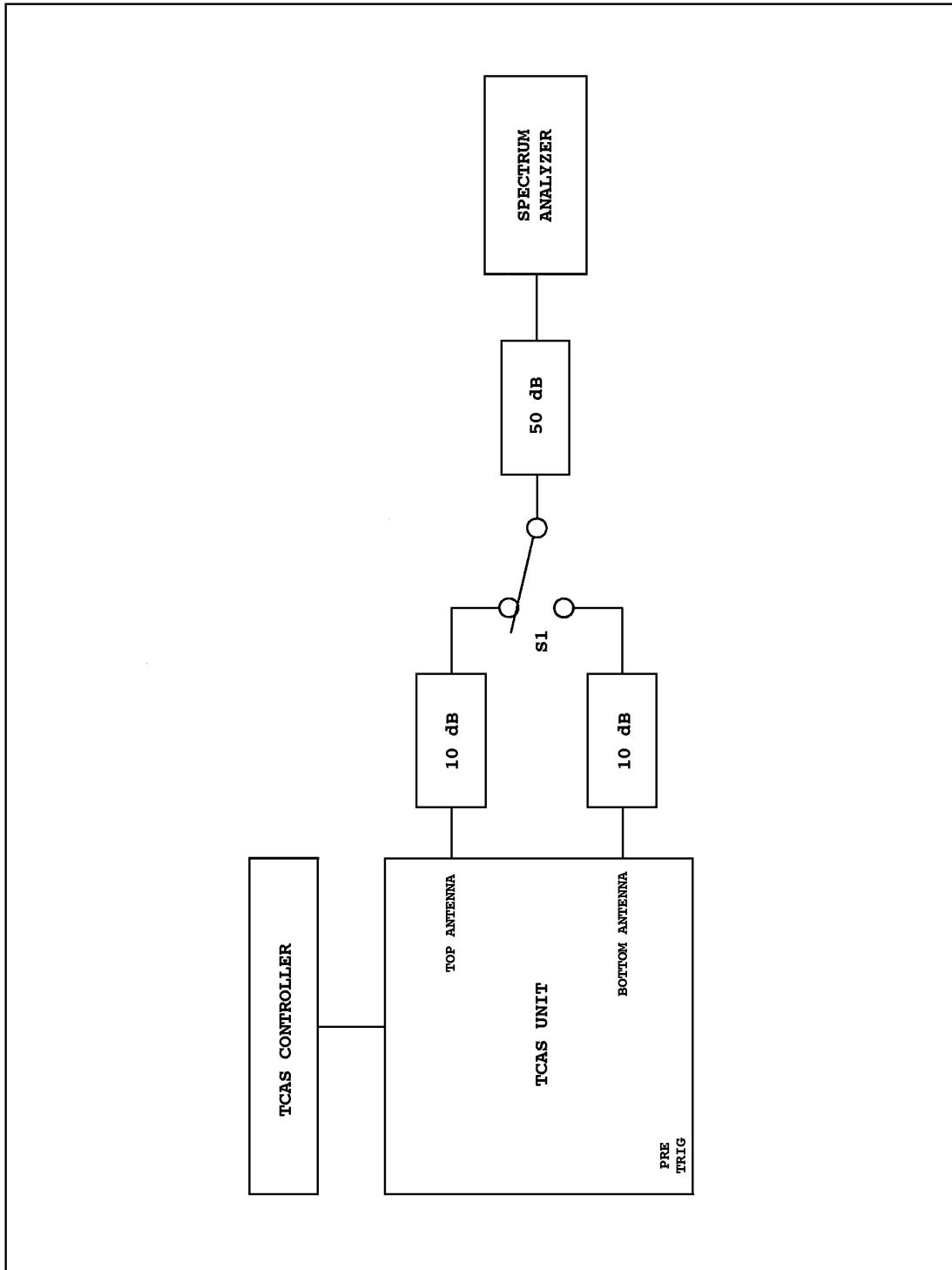


Figure 2-32: Mode S RF Spectrum Test Setup

2.4.2.1.1.6 Unwanted Output Power (§2.2.3.2)

Equipment Required

Detector Video Amplifier (Aertech Model 1204, or equivalent)
Power Supply for DVA.
Reference Signal Generator (HP612A, or equivalent).
Power Meter and Sensor (HP436A and HP8481A, or equivalent).
Oscilloscope (Tektronix 465, or equivalent).
Limiter/Preamplifier (Alpha MT 3280 A2 Limiter and Watkins-Johnson 6203-433 Amplifier, or equivalent).
Two each 10 dB Attenuator (Weinschel Model 39-10, or equivalent).
SPDT switch (RLC Model SM-2-N, or equivalent).

Measurement Procedure

Connect equipment as shown in Figure 2-33. Program ACAS X for the No-Interrogation Test Mode 1 (§2.1.6). Set S1 to top antenna position. Measure Video Detector Amplifier output with oscilloscope. Check for spurious pulse output signals by adjusting trigger level in internal sync mode. All detector output signals (both CW and pulse) **shall** (2306) correspond to less than -70 dBm at the ACAS X RF Reference Point.

Repeat above procedure for bottom antenna output.

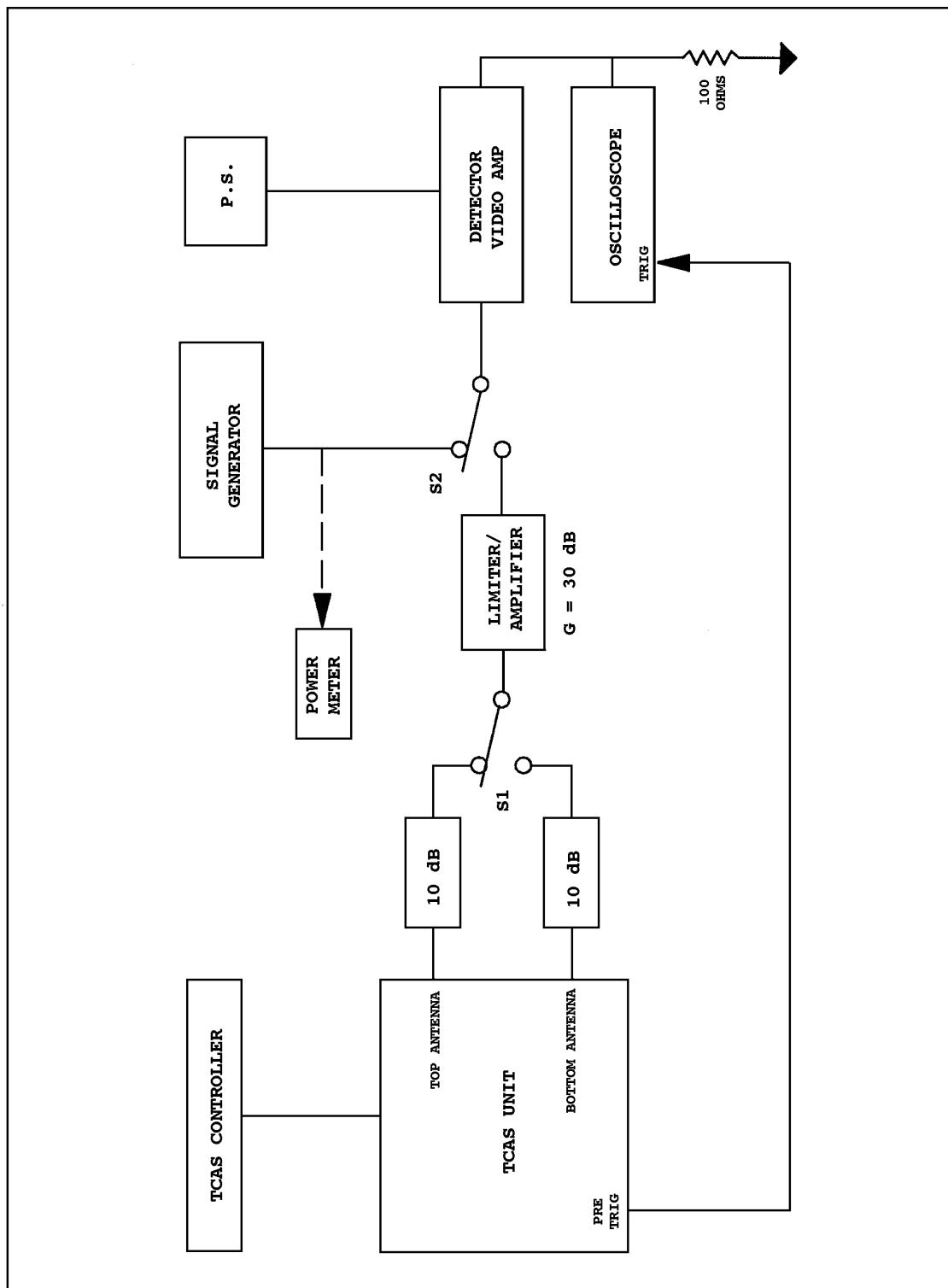


Figure 2-33: Unwanted Output Power Test Setup

2.4.2.1.1.7 Aircraft Suppression Bus (§2.2.3.11)

2.4.2.1.1.7.1 Suppression Pulse Supplied by ACAS X

Equipment Required

Oscilloscope (Tektronix 465, or equivalent).

Measurement Procedure

Connect oscilloscope to ACAS X unit suppression pulse output. Connect oscilloscope to sync on ACAS X unit pretrigger output. Measure and verify proper suppression pulse width associated with both top and bottom antenna interrogations.

2.4.2.1.1.7.2 Suppression Pulse Supplied by Other Avionics

Verify that ACAS X is able to receive suppression pulses generated by other avionics equipment on board ownship and employs these pulses to prevent unwanted ACAS X transmissions in response to transmissions by the suppressing equipment, specifically the Mode S transponder. Using Test §2.4.2.1.2.1.1, Ability to Operate Over the Frequency Band 1087 to 1093 MHz, verify that ACAS X regains normal sensitivity, within three dB, not later than 15 μ s after the end of the applied suppression pulse.

2.4.2.1.1.8 Interrogation Repetition Interval (§2.2.4.1) and Jitter (§2.2.3.4)

Note: This test uses equipment and set-up described in §2.4.2.1.1.3, whisper-shout operation, except for the use of a storage oscilloscope. The TCAS Broadcast Interrogation measurement for directional antenna system is covered in §2.4.2.1.10.2.

Measurement Procedure

Set S1 to top antenna. Program ACAS X for normal Mode C interrogations. Set digital delay and sweep rate to observe rate and jitter of interrogation pulse sequence. Rate and peak-to-peak jitter **shall** (2307) be as specified in §2.2.4.1 and §2.2.3.4. Repeat measurement using normal TCAS broadcast interrogations.

2.4.2.1.2 Receiver Characteristics (§2.2.4.4.1)

Many of the ACAS X receiver characteristics are specified in probabilistic terms (expected probability of success) that can be verified using the following procedures. The following definitions and conventions apply to all of the procedures in §2.4.2.1.2 for which an expected result (success) is given in terms of a required probability of success:

- a. Each test scenario **shall** (2308) be executed a sufficient number of times to establish a test result with 90 percent confidence:
 1. Test Passed: The equipment under test passes if it performs with at least the probability of success specified under the conditions of the given scenario.
 2. Test Not Passed: The equipment under test fails if it does not perform with the probability of success specified under the conditions of the given scenario.
- b. The minimum number of times a test must be repeated (minimum number of trials, T) in order to establish, at a confidence level of 90 percent, that the test passed is a function of the number of times a test results in a failure. For the situation in which the number of failures is one or greater, the minimum number of trials (T) **shall** (2309) be determined using:

$$T = \frac{F}{(1-P)C}$$

Where T = Minimum number of trials
 P = Required probability of success
 C = Confidence factor from Table 2-72
 F = Number of trials that did not result in success.

Table 2-73 lists the minimum number of trials (T) for the various probability of success values (P) specified in §2.4.2.1.2 for each value of F equal to or greater than one. The table also includes the minimum number of trials required given that the number of failures is zero ($F=0$).

In test scenarios where success is defined but no expected probability of success is stated, only a single trial is required. The test is passed if success occurs and failed otherwise.

Notes:

1. *Tests of this type correspond to non-statistical requirements in §2.2.*
2. *Many of the statistical tests are measuring reply decoding or squitter decoding probability by measuring the probability that a track is established. A manufacturer may design a test that is able to measure reply detection probability directly while still meeting the required 90% statistical confidence level discussed above.*

Generalized Scenario Description

The test scenarios used to verify proper operation of the receiver contain a single simulated Mode C or Mode S target and **shall** (2310) be repeated for each ACAS X RF port. The following set of inputs is described for use throughout §2.4.2.1.2; additional inputs and conditions are specified as required.

Inputs

Intruder Aircraft

Altitude	=	8300 ft
Altitude Rate	=	0 FPM
Range	=	two NM at T=0 seconds
Relative Speed	=	-360 kt (-0.1 NM/s)

ACAS X Aircraft

Altitude	=	8000 ft
Altitude Rate	=	0 FPM
Sensitivity Level Selection	=	Automatic

Mode S Squitter Format

All Mode S squitter transmissions referred to in the subparagraphs of §2.4.2.1.2 **shall** (2311) have the following standard data field values:

DF	=	11
CA	=	0
AA	=	Any valid ICAO 24-bit Aircraft Address

Table 2-72: Confidence Factor Required For A 90% Confidence Level

No. of Failures, F	Confidence Factor, C
1	0.257
2	0.376
3	0.449
4	0.500
5	0.534
6	0.570
7	0.595
8	0.616
9	0.634
10	0.649
11	0.663
12	0.675
13	0.686
14	0.696
15	0.705
16	0.713
17	0.720
18	0.729
19	0.734
20	0.740
30	0.780

Table 2-73: Minimum Number Of Trials Vs. Number Of Failures For The Required Probability Of Success Values Specified In §2.4.2.1.2

F	Required Probability of Success (p)						
	0.97	0.96	0.90	0.72	0.66	0.34	0.27
1	129	97	39	14	11	6	5
2	177	133	53	19	16	8	7
3	223	167	67	24	20	10	9
4	267	200	80	29	24	12	11
5	312	234	94	32	28	14	13
6	351	263	105	38	31	16	14
7	392	294	118	42	35	18	16
8	433	325	130	46	38	20	18
9	473	355	142	51	42	22	19
10	514	385	154	55	45	23	21
11	553	415	166	59	49	25	23
12	593	444	178	64	52	27	24
13	632	474	190	68	56	29	26
14	670	503	201	72	59	30	28
15	709	532	213	76	63	32	29
16	748	561	224	80	66	34	31
17	787	590	236	84	69	36	32
18	823	617	247	88	73	37	34
19	863	647	259	92	76	39	35
20	901	676	270	97	79	41	37
30	1277	958	389	137	113	58	52

2.4.2.1.2.1 In-Band Acceptance

2.4.2.1.2.1.1 Ability to Operate Over the Frequency Band 1087 to 1093 MHz for ATCRBS and 1089 to 1091 MHz for Mode S Signals (§2.2.4.4.1.1)

These tests verify the ability of the ACAS X Receiver to operate over the frequency band 1087 to 1093 MHz for ATCRBS signals and also over the frequency band of 1089 to 1091 MHz for Mode S signals.

If the antenna gain is not as specified in §2.2.4.7.2, each of the reply power levels specified in the tests **shall** (2312) be adjusted to compensate for the difference between the specified antenna gain and the actual antenna gain. For example, if the antenna gain is three dB below the specified value, each of the reply power levels specified in the test **shall** (2313) be lowered by three dB.

Mode C Tests

Inputs

As described in §2.4.2.1.2 and supplemented as follows:

Intruder Aircraft

Equipage	=	Mode C
Reply Frequency		
Scenario A	=	1089 MHz
Scenario B	=	1091 MHz
Reply Power		
Scenario A	=	-72 dBm
Scenario B	=	-72 dBm

Conditions

ACAS X initiated and operating at T=0 seconds. The intruder **shall** (2314) reply exactly once during each of the first four whisper-shout sequences and then remain silent.

Scenario Description

A single Mode C intruder is in the active surveillance region of ACAS X.

Success: The intruder track is displayed within 1.5 second following receipt of the reply to the last of the four Mode C-Only All-Call interrogation sequences. (All scenarios to be terminated at T=10 seconds).

Required Probability of Success

$$(0.9)^4 = 0.66$$

Mode S Tests

Inputs

As described in §2.4.2.1.2 and supplemented as follows:

Intruder Aircraft

Equipage	=	Mode S
Squitter Frequency		
Scenario A	=	1087 MHz
Scenario B	=	1093 MHz
Squitter Power		
Scenario A	=	-72 dBm
Scenario B	=	-72 dBm

Conditions

ACAS X initiated and operating at T=0 seconds. The intruder **shall** (2315) transmit a single squitter during the squitter listening period following the completion of each of the first three whisper-shout sequences and then remain silent.

Scenario Description

A Mode S intruder is at co-altitude with ACAS X and transmits three squitters at a time when the ACAS X Receiver is in a squitter listening mode.

Success: ACAS X transmits an acquisition interrogation addressed to the ICAO 24-bit Aircraft Address contained in the AP field of the squitters within one second following the last squitter (all scenarios terminated at T=10 seconds).

Required Probability of Success

$$(0.9)^3 = 0.73$$

2.4.2.1.2.1.2 Sensitivity and Dynamic Range at 1090 MHz (§2.2.4.4.1.1)

This series of tests will verify the receiver sensitivity and dynamic range requirements for replies at 1090 MHz.

If the antenna gain is not as specified in §2.2.4.7.2, each of the reply power levels specified in the tests **shall** (2316) be adjusted to compensate for the difference between the specified antenna gain and the actual value. For example, if the antenna gain is three dB below the specified value, each of the reply power levels specified in the test **shall** (2317) be lowered by three dB.

Mode C Tests

Inputs

As described in §2.4.2.1.2 and supplemented as follows:

Intruder Aircraft

Equipage	=	Mode C
Reply Frequency	=	1090 MHz
Reply Power		
Scenario A	=	-72 dBm
Scenario B	=	-69 dBm
Scenario C	=	-56 dBm
Scenario D	=	-44 dBm
Scenario E	=	-32 dBm
Scenario F	=	-21 dBm
Scenario G	=	-78 dBm (first of the four replies)
	=	-50 dBm (subsequent replies)
Scenario H	=	-76 dBm

Conditions

ACAS X must be initialized and operating at T=0 seconds. The intruder **shall** (2318) reply exactly once during each of the first four whisper-shout sequences and then remain silent.

Scenario Description

A single Mode C intruder is in the active surveillance region of ACAS X.

Success (Scenarios A to F): An intruder track is established following receipt of the reply to the last of the four Mode C-Only All-Call interrogation sequences. Success (Scenario G and H): An intruder track is not established. (Scenarios to be terminated at T=10 seconds).

Required Probability of Success

$$\begin{aligned}
 \text{Scenario A} &= (0.9)^4 & = & 0.66 \\
 \text{Scenario B-F} &= (0.99)^4 & = & 0.96 \\
 \text{Scenario G} &= 1-(0.1) \times (0.99)^3 & = & 0.90 \\
 \text{Scenario H} &= 1-(0.9)^4 & = & 0.34
 \end{aligned}$$

Mode S Tests

Inputs

As described in §2.4.2.1.2 and supplemented as follows:

Intruder Aircraft 1

$$\begin{aligned}
 \text{Equipage} &= \text{Mode S} \\
 \text{Reply Frequency} &= 1090 \text{ MHz} \\
 \text{Squitter Power} & \\
 \text{Scenario A} &= -72 \text{ dBm} \\
 \text{Scenario B} &= -69 \text{ dBm} \\
 \text{Scenario C} &= -56 \text{ dBm} \\
 \text{Scenario D} &= -44 \text{ dBm} \\
 \text{Scenario E} &= -32 \text{ dBm} \\
 \text{Scenario F} &= -21 \text{ dBm} \\
 \text{Scenario G} &= -78 \text{ dBm (first of the three squitters)} \\
 &= -50 \text{ dBm (subsequent squitters)} \\
 \text{Scenario H} &= -76 \text{ dBm}
 \end{aligned}$$

Conditions

ACAS X initialized and operating at T=0 seconds. The intruder **shall** (2319) transmit a single squitter during the squitter listening period following the completion of each of the first three whisper-shout sequences and then remain silent.

Scenario Description

A Mode S intruder is co-altitude with ACAS X and transmits three squitters at times when the ACAS X receiver should be in squitter listening mode.

Success (Scenarios A-F): ACAS X transmits an acquisition interrogation addressed to the ICAO 24-bit Aircraft Address contained in the AP field of the squitters within one second of the last squitter.

Success (Scenario G and H): ACAS X does not transmit an interrogation. (Scenarios to be terminated at T=10 seconds).

Required Probability of Success

$$\begin{aligned}
 \text{Scenario A} &= (0.9)^3 & = & 0.73 \\
 \text{Scenario B-F} &= (0.99)^3 & = & 0.97 \\
 \text{Scenario G} &= 1-(0.1)(0.99)^2 & = & 0.90 \\
 \text{Scenario H} &= 1-(0.9)^3 & = & 0.27
 \end{aligned}$$

2.4.2.1.2.2 Out-of-Band Rejection (§2.2.4.4.1.2)

This series of tests verifies the proper out-of-band-rejection characteristic of the ACAS X receiver.

Mode C Tests

Inputs

As described in §2.4.2.1.2 and supplemented as follows:

Intruder Aircraft

Equipage	=	Mode C
Reply Frequency	=	
Scenario A	=	1065 MHz
Scenario B	=	1115 MHz
Scenario C	=	1075 MHz
Scenario D	=	1105 MHz
Scenario E	=	1080 MHz
Scenario F	=	1100 MHz
Scenario G	=	1084.5 MHz
Scenario H	=	1095.5 MHz
Reply Power	=	
Scenario A	=	(MTL* + 60 dB)
Scenario B	=	(MTL* + 60 dB)
Scenario C	=	(MTL* + 40 dB)
Scenario D	=	(MTL* + 40 dB)
Scenario E	=	(MTL* + 20 dB)
Scenario F	=	(MTL* + 20 dB)
Scenario G	=	(MTL* + 3 dB)
Scenario H	=	(MTL* + 3 dB)

* Measured MTL value at 1090 MHz.

Conditions

ACAS X initialized and operating at T=0 seconds. The intruder **shall** (2320) reply exactly once during each of the first four whisper-shout sequences and then remain silent.

Scenario Description

A single Mode C intruder is in the active surveillance region of ACAS X. The probability that an intruder track is initiated depends on receiver sensitivity at the specified frequencies.

Success: An intruder track is not established. (All scenarios to be terminated at T=10 seconds).

Required Probability of Success

$$1-(0.9)^4 = 0.34$$

Mode S Tests

Inputs

As described in §2.4.2.1.2 and supplemented as follows:

Intruder Aircraft

Equipage	=	Mode S
Squitter Frequency		
Scenario A	=	1065 MHz
Scenario B	=	1115 MHz
Scenario C	=	1075 MHz
Scenario D	=	1105 MHz
Scenario E	=	1080 MHz
Scenario F	=	1100 MHz
Scenario G	=	1084.5 MHz
Scenario H	=	1095.5 MHz
Squitter Power		
Scenario A	=	(MTL* + 60 dB)
Scenario B	=	(MTL* + 60 dB)
Scenario C	=	(MTL* + 40 dB)
Scenario D	=	(MTL* + 40 dB)
Scenario E	=	(MTL* + 20 dB)
Scenario F	=	(MTL* + 20 dB)
Scenario G	=	(MTL* + 3 dB)
Scenario H	=	(MTL* + 3 dB)

* Measured MTL value at 1090 MHz.

Conditions

ACAS X initialized and operating at T=0 seconds. The intruder **shall** (2321) transmit a single squitter during the squitter listening period following the completion of each of the first three whisper-shout sequences and then remain silent.

Scenario Description

A Mode S intruder is co-altitude with ACAS X and transmits three squitters at a time when the ACAS X receiver should be in squitter listening mode. The probability that ACAS X transmits an acquisition interrogation addressed to the ICAO 24-bit Aircraft Address contained in the AP field of the squitter depends on receiver sensitivity at the specified frequencies.

Success: ACAS X transmits no Mode S interrogation. (All scenarios terminated at T=10 seconds).

Required Probability of Success

$$1-(0.9)^3 = 0.27$$

2.4.2.1.3 Reply Link Interference (§2.2.4.5.1.2)

The following procedures verify the existence of an appropriate Mode C and Mode S dynamic minimum threshold (DMTL) function, which is one means of rejecting low level multipath signals. If another method is used to reject low level multipath signals, the test will need to be appropriately revised.

Generalized Scenario Description

The test scenarios used to verify proper operation of the Mode C and Mode S pulse processor contain two simulated Mode C or Mode S targets. In addition the scenarios contain RF test pulses to simulate a narrow pulse signal, a non-specular Mode S multipath

signal and a TACAN/DME pulse. The following set of inputs describes the standard target parameters for use throughout the subparagraphs of §2.4.2.1.3; additional inputs and conditions are specified as required.

Inputs

Intruder Aircraft

Frequency	= 1090 MHz
Altitude Rate	= 0 FPM
Relative Speed	= -360 kt (-0.1 NM/s)

ACAS X Aircraft

Altitude	= 8000 ft
Altitude Rate	= 0 FPM
Range	= 0 NM
Sensitivity Level Selection	= Automatic

Mode S Squitter Format

All Mode S squitter transmissions referred to in the subparagraphs of §2.4.2.1.3 **shall** (2322) have the following standard data field values:

DF	= 11
CA	= 0
AA	= Any valid ICAO 24-bit Aircraft Address.

2.4.2.1.3.1 Mode C Reply Reception (§2.2.4.4.2, §2.2.4.4.2.1). Narrow Pulse Discrimination (§2.2.4.5.1.2.1) and TACAN and DME Discrimination (§2.2.4.5.1.2.2)

This series of tests will verify the existence and proper operation of a Mode C DMTL.

Inputs

As described in §2.4.2.1.3 and supplemented as follows:

Intruder Aircraft 1 (Scenarios A-G)

Equipage	= Mode C
Reply Power	
Scenario A, D	= -58 dBm
Scenario B, E	= -40 dBm
Scenario C, F, and G	= -21 dBm
Altitude	= 19,000 ft
Range	= 2 NM at T=0 sec

Intruder Aircraft 2 (Scenario A-I)

Frequency	= 1090.3 MHz
Equipage	= Mode C
Reply Power	
Scenario A, D	= -69 dBm
Scenario B, E	= -52 dBm
Scenario C, F	= -33 dBm
Scenario G, H, and I	= -60 dBm
Altitude	= 8300 ft

Range	
Scenario A-C	= Maintained such that the leading edge of the F1 pulse from Intruder 2 occurs $0.7 \pm 0.2 \mu\text{s}$ later than the leading edge of the F1 pulse from Intruder 1 throughout the scenario.
Scenario D-F	= Maintained such that the leading edge of the F1 pulse from Intruder 2 occurs $19.55 \pm 0.2 \mu\text{s}$ later than the leading edge of the F1 pulse from Intruder 1 throughout the scenario.
Scenario G	= Maintained such that the leading edge of the F1 pulse from Intruder 2 occurs $26 \pm 0.2 \mu\text{s}$ later than the leading edge of the F1 pulse from Intruder 1 throughout the scenario.
Scenario H-I	= 2 NM at T=0 seconds.

Narrow Pulse (Scenario H)

Frequency	= 1090 MHz
Signal Power	= -40 dBm
Pulse Width	= $0.2 \pm 0.05 \mu\text{s}$
Pulse Rise Time	= 0.05 to 0.1 μs
Pulse Fall Time	= 0.05 to 0.2 μs
Range	= Maintained such that the leading edge of the pulse occurs $1 \pm 0.2 \mu\text{s}$ earlier than the leading edge of the F1 pulse for Intruder 2 throughout the scenario.

Slow Risetime Pulse (Scenario I)

Frequency	= 1090 MHz
Signal Power	= -40 dBm
Pulse Width	= $3 \pm 0.2 \mu\text{s}$
Pulse Rise Time	= $0.6 \pm 0.1 \mu\text{s}$
Pulse Fall Time	= $0.6 \pm 0.1 \mu\text{s}$
Range	= Maintained such that the leading edge of the pulse occurs $4 \pm 0.2 \mu\text{s}$ earlier than the leading edge of the F1 pulse of Intruder 2 throughout the scenario.

Conditions

ACAS X initialized and operating at T=0 seconds. Intruders 1, 2, and the RF pulse source shall (2323) reply only once during each whisper-shout sequence and to the same interrogation step.

Scenario Descriptions

Scenario A-F: Intruder 1 remains outside of the ACAS X active surveillance region. Intruder 2 is inside the surveillance region but within the DMTL range window established by Intruder 1.

Success: An intruder track is not established. (Scenario terminated at T=10 seconds).

Scenario G: Intruder 1 remains outside of the ACAS X active surveillance region. Intruder 2 is inside the surveillance region and just beyond the recovery region of the Intruder 1 DMTL window.

Success: A track is established on Intruder 2 following receipt of the reply to the 4th Mode C whisper-shout sequence. (Scenario terminated at T=10 seconds).

Scenario H: Intruder 2 is in the ACAS X active surveillance region and immediately preceded by a narrow pulse.

Success: A track is established on Intruder 2 following receipt of the reply to the 4th Mode C whisper-shout sequence. (Scenario terminated at T=10 seconds).

Scenario I: Intruder 2 is in the ACAS X active surveillance region and immediately preceded by a slow rise-time pulse.

Success: A track is established on Intruder 2 following receipt of the reply to the 4th Mode C whisper-shout sequence. (Scenario terminated at T=10 seconds).

2.4.2.1.3.2 Mode S Squitter and Reply Reception (§2.2.4.4.2.2), Narrow Pulse Discrimination (§2.2.4.5.1.2.1) and TACAN and DME Discrimination (§2.2.4.5.1.2.2)

This test verifies the existence and proper operation of a Mode S DMTL.

Inputs

As described in §2.4.2.1.3 and supplemented as follows:

Intruder Aircraft 1 (Scenario A-F)

Equipage	=	Mode S
Squitter Power	=	
Scenario A, E, and F	=	-61 dBm
Scenario B	=	-40 dBm
Scenario C, D	=	-21 dBm
Altitude	=	8000 ft
Range	=	2 NM at T=0 sec

Intruder Aircraft 2 (Scenario D)

Equipage	=	Mode S
Squitter Power	=	-60 dBm
Altitude	=	8000 ft
Range	=	Maintained such that the leading edge of the first preamble pulse from Intruder 2 occurs $125 \pm 1.0 \mu\text{s}$ later than the leading edge of the first preamble pulse from Intruder 1 throughout the scenario.

Simulated Multipath Signal (Scenario A-C)

Frequency	= 1090 MHz
Signal Power	
Scenario A	= -69 dBm
Scenario B	= -49 dBm
Scenario C	= -30 dBm
Pulse Width	= $120 \pm 1.0 \mu\text{s}$
Pulse Rise Time	= 0.05 to 0.1 μs
Pulse Fall Time	= 0.05 to 0.2 μs
Range	= Maintained such that the leading edge of the pulse occurs $0.7 \pm 0.2 \mu\text{s}$ later than the leading edge of the first preamble pulse from Intruder 1 throughout the scenario.

Narrow Pulse (Scenario E)

Frequency	= 1090 MHz
Signal Power	= -40 dBm
Pulse Width	= $0.2 \pm 0.05 \mu\text{s}$
Pulse Rise Time	= 0.05 to 0.1 μs
Pulse Fall Time	= 0.05 to 0.2 μs
Range	= Maintain such that the leading edge of the pulse occurs $1 \pm 0.2 \mu\text{s}$ earlier than the leading edge of the first preamble pulse from Intruder 1 throughout the scenario.

Slow Rise Time Pulse (Scenario F)

Frequency	= 1090 MHz
Signal Power	= -40 dBm
Pulse Width	= $3 \pm 0.2 \mu\text{s}$
Pulse Rise Time	= $0.6 \pm 0.1 \mu\text{s}$
Pulse Fall Time	= $0.6 \pm 0.1 \mu\text{s}$
Range	= Maintained such that the leading edge of the pulse occurs $4 \pm 0.2 \mu\text{s}$ earlier than the leading edge of the first preamble pulse from Intruder 1 throughout the scenario.

Conditions

ACAS X initialized and operating at T=0 seconds. In each scenario except D, Intruder 1 and 2 **shall** (2324) each transmit a single squitter and the RF pulse generator **shall** (2325) transmit the appropriate pulse during the squitter listening period following each of the first three whisper-shout sequences and then remain silent. In scenario D Intruders 1 and 2 **shall** (2326) each transmit squitters during the squitter listening period following each whisper-shout sequence and **shall** (2327) reply with a short special surveillance format (DF=0) to each ACAS X discrete interrogation.

Scenario Description

Scenario A-C: Intruder 1 is co-altitude with ACAS X and transmits three squitters at times when ACAS X is in squitter listening mode. The squitters are accompanied by a simulated non-specular multipath signal which begins shortly after the leading edge of the squitter.

Success: ACAS X transmits an acquisition interrogation addressed to the ICAO 24-bit Aircraft Address contained in the AP field of Intruder 1 within one second following the last squitter. (Scenario terminated at T=10 seconds).

Scenario D: Both Intruders are co-altitude with ACAS X and each transmits squitters when ACAS X is in squitter listening mode and replies to discrete interrogations from ACAS X. Intruder 2 is just beyond the recovery region of the Intruder 1 DMTR window.

Success: ACAS X transmits to each intruder two successive interrogations within two seconds following the third squitter. ACAS X transmits a tracking interrogation to each intruder within three seconds following the third squitter. (Scenario terminated at T=10 seconds).

Scenario E: Intruder 1 is co-altitude with ACAS X and accompanied by a large-amplitude narrow pulse immediately preceding the squitter.

Success: ACAS X transmits an acquisition interrogation to Intruder 1 within one second following the last squitter. (Scenario terminated at T=5 seconds).

Scenario F: Intruder 1 is co-altitude with ACAS X and accompanied by a slow rise time pulse immediately preceding the squitter.

Success: ACAS X transmits an acquisition interrogation to Intruder 1 within one second following the last squitter. (Scenario terminated at T=10 seconds).

2.4.2.1.4

Mode C Reply Reception (§2.2.4.4.2.1)

The following procedures verify the ability of the Mode C reply processor to properly detect and decode reply signals, to resolve overlapping replies and to reject narrow pulses, phantom replies and TACAN/DME signals.

Generalized Scenario Description

The test scenarios used to verify proper operation of the Mode C reply processor contain up to three simulated Mode C targets. The following set of inputs is to be used throughout §2.4.2.1.4; additional inputs and conditions are specified as required.

Inputs

ACAS X Aircraft

Altitude Rate	=	0 FPM
Sensitivity Level Selection	=	Automatic

Intruder Aircraft

Frequency	=	1090 MHz
Equipage	=	Mode C
Altitude Rate	=	0 FPM

2.4.2.1.4.1

Bracket Detection and Reply Decoding (§2.2.4.4.2, §2.2.4.4.2.1)

These tests will verify that the Mode C reply processor correctly detects the presence of a valid Mode C reply whose pulse characteristics are within the allowable limits and rejects a Mode C reply whose bracket pulse spacing and code pulse positions are outside the allowable limit.

Inputs

As described in §2.4.2.1.4 and supplemented as follows:

ACAS X Aircraft

Altitude = 9000 feet

Intruder Aircraft

Altitude = 9600 ft (Altitude Code=6760)

Range = 2 NM at T=0 sec.

Relative Speed = -360 kt (-0.1 NM/s)

Reply Power

Scenario A, C and = -65 dBm

E

Scenario B, D and = -21 dBm

F

Reply Pulse Characteristics

Each bracket and altitude code pulse **shall** (2328) have the following values for rise and fall time, pulse width and pulse position with respect to the nominal value, and amplitude with respect to F1.

Scenarios A and B

<u>Pulse</u>	<u>Rise time</u> <u>μs</u>	<u>Fall time</u> <u>μs</u>	<u>Δ Width</u> <u>μs</u>	<u>Δ Position</u> <u>μs</u>	<u>Δ Amplitude</u> <u>dB</u>
F1	0.1	0.2	+0.1	—	—
A2	0.1	0.2	+0.1	-0.1	0
A4	0.1	0.2	+0.1	-0.1	0
B1	0.1	0.2	+0.1	+0.1	-1
B2	0.1	0.2	+0.1	-0.1	0
B4	0.1	0.2	+0.1	+0.1	-1
C2	0.1	0.2	+0.1	+0.1	-1
C4	0.1	0.2	+0.1	+0.1	-1
F2	0.1	0.2	+0.1	-0.1	0

Scenarios C and D

<u>Pulse</u>	<u>Rise time</u> <u>μs</u>	<u>Fall time</u> <u>μs</u>	<u>Δ Width</u> <u>μs</u>	<u>Δ Position</u> <u>μs</u>	<u>Δ Amplitude</u> <u>dB</u>
F1	0.05	0.05	-0.1	—	—
A2	0.05	0.05	-0.1	+0.1	-1
A4	0.05	0.05	-0.1	+0.1	-1
B1	0.05	0.05	-0.1	-0.1	0
B2	0.05	0.05	-0.1	+0.1	-1
B4	0.05	0.05	-0.1	-0.1	0
C2	0.05	0.05	-0.1	-0.1	0
C4	0.05	0.05	-0.1	-0.1	0
F2	0.05	0.05	-0.1	+0.1	-1

Scenarios E and F

<u>Pulse</u>	<u>Rise time</u> <u>μs</u>	<u>Fall time</u> <u>μs</u>	<u>Δ Width</u> <u>μs</u>	<u>Δ Position</u> <u>μs</u>	<u>Δ Amplitude</u> <u>dB</u>
F1	0.1	0.2	0	—	—
A2	0.1	0.2	0	0	0
A4	0.1	0.2	0	0	0
B1	0.1	0.2	0	0	0
B2	0.1	0.2	0	+0.3	0
B4	0.1	0.2	0	+0.3	0
C2	0.1	0.2	0	0	0
C4	0.1	0.2	0	0	0
F2	0.1	0.2	0	+0.3	0

Conditions

ACAS X is initialized and operating at T=0 seconds. The intruder **shall** (2329) reply once during each whisper-shout sequence.

Scenario

A single Mode C intruder is within the active surveillance region of ACAS X at T=0 seconds.

Success: (Scenarios A-D): The intruder track is established following receipt of the reply to the 4th Mode C whisper-shout sequence. (Scenario to be terminated at T=10 seconds).

Success: (Scenarios E and F): An intruder track is not established. (Scenario terminated at T=10 seconds).

2.4.2.1.4.2 Wide Pulse Detection and Pulse Position Discrimination (§2.2.4.4.2, §2.2.4.4.2.1(c))

This test will verify the ability of the Mode C reply processor to detect the presence of all bracket or code pulses within three replies that have merged to form wide pulses and to correctly associate each pulse with the appropriate reply.

Inputs

As described in §2.4.2.1.4 and supplemented as follows:

ACAS X Aircraft

Altitude = 33,400 ft

Intruder Aircraft 1

Altitude = 8300 ft (Altitude Code=6210)

Range = 2 NM at T=0 seconds.

Relative Speed = -360 kt (-0.1 NM/s)

Reply Power at

Scenario A = -21 dBm

Scenario B = -65 dBm

Intruder Aircraft 2

Altitude = 33,700 feet (Altitude Code=1744)
 Range = Maintained such that the leading edge of the F1 pulse of Intruder 2 occurs 0.40 μ s later than the leading edge of the B2 pulse of Intruder 1 throughout the scenario.

*Note: In order to eliminate decoding uncertainties resulting from relative RF phase and pulse variations all replies **shall** (2330) be combined into a single signal that is derived from a common RF source.*

Relative Speed = -360 kt (-0.1 NM/s)

Reply Power

Scenario A = -21 dBm
 Scenario B = -65 dBm

Intruder Aircraft 3

Altitude = 14,500 feet (Altitude Code=2020)
 Range = Maintained such that the leading edge of the F1 pulse of Intruder 3 occurs 0.40 μ s later than the leading edge of the A2 pulse location of Intruder 2 throughout the scenario.

Relative Speed = -360 kt (-0.1 NM/s)

Reply Power

Scenario A = -21 dBm
 Scenario B = -65 dBm

Conditions

ACAS X is initialized and operating at T=0 seconds. Intruders 1, 2 and 3 **shall** (2331) reply once during each whisper-shout sequence and to the same interrogation step.

Scenario

Intruder 1 and 3 are outside the ACAS X active surveillance region at T=0 second. Intruder 2 is within the ACAS X active surveillance region and its reply is garbled by the replies from Intruder 1 and 3. Specifically the leading and trailing edges of the A2 pulse location from Intruder 2 are obscured respectively by the F2 pulse from Intruder 1 and the F1 pulse from Intruder 3 to form a triple overlap. In addition the leading edge of the F1 pulse of Intruder 2 is obscured by the B2 pulse of Intruder 1 and the trailing edge of the B1 pulse of Intruder 2 is obscured by the A2 pulse of Intruder 3. Also the C4, A1, B2, B4 and D4 code pulses of Intruder 2, although 0.4 μ s from code pulse locations of Intruders 1 and 3, should be associated only with Intruder 2.

Success (Scenario A and B): A track is established on Intruder 2 following receipt of the reply to the 4th Mode C whisper-shout sequence. The reply information **shall** (2332) be examined to indicate the acceptance of replies from all three Intruders. (Scenario to be terminated at T=10 seconds).

2.4.2.1.4.3 Narrow Pulse Rejection (§2.2.4.4.2.1(b), §2.2.4.5.2)

This test will verify the ability of the Mode C reply processor to reject narrow reply pulses.

Inputs

As described in §2.4.2.1.4 and supplemented as follows:

ACAS X Aircraft

Altitude	= 8600 ft
----------	-----------

Intruder Aircraft 1

Altitude	= 10,300 ft (Altitude Code=6110)
----------	----------------------------------

Range	= 2 NM at T=0 seconds.
-------	------------------------

Relative Speed	= -360 kt (-0.1 NM/s)
----------------	-----------------------

Reply Power	
-------------	--

Scenario A	= -21 dBm
------------	-----------

Scenario B	= -65 dBm
------------	-----------

Reply Pulse	
-------------	--

Width	= Nominal
-------	-----------

Intruder Aircraft 2

Altitude	= 4,750 ft (Altitude Code=4240)
----------	---------------------------------

Range	= Maintained such that the leading edge of the F1 pulse from Intruder 2 occurs at the nominal leading edge position of the B2 location of the Intruder 1 reply throughout the scenario.
-------	---

Relative Speed	= -360 kt (-0.1 NM/s)
----------------	-----------------------

Reply Power	
-------------	--

Scenario A	= -21 dBm
------------	-----------

Scenario B	= -65 dBm
------------	-----------

Reply Pulse	
-------------	--

Width	= $0.2 \pm 0.01 \mu\text{s}$
-------	------------------------------

Risetime	= $0.05 \mu\text{s}$
----------	----------------------

Falltime	= $0.05 \mu\text{s}$
----------	----------------------

Conditions

ACAS X is initialized and operating at T=0 seconds. Intruder 1 and 2 **shall** (2333) reply once during each whisper-shout sequence and to the same interrogation step.

Scenario

Intruder 1 and 2 are both within the ACAS X active surveillance region at T=0 seconds. The replies from both are garbled such that the F1 pulse from Intruder 2 occurs at the B2 code pulse position of Intruder 1. If the narrow pulses from Intruder 2 are not rejected Intruder 1 will appear to be 600 feet above ACAS X.

Success (Scenario A and B): A track is established on Intruder 1 at an altitude of 10,300 ft. (Scenario to be terminated at T=10 seconds).

2.4.2.1.4.4 Detection of Garbled Replies (§2.2.4.4.2.1(c))

This test will verify the ability of the Mode C reply processor to achieve a minimum level of overall performance in the detection of reply brackets and codes in which the reply pulses have been detected to varying degrees of garble.

Inputs

As described in §2.4.2.1.4 and supplemented as follows:

ACAS X Aircraft

Altitude = 6000 ft

Intruder Aircraft 1

Altitude = 7000 ft (Altitude Code=6020)

Range = 0 NM

Relative Speed = 0 kt

Reply Power = -60 dBm

Intruder Aircraft 2

Altitude = 6300 ft (Altitude Code=4030)

Range = Maintained such that the leading edge of the F1 pulse from Intruder 2 occurs $4.6 \pm 0.02 \mu s$ later than the leading edge of the F1 pulse from Intruder 1 throughout the scenario.

Reply Frequency = 1087 MHz

Relative Speed = 0 kt

Reply Power = -63 dBm

Intruder Aircraft 3

Altitude = 6000 ft (Altitude Code=4420)

Range = $Ri + N * (0.05 \mu s)$, where: Ri equals the initial range such that the leading edge of the F2 pulse from Intruder 3 occurs $1.0 \pm 0.2 \mu s$ earlier than the leading edge of the F1 pulse from Intruder 1, and N equals a parameter value that is incremented from 0 to 944 in steps of 1 following each set of generated replies from Intruder 1, 2 and 3 beginning with the 4th surveillance update interval.

Reply Frequency = 1093 MHz

Relative Speed = 0 kt for each value of N

Reply Power = -57 dBm

Conditions

ACAS X is initialized and operating at T=0 seconds. Intruder 1, 2 and 3 **shall** (2334) reply once during each whisper-shout sequence and to the same interrogation step. Each intruder **shall** (2335) be simulated by a separate RF source in order to maintain phase incoherency between the three replies.

Scenario

The value of N in the Intruder 3 range equation **shall** (2336) be incremented by 1 after each surveillance update interval beginning with the 4th interval. The scenario **shall** (2337) be terminated following the 948th surveillance update interval. Incrementing N will cause the reply from Intruder 3 to move in $0.05 \mu s$ steps across the Intruder 1 and 2 replies from an earlier non-overlapping range to a later non-overlapping range.

ACAS X **shall** (2338) provide track file information (§2.4.1(i)) on each intruder to indicate whether its track was updated with a valid reply for each surveillance update interval. The

reply probability for each intruder **shall** (2339) be determined by computing the rate of surveillance update intervals containing an updating reply to the total number of intervals within a track file.

Success:

Intruder 1 reply probability $\geq 60\%$

Intruder 2 reply probability $\geq 50\%$

Intruder 3 reply probability $\geq 60\%$

2.4.2.1.4.5 Detection of Interleaved Replies (§2.2.4.4.2.1(c))

This test will verify the ability of the Mode C reply processor to detect brackets and reply codes involving three Mode C replies whose pulses are interleaved.

Inputs

As described in §2.4.2.1.4 and supplemented as follows:

ACAS X Aircraft

Altitude = 12,000 ft

Intruder Aircraft 1

Altitude = 13,000 ft (Altitude Code=2220)

Range = 2 NM at T=0 seconds

Relative Speed = -360 kt (-0.1 NM/s)

Reply Power = -50 dBm

Intruder Aircraft 2

Altitude = 11,000 ft (Altitude Code=2120)

Range = Maintained such that the leading edge of the F1 pulse from Intruder 2 occurs 3.35 μ s later than the leading edge of the F1 pulse from Intruder 1 throughout the scenario.

Relative Speed = -360 kt (-0.1 NM/s)

Reply Power = -50 dBm

Intruder Aircraft 3

Altitude = 12,000 ft (Altitude Code=2360)

Range = Maintained such that the leading edge of the F1 pulse from Intruder 3 occurs 11.05 μ s later than the leading edge of the F1 pulse from Intruder 1 throughout the scenario.

Relative Speed = -360 kt (-0.1 NM/s)

Reply Power = -50 dBm

Conditions

ACAS X is initialized and operating at T=0 seconds. All three Intruders **shall** (2340) reply once during each whisper-shout sequence and to the same interrogation step. In order to eliminate detection and decoding uncertainties arising from relative RF phase and pulse variations all three intruder replies **shall** (2341) be simulated by a combined signal that is derived from a common RF source. The position and width of each pulse in the combined

signal **shall** (2342) be selected, from within allowable tolerances, to assure no pulse overlap.

Scenario

The replies from all three Intruders are interleaved. All three intruders are within the ACAS X active surveillance region.

Success: A track is established for each intruder at the specified range and altitude following receipt of the reply to the 4th Mode C whisper-shout sequence. (Scenario to be terminated at T=10 seconds).

2.4.2.1.4.6 Phantom Rejection (§2.2.4.4.2.1(d))

These tests verify the ability of the Mode C reply processor to reject replies whose brackets could be code pulses of other replies.

Inputs

As described in §2.4.2.1.4 and supplemented as follows:

ACAS X Aircraft

Altitude	=	23,000 ft
----------	---	-----------

Intruder Aircraft 1

Altitude	=	31,300 ft (Altitude Code=1414)
Range	=	2 NM at T=0 seconds
Relative Speed	=	-360 kt (-0.1 NM/s)
Reply Power	=	-50 dBm
SPI Pulse	=	Not present

Intruder Aircraft 2

Altitude	=	14,700 ft (Altitude Code=2040)
Range	=	Maintained such that the F1 pulse of Intruder 2 occurs $17.4 \pm 0.1 \mu\text{s}$ later than the F2 pulse of Intruder 1 throughout the scenario.
Relative Speed	=	-360 kt (-0.1 NM/s)
Reply Power	=	-50 dBm

Conditions

ACAS X initialized and operating at T=0 seconds. Intruder 1 and 2 **shall** (2343) both reply once during each whisper-shout sequence and to the same interrogation step.

Scenario

Intruder 1 and 2 are positioned to form a phantom reply whose bracket pulses are derived from the B4 code pulse of Intruder 1 and the F1 bracket pulse of Intruder 2. Both intruders are within the active surveillance region and the phantom reply also indicates an aircraft within the active surveillance region.

Success: A track is established on Intruder 1 and Intruder 2 at the altitude specified for each intruder. The phantom replies do not initiate a track. (Scenario to be terminated at T=10 seconds).

2.4.2.1.4.7 TACAN and DME Pulse Rejection (§2.2.4.5.3)

This test verifies the ability of the Mode C Reply processor to reject any received pulse with a rise time exceeding 0.5 μ s

Inputs

As described in §2.4.2.1.4 and supplemented as follows:

ACAS X Aircraft

Altitude	= 6200 ft (Altitude Code=4410)
----------	--------------------------------

Intruder Aircraft

Altitude	= 6500 ft (Altitude Code=4020)
----------	--------------------------------

Range	= 2 NM at T=0 seconds
-------	-----------------------

Relative Speed	= -360 kt (-0.1 NM/s)
----------------	-----------------------

Reply Power	= -50 dBm
-------------	-----------

Slow Rise Time Pulse

Frequency	= 1090 MHz
-----------	------------

Pulse Width	= $3.0 \pm 0.2 \mu$ s
-------------	-----------------------

Rise Time	= $0.6 \pm 0.1 \mu$ s
-----------	-----------------------

Fall Time	= $0.6 \pm 0.1 \mu$ s
-----------	-----------------------

Power Level	= -50 dBm
-------------	-----------

Range	= $R_i + N * (0.05 \mu$ s), where R_i equals the initial range such that the leading edge of the slow rise time pulse occurs $14 \pm 0.1 \mu$ s later than the leading edge of the F1 pulse of the intruder aircraft and N equals a parameter value that is incremented from 0 to 19 by 1 following each repeat of the scenario.
-------	--

Conditions

ACAS X initialized and operating at T=0 seconds. The intruder and the RF pulse source **shall** (2344) both reply once during each whisper-shout sequence and to the same interrogation step.

Scenario

The scenario is repeated for a total of 20 runs. The intruder is within the ACAS X active surveillance region and accompanied by a simulated slow risetime interferer whose position is co-incident with the B2 and D2 code pulse positions of the intruder and incremented by 0.05 μ s each time the scenario is repeated. If the slow risetime pulse is not rejected the altitude of the intruder will be incorrect.

Success: For each run the intruder track is established at the altitude specified following receipt of the reply to the 4th whisper-shout sequence. (Each run is terminated at T=10 seconds).

2.4.2.1.5

Mode S Squitter and Reply Reception (§2.2.4.4.2.2)

The following test procedures verify the signal detection, decoding and false squitter rejection functions of the Mode S reply processor.

Generalized Scenario Description

The following set of inputs is described for use throughout the subparagraphs of §2.4.2.1.5; additional inputs and conditions are specified as required.

Inputs

ACAS X Aircraft

Altitude	=	8000 ft
Altitude Rate	=	0 FPM
Sensitivity Level	=	
Selection	=	Automatic

Intruder Aircraft

Frequency	=	1090 MHz
Equipage	=	Mode S
Altitude Rate	=	0 FPM
Range	=	2 NM at T=0 seconds
Relative Speed	=	-360 kt (-0.1 NM/s)
Mode S Squitter Format		
DF	=	11
CA	=	0
AA	=	Any valid ICAO 24-bit Aircraft Address

Mode S Short Special Surveillance Format

DF	=	0
VS	=	0
RI	=	11 in response to AQ=1 0 in response to AQ=0
AP	=	Any valid ICAO 24-bit Aircraft Address

Mode S Fruit Reply (to ground interrogation)

DF	=	4
FS	=	0
DR	=	0
UM	=	0
AP	=	Any valid ICAO 24-bit Aircraft Address

Mode S Fruit Reply (to other ACAS X interrogation)

DF	=	0
VS	=	0
RI	=	0
AP	=	Any valid ICAO 24-bit Aircraft Address

2.4.2.1.5.1 Mode S Preamble Reception (§2.2.4.4.2.2(b))

These tests verify that the Mode S reply processor correctly detects the presence of a valid Mode S preamble whose pulse characteristics are within the allowable limits and rejects preambles whose pulse spacings, positions and leading edge characteristics are outside the allowable limit.

Inputs

As described in §2.4.2.1.5 and supplemented as follows:

Intruder Aircraft

Altitude	=	8000 ft
Squitter Power		
Scenarios A, C, E, G and I	=	-65 dBm
Scenarios B, D, F, H and J	=	-21 dBm

Preamble Characteristics

Each of the four pulses within the Mode S preamble **shall** (2345) have the following values for rise and fall time, width and position difference from the nominal value and amplitude difference from the amplitude of the first pulse.

Scenarios A and B

<u>Pulse</u>	<u>Rise time</u> μs	<u>Fall time</u> μs	<u>Δ Width</u> μs	<u>Δ Position</u> μs	<u>Δ Amplitude</u> dB
1	0.1	0.2	+0.05	—	—
2	0.1	0.2	-0.05	-0.125	+2
3	0.1	0.2	+0.05	+0.125	+2
4	0.1	0.2	-0.05	-0.125	0

Scenarios C and D

<u>Pulse</u>	<u>Rise time</u> μs	<u>Fall time</u> μs	<u>Δ Width</u> μs	<u>Δ Position</u> μs	<u>Δ Amplitude</u> dB
1	0.05	0.05	-0.05	—	—
2	0.05	0.05	+0.05	+0.125	0
3	0.05	0.05	-0.05	-0.125	0
4	0.05	0.05	+0.05	+0.125	+2

Scenarios E and F

<u>Pulse</u>	<u>Rise time</u> μs	<u>Fall time</u> μs	<u>Δ Width</u> μs	<u>Δ Position</u> μs	<u>Δ Amplitude</u> dB
1	0.1	0.2	-0.3	—	—
2	0.1	0.2	-0.3	0	0
3	0.1	0.2	-0.3	0	0
4	0.1	0.2	-0.3	0	0

Scenarios G and H

<u>Pulse</u>	<u>Rise time</u> μs	<u>Fall time</u> μs	<u>Δ Width</u> μs	<u>Δ Position</u> μs	<u>Δ Amplitude</u> dB
1	0.1	0.2	0	—	—
2	0.1	0.2	0	+0.2	0
3	0.1	0.2	0	+0.2	0
4	0.1	0.2	0	+0.2	0

Scenarios I and J

Pulse	Rise time μs	Fall time μs	Δ Width μs	Δ Position μs	Δ Amplitude dB
1	0.05 - 0.1	0.05 - 0.2	+4.5	—	—
2					
3		Not Present			
4					

Conditions

ACAS X is initialized and operating at T=0 seconds. The intruder **shall** (2346) transmit a squitter during the squitter listening period following the completion of each of the first three whisper-shout sequences and then remain silent.

Scenario Description

The intruder is within the ACAS X active surveillance region at T=0 seconds and transmits three squitters at a time when ACAS X is in squitter listening mode.

Scenarios A-D: The Mode S preamble pulse characteristics are set to the extreme limits of their tolerance range.

Success: ACAS X transmits an acquisition interrogation addressed to the ICAO 24-bit Aircraft Address contained in the AP field of the intruder within one second following the last squitter. (Scenarios terminated at T=10 seconds)

Scenarios E-H: The Mode S preamble pulse widths (Scenario E and F) and positions (Scenario G and H) are set to out-of-tolerance values that will cause rejection of the preamble.

Success: ACAS X does not transmit an acquisition interrogation. (Scenarios terminated at T=10 seconds).

Scenarios I and J: A single wide pulse is used to simulate a preamble that contains only one clear leading edge but otherwise satisfies the criteria for preamble acceptance.

Success: ACAS X does not transmit an acquisition interrogation. (Scenarios terminated at T=10 seconds).

2.4.2.1.5.2

Mode S Squitter and Fruit Reply Reception (§2.2.4.4.2.2(c))

This test will verify the ability of the Mode S reply processor to recognize and accept valid Mode S Squitter and fruit replies and reject a fruit reply whose data block content is corrupted to the extent that it could result in an incorrect altitude.

Inputs

As described in §2.4.2.1.5 and supplemented as follows:

Intruder Aircraft 1

Altitude	=	8,500 ft
Squitter/Fruit Power	=	-50 dBm
Data Field (Fruit Reply Only)		

The normal data block content **shall** (2347) be modified to contain energy throughout both halves (chips) of the first seven bit positions following the DF field. The amplitude of the pulse in the half that would ordinarily contain no energy **shall** (2348) be three dB below the amplitude of the normal pulse in the other half of the bit position.

Intruder Aircraft 2

Altitude	=	9000 ft
Squitter/Fruit Power	=	-50 dBm
Data Field (Fruit Reply Only)		

As described for Intruder 1 except that energy is contained throughout both halves of the first eight bit positions following the DF field.

Intruder Aircraft 3

Altitude	=	7000 ft
Squitter/Fruit Power	=	-50 dBm
Data Field (Fruit Reply Only)		
DF	=	5

Intruder Aircraft 4

Data Field (Fruit Reply Only)		
AC	=	6671
Squitter/Fruit Power	=	-50 dBm

Intruder Aircraft 5

Altitude	=	8000 ft
Squitter/Fruit Power	=	-50 dBm
Data Field (Fruit Reply Only)		
Bit 26	=	"1"

Conditions

ACAS X is initialized and operating at T=0 seconds. All five Intruders **shall** (2349) transmit a squitter during the squitter listening period following completion of the first whisper-shout sequence and a Mode S fruit reply during the squitter listening period following completion of the second and third whisper-shout sequences and then remain silent. The Mode S fruit replies for Intruders 1, 2, 4 and 5 **shall** (2350) have a DF=0 for the first fruit reply and a DF=4 for the second fruit reply. Both fruit replies from Intruder 3 **shall** (2351) have a DF=5.

Scenario

All five Intruders are within the active surveillance region of ACAS X and each transmits one squitter and two fruit replies at a time when ACAS X is in the squitter listening mode.

Success: ACAS X transmits an acquisition interrogation addressed to the ICAO 24-bit Aircraft Address contained in the AP field of Intruder 1 within one second following the last fruit reply. ACAS X does not transmit an acquisition interrogation addressed to any other ICAO 24-bit Aircraft Address. (Scenario to be terminated at T=10 seconds).

2.4.2.1.5.3 **Mode S Extended Squitter Reception (§2.2.3.9.8, §2.2.3.9.2.1, §2.2.4.6.4.2, §2.2.4.4.2.2(c))**

This test will verify that the Mode S reply processor is:

- a. able to recognize and accept all valid Mode S extended squitters with DF=17 and to use these squitters for Mode S squitter processing and Mode S acquisition; only applicable if optional DF=17 message processing for Mode S track acquisition is implemented.

- b. able to recognize, accept, and decode valid Mode S extended squitters with DF=17 that contain the aircraft identification message in the ME field, if implemented.

Inputs

As described in §2.4.2.1.5 and supplemented as follows:

Intruder Aircraft 1

(Only applicable if optional DF=17 message processing is implemented)

Altitude	=	8000 ft
Squitter Power	=	-50 dBm
Extended Mode S Squitter Format		
DF	=	17
CA	=	0
AA	=	Same address as DF=11 squitter

Intruder Aircraft 2

Altitude	=	8000 ft
Squitter Power	=	-50 dBm
Extended Mode S Squitter Format		
DF	=	17
CA	=	0
AA	=	Same address as DF=11 squitter
ME	=	
Bits 33-37	=	Format Type Code 4
Bits 38-40	=	Aircraft Type Code 3
Bits 41-88	=	ICAO Aircraft Identification, which can be any airline code and flight number encoded according to Ref. G, Ref. M.

Conditions

ACAS X is initialized and operating at T=0 seconds. Intruder 1 **shall** (2352) transmit one DF=11 squitter during the squitter listening period following completion of the first whisper-shout sequence and two DF=17 squitters spaced 0.5 seconds apart during each of the squitter listening periods following the second and third whisper-shout sequences and then remain silent. Intruder 2 **shall** (2353) transmit DF=11 squitters during the squitter listening period following completion of each whisper-shout sequence. Intruder 2 **shall** (2354) also transmit DF=17 squitters containing aircraft identity at a nominal rate of once every five seconds beginning at T=0 seconds such that the DF=17 identity squitter occurs during a squitter listening period.

Scenario

Intruder 1 is within the active surveillance region of ACAS X and transmits one DF=11 squitter followed by four DF=17 squitters when ACAS X is in the squitter listening mode. Intruder 2 is within the altitude protection volume of ACAS X and transmits DF=11 squitters each second and DF=17 identity squitters once every five seconds when ACAS X is in the squitter listening mode.

Success: ACAS X transmits an acquisition interrogation to Intruder 1 within one second following the second extended squitter. ACAS X transmits an acquisition interrogation to Intruder 2 within one second following the third DF=11 squitter and correctly decodes the DF=17 identity squitter containing the Intruder 2 ICAO Aircraft Identification.

2.4.2.1.5.4 Mode S Error Correction (§2.2.4.4.2.2(d))

This test will verify that the equipment makes use of the parity coding in the Mode S reply to correct errors in the received message.

Inputs

As specified in §2.4.2.1.7 and supplemented as follows:

ACAS X Aircraft

Altitude	= 8,000 ft
Sensitivity Level Selection	= Automatic

Intruder Aircraft

Equipage	= Mode S
Altitude	= 8,000 ft (Altitude Code=6620)
Altitude Rate	= 0 FPM
Range	= 4 NM
Relative Speed	= -360 kt (-0.1 NM/s)
Data Field	
RI	= 11

Error Simulation

The data field content of the short special surveillance reply (DF=0) **shall** (2355) be modified following generation of the parity bit sequence and prior to PPM modulation in the following manner:

Data bit number 32 **shall** (2356) contain energy throughout both halves (chips) of its bit position.

The amplitude of the signal in the half that would ordinarily contain energy **shall** (2357) be three dB below the amplitude of the signal in the other half of the bit position.

Conditions

ACAS X initialized and operating at T=0 seconds. The intruder **shall** (2358) transmit a squitter during the squitter listening period following completion of each whisper-shout sequence and **shall** (2359) reply with the modified short special surveillance reply (DF=0) to each ACAS X discrete interrogation.

Scenario Description

The intruder is within the active surveillance region of ACAS X, transmits squitters when ACAS X is in squitter listening mode, and replies to discrete interrogations from ACAS X.

Success: ACAS X transmits an acquisition interrogation addressed to the intruder within one second following the third squitter.

2.4.2.1.6

Mode C Target Surveillance Performance (§2.2.4.6.4)

The following test procedures apply to the Mode C surveillance function of the ACAS X unit and verify proper surveillance of Mode C-equipped aircraft. The test procedures **shall** (2360) verify that ACAS X is capable of the timely acquisition and surveillance of such targets in the specified environments.

Generalized Scenario Description

The test scenarios used to verify Mode C surveillance contain simulated Mode C targets. For each target, the barometric altitude, altitude rate, range (relative to ACAS X), speed (relative to ACAS X) and the reply power of each target is specified. In addition, multipath delay, multipath reply power and whisper-shout level are specified, when applicable.

The target scenarios are not probabilistic; that is, reply probabilities less than one are not explicitly permitted. Instead, the effect of an intermittent target is expressed, when appropriate, by a predetermined sequence of replies and missed scans. Also, fruit replies, when included in the simulation, are generated in a deterministic fashion.

2.4.2.1.6.1 Mode C Surveillance Initiation (§2.2.4.6.4.1.2, §2.2.4.5, §2.2.4.6)

This test procedure verifies that the Mode C surveillance acquisition range is adequate and that parameters of the acquisition logic are correct.

Test Scenarios

Scenarios A, B and C each consist of an Intruder 1 aircraft on a horizontal collision course. The threat aircraft flight is level. In scenario A, the target replies on each surveillance update interval. In scenario B, the target replies for only two consecutive surveillance update intervals and then intermittently (every other surveillance update interval). Since the track acquisition logic requires three consecutive replies a track cannot be established.

In Scenario C, beginning with the fourth reply, the target reports an altitude that is 300 ft different from the altitude reported in the first three replies. The target replies intermittently following the first three replies to prevent track initiation on replies received following the altitude jump. Since the fourth reply altitude is beyond the correlation window a track should not be established.

Inputs

Intruder Aircraft

Initial Range	=	13 NM
Relative Speed	=	-1150 ±50 kt (-0.32 ±0.01 NM/s)
Altitude		
Scenario A and B	=	11,300 ft
Scenario C	=	11,300 ft for the first three replies
	=	11,000 ft for the fourth and succeeding replies
Altitude Rate	=	0 FPM
Equipage	=	Mode C
Reply power	=	-65 dBm
Reply Sequence:		
Scenario A	=	Once every surveillance update interval beginning at T=0 seconds
Scenario B and C	=	Replies for the first two surveillance update intervals then alternating intervals (replies at T=3, 5, 7, etc.)

TCAS-Equipped Aircraft

Sensitivity Level		
Selection	=	Automatic
Altitude	=	11,000 ft

Conditions

ACAS X initialized and operating (with no inputs other than those specified) at or before T=0 seconds.

Success Criteria

Scenario A: A continuous target is displayed beginning at T=4 and persisting throughout the scenario.

Scenario B: No target is displayed during the encounter

Scenario C: No target is displayed between T=0 and T=7.

2.4.2.1.6.2 Mode C Surveillance Extension (§2.2.4.6.4.1.3, §2.2.4.6.4.1.4)**2.4.2.1.6.2.1 Elimination of Multipath False Tracks**

This test procedure **shall** (2361) verify that ACAS X eliminates false tracks created by ground-bounce multipath.

Multipath from reflective surfaces can create false tracks at two ranges (Figure 2-34). This is due to:

- a. Multipath only on the interrogation link or only on the reply link.
- b. Multipath on both the interrogation link and on the reply link.

In this test procedure, the false tracks due to non-sea level multipath from an Intruder A aircraft will be simulated using an additional Intruder B. Unlike true multipath, the replies from Intruder A and B will have different altitude codes. This allows Intruder A to be outside the ACAS X threat zone, while its simulated multipath false track, Intruder B, resides within the threat zone, as shown in Figure 2-35 and Figure 2-36. If ACAS X is eliminating false tracks, there should only be a single intruder in the STM report with the altitude of intruder A. The scenario will be run twice to simulate the two cases of multipath mentioned above.

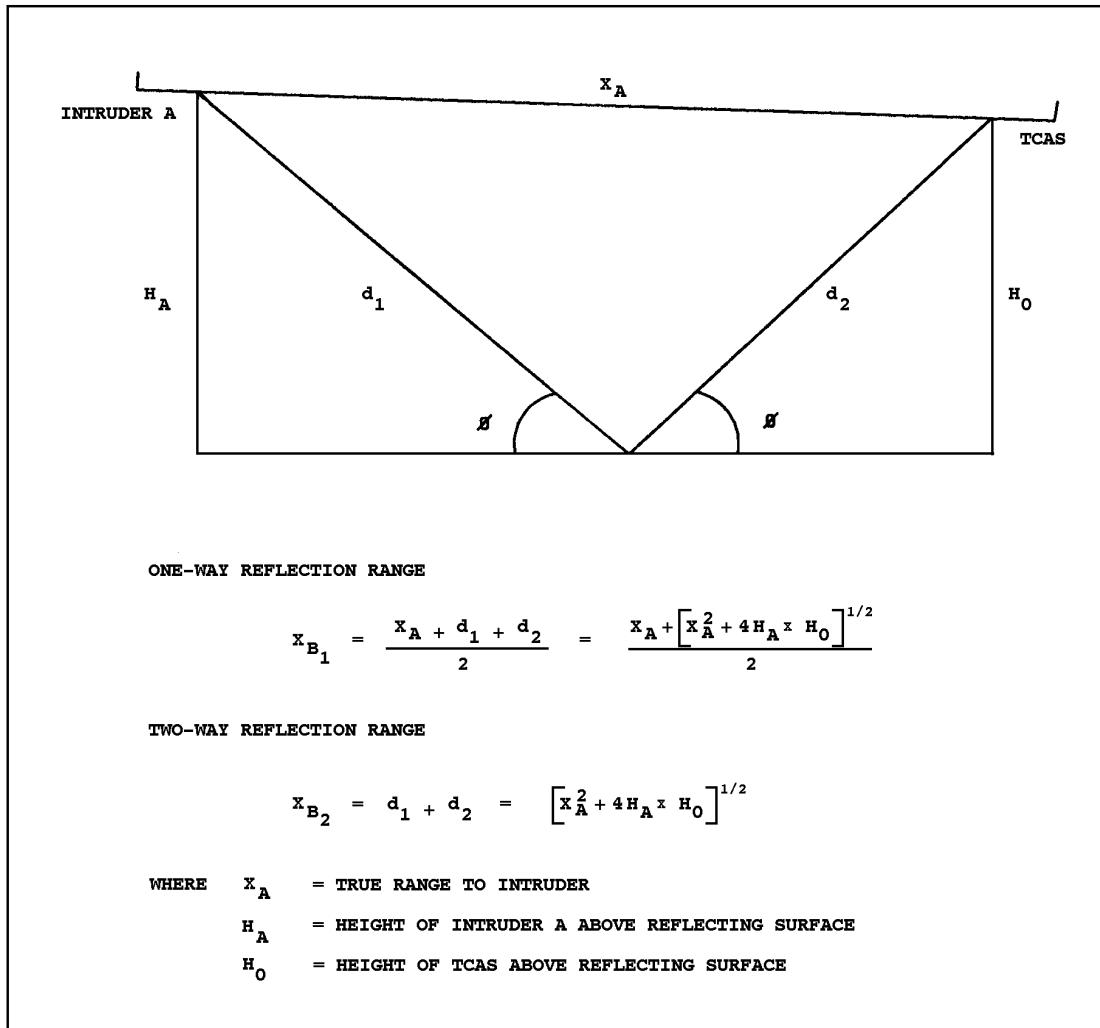


Figure 2-34: Geometrical Illustration Of Multipath

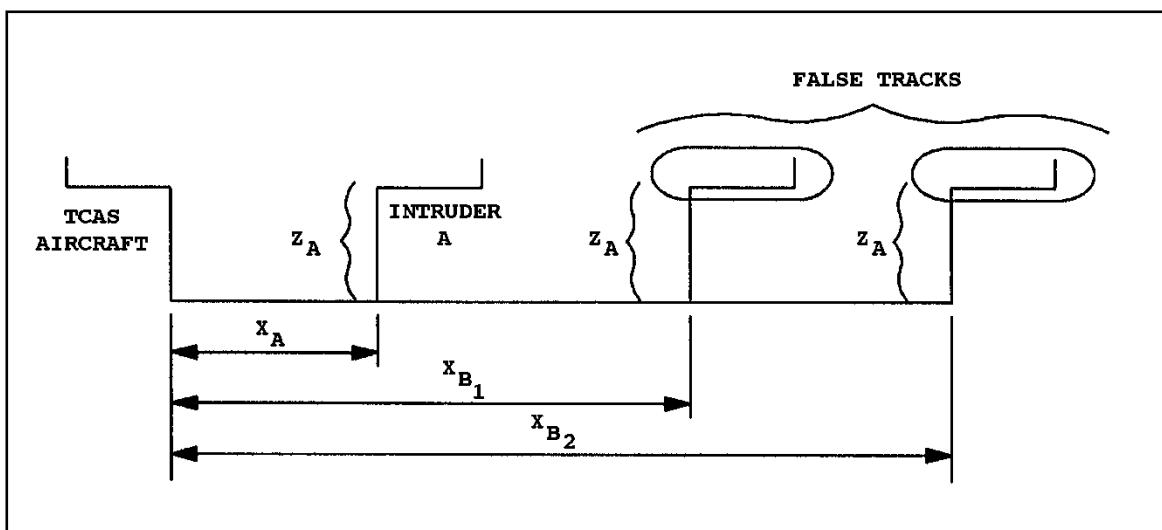


Figure 2-35: Real Multipath

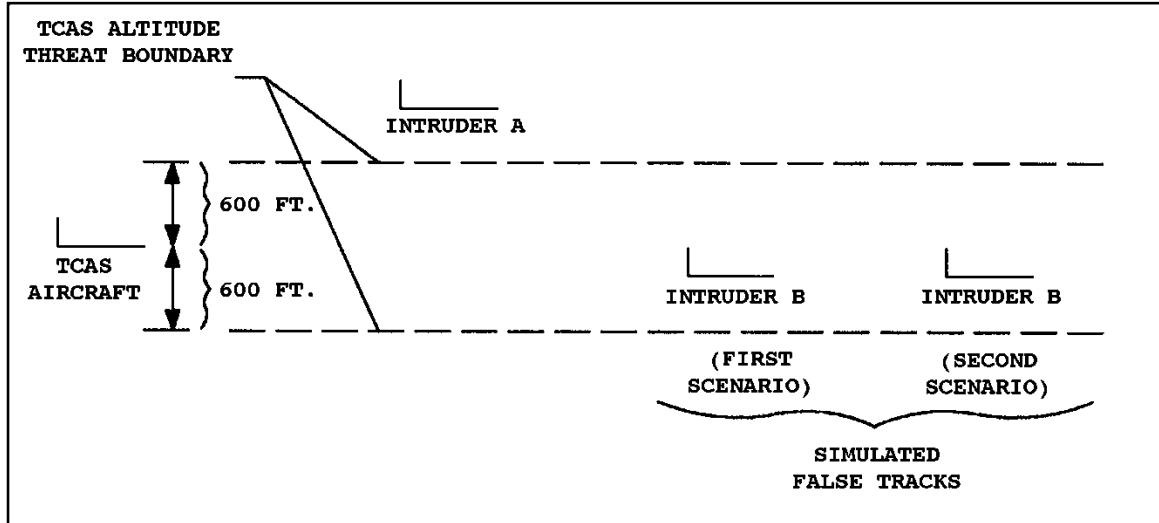


Figure 2-36: Simulated Multipath

Scenario

The scenario consists of an ACAS X-equipped aircraft and two Mode C intruders, Intruders A and B. Intruder B is at an altitude inside the ACAS X threat boundary while Intruder A is not. The range and velocity of Intruder B is identical to that of a multipath generated false track from Intruder A. If ACAS X is eliminating false targets, the STM report will contain only a single intruder with the altitude of intruder A.

Intruders A and B are made to respond to two different whisper-shout levels to eliminate effects of garbling.

Inputs

ACAS X Aircraft

Sensitivity Level	
Selection	= Automatic
Altitude	= 11,000 ft
Height Above	
Reflecting Surface (H_0)	= 9,000 ft

Intruder Aircraft A

Altitude	= 11,700 ft
Altitude Rate	= 0 FPM
Height Above	
Reflecting Surface (H_A)	= 9,850 ft
Initial Range	= 8 NM at T=0 seconds
Relative Speed (X_A)	= -1080 kts (-0.3 NM/s)
Power Level	= -50 dBm
Equipage	= Mode C
Whisper-Shout	
Interrogation No.	= 1

False Target (Aircraft B)

Altitude	=	11,500 ft
Altitude Rate	=	0 FPM
Initial Range	X_{B_1}	= 8.29 NM at T=0 seconds (1st scenario)
	X_{B_2}	= 8.58 NM at T=0 seconds (2nd scenario)
Relative Speed	\dot{X}_{B_1}	$\dot{X}_{B_1} = \frac{X_{B_1} \dot{X}_A}{2X_{B_1} - X_A}$ (1st scenario)
	\dot{X}_{B_2}	$\dot{X}_{B_2} = \frac{X_A \dot{X}_A}{X_{B_2}}$ (2nd scenario)

where X_A , X_{B_1} , and X_{B_2} are defined in Figure 2-34.

Power Level	=	-50 dBm
Equipage	=	Mode C
Whisper-Shout		
Interrogation No.	=	2

Time Sequence:

T=0 seconds	Scenario initialized.
T=0 - 6 seconds	Sufficient time for aircraft B to be acquired.
T=5 - 10 seconds	The STM report should contain only a single intruder with the altitude of intruder A.
T=10 seconds	Repeat scenario with new values for X_B and \dot{X}_B .
T=15 - 20 seconds	The STM report should contain only a single intruder with the altitude of intruder A.
T=20 seconds	Scenario terminated.

2.4.2.1.6.2.2 Range Correlation

The purpose of this test is to demonstrate that the surveillance processor correctly fails to update a track with replies that lie outside the proper range correlation box.

The size of the correlation box is a function of several track parameters and may take on numerous values. Rather than testing exhaustively over all possible cases, this test uses two scenarios to verify the performance in the two altitude regimes (above and below 10,000 ft).

Test Scenarios

Scenarios A and B are qualitatively similar. In each, a single threat aircraft on a horizontal collision course (Intruder 1) is used, beginning at time T=0. Beginning at T=T1 after the track on Intruder 1 is firmly established, but before the threat volume is entered, a range error is introduced so as to cause correlation failure with respect to the established track. This is accomplished by terminating Intruder 1 and introducing Intruder 2 which is identical to 1 except for a range offset equivalent to the intended error value. During the period in which Intruder 2 is present, updates are made only every other surveillance update interval. (This prevents a new track from being established after the original one has been dropped). The period of simulated range error continues until after the threat volume has been entered. Therefore, an advisory will be displayed if the processor has incorrectly

correlated through the onset of the range bias. Figure 2-37 and Figure 2-38 illustrate the intended reply sequence for Scenarios A and B respectively.

Note: T1 is actually chosen so that the last correlating reply occurs more than six seconds before the track would be predicted to penetrate the threat boundary, i.e., surveillance will have deleted the track according to §2.2.4.6.4.1.3. For altitudes above 10,000 feet (Scenario A), the threat boundary is determined by the requirement for 30 seconds warning as in previous tests. For altitudes below 10,000 feet (Scenario B), the corresponding requirement is 25 seconds warning before a 0.3 NM miss.

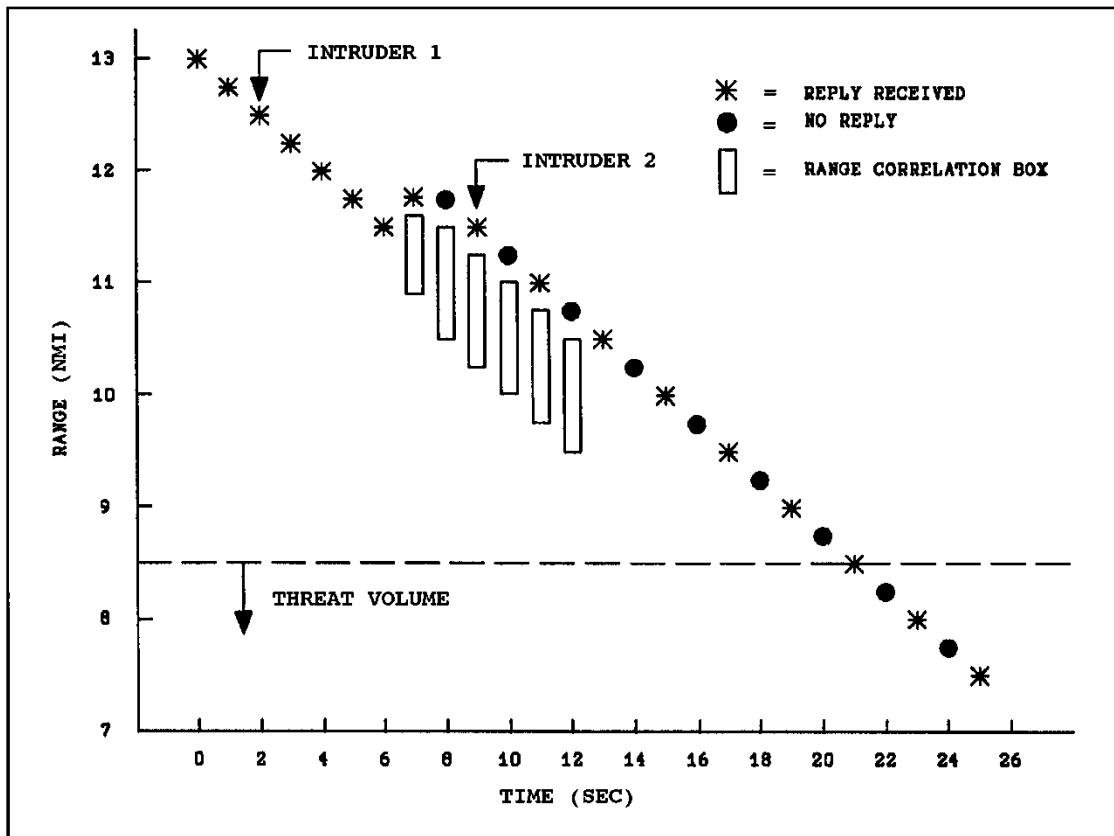


Figure 2-37: Reply Sequence For Range Correlation Test Scenario A (Above 10 Kft)

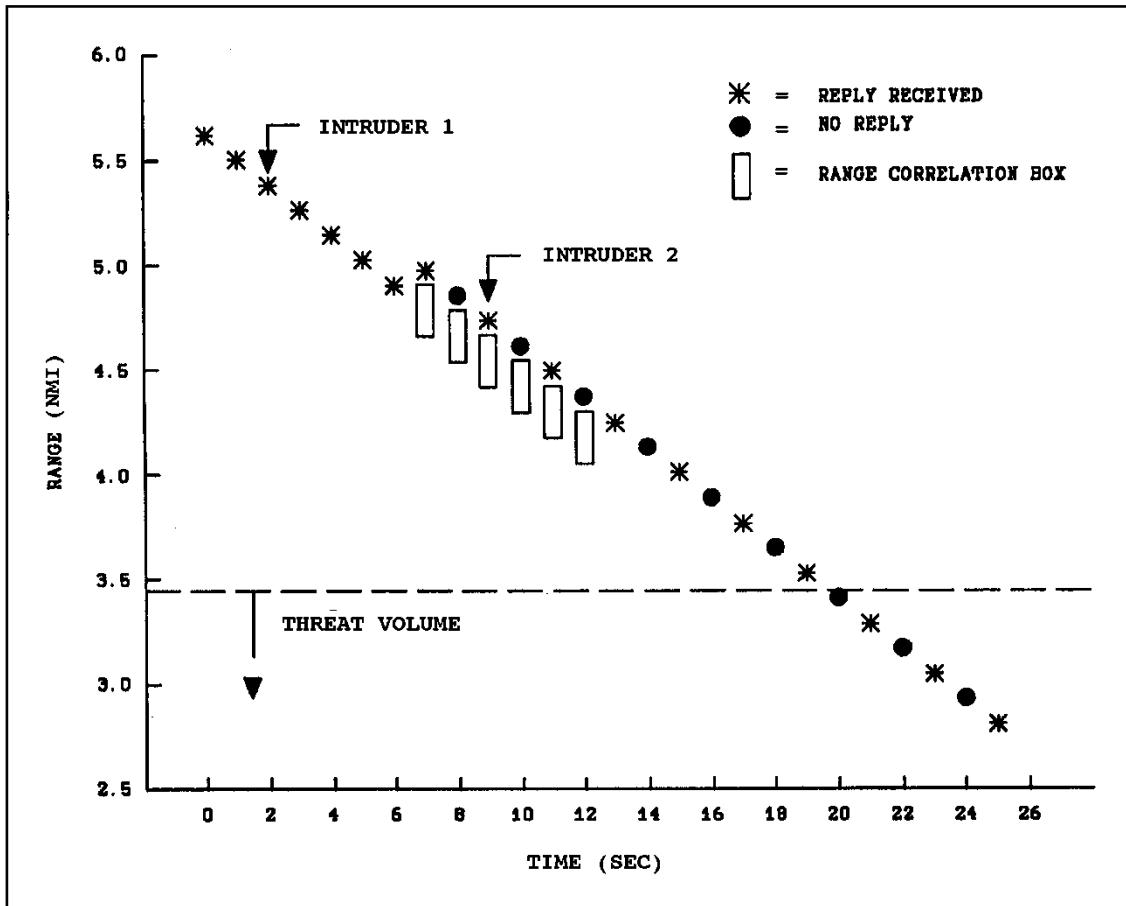


Figure 2-38: Reply Sequence For Range Correlation Test Scenario B (Below 10 Kft)

Inputs

Intruder Aircraft 1

Initial Range

Scenario A = 13 NM at T=0 seconds

Scenario B = 5.75 NM at T=0 seconds

Relative Speed

Scenario A = -900 kt (-0.25 NM/s)

Scenario B = -450 kt (-0.125 NM/s)

Altitude

Scenario A = 11,000 ft

Scenario B = 9,000 ft

Altitude Rate

= 0 FPM

Equipage

= Mode C

Reply Power

= -65 dBm

Reply Sequence

= Once per surveillance update interval from T=0

seconds through T=6 seconds and inhibited

thereafter.

Intruder Aircraft 2

Initial Range	
Scenario A	= 12 NM at T1=7 seconds
Scenario B	= 5.075 NM at T1=7 seconds
Relative Speed	
Scenario A	= -900 kt (-0.25 NM/s)
Scenario B	= -450 kt (-0.125 NM/s)
Altitude	
Scenario A	= 11,000 ft
Scenario B	= 9,000 ft
Equipage	= Mode C
Reply Power	= -65 dBm
Reply Sequence	= Inhibited from T=0 seconds to T=6 seconds. Thereafter alternating surveillance update intervals only (i.e., replies at T=7, 9, 11 seconds, etc.).

ACAS X Aircraft

Sensitivity	
Level Selection	= Automatic
Altitude	= 11,000 ft for scenario A 9,000 ft for scenario B

Conditions

ACAS X initialized and operating (with no inputs other than those specified) at or before T=0 seconds.

Success Criteria

The target drops 6 seconds after the introduction of the range error.

2.4.2.1.6.2.3 Altitude Correlation

This test verifies that the surveillance processor correctly fails to update a track with replies which are outside their proper altitude correlation box of ± 200 feet.

Test Scenario

At T=0 seconds a threat aircraft (Intruder 1) flies a horizontal approach course with an altitude which differs by 300 feet from the ACAS X altitude. Beginning at T=T1, after the track is well established but before the threat volume is entered, an altitude error of 300 feet is introduced so as to cause correlation failure. This is accomplished by terminating Intruder 1 and introducing Intruder 2 which is identical to 1 except for an altitude offset equivalent of the intended error value. If correlation succeeds, the extended track will be within the vertical threat limits and an advisory will be displayed when the horizontal threat boundary is penetrated. During the period in which Intruder 2 is present, updates are made only every other scan, to prevent a new track from being established. The intended reply sequence is illustrated in Figure 2-39.

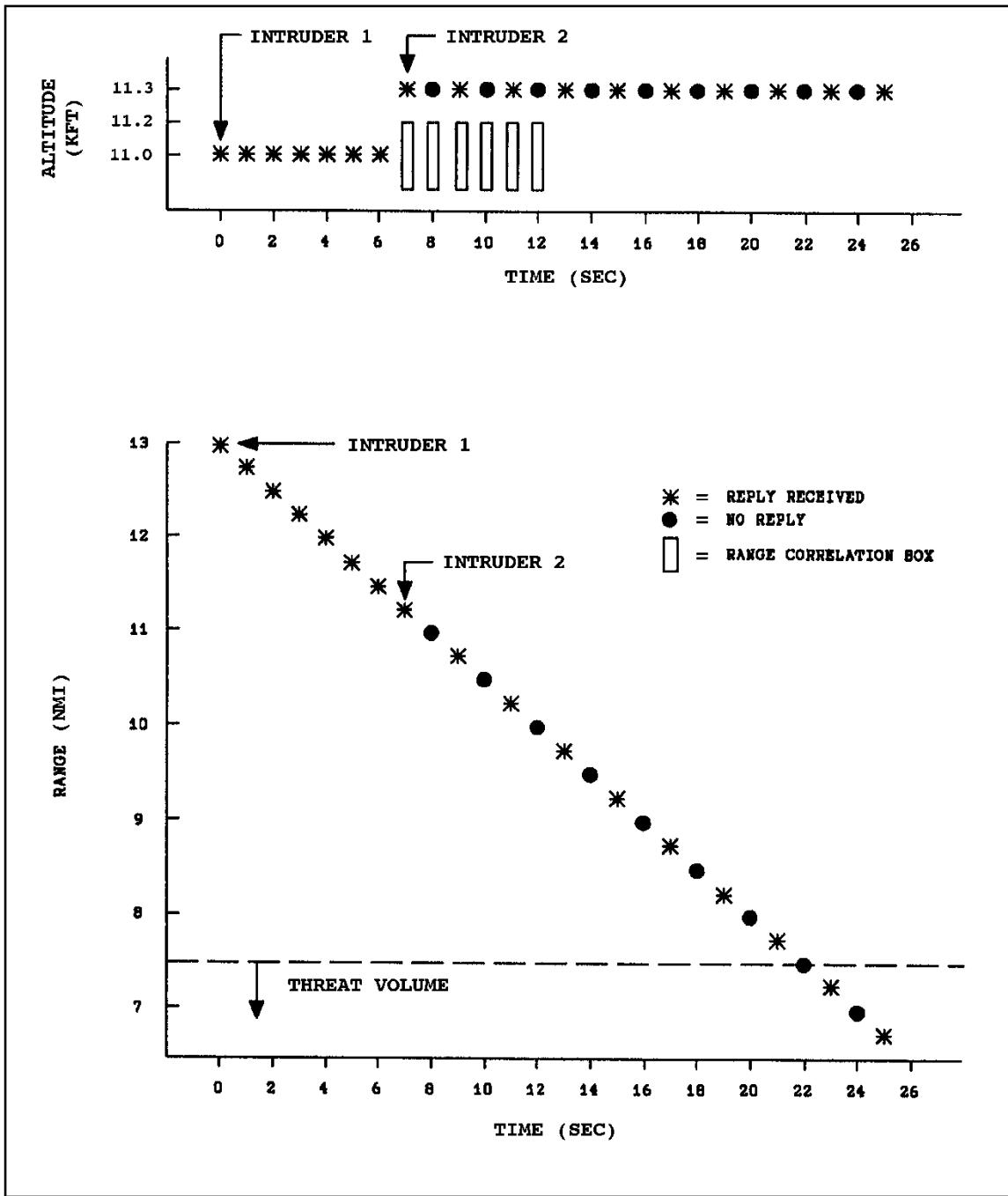


Figure 2-39: Reply Sequence For Altitude Correlation Test

InputsIntruder Aircraft 1

Initial Range	= 13 NM at T=0 seconds
Relative Speed	= -900 kt (-0.25 NM/s)
Altitude	= 11,000 ft
Altitude Rate	= 0 FPM
Equipage	= Mode C
Reply Power	= -65 dBm
Reply Sequence	= Once per surveillance update interval beginning at T=0 seconds through T=6 seconds and inhibited thereafter.

Intruder Aircraft 2

Intruder Range	= 11.25 NM at T1=7 seconds
Relative Speed	= -900 kt (-0.25 NM/s)
Altitude	= 11,300 ft
Altitude Rate	= 0 FPM
Equipage	= Mode C
Reply Power	= -65 dBm
Reply Sequence	= Inhibited from T=0 to T=6 seconds. Thereafter alternating surveillance update intervals only (i.e., replies at T=7, 9, 11 seconds, etc.).

ACAS X-Equipped Aircraft

Sensitivity Level	
Selection	= Automatic
Altitude	= 11,300 ft

Conditions

ACAS X initialized and operating (with no inputs other than those specified) at or before T=0 seconds.

Success Criteria

The target drops 6 seconds after the introduction of the altitude error.

2.4.2.1.6.2.4 Altitude Code Processing

This test verifies that ACAS X properly processes altitude information received from both the intruder aircraft and own transponder over the full range of altitude codes.

InputsACAS X Aircraft

Altitude Rate	= 0 FPM
Range	= 0 NM
Relative Speed	= 0 kt
Sensitivity Level Selection	= Automatic

Altitude

Scenario A	= -1000 ft (Altitude Code=0020)
Scenario B	= -500 ft (Altitude Code=0420)
Scenario C	= 500 ft (Altitude Code=0220)
Scenario D	= 2500 ft (Altitude Code=0120)
Scenario E	= 6500 ft (Altitude Code=4020)
Scenario F	= 14,500 ft (Altitude Code=2020)
Scenario G	= 30,200 ft (Altitude Code=1410)
Scenario H	= 62,200 ft (Altitude Code=0414)
Scenario I	= 125,000 ft (Altitude Code=0222)

Intruder Aircraft

Frequency	= 1090 MHz
Equipage	= Mode C
Altitude Rate	= 0 FPM
Range	= 2 NM at T=0 seconds
Relative Speed	= -360 kt (-0.1 NM/s)
Reply Power	= -50 dBm
Altitude	
Scenario A	= -700 ft (Altitude Code=0410)
Scenario B	= -200 ft (Altitude Code=0640)
Scenario C	= 800 ft (Altitude Code=0340)
Scenario D	= 2800 ft (Altitude Code=4140)
Scenario E	= 6800 ft (Altitude Code=6040)
Scenario F	= 14,800 ft (Altitude Code=3040)
Scenario G	= 30,500 ft (Altitude Code=1020)
Scenario H	= 62,500 ft (Altitude Code=0024)
Scenario I	= 125,300 ft (Altitude Code=0612)

Conditions

ACAS X is initialized and operating at T=0 seconds. The intruder **shall** (2362) reply once during each whisper-shout sequence.

Scenario

For each scenario a single Mode C intruder is within the threat zone of ACAS X at T=0 seconds.

Success (Scenarios A-I): Upon receipt of the reply to the 4th Mode C whisper-shout sequence, an intruder track file is established with tracked altitude equal to the value stated in the Inputs section above.

2.4.2.1.6.3**Missing Mode C Replies (§2.2.4.6.4.1.3)**

This test procedure will verify that the Mode C surveillance processor correctly coasts a track when replies are missing for several consecutive surveillance update intervals and drops a coasted track when the maximum coast length is exceeded.

Test Scenarios

Two scenarios are used, each with a single threat aircraft on a horizontal collision course. In Scenario A, the track should coast for 5 surveillance update intervals after the reply data stream is terminated.

In Scenario B, the track should coast for 5 surveillance update intervals after the reply data stream is terminated.

Inputs

Intruder Aircraft, Scenario A

Initial Range	= 13 NM
Relative Speed	= -1200 kt (-0.33 NM/s)
Altitude	= 11,300 ft
Altitude Rate	= 0 FPM
Equipage	= Mode C
Reply Power	= -65 dBm
Reply Sequence	= Once per surveillance update interval beginning at T=0 with final reply at T=13 seconds.

Intruder Aircraft, Scenario B

Initial Range	= 13 NM
Relative Speed	= -900 kt (-0.25 NM/s)
Altitude	= 11,300 ft
Altitude Rate	= 0 FPM
Equipage	= Mode C
Reply Power	= -65 dBm
Reply Sequence	= Once per surveillance update interval beginning at T=0 seconds with final reply at T=20 seconds.

ACAS X-Equipped Aircraft

Sensitivity Level Selection	= Automatic
Altitude	= 11,600 ft

Conditions

ACAS X initialized (with no inputs other than those specified) at or before T=0 seconds.

Success Criteria

Scenario A: The track should coast for 5 surveillance update intervals after the reply data stream is terminated.

Scenario B: The track should coast for 5 surveillance update intervals after the reply data stream is terminated.

2.4.2.1.6.4

Surveillance Target Capacity (Mode C) (§2.2.4.6.1)

This test verifies that the Mode C surveillance processor has sufficient storage and processing capacity to provide surveillance on 30 Mode C aircraft when operating in a Mode C fruit environment of 30,000 fruit replies/second. The 30 Mode C aircraft are distributed such that 22 altitude reporting and two non-altitude reporting aircraft are within a range of five NM and an additional six altitude reporting aircraft are beyond five NM. This test also verifies that a single track is established on a target that generates more than one reply within a surveillance update interval.

Success of the test is determined by inspection of the number of tracks at various times during the test.

Inputs

ACAS X Aircraft

Sensitivity Level Selection = Automatic
 Altitude = 8,000 ft

Intruder Aircraft

The following values apply to all intruder aircraft:

Altitude Rate = 0 FPM
 Equipage = Mode C
 Reply Power = -65 dBm

All intruders **shall** (2363) be separated by at least 300 feet from each other in altitude and **shall** (2364) be within the surveillance altitude volume of ±10,000 feet relative to ownship.

Intruder Aircraft 1

Range = 4.05 NM at T=0 seconds
 Relative Speed = -120 kt (-0.033 NM/s)
 Whisper-Shout Step Number = 1 and 25

Intruder Aircraft 2

Range = 5.25 NM at T=0 seconds
 Relative Speed = -150 kt (-0.042 NM/s)
 Whisper-Shout Step Number = 2 and 25

Intruder Aircraft 3-24

Odd-numbered intruders have ranges and relative speeds identical to Intruder 1 and even-numbered intruders have ranges and relative speeds identical to Intruder 2. Intruders 23-24 are non-altitude reporting, i.e., they reply with brackets only. The applicable whisper-shout step number for reply response and the time of introduction into the scenario are as follows:

<u>Intruder</u>	<u>Whisper-Shout Step Number</u>	<u>Initial Time (seconds)</u>
3	3 and 4	10
4	5 and 6	10
5	7 and 8	20
6	9 and 10	20
7	11 and 12	30
8	13 and 14	30
9	15 and 16	40
10	17 and 18	40
11	19 and 20	50
12	21 and 22	50
13	23 and 24	60
14	25 and 27	60
15	26 and 28	70

16	29 and 31	70
17	30 and 32	80
18	33 and 35	80
19	34 and 36	90
20	37 and 39	90
21	38 and 40	100
22	41 and 43	100
23 Non-Altitude Reply	42 and 44	110
24 Non-Altitude Reply	45 and 47	110

The whisper-shout interrogation numbers relate to the step numbers illustrated in Figure 2-10, High-Density Whisper-Shout Sequence. The intent is to generate replies from the same target on two adjacent whisper-shout steps of a sequence within the same quadrant.

Intruder Aircraft 25-30

Range	= distributed uniformly in range between 6 NM and 16 NM so as to maintain a 2 NM spacing between intruders
Relative Speed	= 0 kt

Mode C Fruit Replies

Average Fruit Rate	= 30,000 replies/second
Altitude Codes	= Selected randomly from all possible valid codes.

Reply Power Level Distribution

Individual fruit replies **shall** (2365) be generated at the rates and power levels indicated below and interspersed to create an average fruit rate of 30,000/ second.

<u>Rate</u>	<u>Level</u>
15,000/s	-71 dBm
7,500/s	-68 dBm
7,500/s	-65 dBm

The timing of fruit replies exceeding -71 dBm **shall** (2366) be adjusted to prevent overlap or garble of any other signals generated by the test unit.

Conditions

ACAS X initialized and operating at T=0 seconds. Each intruder **shall** (2367) reply only to its assigned step number(s) except that Intruders 25-30 can reply to any step number.

Scenario

The scenario consists of 30 Mode C aircraft at various closing rates and relative altitudes. The first 24 targets have motion relative to ACAS X and are generated by repeating Intruders 1 through 2 at 10-second intervals between T=0 seconds and T=110 seconds. Each of the first 24 intruders **shall** (2368) respond to a different step number in the Mode C whisper-shout sequence to prevent garbling replies. Figure 2-40 illustrates the range-time relationship of the first and last of the 12 groups of the first 24 intruders. The first group (Intruders 1-2) and Intruders 25-30 are introduced at T=0 seconds and continue for the duration of the scenario. The last group (Intruders 23-24) is introduced at T=110 seconds and continues for the duration of the scenario. The remaining 10 intermediate groups are

introduced at T=10 seconds, 20 seconds and 100 seconds respectively and continue for the duration of the scenario. Between T=120 seconds and 240 seconds, the scenario contains 24 Mode C targets within five NM of ACAS X and an additional 6 Mode C targets between 6 and 16 NM of ACAS X. In addition a Mode C environment of 0.3 aircraft/NM² within 30 NM of ACAS X is simulated by the generation of Mode C fruit at a rate of 30,000 fruit replies/second. Since the presence of fruit is intended only to verify the processing capacity of ACAS X, the individual fruit reply times and altitude codes are selected to prevent interference and generation of false tracks. Surveillance track file data (§2.4.1i) **shall** (2369) be recorded to provide information on all tracks at T=2 minutes, three minutes and four minutes.

Success: The track file information indicates the correct number of tracks. Scenario terminated at T = five minutes.



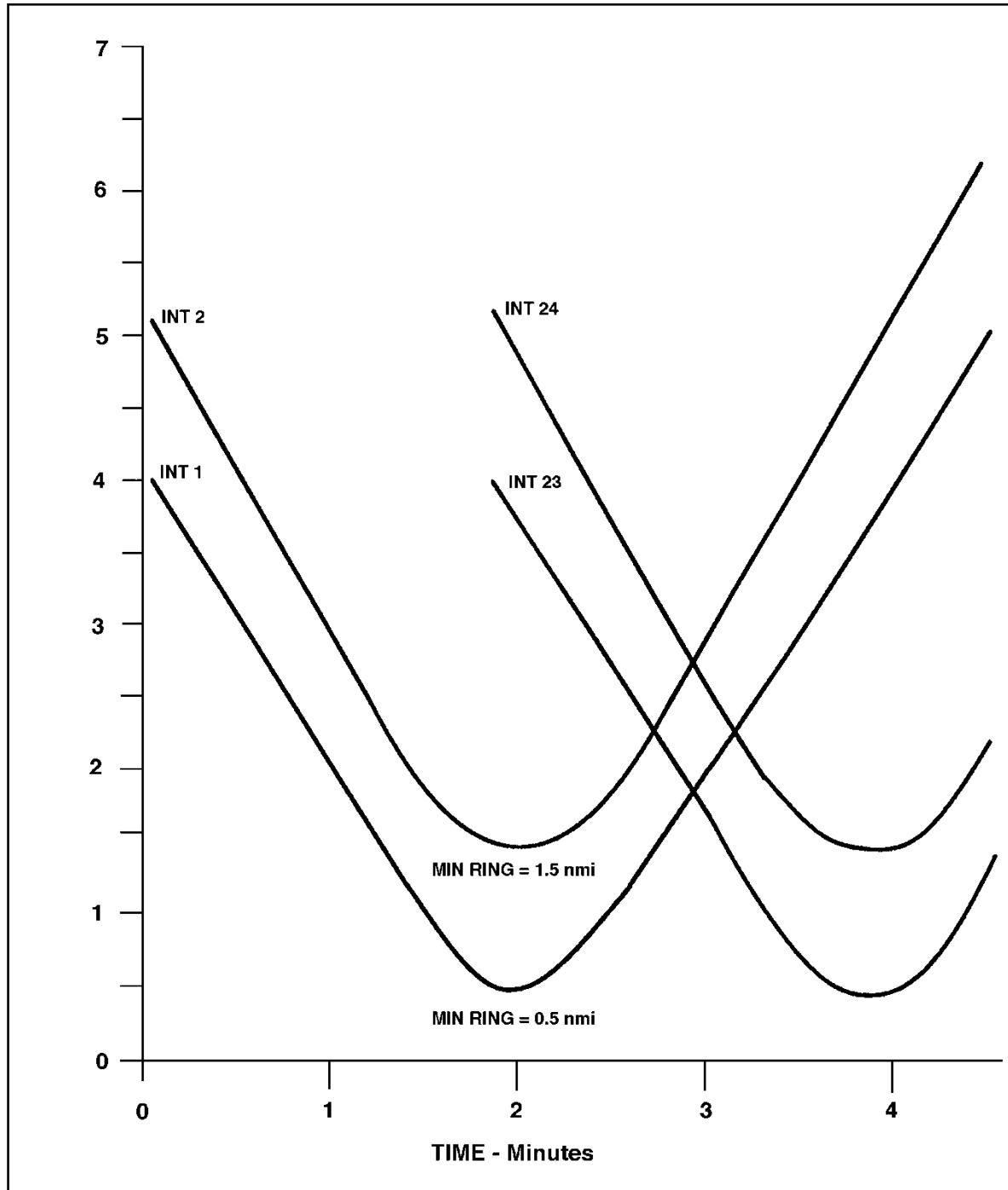


Figure 2-40: Range Vs. Time Plot For Intruders 1, 2, 23, And 24

2.4.2.1.6.5 Surveillance Overload (§2.2.4.6.1.1)

The purpose of this test is to verify that, when the track capacity of §2.2.4.6.1 is exceeded, the Mode C surveillance processor correctly drops tracks according to the criteria of §2.2.4.6.1.1. That is, targets **shall** (2370) be deleted in decreasing range order.

Inputs

As described in §2.4.2.1.6.4 and supplemented as follows:

Intruder Aircraft 31

Range	=	6 NM at T=0 seconds
Relative Speed	=	-120 kt (-0.033 NM/s)
Whisper-Shout Step Number	=	48 and 49

Intruder Aircraft 32-34

Intruders 32-34 have ranges and relative speeds identical to Intruders 1, 2 and 31 respectively. The applicable whisper-shout step number for reply response and the time of introduction into the scenario are identical to Intruders 13, 14, and 15 respectively.

All intruders **shall** (2371) be separated by at least 300 feet from each other in altitude and **shall** (2372) be within the surveillance altitude volume of $\pm 10,000$ feet relative to ownship.

Conditions

As described in §2.4.2.1.6.4.

Scenario:

As described in §2.4.2.1.6.4 except that four additional intruders are present.

Success: The track file information indicates that the four intruders furthest in range have been deleted from the track file.

2.4.2.1.7**Mode S Target Surveillance Performance (§2.2.4.6.4.2, §2.2.4.4.2.2)**

The following tests are designed to verify the proper operation of the ACAS X Mode S surveillance processor.

Generalized Scenario Description

The test scenarios used to verify proper operation of the Mode S surveillance processor contain simulated Mode S and Mode C targets. Many of the position and rate values associated with the ACAS X unit under test and the Mode S message field contents associated with the simulated targets can be represented by a single standard input description. The following set of inputs is applicable for each of the tests in §2.4.2.1.7. Additional inputs and conditions relating to each test are specified as required.

InputsACAS X Aircraft

Altitude Rate	=	0 FPM
---------------	---	-------

Intruder Aircraft

Squitter/Reply Frequency	=	1090 MHz
Squitter/Reply Power	=	-50 dBm

Mode S Squitter Format

DF	=	11
CA	=	0
M	=	Any valid ICAO 24-bit Aircraft Address

Mode S Short Special Surveillance Format

DF	=	0
----	---	---

VS	=	0
AP	=	Any valid ICAO 24-bit Aircraft Address

Mode S Long Special Surveillance Format

DF	=	16
VS	=	0
MV		
VDS1	=	3
VDS2	=	0
Remainder of Field	=	0s
AP	=	Any valid ICAO 24-bit Aircraft Address

Mode S Fruit Reply (to ground interrogation)

DF	=	4
FS	=	0
DR	=	0
UM	=	0
AP	=	Any valid ICAO 24-bit Aircraft Address

Mode S Fruit Reply (to other TCAS interrogation)

DF	=	0
VS	=	0
RI	=	0
AP	=	Any valid ICAO 24-bit Aircraft Address

2.4.2.1.7.1 Mode S Surveillance Initiation (§2.2.4.6.2.2.2, §2.2.4.6.4.2.1)

Test 1 Mode S Surveillance Initiation

This test will verify that ACAS X is designed to:

- a. Make use of the parity coding in squitter transmissions to detect and correct errors in the received message.
- b. Employ a squitter processing technique to minimize the number of interrogations necessary for target acquisition.
- c. Initiate acquisition of targets that are within an altitude of $\pm 10,000$ ft relative to ACAS X.
- d. Ensure timely acquisition of targets that transition the $\pm 10,000$ ft relative altitude boundary.
- e. Prevent acquisition interrogations to targets with an RTCA/DO-181A or later Mode S transponder whose squitter CA field indicates on the ground.
- f. Reject squitters that contain an all 0s or all 1s ICAO 24-bit Aircraft Address or an address identical to that of ownship transponder.

Inputs

As described in §2.4.2.1.7 and supplemented as follows:

ACAS X Aircraft

Altitude	=	8000 ft
Sensitivity Level Selection	=	Automatic

Intruder Aircraft 1

Equipage	=	Mode S
Altitude	=	8000 ft
Altitude Rate	=	0 FPM
Range	=	6 NM at T=0 seconds
Relative Speed	=	-360 kt (-0.1 NM/s)

Intruder Aircraft 2

Equipage	=	Mode S
Altitude	=	8000 ft
Altitude Rate	=	0 FPM
Range	=	6 NM at T=0 seconds
Relative Speed	=	-360 kt (-0.1 NM/s)

Data Field Error Simulation:

The data field content of the squitter **shall** (2373) be modified following generation of the parity bit sequence and prior to PPM modulation in the following manner:

- a. Data bit number 32 **shall** (2374) contain energy throughout both halves (chips) of its bit position.
- b. The amplitude of the signal in the half that would ordinarily contain energy **shall** (2375) be three dB below the amplitude of the signal in the other half of the bit position.

Intruder Aircraft 3

Equipage	=	Mode S
Altitude	=	17,800 ft
Altitude Rate	=	0 FPM
Range	=	6 NM at T=0 seconds
Relative Speed	=	-360 kt (-0.1 NM/s)

Intruder Aircraft 4

Equipage	=	Mode S
Initial Altitude	=	20,000 ft at T=0 seconds
Altitude Rate	=	-9,000 FPM
Range	=	6 NM at T=0 seconds
Relative Speed	=	-360 kt (-0.1 NM/s)

Intruder Aircraft 5

Equipage	=	Mode S (RTCA/DO-181A or later transponder)
CA	=	4 at T=0 seconds
	=	5 at T=10 seconds

Intruder Aircraft 6

Equipage	= Mode S
Altitude	= 8000 ft
Altitude Rate	= 0 FPM
Range	= 6 NM at T=0 seconds
Relative Speed	= -360 kt (-0.1 NM/s)
Mode S Squitter Format	
AA	= all 0s

Intruder Aircraft 7

Equipage	= Mode S
Altitude	= 8000 ft
Altitude Rate	= 0 FPM
Range	= 6 NM at T=0 seconds
Relative Speed	= -360 kt (-0.1 NM/s)
Mode S Squitter Format	
AA	= all 1s

Intruder Aircraft 8

Equipage	= Mode S
Altitude	= 8000 ft
Altitude Rate	= 0 FPM
Range	= 6 NM at T=0 seconds
Relative Speed	= -360 kt (-0.1 NM/s)
Mode S Squitter Format	
AA	= ownship ICAO 24-bit Aircraft Address

Conditions

ACAS X initialized and operating at T=0 seconds.

Intruder 1 **shall** (2376) transmit a single squitter during the squitter listening period following completion of the 1st, 23rd and 24th whisper-shout sequences and then remain silent.

Intruder 2 **shall** (2377) transmit a squitter with an error-modified data field during the squitter listening period following completion of each of the first three whisper-shout sequences and then remain silent.

Intruder 3 **shall** (2378) transmit a squitter during the squitter listening period following completion of the 1st and 3rd whisper-shout sequences and a fruit reply (DF=4) following completion of the 2nd whisper-shout sequence and then remain silent.

Intruder 4 **shall** (2379) transmit a squitter following completion of each whisper-shout sequence.

Intruder 5 **shall** (2380) transmit a squitter following completion of each whisper-shout sequence.

Intruders 6, 7 and 8 **shall** (2381) transmit a squitter during the squitter listening period following completion of each of the first three whisper-shout sequences and then remain silent.

Scenario Description

Intruder 1 is within the altitude protection volume of ACAS X and transmits the 2nd squitter sufficiently delayed from the 1st squitter to cause ACAS X to terminate squitter processing for that target.

Success: ACAS X transmits no acquisition interrogations to Intruder 1.

Intruder 2 is within the ACAS X protection volume and has an error in the message field.

Success: ACAS X transmits an acquisition interrogation addressed to Intruder 2 within one second following receipt of the third squitter.

Intruder 3 is within the altitude protection volume of ACAS X.

Success: ACAS X transmits an acquisition interrogation to Intruder 3 within 1 second following receipt of the 2nd squitter.

Intruder 4 is initially outside of the altitude protection volume of ACAS X but has an altitude rate that will cause it to be within the altitude protection volume of ACAS X at T=14 seconds.

Success: ACAS X transmits an acquisition interrogation addressed to Intruder 4 for track establishment no later than T=16 seconds.

Intruder 5 is within the protection volume of ACAS X and initially on the ground as indicated by the squitter CA field. At T=10 seconds the CA field indicates Intruder 5 to be airborne.

Success: ACAS X transmits an acquisition interrogation to Intruder 5 no later than T=11 seconds.

Intruders 6, 7 and 8 are within the ACAS X protection volume and have unacceptable ICAO 24-bit Aircraft Addresses.

Success: ACAS X transmits no acquisition interrogations addressed to Intruders 6, 7 and 8.

Test 2 Use of Extended Squitter Altitude

This test verifies that the ACAS X correctly uses the DF=17 encoded altitude for the purposes of monitoring altitude.

Scenario Description

- Intruder 1 verifies §2.2.4.6.4.2.1 by having an intruder which squitters DF=17 Airborne Position Messages, not qualifying for Extended Hybrid Surveillance, but with valid altitude just within the 10,000 ft ACAS X surveillance volume.
- Intruder 2 verifies §2.2.4.6.4.2.1 by having an intruder which squitters DF=17 Airborne Position Messages, not qualifying for Extended Hybrid Surveillance, but with valid altitude just outside the 10,000 ft ACAS X surveillance volume.

ACAS X Aircraft

Altitude	= 12,000 ft
Altitude Rate	= 0 FPM
Position	= London

Intruder Aircraft #1

DF=11 transmitted at T=3 and 4
Altitude = 21,900 ft
Altitude Rate = 0 FPM
Range = 2.9 NM at T=0 sec
Relative Speed = 0

DF=17 extended squitters contain invalid position data (format type code is set to 0) or with a reported NACp value < 7, not qualifying for Extended Hybrid Surveillance.

Intruder Aircraft #2

DF=11 transmitted at T=3 and 4
Altitude = 22,100 ft
Altitude Rate = 0 FPM
Range = 2.9 NM at T=0 sec
Relative Speed = 0

DF=17 extended squitters contain invalid position data (format type code is set to 0) or with a reported NACp value < 7, not qualifying for Extended Hybrid Surveillance.

Success Criteria**Intruder 1**

The intruder is placed into track by T=8 sec because it is within the altitude surveillance volume.

Intruder 2

The intruder is NOT placed into track. No UF=0 interrogations to this intruder are made by ACAS X because the intruder is outside the ACAS X altitude surveillance volume.

2.4.2.1.7.2**Mode S Range Acquisition (§2.2.4.6.2.2.2, §2.2.4.6.4.2.2.1, §2.2.4.2, §2.2.4.7, §2.2.4.7.4.1)**

This test verifies that the ACAS X unit is able to:

- a. Acquire and establish track on a high closing speed intruder prior to its penetration into the ACAS X threat volume.
- b. Adjust its re-interrogation rate to a potential intruder following unsuccessful acquisition in order to minimize unnecessary interrogations.
- c. Periodically monitor the VS field in replies from an intruder that indicates on the ground in order to provide timely acquisition when the intruder indicates airborne.
- d. Use reliable Mode S reply altitude data during acquisition to establish a track.

Inputs

As described in §2.4.2.1.7 and supplemented as follows:

ACAS X Aircraft

Altitude = 15,300 ft
 Sensitivity Level Selection = Automatic

Intruder Aircraft 1

Equipage = Mode S
 Altitude = 15,300 ft
 Altitude Rate = 0 FPM
 Range = 13 NM at T=0 seconds
 Relative Speed = -1,200 kt (-0.333 NM/s)
 Data Field
 RI = 13 in response to AQ=1
 = 0 in response to AQ=0

Intruder Aircraft 2

Equipage = Mode S
 Altitude = 12,000 ft
 Altitude Rate = 0 FPM
 Range = 13 NM at T=0 seconds
 Relative Speed = -180 kt (-0.05 NM/s)

Intruder Aircraft 3

Equipage = Mode S
 Altitude = 6000 ft at T=0 seconds
 Altitude Rate = 0 FPM at T=0 seconds
 = +1000 FPM at T=20 seconds
 Range = 7 NM at T=0 seconds
 Relative Speed = -400 kt (-0.111 NM/s) at T=0 seconds
 = -600 kt (-0.167 NM/s) at T=20 seconds
 Data Field
 RI = 13 in response to AQ=1
 = 0 in response to AQ=0
 VS = 1 at T=0 seconds
 = 0 at T=20 seconds

Intruder Aircraft 4

Equipage = Mode S
 Altitude
 1st reply = 15,900 ft with Q-bit = 1
 2nd reply = 15,300 ft with Q-bit = 0
 Remaining replies = 15,300 ft with Q-bit = 1
 Altitude Rate = 0 FPM
 Range = 13 NM at T=0 seconds
 Relative Speed = -180 kt (-0.05 NM/s)

Conditions

ACAS X initialized and operating at T=0 seconds. Intruder 1, 3 and 4 shall (2382) transmit a squitter during the squitter listening period following the completion of each whisper-shout sequence and shall (2383) reply with a short special surveillance format (DF=0) to each ACAS X discrete interrogation addressed to it.

Intruder 2 **shall** (2384) not reply to interrogations. Intruder 2 **shall** (2385) transmit a squitter during the squitter listening period following the completion of each of the following whisper-shout sequences:

- 1st through 3rd whisper-shout sequence
- 13th through 16th whisper-shout sequence
- 26th through 31st whisper-shout sequence
- 41st through 51st whisper-shout sequence
- 61st through 71st whisper-shout sequence

Scenario Description

The scenario **shall** (2386) be run with the test unit connected to both the top and bottom antenna RF ports of the ACAS X equipment.

Intruder 1 is within the altitude protection volume of ACAS X, transmits squitters at a time when ACAS X is in squitter listening mode and replies to discrete interrogations from ACAS X.

Success: ACAS X transmits an acquisition interrogation addressed to the ICAO 24-bit Aircraft Address contained in the AP field of the reply by Intruder 1 within one second following the third squitter. ACAS X transmits a tracking interrogation addressed to Intruder 1 within two seconds following the third squitter and a track is established on the intruder after the reply to the tracking interrogation is received.

Intruder 2 is within the altitude protection volume of the ACAS X, transmits squitters often enough to cause repeated ACAS X interrogation attempts but does not respond to interrogations.

Success: During the 6-update-interval period following the 3rd squitter, ACAS X transmits a maximum of nine acquisition interrogations total to Intruder 2 with no more than three acquisition interrogations in any single update interval. During each of the 6 update-interval periods following the 16th and 31st squitters, ACAS X transmits a maximum of six acquisition interrogations total to Intruder 2 with no more than one acquisition interrogation in any single update interval. During each of the 6 update-interval periods following the 51st and 71st squitters, ACAS X transmits a maximum of one acquisition interrogation to Intruder 2. (Scenario terminated at T=80 seconds).

Intruder 3 is within the altitude protection volume of ACAS X, initially on the ground, transmits squitters at a time when ACAS X is in squitter listening mode and replies to discrete interrogations from ACAS X. At T=20 seconds, Intruder 3 becomes airborne.

Success: ACAS X transmits an acquisition interrogation addressed to Intruder 3 within one second following the third squitter. ACAS X continues to monitor the vertical status of Intruder 3 every 5 seconds until a reply indicates Intruder 3 airborne. ACAS X transmits a tracking interrogation to Intruder 3 for track establishment within two seconds of receipt of the first reply following T=20 seconds in which the VS field indicates Intruder 3 airborne.

Intruder 4 is within the altitude protection volume of ACAS X, transmits squitters at a time when ACAS X is in squitter listening mode and replies to discrete interrogations from ACAS X.

Success: ACAS X transmits an acquisition interrogation addressed to the ICAO 24-bit Aircraft Address contained in the AP field of the reply from Intruder 4 within one second

following the third squitter. ACAS X transmits three additional interrogations to Intruder 4 before track is established with an Intruder 4 altitude of 15,300 ft.

2.4.2.1.7.3 Maintenance of Established Mode S Tracks (§2.2.4.6.4.2.3.2.5, §2.2.4.7, §2.2.5.5.6)

This test will verify the ability of the Mode S surveillance processor to:

- a. Make a reasonable effort to update an established track file for a target having poor link reliability.
- b. Properly extrapolate an established track file in the absence of updating replies.
- c. Terminate an established track file after a period of unsuccessful updates.
- d. Select the appropriate interrogation rate of the Mode S intruder.
- e. Maintain a Mode S track on an airborne intruder through momentary indications of an on ground condition.
- f. Filter momentary spikes and jumps in the intruder altitude data.

Inputs

As described in §2.4.2.1.7 and supplemented as follows:

ACAS X Aircraft

Altitude	= 12,000 ft
Sensitivity Level Selection	= Automatic

Intruder Aircraft 1

Equipage	= Mode S
Altitude	= 10,500 ft at T=0 seconds
Altitude Rate	= +3000 FPM
Range	= 9 NM at T=0 seconds
Relative Speed	= -720 kt (-0.2 NM/s)
Data Field	
RI	= 13 in response to AQ=1 = 0 in response to AQ=0

Intruder Aircraft 2 - 3

Equipage	= Mode S
Altitude	= 12,000 ft
Altitude Rate	= 0 FPM
Range	
Int 2	= 16 NM at T=0 seconds
Int 3	= 11 NM at T=0 seconds
Relative Speed	
Int 2	= -720 kt (-0.2 NM/s)
Int 3	= +720 kt (+0.2 NM/s)
Data field	
RI	= 13 in response to AQ=1 = 0 in response to AQ=0

Intruder Aircraft 4

Equipage	= Mode S
Altitude	= 12,000 ft

Altitude Rate	= 0 FPM
Range	= 9 NM at T=0 seconds
Relative Speed	= -720 kt (-0.2 NM/s)
Data Field	
RI	= 13 in response to AQ=1 = 0 in response to AQ=0
VS	= 0 at T=0 seconds = 1 for 2 seconds starting at T=7 seconds = 1 for 5 seconds starting at T=15 seconds

Intruder Aircraft 5

Equipage	= Mode S
Altitude	= 12,000 ft from T=0 to T=10 seconds
	= 12,600 ft from T=11 seconds on
Altitude Rate	= 0 FPM
Range	= 9 NM at T=0 seconds
Relative Speed	= -720 kt (-0.2 NM/s)
Data Field	
RI	= 13 in response to AQ=1 = 0 in response to AQ=0

Conditions

ACAS X initialized and operating at T=0 seconds. Intruder 1 **shall** (2392) transmit a squitter during the squitter listening period following the completion of each whisper-shout sequence and **shall** (2393) reply with a short special surveillance format (DF=0) to each ACAS X interrogation addressed to it that occurs prior to the 11th whisper-shout sequence and then remain silent. Intruder 2, 3, 4 and 5 **shall** (2394) transmit a squitter during the squitter listening period in each surveillance update interval and **shall** (2395) reply with a short special surveillance format (DF=0) to each ACAS X interrogation addressed to it except that Intruder 3 **shall** (2396) not reply to interrogations beginning at T=17 seconds.

Scenario Description for Intruder 1

The intruder is within the altitude protection volume of ACAS X and transmits squitters at a time when ACAS X is in the squitter listening mode. Tracking surveillance is established on the intruder within two seconds following the third squitter as noted by the occurrence of a ACAS X tracking interrogation. At T=10 seconds (following the 10th whisper-shout sequence) the intruder stops replying.

Success: ACAS X transmits a minimum of two tracking interrogations per surveillance update interval addressed to the intruder following the last reply from the intruder. ACAS X transmits a maximum of five interrogations per surveillance update interval and 16 interrogations in a 6-second period following the last reply. Transmission of interrogation pairs containing two successive tracking interrogations alternate between the top and bottom antenna ports beginning with the port that last received a reply. (Scenario is terminated at T=31 seconds).

Scenario Description for Intruder 2

Intruder 2 is initially outside of the 60-second TAU boundary of ACAS X, closing in range and transmits squitters when ACAS X is in the squitter listening mode.

Success: Tracking surveillance at a rate of one interrogation every 5 seconds is established on the intruder no later than five seconds after receipt of the third squitter. The intruder

crosses the 60-second TAU boundary between T=15 and T=18 seconds at which time ACAS X switches to an interrogation rate of one interrogation every second. (Scenario is terminated at T=25 seconds).

Scenario Description for Intruder 3

Intruder 3 has an increasing range and transmits squitters when ACAS X is in the squitter listening mode and stops replying to ACAS X at T=17 seconds.

Success: Tracking surveillance at a rate of one interrogation every 5 seconds is established on the intruder no later than two seconds following the third squitter.

Following the second interrogation at the once per 5-second rate to which no reply is received, ACAS X transmits a minimum of two tracking interrogations per surveillance update interval addressed to the intruder. ACAS X transmits a maximum of five interrogations per surveillance update interval and 16 interrogations in a 6-second period following the second interrogation before the intruder track is dropped. The total number of coasted surveillance update periods is nine. Transmission of interrogation pairs containing two successive tracking interrogations alternate between the top and bottom antenna ports beginning with the port that last received the reply. (Scenario is terminated at T=30 seconds).

Scenario Description for Intruder 4

Intruder 4 is within the threat volume of ACAS X and transmits squitters at a time when ACAS X is in a squitter listening mode. The intruder indicates a two-scan on-the-ground condition via the VS field at T=7 seconds and a seven-scan on-the-ground condition at T=15 seconds.

Success: Tracking surveillance is established on Intruder 4 within two seconds following the third squitter as noted by the occurrence of ACAS X tracking interrogations. A continuous track is maintained on Intruder 4 until track is dropped between T=19 and T=22 seconds. Verify that replies indicating through the VS field that the aircraft is not on the ground are not passed to the STM function ReceiveDF0. (Scenario is terminated at T=25 seconds).

Scenario Description For Intruder 5

Intruder 5 is within the threat volume of ACAS X and transmits squitters at a time when ACAS X is in a squitter listening mode. Beginning at T=11 seconds, Intruder 5 altitude value changes by 600 ft.

Success: Tracking surveillance is established on Intruder 5 within two seconds following the third squitter as noted by the occurrence of ACAS X tracking interrogations. A continuous track is maintained on Intruder 5 until track is dropped between T=15 and T=18 seconds followed by the reestablishment of a track on Intruder 5 at the new altitude. (Scenario is terminated at T=25 seconds).

Note: This test ensures that DF=0 replies that do not meet the requirements of §2.2.4.6.4.2.3.2.5 for the maintenance of established tracks are not provided to the STM, as per the requirements of §2.2.5.5.6.

2.4.2.1.7.4 Interference Limiting (§2.2.3.6)

2.4.2.1.7.4.1 Interrogation Control of Airborne ACAS X (§2.2.3.6.1, §2.2.3.6.2, §2.2.3.6.4)

This test verifies that the equipment is able to determine both the number of Active CAS aircraft within its nominal detection range and the distribution of Active CAS aircraft within 6 NM range and, based on this information, adjust its sensitivity, interrogation rate

and power level to minimize interference to other interrogators. It will also verify that ACAS X is not subject to the 3 inequalities of §2.2.3.6.1, and uses the relaxed inequalities 4 and 5 of §2.2.3.6.4, if it is above 18,000 ft altitude.

Inputs

As described in §2.4.2.1.7 and supplemented as follows:

ACAS X Aircraft

Altitude	
Scenario A&B	= 8,000 ft
Scenario C	= 19,000 ft
Sensitivity Level	
Selection	= Automatic

Intruder Aircraft 1 Through 18

Equipage	= TCAS II
Altitude	= Uniformly distributed within ±3000 ft of ACAS X aircraft
Altitude Rate	= 0 FPM
Range	
Scenario A	= uniformly distributed in range between 0 and 12 NM such that the number of TCAS II aircraft vs. range is equal to 1.5 times the range in NM.
Scenario B&C	= uniformly distributed in area between 6 and 12 NM such that the number of TCAS II aircraft vs. range is equal to 0.125 times the square of the range in NM.
Relative Speed	= 0 kt
Data Field	
RI	= 3 in response to AQ=0
TCAS Broadcast	
Interrogation Power	= -50 dBm

Intruder Aircraft 19

Equipage	= TCAS II
Altitude	= 8,000 ft
Altitude Rate	= 0 FPM
Range	= 9.55 NM
Relative Speed	= -180 kt (-0.05 NM/s)
Data Field	
RI	= 3 in response to AQ=0
TCAS Broadcast	
Interrogation Power	= -50 dBm

Intruder Aircraft 20 through 38

Equipage	= TCAS II
Altitude	= Uniformly distributed within ± 3000 feet of ACAS X aircraft
Altitude Rate	= 0 FPM
Range	= Uniformly distributed between 14 NM and 30 NM
Relative Speed	= 0 kt
Data Field	
RI	= 3 in response to AQ=0
TCAS Broadcast	
Interrogation Power	= -50 dBm

Intruder Aircraft 39 through 46

Equipage	= Mode C
Altitude	= Uniformly distributed within ± 3000 ft of ACAS X
Altitude Rate	= 0 FPM
Range and Azimuth	
Int 39 and 40	= 5 & 6 NM respectively and in forward azimuth beam
Int 41 and 42	= 3 & 3.5 NM respectively and in left azimuth beam
Int 43 and 44	= 3 & 3.5 NM respectively and in right azimuth beam
Int 45 and 46	= 1.2 & 1.5 NM respectively and in rear azimuth beam
Relative Speed	= 0 kt
Reply Power	= -50 dBm

Conditions

For each scenario, ACAS X initialized and operating at T=0 seconds. Beginning at T=0 seconds, Intruders 1 through 19 shall (2397) transmit a TCAS Broadcast Interrogation once every 10 seconds, a squitter during the squitter listening period following the completion of each whisper-shout sequence, and shall (2398) reply with a short special surveillance format (DF=0) to each discrete ACAS X interrogation addressed to it. Intruders 20 through 38 are present between T=100 seconds and T=200 seconds and are required to transmit only squitters and TCAS Broadcast Interrogations as above. Intruders 39 through 46 are present during the entire time and are required to respond each scan to a whisper-shout step.

Scenario Description

The following scenario description is applicable to Scenario A, B and C, except that Intruder 19 is not present during Scenario C. For convenience in running the test all intruder aircraft except Intruder 19 are specified to be stationary with respect to the ACAS X equipment. Intruders 1-19 are present throughout the entire scenario and will result in a certain amount of ACAS X interrogation power-rate limiting. For Scenario B Intruder 19 will be within 6 NM at T=71 causing α to drop below 1. Intruders 20-38 are generated during the interval between T=100 seconds and T=200 seconds. The introduction of Intruders 20-38 should result in an increased count of the number of TCAS aircraft. This in turn should result in a further reduction in the ACAS X interrogation power-rate

parameters. Intruders 39 through 46 are positioned to cause ACAS X to select the highest level whisper-shout sequence in each beam and therefore to transmit the maximum number of Mode C interrogations each scan.

Intruder 19 is moving relative to ACAS X and will penetrate the threat boundary at T=162 seconds when all 38 intruders are present. Since Intruder 19 is TCAS-equipped, it will cause the ACAS X under test to generate air-to-air coordination interrogations between T=163 and T=196 seconds. These interrogations are not subject to interference limiting constraints and are generated at full power.

The removal of Intruders 20-38 should result in the recovery of the interrogation limits to their original values. At T=60, 160 and 260 seconds the value of the left-hand side of inequality 3 of §2.2.3.6.1 **shall** (2399) be computed by measuring the transmitted power level of each Mode C interrogation in a whisper-shout sequence.

The effects of the 8-second freeze are apparent as the inequalities fail when 18 new Intruders are introduced and ACAS X has not yet reduced the power enough to account for the unexpected change. The interference limiting parameters should be nearly stable by the end of each 100 second interval. The effects of the 5 second update and the 6 second retry on acquisitions to Intruders 20-38 may cause inequality 3 to be exceeded when they line up during the 8-second freeze.

Success for Scenario A&B:

$$T=60 \text{ and } 260 \text{ sec}, \sum_{k=1}^K \left[\frac{PA(k)}{250 \text{ watts}} \right] \leq 3$$

$$T=162 \text{ seconds}, \sum_{k=1}^K \left[\frac{PA(k)}{250 \text{ watts}} \right] \leq 2.05$$

$$T=92 \text{ and } 292 \text{ sec}, \sum_{i=1}^I \left[\frac{P(i)}{250 \text{ watts}} \right]^\alpha \leq \begin{array}{l} 14 \text{ for Scenario A} \\ \text{or} \\ 11 \text{ for Scenario B} \end{array}$$

$$T=192 \text{ sec}, \sum_{i=1}^I \left[\frac{P(i)}{250 \text{ watts}} \right]^\alpha \leq 7.18 \text{ for Scenario A \& B}$$

$$T=92, 192, \text{ and } 292 \text{ sec}, \sum_{i=1}^I \left[\frac{P(i)}{250 \text{ watts}} \right]^\alpha \leq \text{the smaller of } \left[\frac{280}{NTA+1}, \frac{11}{\alpha^2} \right]$$

The air-to-air coordination interrogations to Intruder 19 between T=163 and T=196 seconds **shall** (2400) be transmitted at full power. Scenario is terminated at T=300 seconds.

Success for Scenario C

Because of its altitude, ACAS X is not subject to interference limiting inequalities 1, 2, and 3, but is still subject to inequalities 4 and 5.

Scenario is terminated at T=300 seconds.

$$T=92, 192, \text{ and } 292 \text{ sec}, \sum_{i=1}^I \left[\frac{P(i)}{250 \text{ watts}} \right] \leq [11]$$

$$T = 60, 162, \text{ and } 260 \text{ sec}, \quad \frac{1}{B} \sum_{k=1}^K \left[\frac{PA(k)}{250 \text{ watts}} \right] \leq [3]$$

2.4.2.1.7.4.2 Interrogation Control of ACAS X On The Ground (§2.2.3.6.3)

This test verifies that the NTA value is automatically set to three times the measured value whenever ownship indicates on the ground and that ACAS X quickly recovers its nominal airborne mode surveillance range when transitioning from an on-ground status to airborne status.

Inputs

25 TCAS aircraft are present within 10 to 30 NM of own ACAS X.

$$\begin{aligned} \text{Air/ground status} &= \text{airborne at } T=0 \text{ seconds and } 100 \text{ seconds} \\ &= \text{on-the-ground at } T=20 \text{ seconds.} \end{aligned}$$

Conditions

ACAS X initialized and operating at T=0 seconds. Beginning at T=0 seconds ACAS X indicates airborne. Beginning at T=20 seconds ACAS X indicates on the ground. Beginning at T=100 seconds ACAS X indicates airborne.

Scenario Description

The scenario **shall** (2401) be run with the test unit connected simultaneously to the top and bottom antenna ports of the ACAS X equipment. Between T=0 seconds and T=20 seconds when ACAS X is airborne and 25 other TCAS intruders are present, interference limiting is not invoked and ACAS X is transmitting full power. At T=19 seconds, ACAS X transmit power is as follows:

Success:

$$\begin{aligned} \sum_{i=1}^I \left[\frac{P(i)}{250 \text{ watts}} \right]^\alpha &\leq 280/26 \\ \sum_{k=1}^K \left[\frac{PA(k)}{250 \text{ watts}} \right] &\leq 80/26 \end{aligned}$$

At T=20 seconds, ACAS X is on the ground, the NTA count is set to 75. At T=80 seconds, ACAS X transmit power is reduced as follows:

Success:

$$\begin{aligned} \sum_{i=1}^I \left[\frac{P(i)}{250 \text{ watts}} \right]^\alpha &\leq 280/76 \\ \sum_{k=1}^K \left[\frac{PA(k)}{250 \text{ watts}} \right] &\leq 80/76 \end{aligned}$$

At T=100 seconds, ACAS X is again airborne and is transmitting full power by T=103 seconds.

Success:

$$\sum_{i=1}^I \left[\frac{P(i)}{250 \text{ watts}} \right]^\alpha \leq 280/26$$

$$\sum_{k=1}^K \left[\frac{PA(k)}{250 \text{ watts}} \right] \leq 80/26$$

Scenario terminated at T=120 seconds.

2.4.2.1.7.4.3 Correct Content of Transmitted TCAS Broadcast Interrogation Messages (§2.2.3.8.3.2.4.2, §2.2.3.9.2.4)

This test verifies that the ACAS X unit under test transmits correctly coded TCAS Broadcast Interrogation Messages for the entire period that the ACAS X unit is powered on.

Conditions:

ACAS X initialized and operating.

Expected Output:

Msgs: UF=16, with UDS1 (bits 33-36) = 3, UDS2 (bits 37-40) = 2, MID (bits 65-88) = own ICAO 24-bit Aircraft Address,
AP address*= FFFFF1_6
* i.e., address before parity overlay added and after parity overlay removed

The UF=16 messages are transmitted such that, for any other TCAS II aircraft within 30 NM and at any azimuth, the nominal rate of own TCAS Broadcast Interrogation Messages arriving at that TCAS II is 1 per 8 to 10 seconds.

Note: Use of directional antenna for TCAS Broadcast Interrogation Messages is tested in §2.4.2.1.10.2.

2.4.2.1.7.4.4 Surveillance Special Functionality Test (§2.2.3.6.2)

This test will verify that ACAS X is designed to:

- a. Initialize interference limiting at 10dB power attenuation for Mode S surveillance and 7dB for Mode C surveillance when powered on while on the ground, §2.2.3.6.3
 - b. Limit acquisition of targets to those within an altitude of +/- 3000 feet relative to ACAS X when operating on the ground, §2.2.4.6.4.2.1 and §2.2.4.6.4.2.2
 - c. Properly interrogate Active CAS targets on the ground when ownship is airborne and at or below 2000 feet AGL and set NTA3 and NTA6 appropriately, §2.2.3.6.1 and §2.2.4.6.4.2.2

InputsACAS X Aircraft

Altitude / Radio Altitude	= 0 ft at T=0 seconds
Altitude Rate	= 0 FPM at T=0 seconds
	= 1000 FPM at T=45 seconds
	(gradual acceleration to 1000 FPM permitted)
On Ground	= ACAS X on ground at T=0 seconds
In Air	= ACAS X in-air after T=45 seconds

Intruder Aircraft 1

Equipage	= TCAS II
Altitude	= 0 ft
Altitude Rate	= 0 FPM
Range	= 0.2 NM at T=0 seconds
Relative Speed	= 0 kt at T=0 seconds
	= 200 kt at T=45 Seconds

Intruder Aircraft 2-6

Equipage	= Mode S
Altitude	= 500 - 4500 ft in 1000 ft increments
Altitude Rate	= 0 FPM
Range	= 15 NM
Squitter Power	= -68 dBm
Relative Speed	= 0 kt

Conditions

ACAS X initialized and operating at T=0 seconds.

Scenario Description

ACAS X initializes to 10dB attenuation on interrogation power and squitter reception sensitivity.

Success: ACAS X does not send acquisition interrogations to targets 2-4 until T=17 seconds and should send acquisition interrogations no later than T=33 seconds. Verify that the Mode C interrogation sequence power/steps has been reduced by 7dB at the start of the test. Verify that by T=81, the Mode C interrogation sequence power/steps has been restored to full power.

ACAS X does not interrogate targets for acquisition outside +/- 3000 ft of own altitude when on the ground.

Success: ACAS X does not send acquisition interrogations to targets 5-6 before T=45 seconds.

ACAS X properly interrogates TCAS targets on the airport surface and sets NTA3 and NTA6 appropriately.

Success: ACAS X does not send range monitoring interrogations to target 1 before T=45 seconds (because ownship is on the ground). NTA6=0 from T=0 to 44 seconds; NTA6=1 from T=45 to 152 seconds; NTA6=0 after T=153 seconds. NTA3=0 from T=0 to 44 seconds; NTA3=1 from T=45 to 98 seconds; NTA3=0 after T=99 seconds. Range monitoring interrogations to target 1 **shall** (2402) not be sent after T=165 seconds. The NTA numbers expected above can transition up to 5 seconds later than specified above.

2.4.2.1.7.4.5 Interference Limiting and Required Interrogations

This test will verify that ACAS X is designed to send interrogations as required in §2.2.4.6 while in maximum interference limiting.

Inputs and Conditions

Create an aircraft environment that will cause ACAS X to enter the maximum interference limiting, 10dB for Mode S. This shall (2410) be done by having a sufficiently high NTA count, sufficient Mode C traffic to require a high resolution whisper shout sequence in all four antenna quadrants, and multiple Mode S targets in communication range.

Scenario Description

ACAS X must be airborne for this test. ACAS X will follow the interference limiting procedures until 10dB of Mode S interrogation power and receiver sensitivity attenuation is reached. The conditions below must then be met.

Success: The power sum for the left hand side of interference limiting inequality 1 consistently exceeds the right hand side determined by NTA. All targets within communication range are interrogated and tracked as required in §2.2.4.6.

2.4.2.1.7.5 Surveillance Target Capacity and Overload (Mode S) (§2.2.4.6.1, §2.2.4.6.1.1)

This test verifies that the Mode S surveillance processor has sufficient storage and processing capacity to process a maximum of 150 squitter signals from Mode S targets within 30 NM of ACAS X while providing surveillance on 30 Mode S targets (24 within five NM of ACAS X and six beyond five NM) when operating in a traffic density of 0.3 aircraft/NM² and a Mode C fruit environment of 30,000 replies/second.

This test also verifies that the Mode S surveillance processor retains tracks according to the criteria of §2.2.4.6.1.1 when the track capacity of §2.2.4.6.1 is exceeded. That is, targets **shall** (2403) be deleted in decreasing range order.

This test also verifies that a ACAS X, operating in a Mode S capacity environment, does not degrade the reply probability of a collocated Mode S transponder more than one percent, and that collocated Mode S transponder transmissions do not interfere with ACAS X data processing.

Inputs

As specified in §2.4.2.1.7 and supplemented as follows:

ACAS X Aircraft

Altitude = 8,000 ft
 Sensitivity Level Selection = Automatic

Intruder Aircraft

The following values apply to all intruder aircraft:

Altitude Rate = 0 FPM
 Equipage = Mode S

Intruder Aircraft 1-24

Altitude	= As specified for corresponding intruders in §2.4.2.1.6.4
Range	= As specified for corresponding intruders in §2.4.2.1.6.4
Relative Speed	= As specified for corresponding intruders in §2.4.2.1.6.4
Data Field	
RI	= 12 in response to AQ=1 = 0 in response to AQ=0
Squitter/Reply Probability	= 1

Intruder Aircraft 25-150

Altitude	= As specified in Table 2-74
Range	= As specified in Table 2-74
Relative Speed	= 0 kt
Data Field	
RI	= As specified in Table 2-74 in response to AQ=1 = 0 in response to AQ=0
AP	= Each of the intruders from 25 through 150 is assigned a block of 100 ICAO 24-bit Aircraft Addresses. The first address in the block is the "correct" address for that intruder and is used whenever a valid squitter from that intruder is requested. An incorrect squitter address is taken from the remaining addresses each time an incorrect squitter is requested. The sequence begins again at the first incorrect aircraft address when the address block is exhausted.
Squitter/Reply Probability	= As specified in Table 2-74. The specified values of squitter and reply probability are achieved by generating valid squitters and replies in the following patterns (for each five surveillance update interval cycle):

Surveillance Update Interval No.

<u>Probability</u>	<u>1</u>	<u>2</u>	<u>3</u>	<u>4</u>	<u>5</u>
1.0	X	X	X	X	X
0.8	0	X	X	X	X
0.6	X	0	X	0	X
0.4	0	X	0	X	0
0.2	X	0	0	0	0
0.0	0	0	0	0	0

where "X" indicates generation of either a squitter with a correct address or a valid reply in response to an ACAS X interrogation. A "0" indicates generation of a squitter with an incorrect address or no generation of a reply in response to an ACAS X interrogation.

Table 2-74: Squitter And Reply Probability Vs. Range And Altitude

Intruder	Range NM	Altitude Rel. to ACAS X	Squitter Probability	Reply Probability	Data Field RI
25-27	6	+5000	1.0	1.0	13
28-30	6	+2000	1.0	1.0	11
31-33	7	-5000	1.0	1.0	11
34-36	7	0	1.0	1.0	10
37-39	8	+4000	1.0	1.0	11
40-42	8	-2000	1.0	1.0	11
43-45	9	-4000	1.0	1.0	12
46-48	9	+2000	1.0	1.0	11
49-51	10	+5000	1.0	1.0	11
52-54	10	+3000	1.0	1.0	10
55-57	11	0	1.0	1.0	13
58-60	11	-5000	1.0	1.0	11
61-63	12	-2000	1.0	1.0	11
64-66	12	+4000	1.0	1.0	10
67-69	13	+2000	1.0	1.0	11
70-72	13	-4000	1.0	1.0	11
73-75	14	0	1.0	1.0	12
76-78	14	+5000	1.0	1.0	11
79-81	15	-2000	1.0	1.0	11
82-84	15	-4000	1.0	1.0	10
85-87	16	-5000	1.0	1.0	13
88-90	16	+4000	1.0	0.8	11
91-93	17	0	1.0	0.8	10
94-96	17	-4000	1.0	0.8	11
97-99	18	-2000	1.0	0.8	11
100-102	18	+5000	1.0	0.8	12
103-105	19	+3000	1.0	0.8	10
106-108	19	0	1.0	0.8	13
109-111	20	+4000	1.0	0.8	11
112-114	20	-2000	1.0	0.8	11
115-117	21	-4000	1.0	0.8	10
118-120	21	+5000	1.0	0.8	11
121-123	22	0	1.0	0.8	12
124-126	22	-2000	1.0	0.8	11
127-129	23	-4000	1.0	0.8	10
130-132	24	+4000	0.8	0.6	13
133-135	25	0	0.8	0.6	10
136-138	26	+5000	0.8	0.6	11
139-141	27	-2000	0.6	0.7	11
142-144	28	-5000	0.6	0.6	12
145-147	29	+2000	0.6	0.6	11
148-150	30	0	0.6	0.6	13

Intruder	Range NM	Altitude Rel. to ACAS X	Squitter Probability	Reply Probability	Data Field RI
----------	----------	-------------------------	----------------------	-------------------	---------------

Mode S Transponder Interrogation Signal

Frequency = 1030 MHz
 Interrogation Type = Mode C
 Interrogation Rate = 75 per second
 Interrogation Power = -50 dBm

Mode C Fruit Replies

Average Rate = 30,000 replies/second.
 Altitude Codes = Selected randomly from all possible valid codes.

Fruit Reply Power Level Distribution

Individual fruit replies **shall** (2404) be generated at the rates and power levels indicated below and interspersed to create an average fruit rate of 30,000/second.

<u>Rate</u>	<u>Level</u>
15,000/s	-71 dBm
7,500/s	-68 dBm
7,500/s	-65 dBm

The timing of individual fruit replies **shall** (2405) be such as to prevent overlap or garble of any other signal generated by the test unit.

Conditions

ACAS X is initialized and operating at T=0 seconds. Each intruder **shall** (2406) transmit a squitter during the squitter listening period following the completion of each whisper-shout sequence with the specified address and **shall** (2407) reply as specified with a short special surveillance format (DF=0) to each discrete ACAS X interrogation addressed to it. In addition, the test unit **shall** (2408) cause the collocated ACAS X Mode S transponder to be interrogated with a Mode C-Only All-Call interrogation at the level and rate specified above. All squitters, replies and interrogations are input to both ACAS X antenna ports simultaneously.

Scenario

Intruder aircraft 1 through 24, which are designed to be within the five NM surveillance range of ACAS X during the scenario, represent a peak target load equivalent to a density of 0.3 aircraft/NM². All of these intruders are within the ACAS X altitude protection volume and will be subject to tracking surveillance. Intruder aircraft 25 through 150 represent the minimum squitter capacity required of ACAS X. These additional targets are stationary with respect to ACAS X and are distributed uniformly in area between a six and 30 NM range. All of the targets are within the ACAS X altitude protection volume and will cause ACAS X to generate acquisition interrogations. The squitter and reply probability associated with each of the intruders from 25 through 150 are selected to approximate a reasonable link reliability for the range of the intruder.

The ACAS X surveillance processor **shall** (2409) provide complete surveillance track file information (§2.4.1(i)) on all targets whether maintained in a squitter, acquisition, 1-second tracking, or 5-second tracking state at T=2 minutes, T=3 minutes and T=4 minutes.

Success: The track file information indicates the correct number of tracks and the collocated Mode S transponder reply probability between T=2.5 and T=3.5 minutes is equal to or greater than 0.99. Also, the track file information indicates that the Mode S overload function retains the 30 Mode S intruders closest in range.

2.4.2.1.7.6

Mode S Power Programming (§2.2.4.6.4.2.3.2.6)

This test verifies that ACAS X equipment adjusts the transmit power level of Mode S tracking interrogations to targets within 10 NM of ACAS X according to the criteria specified in §2.2.4.6.4.2.3.2.6.

Inputs and Conditions

As described in §2.4.2.1.7.4.1, Scenario A, except that Intruder aircraft 1 through 18 are Mode S-equipped and aircraft 20 through 46 are not present.

Scenario Description

As described in §2.4.2.1.7.4.1 except that Intruder aircraft 1 through 18 are not Active CAS-equipped, Intruder aircraft 1 does not reply to the first interrogation of each surveillance update interval, Intruder aircraft 20 through 46 are not present, interference limiting is not activated and the scenario is terminated at T=200 seconds. Between T=160 and T=190 seconds the transmitted power level from the ACAS X to each of Intruders 1 through 19 **shall** (2411) be measured.

Success: The transmitted power level of each of the tracking interrogations to Intruder aircraft 1 through 18 **shall** (2412) be in accordance with the power programming criteria specified in §2.2.4.6.4.2.3.2.6. The coordination interrogations to Intruder 19 **shall** (2413) be transmitted at full power. At least one retry interrogation to intruder aircraft 1 is transmitted at full power during each surveillance update interval. Scenario terminated at T=200 seconds.

2.4.2.1.8

Combined Mode S and Mode C Surveillance

2.4.2.1.8.1

Surveillance Target Capacity (§2.2.4.6.1)

This test verifies that ACAS X equipment has sufficient storage and processing capacity to process all signals received from targets within 30 NM of ACAS X and to provide surveillance on an equal mix of Mode S and Mode C targets within five NM of ACAS X when operating in a traffic density of 0.3 aircraft per NM².

This test also verifies that an ACAS X, operating in a Mode S and Mode C capacity environment, does not degrade the reply probability of a collocated Mode S transponder more than one percent, and that collocated Mode S transponder transmissions do not interfere with ACAS X data processing.

Inputs

As specified in §2.4.2.1.7 and supplemented as follows:

ACAS X Aircraft

Altitude	=	8,000 ft
Sensitivity Level Selection	=	Automatic

Intruder Aircraft

The following values apply to all intruder aircraft:

Altitude Rate	=	0 FPM
Reply/Squitter Power	=	-65 dBm

Intruder Aircraft 1-24

Altitude	=	As specified for corresponding intruders in §2.4.2.1.6.4
Range	=	As specified for corresponding intruders in §2.4.2.1.6.4
Relative Speed	=	As specified for corresponding intruders in §2.4.2.1.6.4
Equipage	=	
Odd Numbered Intruders	=	Mode C
Even Numbered Intruders	=	Mode S
Whisper-Shout Step Number for Mode C Intruders	=	As specified for corresponding intruders in §2.4.2.1.6.4
Data Field for Mode S Intruders	=	
RI	=	12 in response to AQ=1
	=	0 in response to AQ=0

Intruder Aircraft 25-150 (Even numbered intruders only)

Altitude	=	As specified in Table 2-74
Range	=	As specified in Table 2-74
Relative Speed	=	0 kt
Data Field	=	
RI	=	As specified in Table 2-74 in response to AQ=1
	=	0 in response to AQ=0
AP	=	As specified in §2.4.2.1.7.5
Squitter/Reply Probability	=	As specified in §2.4.2.1.7.5

Mode S Transponder Interrogation Signal

As specified in §2.4.2.1.7.5

Mode C Fruit Replies

As specified in §2.4.2.1.7.5

Conditions

ACAS X is initialized and operating at T=0 seconds. Each Mode S intruder **shall** (2414) transmit a squitter during the squitter listening period following the completion of each whisper shout sequence with the specified address and **shall** (2415) reply, as specified, with a short special surveillance format (DF=0) to each discrete ACAS X interrogation addressed to it. Each Mode C intruder **shall** (2416) reply only to its assigned whisper-shout step number(s). In addition, the test unit **shall** (2417) cause the Mode S transponder to be interrogated with a Mode C interrogation at the level and rate specified. All squitters, replies and interrogations are input to both ACAS X antenna ports simultaneously.

Scenario

Intruders 1 through 24 simulate an equal mixture of Mode C and Mode S targets within the five NM surveillance range of ACAS X. The quantity of targets represents a peak condition for a traffic density of 0.3 aircraft/NM². All of these targets are within the altitude protection volume and will be subject to tracking surveillance.

The remaining intruders are uniformly distributed in area between five NM and 30 NM range and represent the Mode S half of the targets in a density of 0.3 aircraft/NM². All of these targets are within the ACAS X altitude protection volume and will cause ACAS X to generate acquisition interrogations. The squitter and reply probability specified for these intruders approximates realistic values. The existence of Mode C targets within a five NM to 30 NM range is simulated by generation of Mode C fruit.

The ACAS X surveillance processor **shall** (2418) provide complete surveillance track file information (§2.4.1i) on all targets whether maintained in a squitter, acquisition, 1-second tracking, or 5-second tracking state at each of the following scenario times: T=2 minutes, 3 minutes and 4 minutes.

Success: The track file information (provided as output of the STM to the display) indicates the current number of tracks and, the collocated Mode S transponder reply probability, measured between T=2.5 minutes and T=3.5 minutes, is equal to or greater than 0.99.

2.4.2.1.8.2 Altitude and Range Tracking of Mode C and Mode S Targets for TRM (§2.2.4.6)

2.4.2.1.8.2.1 Range Tracking Accuracy

This test verifies that the range tracker is able to successfully track through transients in airspeed caused by sudden aircraft maneuvers. In the scenario, a non-closing aircraft abruptly undergoes a change in rate that makes it threatening. Proper performance of the tracker will result in a prompt RA.

Note: The scenario simulates an actual situation in which ownship and the intruder aircraft are initially flying a parallel course at 600 kt each with the intruder slightly above the ACAS X aircraft. Both aircraft then make converging 90 degree turns at a turn rate of 3.6 degrees/second.

Test Scenario

In the scenario a single intruder aircraft flies at a slightly higher altitude than ownship. The intruder is initially flying at constant range. It then is given a closing rate such that the threat volume is soon penetrated. If the tracker does not lag excessively, an advisory will be generated by the proper time. A late advisory constitutes failure.

Inputs

Intruder Aircraft

Range	= 10 NM from T=0 s to T=6 s and = $\left[10 - \frac{100}{6\pi} (1 - \cos 3.6N) \right]$ NM from T=6 s to T=31 s.
	N equals a parameter value that is incremented by one each surveillance update interval, from 0 to 25 following each intruder reply between T=6 s and T=31 s. From T=32 s the range is determined according to the air speed specified below.
Relative Speed	= 0 kt from T=0 s to T=6 s. = As implied by successive range reports for each value of N from T=6 s to T=31 s. = -1200 kt (-0.333 NM/s) from T=32 s.
Altitude	= 11,200 ft
Altitude Rate	= 0 FPM
Equipage	= Mode C
Reply Power	= -65 dBm

ACAS X-Equipped Aircraft

Sensitivity Level	
Selection	= Automatic
Altitude	= 11,000 ft

Conditions

ACAS X initialized and operating (with no inputs other than those specified) at or before T=0 seconds.

Time Sequence of Events

T = 0 seconds	Scenario initiated.
T = 20 to 40 seconds	Descend advisory displayed.
T = 40 seconds	Scenario terminated.

2.4.2.1.8.2.2 Altitude Tracking Accuracy

This test verifies that the altitude tracker successfully tracks through transients in altitude rate. The scenarios use the abrupt termination and initiation, respectively, of a closing altitude rate in an otherwise constant-altitude track (similar to the range-rate tests of §2.4.2.1.6.2.2.).

Test Scenarios

Two scenarios are used, each with a single intruder aircraft which is closing in range at a constant rate. In Scenario A, the intruder is initially below ownship and climbing at a high rate; this rate abruptly goes to zero. In Scenario B, the intruder is initially above ownship with an altitude rate of 0; this rate abruptly goes to -4000 FPM.

Inputs

Intruder Aircraft, Scenario A

Initial Range	= 10 NM
Relative Speed	= -900 kt (-0.25 NM/s)
Initial Altitude	= 11,100 ft
Final Altitude	= 11,433 ft
Altitude Rate	= +4000 FPM from T=0 to T=5 seconds
	= 0 from T=5 seconds to T=20 seconds
Equipage	= Mode C
Reply Power	= -65 dBm

Intruder Aircraft, Scenario B

Initial Range	= 10 NM
Relative Speed	= -900 kt (-0.25 NM/s)
Initial Altitude	= 13,200 ft
Altitude Rate	= 0 FPM from T=0 to T=10 seconds
	= -4,000 FPM from T=10 seconds to T=20 seconds
Equipage	= Mode C
Reply Power	= -65 dBm

ACAS X-Equipped Aircraft

Sensitivity Level	= Automatic
Selection	
Altitude	= 12,000 ft

Conditions

ACAS X initialized and operating (with no inputs other than those specified) at or before T=0 seconds.

Time Sequence of Events

Success: In both scenarios, a track on the intruder is established after receipt of the fourth correlating reply at T=3 seconds and is maintained until the scenario ends at T=20 seconds.

2.4.2.1.9 Bearing Estimation

Bearing Accuracy with Standard Ground Plane (§2.2.4.6.6.1, §2.2.4.6.6.2)

The following procedures verify the target bearing accuracy of the bearing antenna and associated processing equipment on an antenna range.

Equipment Required

- Antenna pattern range.
- Ground plane four feet in diameter or larger.
- Reference quarter wave stub.
- Test equipment per §2.4.1.2.
- Means to read the smoothed bearing estimation for the test targets.

Set-Up/Calibration for Bearing Accuracy Tests

Install a vertically polarized quarter wave stub antenna in the center of the ground plane. Attach the ground plane to the antenna range turntable. Set the power of the 1090 MHz test

source so that the received signal strength as measured at the stub antenna connector is at the desired power level specified in the test for an elevation of zero degrees.

Replace the stub with the antenna and connect it to the ACAS X receiver with cables that simulate the actual aircraft installation. The RF port to the unused bottom antenna **shall** (2419) be terminated in a matched load.

Transmit simulated Mode C or Mode S replies from the test source as required by each of the following tests.

Accuracy, -10 degrees to +20 degrees Elevation

This test will verify the bearing accuracy over all azimuths, and over two elevation zones (-10 to +10 degrees and +10 to +20 degrees) using replies that vary in frequency.

Inputs

ACAS X Aircraft

Altitude	=	9,600 ft
Altitude Rate	=	0 FPM
Sensitivity Level	=	Automatic

Intruder Aircraft

Equipage	=	Mode C
Altitude	=	9,600 ft
Altitude Rate	=	0 FPM
Range	=	2 NM
Relative Speed	=	0 kt

Conditions

ACAS X and the test equipment are initiated in a mode that provides one reply per whisper-shout sequence and smoothed bearing estimate for all targets.

Scenarios

For each combination of the following conditions, record the error between the smoothed bearing estimate and the true bearing at zero degrees + m*15 degrees, m=0, 1, ..., 23, azimuth values using 20 whisper-shout sequences at each bearing. The 20 whisper shout sequences chosen **shall** (2420) be such that the bearing filter output has had time to attain a steady state value.

Received Level	=	-51 dBm
Received Frequency	=	1087 and 1093 MHz
Elevation Angle	=	-10, 0, +10, and +20 degrees

Compute the overall RMS error using the $24 * 20 * 2 = 960$ individual errors at each elevation angle.

Success (-10, 0 and +10 degrees elevation)
RMS error \leq 9 degrees
Peak error \leq 27 degrees

Success (+20 degrees elevation)
RMS error \leq 15 degrees
Peak error \leq 45 degrees

Note: The smoothed bearing estimate is a filtered estimate of reply bearing measurements, quantified to 5.625 degrees. The rms error due to the 5.625 degrees quantization is 1.6 degrees and is accounted for in the performance requirements in §2.2.4.6.6.

2.4.2.1.9.2 Reply Processing (§2.2.4.6.6.3)

The following procedures verify correct processing of the individual pulse bearing data during the formation of reply bearings. The tests are conducted using the test equipment described in §2.4.1.2, modified to provide two replies having different bearings.

*Note: These tests require that two replies at different bearings from an ideal antenna be simulated and inserted into the RF ports of the equipment. A zero degree elevation angle **shall** (2421) be assumed. Systematic bias effects should not be simulated for these tests.*

Equipment Required

Test equipment per §2.4.1.2 modified to provide the inputs described in the following tests. Means to read the smoothed bearing estimates for the test targets.

2.4.2.1.9.2.1 Mode C Interleaved Replies (§2.2.4.6.6.3.1)

This procedure will verify that pulse bearing measurements are correctly associated with replies in interleaved garble situations.

Inputs

ACAS X Aircraft

Altitude	=	33,800 ft
Altitude Rate	=	0 FPM
Sensitivity Level	=	Automatic

Intruder #1

Bearing	=	0 degrees
Altitude	=	34,800 ft (Altitude Code=5144)
Range	=	2 NM
Reply Power	=	20 dB above MTL
Reply Frequency	=	1090 MHz

Intruder #2

Bearing	=	90 degrees
Altitude	=	34,800 ft (Altitude Code = 5144)
Range	=	2.53 NM
Reply Power	=	20 dB above MTL
Reply Frequency	=	1090 MHz

Conditions

ACAS X initialized and operating at T=0 seconds. Intruder 1 and 2 **shall** (2422) reply once during each of 20 whisper-shout sequences. The 20 whisper-shout sequences chosen **shall** (2423) be such that the bearing filter output has had time to attain a steady state value.

Scenarios

Measure and record the bearing of Intruder 1 without the presence of Intruder 2. Measure and record the bearing of Intruder 2 without the presence of Intruder 1. Measure and record the bearing for Intruders 1 and 2 interleaved.

Success: The RMS target bearing error over the 20 whisper-shout sequences for each of the intruders **shall** (2424) not increase by more than one degree as a result of interleaving.

2.4.2.1.9.2.2 Mode C Overlapped Replies (§2.2.4.6.6.3.2)

This test will verify that pulse bearing measurements are correctly selected and associated with replies in overlapped garble situations.

Inputs

Intruder #1

Bearing	= 0 degrees
Altitude	= 35,800 ft (Altitude Code=5744)
Range	= 2 NM
Reply Power	= 20 dB above MTL
Reply Frequency	= 1090 MHz

Intruder #2

Bearing	= 90 degrees
Altitude	= 35,800 ft (Altitude Code=5744)
Range	= 3.41 NM
Reply Power	= 20 dB above MTL
Reply Frequency	= 1090 MHz

Conditions

ACAS X initialized and operating at T=0 seconds. Intruders 1 and 2 **shall** (2425) reply once during each of 20 whisper-shout sequences. The 20 whisper-shout sequences chosen **shall** (2426) be such that the bearing filter output has had time to attain a steady state value.

Scenarios

Measure and record the bearing for Intruders 1 and 2 overlapped. Calculate the bearing error as in §2.4.2.1.9.2.1, using the measured bearing of Intruder 1 without 2 and Intruder 2 without Intruder 1.

Success: The RMS target bearing error over 20 whisper-shout sequences for each of the intruders **shall** (2427) not increase by more than one degree as a result of overlapping.

2.4.2.1.9.2.3 Mode S Overlapped Replies (§2.2.4.6.6.3.3)

This procedure will verify that the Mode S target bearing estimate is not significantly affected by a single Mode A fruit reply.

InputsACAS X Aircraft

Altitude	=	34,800 ft
Altitude Rate	=	0 FPM
Sensitivity Level Selection	=	Automatic

Intruder #1

Mode S Short Reply	=	
Bearing	=	0 degrees
Range	=	4 NM
Altitude	=	35,800 ft (Altitude Code=5744)
Reply Power	=	-51 dBm

Intruder #2

Mode A	=	
Bearing	=	90 degrees, 180 degrees
Range	=	5 NM
Code	=	7777
Reply Power	=	-58 dBm

Conditions

ACAS X initialized and operating at T=0 seconds. Intruders 1 and 2 reply once during each of 20 Mode S interrogation cycles. The 20 cycles chosen **shall** (2428) be such that the bearing filter output has had time to attain a steady state value.

Scenarios

Measure and record the bearing of Intruder 1 without the presence of Intruder 2. Measure and record the bearing of Intruder 1 when interfered with by Intruder 2.

Success: For both values of Intruder 2 bearing, the Intruder 1 RMS target bearing error over 20 interrogation sequences **shall** (2429) not increase by more than three degrees when interfered with by Intruder 2.

2.4.2.1.9.3 Bearing Filter Performance (§2.2.4.6.6.4)

The following procedures verify proper operation of the bearing surveillance processor with respect to acquisition, track, coasts and time lag performance.

Equipment Required

Same as §2.4.2.1.9.2.

2.4.2.1.9.3.1 Bearing Track and Coast (§2.2.4.6.6.4.1)

This procedure will verify that the bearing processor correctly acquires, tracks and coasts a track's bearing data.

InputsACAS X Aircraft

Altitude	=	33,800 ft
Altitude Rate	=	0 FPM
Sensitivity Level Selection	=	Automatic

Intruder (Mode C)

Bearing	= 80 degrees (at T=0)
Bearing Rate	= 1 degree/second
Altitude	= 34,800 ft (Altitude Code=5144)
Range	= 2 NM
Range Rate	= 0
Reply Power	= 20 dB above MTL

Conditions

ACAS X initialized and operating at T=0 seconds. Intruder 1 replies once during each whisper-shout sequence for 20 surveillance update intervals and then does not reply for the next 10 update intervals.

Scenarios

The target bearing estimate is recorded from the internal memory each surveillance update interval for the 30 update interval test.

Success: The bearing estimate **shall** (2430) have an RMS error less than five degrees during the interval T=5 to T=20 seconds. The peak error at T=5 and T=25 seconds **shall** (2431) be less than ± 15 degrees.

2.4.2.1.9.3.2 Filter Lag (§2.2.4.6.6.4.1)

This procedure will verify proper operation when the intruder bearing rate is three degrees/second.

Input and Conditions

Same as §2.4.2.1.9.3.1 except that the bearing rate equals three degrees per second and the intruder replies for 30 consecutive surveillance update intervals.

Scenarios

Same as §2.4.2.1.9.3.1. The target bearing estimate is recorded for each surveillance update interval for the 30 update interval test.

Success: The peak bearing lag, defined as the time between when a specific true azimuth exists and when a second order best fit of the target bearing estimate equals that azimuth, **shall** (2432) not exceed five seconds for the 7th through 30th interval.

Note: The effect of aircraft roll during a turn is ignored for the purposes of this test.

2.4.2.1.9.4 Radiation Pattern (§2.2.4.7.2, §2.2.4.7.2.1)**Equipment Required**

- Antenna pattern range.
- Ground plane four feet in diameter or larger.
- Reference quarter wave stub.
- Antenna pattern recorder and L-Band receiver.
- L-Band source antenna and RF generator.

Test

Install the TCAS bearing antenna in the center of the ground plane. Attach the ground plane to the antenna range turntable. Record 360 degree azimuth patterns at 1087, 1093, and 1030 MHz at elevation angles of -15, -10, -5, 0, +5, +10, +15 and +20 degrees.

Repeat the measurements with the reference quarter wave stub installed in the ground plane.

The gain of the TCAS bearing antenna **shall** (2433) meet the requirements of §2.2.4.7.2.1.

2.4.2.1.9.5 Mode C Azimuth Filtering (§2.2.4.6.4.1.2)

This test will verify that ACAS X rejects replies to Mode C interrogations that are received outside of the three dB beam width of the directional beam used for the interrogation.

Inputs

ACAS X Aircraft

Sensitivity Level	=	Automatic
Selection	=	Automatic
Altitude	=	8,600 ft

Intruder Aircraft

Altitude	=	8,300 ft
Altitude Rate	=	0 FPM
Equipage	=	Mode C
Range	=	2 NM
Relative Speed	=	-360 kt (-0.1 NM/s)
Reply Power	=	-50 dBm
Whisper-shout	=	
Step Number	=	1, 25, 26 and 65
Reply Bearing	=	The replies from the intruder shall (2434) be generated and inserted in the TCAS RF ports in a manner that simulates the actual antenna output signals that would result if the intruder bearing were 62 degrees in azimuth from the direction of each interrogation.

Conditions

ACAS X initialized and operating at T=0 seconds. The intruder **shall** (2435) reply only to its assigned whisper-shout step number(s).

Scenario

A single intruder is within the ACAS X threat volume at T=0 seconds and responds to sidelobe interrogations.

Success: An RA is not generated. Scenario terminated at T=10 seconds.

2.4.2.1.10 ACAS X Antenna System

2.4.2.1.10.1 Use of Directional Interrogations for Mode C Surveillance (2.2.4.5.4.2.1, §2.2.4.7.2.1) and Bearing Receive Radiation Pattern (§2.2.4.5.4.2.2, §2.2.4.7.2.2)

The following tests are designed to verify that the directional interrogation antenna system meets the specifications stated in §2.2.4.7.2.1 for a four-beam directional antenna and that the receive radiation pattern meets the requirements of §2.2.4.7.2.2.

Note: The following tests assume that the top mounted directional antenna system contains the antenna elements as well as all necessary RF components to form the directional interrogation patterns, the control pattern and the omnidirectional receive pattern, and that the interface to the remainder of ACAS X consists of a directional interrogation RF port (for P1, P3 and P4), a side-lobe-suppression RF port (for P2), RF receive ports and a control input for directional beam selection. The tests also assume antenna reciprocity in that it is able to receive as well as transmit via the directional and control beams.

Equipment Required

- Antenna pattern range.
- Ground plane four feet in diameter or larger.
- Reference quarter wave stub antenna.
- Antenna pattern recorder and L-Band receiver.
- L-Band source antenna and RF generator.

Measurement Procedure

Step 1 - Directional Bandwidth

Install the directional antenna in the center of the four-foot diameter or larger ground plane and mount the ground plane on the antenna range pedestal. Record 360° azimuth patterns at 1030 MHz for each directional interrogation beam at elevation angles of -15, -10, -5, 0, +5, +10, +15 and 20 degrees.

Record the following information for use in later steps:

- a. RF source generator power output level.
- b. Azimuth position of the peak-of-beam of each directional beam at +10 degree elevation.
- c. Azimuth position of the cross-over point of each pair of adjacent beams.

Success: The three dB azimuth beam width of each directional beam at elevation angles of -15, -10, -5, 0, +5, +10, +15, and +20 degrees **shall** (2436) be less than 100 degrees.

Step 2 - Control Pattern Characteristic

Readjust the RF source generator power output level by an amount equivalent to the relative value of P2 that has been established by the manufacturer as appropriate for the directional interrogation antenna system. Record 360 degree azimuth patterns at 1030 MHz at elevation angles of -15, -10, -5, 0, +5, +10, +15 and +20 degrees for the control pattern associated with each directional interrogation beam.

Success: At each of the azimuth cross-over positions determined in Step 1, the amplitude of the control pattern **shall** (2437) be at least three dB (9 dB for the moderate whisper-shout sequence) below the amplitude of its associated directional interrogation beam to insure adequate coverage against maximum-suppression transponders. Beyond the first

nulls of the main beam of the directional pattern, the amplitude of the control pattern **shall** (2438) be greater than the amplitude of any side- or back-lobe of its associated directional interrogation beam to ensure that minimum-suppression transponders do not respond to more than two adjacent directional beams. If in the moderate whisper-shout sequence the control pattern is not always larger than any side- or back-lobe of its associated directional beam then there should be tests which demonstrate that the ACAS X rejects replies which are received as a result of these unsuppressed side-lobes.

Step 3 - Directional Gain and Antenna Loss

Record an elevation pattern from -15 to +20 degrees at 1030 MHz at the RF generator power output level used in Step 1 for each directional interrogation beam at the peak-of-beam azimuth position determined in Step 1.

Replace the directional antenna on the ground plane with the reference quarter wave stub antenna. Using the same RF generator output level as Step 1, record 360 degree azimuth patterns at 1087 MHz and 1093 MHz at elevation angles of -15, -10, -5, 0, +5, +10, +15 and +20 degrees. Retain for Step 4. Using the same RF generator output level as Step 1, record an elevation pattern from -15 to +20 degrees at 1030 MHz at each of the peak-of-beam azimuth positions determined in Step 1.

Compute a relative gain for each directional beam as follows:

Using the elevation pattern of the directional beam and the elevation pattern of the reference stub taken at the same azimuth position, determine the average value of the relative amplitudes of the two patterns at -15, -10, -5, 0, +5, +10, +15 and +20 degrees elevation.

Compute the fractional beamwidth of each directional beam by dividing its measured three dB beamwidth from Step 1 by 360 degrees. Compute the maximum total radiated power for each directional beam by determining the product of its relative gain, its fractional beamwidth and the peak transmitted power for that beam as measured at the RF reference point.

Success: The maximum total radiated power associated with each directional interrogation beam **shall** (2439) be within ± 2 dB relative to the nominal maximum value specified in §2.2.4.7.2.1.

Step 4 - Bearing Receive Radiation Pattern

Replace the reference quarter wave stub antenna on the ground plane with the directional interrogation antenna. Using the same RF generator output level as Step 1, record 360 degrees azimuth patterns at 1087 MHz and 1093 MHz at elevation angles of -15, -10, -5, 0, +5, +10, +15 and +20 degrees. Compare with the reference stub azimuth patterns recorded in Step 3.

Success: The gain **shall** (2440) meet the requirements of §2.2.4.7.2.2.

2.4.2.1.10.2 Use of Directional Antenna for TCAS Broadcast Interrogations (§2.2.4.7.3, §2.2.3.9.2.4)

These tests are designed to verify that a directional antenna system used for transmission of TCAS Broadcast Interrogations meets the specifications stated in §2.2.4.7.3 and §2.2.3.9.2.4.

2.4.2.1.10.2.1 Total Radiated Power

Compute the maximum total radiated TCAS Broadcast Interrogation power for each directional beam by determining the product of the relative gain and fractional beam width

derived in §2.4.2.1.10.1, Step 3, and the peak transmitted TCAS Broadcast Interrogation power from that beam as measured at the RF reference point.

Success: The maximum total radiated TCAS Broadcast Interrogation power associated with each directional interrogation beam **shall** (2441) be 49 ± 2 dBm.

2.4.2.1.10.2.2 Interrogation Repetition Interval and Jitter

Equipment Required

This test uses the equipment and set-up described in §2.4.2.1.1.3 (whisper-shout operation) except that a storage oscilloscope is required for this test.

Measurement Procedure

Set S1 to top antenna. Program ACAS X for directional TCAS Broadcast Interrogations. This measurement procedure requires that each directional TCAS Broadcast Interrogation be associated with a directional beam. If separate antenna beam outputs are not used (i.e., beam switching occurs inside the antenna subsystem), the interrogation-to-beam association **shall** (2442) be accomplished by simultaneously observing the beam switching control signal and the interrogation waveform.

Set the oscilloscope digital delay and sweep rate to observe the rate and jitter of the TCAS Broadcast Interrogations.

Success: The TCAS Broadcast Interrogation rate, beam sequencing, and jitter **shall** (2443) be as specified in §2.2.4.7.3 and §2.2.3.9.2.4.

2.4.2.2 Test Suite Test Procedures

This section describes the tests per Test Suite for the Aircraft Collision Avoidance System (ACAS) X as specified in the ACAS X Algorithm Design Description (ADD [Volume II]). Detailed test input and expected output files are provided.

The main purpose of the Test Suite is to validate correct implementation of the ADD, specified in the MOPS as Julia code. In addition, several test encounters were generated as MOPS tests for functionality validation, e.g. coordination messages with different combinations of the parity bits, Gillham altitude decoding and correct setting of the ARA field values during RAs. Complete functional validation of the ADD is not covered in this Test Suite but has been accomplished through other algorithm validation activities as indicated in the Metrics Matrix (Ref. Y).

The test files are provided in electronic form to allow, to the maximum extent possible, the use of automation in preparing test inputs and comparing test outputs. §2.4.2.2.1 and §2.4.2.2.2 describe in detail the test input files and expected output files to test an implementation of the Collision Avoidance System. The location of these files is described in §1.6.2.

The files of the Test Suite ensure functional coverage as well as branch coverage for each branch of the ADD. Table 2-75 shows the functional areas and branch coverage included in the encounter set.

Table 2-75: Functional Areas Covered by Test Suite

Test Group Number	Functionality	Encounter Numbering
10	Different RA combination (MOPS Table 2-49), ARA field, RA broadcast and RA ground message	1000-1999
20	Test TID field in multithreat encounters with different equipage combinations	2000-2999
3x	Coordination tests	
31	Coordination tests per hazard and operability study (HAZOP)	3100-3199
32	Coordination Parity Check	3200-3299
33	Coordination Reach-back tests	3300-3399
4x	Mode C	
41	Mode C Gillham Decode	4100-4199
42	Mode C TID field during non-zero vertical rate	4200-4299
5x	Specific geometries and combinations	
51	Turning	5100-5199
52	Long Forward Prediction	5200-5299
53	Altitude inhibit	5300-5399
54	Ownship discretes	5400-5499
55	Improper or degraded inputs	5500-5599
56	Precision Encounters	5600-5699
60	Xo Designation	6000-6999
70	Branch Coverage	7000-7999

In addition to the ownship information, intruder information is provided as Mode C replies, Mode S replies, or ADS-B and ADS-R messages.

The Test Suite consists of 422 test scenarios. A detailed list of encounters is shown in Appendix F.

The input for the tests is provided in the form of scenarios which are defined in the Encounter Input Files. These scenarios specify the surveillance and sensor inputs needed to execute the ACAS X logic. A detailed description of the Encounter Input Files is given in §2.4.2.2.1.

The expected outputs of the tests are defined in the STM Report File, the TRM Report File, and the CostFile. Detailed descriptions of those output files are given in §2.4.2.2.2. All files are delivered in JSON format.

The ADD contains prescriptive or mandatory algorithms and suggestive or optional algorithms. The set of mandatory encounters always has to be run, the set of optional encounters only if the suggestive algorithms were implemented as suggested in the ADD. See §2.4.2.2.4 for more details.

The Encounter Input Files can be run using ASIM, a simulation tool that directly executes the ACAS X ADD Julia code and produces output files matching the Test Suite expected output formats. Arbitrary encounters conforming to the Encounter Input File format may be run in ASIM to produce comparison output files for further testing. Information on how to obtain ASIM is given in §1.6.2.

2.4.2.2.1 Encounter Input Files

This file contains aircraft sensor data to be passed directly into the ACAS X STM. This includes ownship discretes, ownship altitude and position data, active (Mode C and Mode S) and passive (i.e. ADS-B, ADS-R) intruder surveillance data, intruder designation, capability and coordination information. Each observation contains the time a message was received (“report_time”) and the time of applicability (“toa”) of the message. In the Test Suite, report time and toa of a message will always be the same. The report time indicates exactly when during the encounter this information will be sent to the STM. Values and details of the Input file are described in Table 2-76.

Table 2-76: Input File Details

Field	Component	Description
airborne_position_report	toa	time of applicability (seconds)
	address	ICAO address
	non_icao	flag used to indicate if the address is a non-standard ICAO address (anonymous)
	rebroadcast	flag used to indicate if the message is ADS-B (0) or ADS-R (1)
	lat_deg	encoded latitude (degrees)
	lon_deg	encoded longitude (degrees)
	baro_alt_ft	barometric altitude (ft)
	quant_ft	altitude quantization, typically 25 or 100 (ft)
	nic	Navigation Integrity Category (NIC), 0-11
airborne_velocity_report	toa	time of applicability (seconds)
	address	ICAO address
	non_icao	flag used to indicate if the address is a non-standard ICAO address (anonymous)
	rebroadcast	flag used to indicate if the message is ADS-B (0) or ADS-R (1)
	vel_ns_kts	velocity in the North/South dimension (kts)
	vel_ew_kts	velocity in the East/West dimension (kts)
	nic	Navigation Integrity Category (NIC), 0-11
baro_alt_obs	toa	time of applicability (seconds)
	baro_alt_ft	observed ownship barometric altitude (ft)
capability_report	toa	time of applicability (seconds) (unused by ACAS-X)
	address	ICAO address
	adsb_version	0= RTCA/DO-260/242, 1= RTCA/DO-260A/242A, 2= RTCA/DO-260B/242B
	ca_operational	0=not operational, 1=operational
	sense	vertical and/or horizontal sense capability (3 bits)
	type_capability	CAS capability/type (3 bits)
	priority	priority (2 bits)
	daa	DAA capability/type (2 bits in OCM)
df0	toa	time of applicability (seconds)
	mode_s	Mode S address in the reply message
	r_slant_ft	slant range (ft)
	Chi_rel_rad	azimuth, ie bearing (radians)
	baro_alt_ft	barometric altitude (ft)

Field	Component	Description
	quant_ft	altitude quantization (ft)
	ri	RI field, indicates whether there is no TCAS, TCAS TA only, TCAS with RA
	surv_mode	surveillance mode used to elicit the reply
heading_obs	toa	time of applicability (seconds)
	heading_true_rad	observed ownship heading from true north (radians)
	heading_degraded	ownship heading is degraded
mode_c_reply	toa	time of applicability (seconds)
	external_ID	optional external ID set by a front-end tracker
	coded_alt	Gillham coded altitude (coded ft)
	confidence	altitude confidence bits
	r_slant_ft	slant range (ft)
	Chi_rel_rad	azimuth, ie bearing (radians)
mode_status_report	toa	time of applicability (seconds) (unused by ACAS-X)
	address	ICAO address
	non_icao	flag used to indicate if the address is a non-standard ICAO address (anonymous)
	rebroadcast	flag used to indicate if the message is ADS-B (false, default) or ADS-R (true)
	adsb_version	0= RTCA/DO-260/242, 1= RTCA/DO-260A/242A, 2= RTCA/DO-260B/242B
	nacp	Navigation Accuracy Position (NACp), 0-11
	nacv	Navigation Accuracy Velocity (NACv), 0-4
	sil	Source Integrity Level (SIL), 0-3
	sda	System Design Assurance (SDA), 0-2
ownship_discretes	toa	time of applicability (seconds) (unused by ACAS-X)
	address	24-bit address of ownship
	mode_a	Mode A code for ownship
	opflg	system operational
	manual_SL	sensitivity level from an external source
	own_ground_display_mode_on	indicates if display is allowed when own on the ground
	on_surface	indicates whether ownship is operating on the surface
	aoto_on	indicates if ADS-B Only TA-Only is allowed
	is_coarsely_quant	indicates if ownship altitude is 100 foot quantized
rad_alt_obs	toa	time of applicability (seconds) (unused by ACAS-X)
	rad_alt_ft	observed ownship radio altitude (ft)
target_designation	mode_s	mode s address of the target aircraft
	designation	Xo mode selected for the target aircraft
uf16uds30	toa	time of applicability (seconds)
	mode_s	Mode S address in the reply message
	cvc	cancel vertical complement 0-3
	vrc	vertical resolution complement 0-3
	vsb	vertical sense bit
wgs84_obs	toa	time of applicability (seconds)
	lat_deg	latitude ownship (degrees)
	lon_deg	longitude ownship (degrees)

Field	Component	Description
	vel_ew_kts	velocity ownership in the East/West dimension (kts)
	vel_ns_kts	velocity ownership in the North/South dimension (kts)

2.4.2.2.1.1 Rounding

All latitude and longitude values of the inputs are rounded to 8 significant digits, all other input values are rounded to 5 significant digits.

2.4.2.2.1.2 Timing

The JSON input files contain two types of time information: the time a message was received (“report_time”), and the time of applicability (“toa”) of the message. Both time data fields have to agree within each message.

The STM-TRM cycle runs at every full second. All messages with a timestamp up to (and including) the time of the STM-TRM cycle are processed in that cycle. For example the STM-TRM cycle that runs at t=10.0 seconds processes all messages with t in the interval (9,10] seconds. The order of messages must be sequential.

The note in §2.2.3.9.3.4 - T4 mentions the coordination reach-back concept of the ADD which allows the TRM to request the most recent VRC at any time following receipt of the STM Report up until the beginning of the algorithm "Online Cost Estimation." To test this mechanism, the Test Suite contains specific coordination reach-back tests in Testgroup 33. For these tests, messages are not only read at the beginning of the STM-TRM cycle, but coordination messages (uf16uds30) are also read directly before the TRM is executed. For the tests in the Test Suite, this has to be simulated at 600 msec into each cycle. For example, if the STM-TRM cycle runs at t=10.0 seconds, the coordination messages up to (and including) 10.6 seconds are read after the STM is run, but before the TRM is executed. If there is a coordination message at 10.5 seconds, then this coordination message gets used for the execution of the TRM. If the message arrives at 10.8 seconds, then it is not used. The descriptive text of each encounter contains instructions if the reach-back function has to be used for the test. If no such information is provided, the test should be run without reach-back mechanism.

The following rules are applied to the input files of all Test Suite encounters:

- Messages are at least 1 msec apart from each other, they never arrive at the same time.
- Time of applicability is rounded to full msec.
- Each cycle contains maximal one ownership message of the same type, no duplicates.

There are several places in the ADD where the condition “if a time difference is < 10ms” or “if a time difference is \geq 10 msec” determines if a message might get ignored or used, or if the values of the message are used directly or their extrapolated values are used instead. Due to floating point representation and compiler differences / compiler settings, a calculated dt of 10 msec might be represented as a number slightly below 10 msec, hence the comparison in the code will have a different result and a different action is performed, e.g., the position data will not be forward extrapolated. This can cause a different result in the output.

To avoid this issue, the input files were designed so that this 10ms comparison issue is avoided. This approach seems reasonable because the problem is a compiler-generated issue and not an implementation issue.

2.4.2.2.2 Encounter Output Files

There are three output files used to compare expected results to the test results. The STM report contains surveillance and track information, the TRM report contains the collision avoidance advisories and the CostFile includes different cost values that are used as interim values to determine the TAs or RAs given by the system. The following sections describe these files in detail.

2.4.2.2.2.1 STM Report

This output file represents the way ACAS X views the current state of the encounter at a nominal 1 Hz rate. This includes ownship heading and vertical state estimate as well as reports for each intruder. The intruder reports include coordination information and the distributions of the intruder's relative position and velocity. These values are used as input for the TRM.

Values and details of the STMreport are described in Table 2-77.

Table 2-77: STM Report Details

Field	Component	Description
trm_input	own	Information about ownship
	intruder (multiple)	Information for each intruder
own	h	altitude above ground as measured by radar altimeter (ft)
	psi	heading (radians, north is 0, clockwise is positive)
	mode_s	Mode S address (24 bit)
	mode_a	Mode A id (24 bit)
	opmode	ownship vertical CA mode setting
	ground_speed	Xu only, ownship ground speed (ft/s)
	degraded_own_surveillance	Bit vector indicating types of surveillance
	xo_availability (multiple)	Availability for each Xo mode
	belief_vert (multiple)	Vertical beliefs
intruder	valid	if this is false, ignore this intruder for TRM processing
	id	id of the intruder, must be unique
	address	ICAO (e.g., Mode S id) or non-ICAO address (24 bit)
	is_icao	address field is an ICAO address
	vrc	vertical resolution complement
	equipage	CAS equipment
	active_cas_version	Type of active CAS
	coordination_msg	Coordination message type
	source	surveillance source the track is derived from

Field	Component	Description
	degraded_surveillance	Bit vector indicating types of surveillance of degraded quality
	is_proximate	Indicates if proximate to ownship used for proximity advisory)
	designated_mode	requested designation
	protection_mode	protection mode code (for support for Xa/Xo)
	dna	flag indicating the target has been designated for No Alerts
	xo_valid (multiple)	Xo validity for each Xo mode
	xo_status	Xo status information
	stm_display	STM display data for onboard display, contains range and bearing
	belief_vert (multiple)	all states in the belief with nonzero weight
belief_vert	z	barometric altitude (ft)
	dz	vertical rate (ft/s)
	weight	weight of this element in the belief
belief_horiz	x_rel	relative position (feet)
	y_rel	relative position (feet)
	dx_rel	relative speed (feet/second)
	dy_rel	relative speed (feet/second)
	weight	weight of this element in the belief
transponder	ri	air-to-air reply information
	sl	sensitivity level
	vi	version indicator
	bit48	data link subfield bit 48 in MB
	bit69	data link subfield bit 69 in MB
	bit70	data link subfield bit 70 in MB
	bit71	data link subfield bit 71 in MB
	bit72	data link subfield bit 72 in MB
display	arrow	display vertical rate arrow
	alt_reporting	intruder altitude reporting
	bearing_valid	bearing for intruder is available
	id	track file intruder number
	z_rel	relative altitude (ft)
	r_ground	tracked ground-range (ft)
	Chi_rel	bearing relative to own airframe (rad)
	mode_s	intruder MSID
	is_icao	indication whether 24-bit address is ICAO compliant

2.4.2.2.2 TRM Report

This file contains the output values of ACAS X, also at a nominal 1 Hz rate and synchronized with the STM 1 Hz rate, which are the TAs and RAs that are to be issued at each time step, values used for the ownship display, ground message and broadcast data, and the coordination data generated for each intruder.

Values and details of the TRM report are described in Table 2-78.

Table 2-78: TRM Report Details

Field	Component	Description
display	cc	combined control (label 270)
	vc	vertical control (label 270)
	ua	up advisory (label 270)
	da	down advisory (label 270)
	target_rate	vertical rate to target from RA (ft/s)
	turn_off_aurals	Low altitude cutoff for aurals
	crossing	RA will result in a crossing
	alarm	RA should be annunciated
	intruder (multiple)	Per-intruder information for the pilot display
intruder (for display)	id	id of the intruder, must be unique
	mode_s	Mode S address of the intruder, may not be unique
	is_icao	indication whether 24-bit address is ICAO compliant
	tds	track display score
	code	track display code
coordination (multiple)	id	id of the intruder, must be unique
	cvc	cancel vertical complement
	vrc	vertical resolution complement
	vsb	vertical sense bit
	chc	cancel horizontal complement
	hrc	horizontal resolution complement
	hsb	horizontal sense bit
	mtb	multiple threats for this RA
	mid	Mode S address of ownship, may not be unique
	taa	Mode S address of the intruder, may not be unique
designation	coordination_msg	Coordination message mechanism
	availability (multiple)	Availability for each Xo mode
	intruder (multiple)	Designation info for each intruder
intruder (for designation)	id	id of the intruder, must be unique
	address	24-bit address of the intruder, may not be unique
	is_icao	Value in address field is an ICAO address
	designated_mode	requested designation
	logic_mode	logic processing
	active_ra	Intruder has an RA (that may be suppressed)
	suppressed_ra	RA was suppressed to the pilot display

Field	Component	Description
	multithreat	There is a multithreat RA
	valid (multiple)	Xo validity flags
	status	Xo status information
logic_mode	processing	TRM processing mode for intruder
	dna	flag indicating the target has been designated for No Alerts
	protection_mode	Protection mode used when processing the intruder
broadcast	ra_data	Contains ARA, RMF, RAC, RAT, and MTE fields
	spi	Suppression indicator (1 bit)
	aid	Own Mode A identifier code (13 bits)
	cac	Own Mode C altitude code (13 bits)
ground_msg	ra_data	Contains ARA, RMF, RAC, RAT, and MTE fields
	tti	Threat type indicator for most recent threat (1 bit)
	tid	Threat identifier for most recent threat (24 bits, address or altitude-range-bearing)
	dsi	TID designation indicator (1 bit)
	spi	TID suppression indicator (1 bit)
ra_data (for broadcast and ground message)	avra_single_intent	Active Vertical RA (AVRA) (7 bits) Single intent/Conflicting intents (1 bit)
	avra_crossing	Crossing/ Not crossing (1 bit)
	avra_down	Up/Down (1 bit)
	avra_strength	Strength bits as integer (4 bits)
	ldi	Low-level Descend Inhibit (2 bits)
	rmf	RA Message Format (2 bits)
	rac (multiple)	RA Complements (4 bits)
	rat	RA Terminated (1 bit)
	mte	There are multiple threats for this RA (1 bit)
tid	tid_type	Whether TID is address or altitude-range-bearing
	address (one of)	Address representation of TID
	altrngbrg (one of)	Encoded altitude-range-bearing representation of TID
address	tid	TID address (24 bits)
altrngbrg	tida	TID encoded altitude (ft) (11 bits)
	tidr	TID encoded range (ft) (7 bits)
	tidb	TID encoded bearing (deg) (6 bits)
debug	alert	TA active (debug)
	dz_min	minimum vertical rate for RA compliance (ft/s) (debug)
	dz_max	maximum vertical rate for RA compliance (ft/s) (debug)
	ddz	intended acceleration (ft/s/s) (debug)
	intruder (multiple)	Per-intruder debugging information
intruder (for debug)	id	id of the intruder, must be unique

Field	Component	Description
	address	ICAO or non-ICAO address of the intruder, may not be unique
	is_icao	Value in address field is an ICAO address
	sense	Sense of the intruder RA

2.4.2.2.2.3 CostFile

This file contains the cost values for each action within the TRM. Values and details of the CostFiles are described in Table 2-79.

Table 2-79: CostFile Details

Field	Component	Type	Description
Encounter	time	double	toa in seconds
	own_mode_s	uint32	Ownship ICAO 24-bit Aircraft Address
	data	➔	Cost data
data	Individual	➔	Individual cost values Omitted if no intruders
	Combined	➔	Merged cost values Omitted if no intruders
Combined	UnequippedCostFusion	➔	Single fused cost for each action for all unequipped intruders Omitted if only one intruder or no unequipped intruders
UnequippedCostFusion	{MAINTAIN, SCL2500, DES1500, SDES2500, SDES1500, DND, DNC, COC, CL1500, SCL1500, MTLO}	double	specific fused online cost value
Individual	int_mode_s	uint32	Intruder ICAO 24-bit Aircraft Address
	values	➔	Cost values for specified intruder associated with each action in the offline cost table
	int_id	uint32	Intruder identifier (track id)
values	MAINTAIN	➔	Cost values for maintain vertical rate action
	SCL2500	➔	Cost values for increase climb action
	DES1500	➔	Cost values for initial descend action
	SDES2500	➔	Cost values for increase descend action
	SDES1500	➔	Cost values for subsequent descend action
	DND	➔	Cost values for DND action
	DNC	➔	Cost values for DNC action
	COC	➔	Cost values for CoC action
	CL1500	➔	Cost values for initial climb action
	SCL1500	➔	Cost values for subsequent climb action

Field	Component	Type	Description
	MTLO	→	Cost values for MTLO action
{MAINTAIN, SCL2500, DES1500, SDES2500, SDES1500, DND, DNC, COC, CL1500, SCL1500, MTLO}	Online TA	double	Cost value for TA (ta_altitude_dependent_coc)
	Offline	double	Offline cost value
	Total	double	Total cost value
	Online Itemized	→	List of itemized online costs
	Online	double	Total Online cost value
	Online Subset	double	Subset of online cost
	Multithreat	→	Costs applied when there are multiple threats
Multithreat	C_restrict_mtlo	double	Specific online cost value applied only to MTLO action Omitted if not applied
	sandwich_prevention	double	Total online cost value for SandwichPreventionCost Omitted if SandwichPreventionCost not used
Online Itemized	time_based_non_compliance	double	Specific online cost value for TimeBasedNonComplianceCost
	coord_ra_deferral	double	Specific online cost value for CoordinatedRADeferralCost
	safe_crossing_ra_deferral	double	Specific online cost value for SafeCrossingRADeferralCost
	max_reversal	double	Specific online cost value for MaxReversalCost
	low_alt_parallel_ra_deferral	double	Specific online cost value for LowAltitudeParallelRADeferralCost applied for RAs
	prevent_early_weakening	double	Specific online cost value for PreventEarlyWeakeningCost
	SA01	double	Specific online cost value for SA01Heuristic
	altitude_inhibit	double	Specific online cost value for AltitudeInhibitCost
	bad_transition	double	Specific online cost value for BadTransitionCost
	restrict_coc_due_to_reversal	double	Specific online cost value for RestrictCOCDueToReversal
	ta_altitude_dependent_coc	double	Specific online cost value for AltitudeDependentCOCCost applied for TAs
	critical_interval_protection	double	Specific online cost value for CriticalIntervalProtectionCost
	ta_low_alt_parallel_ra_deferral	double	Specific online cost value for LowAltitudeParallelRADeferralCost applied for TAs
	advisory_restart	double	Specific online cost value for AdvisoryRestartCost
	altitude_dependent_coc	double	Specific online cost value for AltitudeDependentCOCCost applied for RAs

Field	Component	Type	Description
	compatibility	double	Specific online cost value for CompatibilityCost
	coord_delay	double	Specific online cost value for CoordinationDelayCost
	initialization	double	Specific online cost value for InitializationCost
	C_force_alert	double	Specific online cost value
	C_restrict_mtlo	double	Specific online cost value applied only to MTLO action Omitted if not applied
	prevent_early_coc	double	Specific online cost value applied to DNC, DND, and COC only
	crossing_no_alert	double	Specific online cost value for CrossingNoAlertCost

2.4.2.2.3 Pass / Fail Criteria

Each encounter of the Test Suite must be run with the vendor implementation. The following values must be extracted from the observed outputs of the vendor implementation and compared to the expected outputs provided in the Test Suite. If the differences for all values are within the tolerances (also considering wrap-around values for angular comparisons), then the test passes, otherwise it fails.

For the test encounters, the expected results are listed in the STMreport, the TRMreport and in the CostFile. The values that have to be checked to determine if a test passes or fails and the acceptable tolerances are listed in §2.4.2.2.3.1, §2.4.2.2.3.2, and §2.4.2.2.3.3.

2.4.2.2.3.1 STM Report

The vendor implementation output values must be compared to the expected results of the STM report for the values shown in Table 2-80.

Table 2-80: STM Report Test Values and Tolerances

Value description	Tolerance
display:	
arrow (integer)	0
alt_reporting: indicator (bool)	0
bearing_valid: indicator (bool)	0
id: Intruder track number (integer)	0
z_rel: Relative altitude (float)	0.1
r_ground: Tracked ground-range (float)	0.1
Chi_rel: Relative bearing (float)	0.1
mode_s: Intruder ICAO Aircraft Address (24-bit)	0
is_icao: indicator whether 24-bit address is ICAO compliant (bool)	0

2.4.2.2.3.2 TRM Report

The vendor implementation output values must be compared to the expected results of the TRM report for the values shown in Table 2-81.

Table 2-81: TRM Report Test Values and Tolerances

Value description	Tolerance
display:	
cc, vc, ua, da (int)	0
target_rate (float)	0.1
turn_of_aurals, crossing, alarm (bool)	0
intruder:	
id: Intruder track number (integer)	0
mode_s: Intruder Mode S address (24-bit)	0
is_icao: indicator whether 24-bit address is ICAO compliant	0
tds: track display score (float)	0.1
code: track display code (integer)	0
coordination:	
id: Intruder track number (integer)	0
cvc, vrc, vsb, chc, hrc, hsb (int)	0
mtb: multithreat indicator (bool)	0
mid: ownship Mode S address (24-bit)	0
taa: intruder Mode S address (24-bit)	0
coordination_msg (integer)	0
designation:	
availability: for each Xo mode ([integer, integer])	0
intruder:	
id: Intruder track number (integer)	0
mode_s: Intruder Mode S address (24-bit)	0
is_icao: indicator whether 24-bit address is ICAO compliant	0
designated_mode (integer)	0
logic_mode:	
processing: mode for intruder (integer)	0
dna: indicator for No Alerts (bool)	0
protection_mode: used for intruder (integer)	0
active_ra: maybe suppressed (bool)	0
suppressed_ra: RA suppressed to pilot display (bool)	0
multithreat (bool)	0
valid: flags for Xo validity ([bool, bool])	0
status: Xo status information (integer)	0
broadcast:	
ra_data:	
Active vertical RA (avra): single_intent (bool), crossing (bool), down (bool), strength (4 bit),	0

Value description	Tolerance
ldi: Low-level Descend Inhibit (2 bits)	0
rmf: RA message format (2 bits)	0
rac: RA complements ([bool, bool, bool, bool])	0
rat: RA terminated (bool)	0
mte: multiple threats (bool)	0
spi: Suppression indicator (bool)	0
aid: Ownship Mode A identifier code (13 bits)	0
cac: Ownship Mode C altitude code (13 bits)	0
ground_msg:	
ra_data:	
Active vertical RA (avra): single intent (bool), crossing (bool), down (bool), strength (4 bit),	0
ldi: Low-level Descend Inhibit (2 bits)	0
rmf: RA message format (2 bits)	0
rac: RA complements ([bool, bool, bool, bool])	0
rat: RA terminated (bool)	0
mte: multiple threats (bool)	0
tti: threat type indicator for most recent threat (bool)	0
tid:	
tid_type (bool)	0
address or altrngbrg (24 bit)	0
dsi: TID designation indicator (bool)	0
spi: TID suppression indicator (bool)	0

2.4.2.2.3.3 CostFile

All CostFile values as listed in Table 2-79 must be compared for each second of the encounter, each intruder, and each possible action which are "MAINTAIN", "SCL2500", "DES1500", "SDES2500", "SDES1500", "DND", "DNC", "COC", "CL1500", "SCL1500", and "MTLO".

Table 2-82: Cost Value Tolerances

Value description	Tolerance
Cost value (all)	1E-5 (0.00001)

2.4.2.2.4 Prescriptive/Mandatory Vs. Suggestive/Optional Algorithms

The ADD consists of two kinds of algorithms, prescriptive and suggestive, also called mandatory and optional, respectively. The prescriptive algorithms are mandatory and have to be implemented exactly as specified in the ADD. The suggestive algorithms are optional. The suggestive algorithms or a manufacturer's own implementation must meet the correlation requirements specified in §2.2.5.4.

If an equipment manufacturer chooses to implement the optional algorithms according to the suggestions in the ADD, the Test Suite encounters for optional algorithms can be used to demonstrate compliance. If an equipment manufacturer's own implementation is used, the manufacturer must provide testing for those algorithms.

Accordingly, the Test Suite contains tests that are marked as "prescriptive". Those tests are mandatory, every equipment manufacturer has to run and pass these. In addition,

Test Suite contains tests that are for the suggestive algorithms. These are marked as “non-prescriptive”. The indication of “prescriptive” and “non-prescriptive” encounters is provided in Tables F-1 to F-7 in Appendix F.

If an equipment manufacturer’s implementation of the optional algorithms differs from the suggestions in the ADD, only the “prescriptive” test encounters can be used for testing. In this case, not all branches of the prescriptive algorithms are covered and additional test encounters have to be produced to make sure these branches are tested. Those branches of the prescriptive part of the ADD that are not covered by running the “prescriptive” test encounters only are listed in Table 2-83.

Table 2-83: Branches of Prescriptive Algorithms Not Covered By Prescriptive Test Encounters

Algorithm	BranchType	LineNumber
AddModeCTrackToReport	THEN	6
AddModeCTrackToReport	THEN	8
AddModeCTrackToReport	IMP_ELSE	8
AddModeCTrackToReport	THEN	14
ConvertToCartesian	THEN	4
HeadingAtToa	THEN	7
RemoveStaleModeCTracks	THEN	12
RemoveStaleModeCTracks	IMP_ELSE	15
RemoveStaleModeCTracks	THEN	28
SurroundingPoints	THEN	3
IsTargetDesignationActive	IMP_ELSE	15
IsTargetDesignationActive	IMP_ELSE	18
SetIntruderDesignationData	THEN	37

2.4.2.2.5 Defensive Code Branches

The code in the ADD contains some branches that are unreachable. Those are considered useful as defensive code and will not be removed. It is acceptable not to have these branches covered. The unreachable branches in Prescriptive Algorithms are listed in Table 2-84, unreachable branches in Suggestive Algorithms are listed in Table 2-85. Branch types are in one of three categories: THEN - the *then* part of a conditional branch, ELSE - the *else* part of a conditional branch, and IMP ELSE - an implicit else branch, where there is no explicit *else* in the code.

Table 2-84: Unreachable Branches (Defensive Code) in Prescriptive Algorithms

Algorithm	BranchType	LineNumber
IntruderPrep	ELSE	36
GetModifiedGlobalRates	ELSE	16
MaintainRates	THEN	17
ArbitrateConflictingSenses	ELSE	41
DetermineIntruderAlert	IMP_ELSE	33
DetermineScore	IMP_ELSE	29
ConvertAdvisory	IMP_ELSE	27
DroppedIntrudersAdjustment	IMP_ELSE	39
DetermineDSI	IMP_ELSE	5
EncodeCAC	IMP_ELSE	13
EncodeCAC	IMP_ELSE	4
EncodeTIDAltitude	THEN	5
EncodeTIDAltitude	THEN	9
EncodeTIDAltitude	THEN	13
EncodeTIDAltitude	THEN	15
EncodeTIDRange	THEN	6
EncodeTIDRange	THEN	14
EncodeTIDRange	THEN	16
EncodeTIDBearing	IMP_ELSE	4
DetermineTTIAndTID	IMP_ELSE	5
ConvertToAzAndXRangeRate	THEN	15
FilterTracksForTRM	IMP_ELSE	13
GillhamEncode	IMP_ELSE	5
GroundRange	THEN	3

Algorithm	BranchType	LineNumber
ConvertECEFToWGS84	THEN	27
AdvanceADSBTrackPosition	THEN	9
UpdateModeCTrack	IMP_ELSE	5
UpdateDroppedTargetDesignationState	IMP_ELSE	35
CheckTargetDesignationTimers	IMP_ELSE	10
CheckTargetDesignationTimers	IMP_ELSE	22
SetPendingTargetDesignation	IMP_ELSE	28
SetTargetDesignation	THEN	10
SetTargetDesignation	THEN	12
SetTargetDesignation	IMP_ELSE	12
SetTargetDesignation	THEN	36
SetTargetDesignationModeAvailability	THEN	16
UpdateDNAValidity	THEN	5

Table 2-85: Unreachable Branches (Defensive Code) in Suggestive Algorithms

Algorithm	BranchType	LineNumber
EstablishModeCTrack	THEN	5
EstablishModeCTrack	THEN	10
DecorrelateTargets	IMP_ELSE	31
CorrelateTargets	ELSE	34
CorrelateTargets	THEN	35
CorrelateTargets	IMP_ELSE	35
CorrelateTargets	IMP_ELSE	55
RangeSquared	IMP_ELSE	7
AddDecorrelatedTrackToTarget	IMP_ELSE	5
GenerateLeadTrackList	THEN	20
ExtrapolateTrack	IMP_ELSE	6
MergeTargetDesignations	THEN	21

Algorithm	BranchType	LineNumber
MergeTargetDesignations	IMP_ELSE	20
MergeTargetDesignations	THEN	32
MergeTargetDesignations	ELSE	40
RemoveDecorrelatedTrackFromTarget	IMP_ELSE	9
EvaluateOnGroundModeC	IMP_ELSE	9
HypotheticalTrackTest	THEN	19
CorrelateID	THEN	4
CorrelateID	THEN	9
CorrelateID	THEN	14
CorrelateIndividualTracks	ELSE	36

2.4.2.2.6 Aurals Flag

The ACAS X MOPS includes a test to verify that all different aurals that can be issued during an RA are enunciated. The different aurals are listed in the ACAS X MOPS in Table 2-49. The Test Suite includes a flag indicating the subset of encounters that can be used within a bench test to test all different aurals (“aurals = true” in the encounter header). The use of these tests for the aurals is not required; these tests are just one possibility to make sure all different aurals are tested.

2.4.2.2.7 Detailed Test Suite Encounter List

The detailed Test Suite Encounter List can be found in Tables F-1 through F-7 in Appendix F.

2.4.2.3 External Parameter Selection

The tests in §2.4.2.3, §2.4.2.4, §2.4.2.5 and §2.4.2.6 are end-to-end system tests that verify both the operation of the ACAS X/transponder system with external elements, e.g., Mode S ground sensors and other ACAS X or TCAS-equipped aircraft, and also the operation of ACAS X with its associated Mode S transponder. The tests verify proper implementation of formats and protocols in §2.2.3 and proper coding of some of the software algorithms specified in the Algorithm Description Document (ADD). Test results are considered successful only if all of the generated outputs match the expected results specified for the test scenario. This includes the presence of data output within allowed tolerances (if any), as well as the absence of unspecified data output, except as noted.

In the scenarios that frame these tests, the use of the acronym ‘TCAS II’ for the intruder is used for convenience and should be understood to mean an Active CAS. The choice of Active CAS, e.g., TCAS II vs. ACAS X, is not critical since these tests depend on the signals transmitted; for transmitted coordination interrogations, all Active CAS use the same signals/formats. The one exception, i.e., the one test in which an intruder TCAS/ACAS X distinction is important, is test 2.4.2.4.3.7, “Use of Received Collision Avoidance Coordination Capability Bits (CCCB).

The following abbreviations are used:

Equip	Equipage (Mode C, Mode S, ACAS X, TCAS II)
Int1	Intruder Aircraft 1 (also Int2, Int3, etc.)
MSS	Mode S ground sensor
Msgs	Messages
R	Horizontal Range
RA	Resolution Advisory
RDOT	Relative speed (positive if diverging)
UUT	Unit Under Test
Z	Altitude
ZDOT	Altitude rate (positive if climbing)

Any ACAS X or TCAS intruder should reply that it has no active RA when interrogated by the UUT, unless otherwise specified. A successful reply should be generated for each intruder in response to each interrogation, unless otherwise specified. Subfield values may be either decimal or binary; in the latter case the value is shown as a concatenated string of ones and zeros, whose length is equal to the length of the indicated subfield. For example, the value ‘001100’ represents a bit string.

All the messages generated by an intruder must be addressed to the UUT except in the case of squitter messages.

When a test calls for the UUT to be interrogated by a Mode S sensor, critical field and subfield values in the sensor’s uplink interrogation are specified in the test. Unless otherwise specified, the values for UF=4, UF=5, UF=20 and UF=21 are as follows:

a. UF=4 Message Contents

UF=4 Interrogation Contents				
			SD field	
PC	RR	DI	IIS	LAS
0	0	0	1	0

b. UF=5 Message Contents

UF=5 Interrogation Contents				
			SD field	
PC	RR	DI	IIS	LAS
0	0	0	1	0

c. UF=20 Message Contents

UF=20 Interrogation Contents										
			SD field				MA field			
PC	R R	DI	IIS	RRS	LOS	TMS	DP	MP	M/CH	SLC
0	0	7	1	0	0	0	0	0	5	0

d. *UF=21 Message Contents*

UF=21 Interrogation Contents											
		SD field						MA field			
PC	R R	DI	IIS	RRS	LOS	TMS	DP	MP	M/CH	SLC	
0	0	7	1	0	0	0	0	0	5	0	

Delays between interrogation and response are ignored unless otherwise noted.

Specific timing of advisories will be tested elsewhere so the outcome of these tests is dependent on the content and sequence of the messages associated with the advisory but not the actual timing of the advisory. Because of the various tolerances and measurement errors that can occur in performing these tests, a tolerance of 1.5 seconds is considered acceptable when timing requirements are indicated.

When testing the TIDR and TIDB message fields, a tolerance of ± 1 LSB may be applied.

Unless otherwise noted, tests in §2.4.2.3, §2.4.2.4, §2.4.2.5, and §2.4.2.6 are to be run while the UUT is connected to an actual Mode S transponder.

2.4.2.3.1 Ground Control of Sensitivity Level

This test procedure verifies that ACAS X properly ignores Sensitivity Level Commands from a ground Mode S sensor without changing the ownship ACAS X sensitivity level.

Scenario A

A Mode S ground sensor transmits valid Sensitivity Level Commands to the UUT; proper setting and reporting of sensitivity level and TCAS equipage are verified by having an intruder interrogate the UUT every second during the scenario.

Input:

UUT:	Z	=	21,000 ft	at T=0
	ZDOT	=	0 fpm	from T=0 to 23
Int1:	Equip	=	TCAS II	
	R	=	5.0 NM	at T=0
	RDOT	=	5 kt	from T=0 to 23
	Z	=	21,000 ft	at T=0
	ZDOT	=	0 fpm	from T=0 to 23
	Msgs	=	UF=0	Once per second
MSS:	Msgs	=	UF=20	at T=1, 4, 7, 10, 13, 16, 19

UF=20 Messages transmitted as follows:

UF=20 Interrogation Contents											
		SD field						MA field			
P C	RR	DI	IIS	RRS	LOS	TMS	D P	MP	M/CH	SLC	Timing Sequence
0	19	7	1	0	0	0	0	0	5	1	T=1

0	19	7	1	0	0	0	0	0	5	2	T=4
0	19	7	1	0	0	0	0	0	5	3	T=7
0	19	7	1	0	0	0	0	0	5	4	T=10
0	19	7	1	0	0	0	0	0	5	5	T=13
0	19	7	1	0	0	0	0	0	5	6	T=16
0	19	7	1	0	0	0	0	0	5	7	T=19

Expected Output:

UUT transmits DF=0 replies with the following data:

DF=0 reply contents		
SL (bits 9-11)	RI(bits 14-17)	Timing
011	0011	In response to all UF=0 Interrogations

UUT transmits DF=20 replies with the following data:

DF=20 MB Field Contents RA Report												
BDS	ARA	LDI	RMF	RAC	RAT	MTE	CNT	TTI	TID	DSI	SPI	Sequence
48	0000000000	00	01	0000	0	0	0	0	0	0	0	In response to all UF=20 Interrogations

2.4.2.3.2 TA-Only Mode Selection

Scenario A Selection of TA-Only Mode After RA is annunciated

This test verifies that a manual sensitivity setting of "TA-Only" will inhibit an active RA.

Input:

UUT: Z = 7,900 ft at T=0
ZDOT = 0 fpm from T=0 to 60

Int1: Equip = TCAS II
 R = 6 NM at T=0
 RDOT = -360 kt from T=0 to 60
 Z = 8,000 ft at T=0
 ZDOT = 0 fpm from T=0 to 60
 Msgs = Squitter msg at T=0
 = UF=0 from T=0 to 60
 Other: Manual setting "TA/RA" Until RA is annunciated

Manual setting "TA-Only" After RA is annunciated

Expected Output:

Display: RA is annunciated and then inhibited after the TA-Only Mode is selected.

Msgs:

UUT transmits DF=0 replies with the following data:

DF=0 Reply Contents		
SL (bits 9-11)	RI(bits 14-17)	Timing
011	0011	While TA/RA is selected
011	0010	While TA-Only is selected

Note: Although UUT is in performance level 3, the manual setting of "TA-Only" mode inhibits the display of RAs.

Scenario B Selection of TA-Only Mode Prior to RA condition

This test verifies that a manual sensitivity setting of "TA-Only" will inhibit the issuance of RAs.

Input:

UUT: Z = 7,900 ft at T=0
 ZDOT = 0 fpm from T=0 to 60
 Int1: Equip = TCAS II
 R = 6 NM at T=0
 RDOT = -360 kt from T=0 to 60
 Z = 8,000 ft at T=0
 ZDOT = 0 fpm from T=0 to 60
 Msgs = Squitter msg at T=0
 = UF=0 from T=0 to 60
 Other: Manual setting "TA-Only" from T=0 to 60

Expected Output:

Display: Only a TA is annunciated (No RA is annunciated).

Msgs:

UUT transmits DF=0 replies with the following data:

DF=0 Reply Contents		
SL (bits 9-11)	RI(bits 14-17)	Timing
011	0010	In response to all UF=0 Interrogations

Note: Although UUT is in performance level 3, the manual setting of "TA-Only" mode inhibits the display of RAs.

2.4.2.4 Coordination

See test operation description in §2.4.2.3.

2.4.2.4.1 Transmission of RA Report to Mode S Sensor

This test verifies that the TCAS/transponder system correctly:

- a. Passes RA information from TCAS to the transponder,
- b. Indicates to the ground (DR field in DF=4, DF=5, DF=20, and DF=21 replies) that it has information awaiting downlink,
- c. Transmits this information in DF=20, DF=21 replies
- d. Retains RA information for 18 ± 1 seconds following the end of the RA,
- e. Indicates the end of the RA via the RA Terminated indicator.
- f. Transmits RA data in the ADS-B TCAS RA Broadcast message (Scenario K only)

Scenario A

The UUT and Mode C transponder-equipped threat are on a horizontal collision course. The UUT declares a "Descend" advisory and is asked to communicate it to the Mode S ground sensor.

Input:

UUT: Z	=	12,000 ft	at T=0
ZDOT	=	0 fpm	from T=0 to 88

Int1: Equip	=	MODE C	
R	=	6.2 NM	at T=0
Bearing	=	0 deg	at T=0
Bearing rate	=	0	from T=0 to 88
RDOT	=	-360 kt	from T=0 to 88
Z	=	12,200 ft	from T=0 to 88
ZDOT	=	0 fpm	from T=0 to 88
Msgs	=	None	

MSS: Msgs = UF=4, with RR=0 from T=0 to 88
 UF=4, with RR=19 from T=0 to 88

Expected Output:

Display: "Descend" RA

Msgs: DF=4 replies received every second with the following data:

bit 12 = 0 (T=0 until RA is issued)

bit 12 = 1 (while the RA is active and for 18 ± 1 second after RA termination)

bit 12 = 0 (for the remainder of the scenario)

DF=20 replies received every second with the following data:

DF=20 MB Field Contents														
RA Report														
BDS	ARA	LDI	RMF	RAC	RAT	MTE	CNT	TTI	TIDA	TIDR	TIDB	DSI	SPI	Sequence
48	0000000000	00	01	0000	0	0	0	0	0	0	0	0	0	Prior to RA
48	1010100000	00	01	0000	0	0	0	0	*alt	*rng	*brg	0	0	While RA is active
48	1010100000	00	01	0000	1	0	0	0	*alt	*rng	*brg	0	0	for 18 ± 1 s after RA termination
48	0000000000	00	01	0000	0	0	0	0	0	0	0	0	0	Remainder of scenario

*Verify that the intruder altitude (alt), range (rng) and bearing (brg) are appropriately encoded into the TIDR, TIDA and TIDB fields for each second during which this information is transmitted in accordance with the following tables. When the RA terminates the TIDR, TIDA and TIDB fields are frozen for the subsequent 18 ± 1 seconds.

Time	Altitude (ft)	TIDA	Bearing (deg)	TIDB
0-62	12,200	00010000101	0	000001
63-88	12,200	00010000101	180	011110

Time	Range	TIDR	Time	Range	TIDR	Time	Range	TIDR
0	6.2	0111111	30	3.2	0100001	60	0.2	0000011
1	6.1	0111110	31	3.1	0100000	61	0.1	0000010
2	6	0111101	32	3	0011111	62	0	0000001
3	5.9	0111100	33	2.9	0011110	63	0.1	0000010
4	5.8	0111011	34	2.8	0011101	64	0.2	0000011
5	5.7	0111010	35	2.7	0011100	65	0.3	0000100
6	5.6	0111001	36	2.6	0011011	66	0.4	0000101
7	5.5	0111000	37	2.5	0011010	67	0.5	0000110
8	5.4	0110111	38	2.4	0011001	68	0.6	0000111
9	5.3	0110110	39	2.3	0011000	69	0.7	0001000
10	5.2	0110101	40	2.2	0010111	70	0.8	0001001
11	5.1	0110100	41	2.1	0010110	71	0.9	0001010
12	5	0110011	42	2	0010101	72	1	0001010
13	4.9	0110010	43	1.9	0010100	73	1.1	0001011
14	4.8	0110001	44	1.8	0010011	74	1.2	0001101
15	4.7	0110000	45	1.7	0010010	75	1.3	0001110
16	4.6	0101111	46	1.6	0010001	76	1.4	0001111
17	4.5	0101110	47	1.5	0010000	77	1.5	0010000
18	4.4	0101101	48	1.4	0001111	78	1.6	0010001
19	4.3	0101100	49	1.3	0001110	79	1.7	0010010
20	4.2	0101011	50	1.2	0001101	80	1.8	0010011
21	4.1	0101010	51	1.1	0001100	81	1.9	0010100
22	4	0101001	52	1	0001011	82	2	0010101
23	3.9	0101000	53	0.9	0001010	83	2.1	0010110
24	3.8	0100111	54	0.8	0001001	84	2.2	0010111
25	3.7	0100110	55	0.7	0001000	85	2.3	0011000
26	3.6	0100101	56	0.6	0000111	86	2.4	0011001
27	3.5	0100100	57	0.5	0000110	87	2.5	0011010
28	3.4	0100011	58	0.4	0000101	88	2.6	0011011
29	3.3	0100010	59	0.3	0000100			

Scenario B – J

Note: These scenarios fulfill the test requirements of §2.4.2.1.6.2 D Altitude Code Processing.

Input:

UUT: Z = (See below) at T=0
 ZDOT = 0 fpm from T=0 to 10

Int1: Equip = MODE C
 R = 2 NM at T=0
 Bearing = (See below) at T=0
 Bearing rate = 0 from T=0 to 10
 RDOT = -360 kt from T=0 to 10
 Z = (See below) from T=0 to 10
 ZDOT = 0 fpm from T=0 to 10
 Msgs = None

MSS: Msgs = UF=4, with RR=0 from T=0 to 10
 UF=4, with RR=19 from T=0 to 10

Vary the altitude and bearing as follows:

Scenario	Own Altitude (ft)	Intruder Altitude (ft)	Bearing (deg)
B	-1,000	-700	30
C	-500	-200	60
D	500	800	90
E	2,500	2,800	120
F	6,500	6,800	150
G	14,500	14,800	210
H	30,200	30,500	240
I	62,200	62,500	270
J	125,000	125,300	300

Expected Output:

Display: "Descend" RA

Msgs: DF=4 replies received every second with the following data:

bit 12 = 0 (T=0 until RA is issued)

bit 12 = 1 (while the RA is active)

DF=20 replies received every second with the following data:

DF=20 MB Field Contents														
RA Report														
BDS	ARA	LDI	RMF	RAC	RAT	MTE	CNT	TTI	TIDA	TIDR	TIDB	DSI	SPI	Sequence
48	0000000000	00	01	0000	0	0	0	0	0	0	0	0	0	Prior to RA
48	1010100000	00	01	0000	0	0	0	0	*alt	*rng	*brg	0	0	While RA is active

*Verify that the intruder altitude (alt), range (rng) and bearing (brg) are appropriately encoded into the TIDR, TIDA and TIDB fields for each second during which this information is transmitted in accordance with the following table. (within tolerance of ± 1 lsb)

For Scenarios B - J

Time	Range	TIDR
0	2	0010101
1	1.9	0010100
2	1.8	0010011
3	1.7	0010010
4	1.6	0010001
5	1.5	0010000
6	1.4	0001111
7	1.3	0001110
8	1.2	0001101
9	1.1	0001100
10	1	0001010

Scenario	Time	Altitude (ft)	TIDA	Bearing (deg)	TIDB
B	0-10	-700	00000000100	30	000101
C	0-10	-200	00000001001	60	001010
D	0-10	800	00000010011	90	001111
E	0-10	2,800	00000100111	120	010100
F	0-10	6,800	00001001111	150	011001
G	0-10	14,800	00100111111	210	100011
H	0-10	30,500	001001111100	240	101000
I	0-10	62,500	010011111100	270	101101
J	0-10	125,300	10011110000	300	110010

Scenario K

The UUT and TCAS II-equipped threat are on a horizontal collision course. The threat declares the conflict and selects a "Descend" advisory. Subsequently, the UUT declares the conflict and selects a "Climb". A 2nd Mode C equipped intruder then becomes an RA. The test verifies that the proper RA data is successfully transmitted to the ground sensors first for the Mode S intruder and then for the Mode C equipped intruder.

Input:

UUT: Z = 12,200 ft at T=0
 ZDOT = 0 fpm from T=0 to 90

Int1: Equip = TCAS II
 R = 6.2 NM at T=0
 RDOT = -360 kt from T=0 to 90
 Z = 12,000 ft from T=0 to 90
 ZDOT = 0 fpm from T=0 to 90
 Msgs = UF=16, in MU: UDS=48, MTB=0, CVC=0,
 VRC=1, CHC=0, HRC=0, VSB=14, MID=AAAAAAA from
 T=26 to 60

Int2: Equip = MODE C
 R = 7.2 NM at T=0
 Bearing = 0 deg at T=0
 Bearing rate = 0 from T=0 to 100
 RDOT = -360 kt from T=0 to 100
 Z = 12,000 ft from T=0 to 100
 ZDOT = 0 fpm from T=0 to 100
 Msgs = None

MSS: Msgs = UF=4, with RR=0 from T=0 to 100
 UF=4, with RR=19 from T=0 to 100

Expected Output:

Display: "Climb" RA

Msgs: DF=4 replies received every second with the following data:

Bit 12=0 (until the UUT RA is issued)

Bit 12=1 (while the UUT RA is active and for 18 ± 1 second after RA Termination)

Bit 12=0 (for the remainder of the scenario)

DF=20 replies received every second with the following data:

DF=20 MB Field Contents							
RA Report							
BDS	ARA	RAC	RAT	MTE	TTI	TID	Sequence
48	0000000000	0000	0	0	0	0	prior to UUT or INT1 RA being issued

48	0000000000	1000	0	0	0	0	after INT1 RA declared but prior to UUT RA		
48	1000100000	1000	0	0	1	AAAAAA	while both INT1 RA and UUT RA against INT1 are active		
BDS	ARA	RAC	RAT	MTE	TTI	TIDA	TIDR	TIDB	Sequence
48	1000100000	1000	0	1	0	*alt	*rng	*brg	while UUT RA is active against INT1 and INT2
48	1000100000	0000	0	0	0	*alt	*rng	*brg	After INT1 RA is terminated and UUT RA is active against INT2
48	1000100000	0000	0	0	0	*alt	*rng	*brg	For 18 ± 1 seconds after UUT RA termination
48	0000000000	0000	0	0	0	0	0	0	Remainder of scenario

Altitude (alt) = 00010000011

Range (rng) = per range table in scenario A

Bearing (brg) = 000001

Subfields LDI, CNT, DSI, SPI =0; RMF =01; See full RA Report format in tests 2.4.2.2.3.1A and 2.4.2.2.4.1A above.

DF=17 ADS-B TCAS RA Broadcast Messages received with the following data:

DF=17 ME Field Contents ADS-B TCAS RA Broadcast Message		
TYPE=28 Subtype=2	ME bits 41-88 same as MB bits 41-88 above.	Transmitted every 0.7-0.9s while RA is active and for 24 ± 1 seconds after RA is terminated

2.4.2.4.2 Transmission of RA Broadcast Interrogation

This test verifies that the TCAS properly transmits RA Broadcast Interrogation Messages at 1-second intervals for the period that an RA is active. The RA Broadcast Interrogation Messages describe the most recent RA that existed during the preceding 1-second period.

Scenario A

Input:

UUT:	Z	= 12,000 ft	at T = 0
	ZDOT	= 0 fpm	from T = 0 to 88
	Mode A Code	= 5134 (octal)	

Int1:	Equip	= MODE C	
	R	= 6.2 NM	at T = 0
	Bearing	= 45 deg	at T = 0
	Bearing rate	= 0	from T = 0 to 88
	RDOT	= -360 kt	from T = 0 to 88
	Z	= 12,200 ft	from T = 0 to 88
	ZDOT	= 0 fpm	from T = 0 to 88
	Msgs	= None	

Expected Output

Display: "Descend" RA
UF=16 RA Broadcast Interrogation Messages transmitted with the following data:

UF=16 MU Field Contents										
RA Broadcast Interrogation Message										
U D S	ARA	L D I	R M F	RAC	R A T	M T E	S P I	AID	CAC	Timing
49	1010100000	00	01	0000	0	0	0	1010010011100	0011000101010	Every second during RA.
49	1010100000	00	01	0000	1	0	0	1010010011100	0011000101010	Final transmission the second after RA is terminated

No additional RA Broadcast Interrogation Messages are transmitted following the ‘final’ transmission with $\text{RAT} = 1$.

Scenario B

Input:

UUT: Z = 12,200 ft at T = 0
 ZDOT = 0 fpm from T = 0 to 90
 Mode A Code = 2760 (octal)

Int1:	Equip	= TCAS II	
	R	= 6.2 NM	at T = 0
	RDOT	= -360 kt	from T = 0 to 90
	Z	= 12,000 ft	from T = 0 to 90
	ZDOT	= 0 fpm	from T= 0 to 90
	Msgs	= UF=16, in MU: UDS=48, MTB=0, CVC=0, VRC=1, CHC=0, HRC=0, VSB=14, MID=2 from T=26 to 60	

Expected Output

Display: “Climb” RA

UF=16 RA Broadcast Interrogation Messages transmitted with the following data:

UF=16 MU Field Contents RA Broadcast Interrogation Message										
U D S	ARA	L D I	R M F	RAC	R A T	M T E	S P I	AID	CAC	Timing
49	1000100000	00	01	0000	0	0	0	0101110110000	1001000101010	Every second during RA.
49	1000100000	00	01	0000	1	0	0	0101110110000	1001000101010	Final transmission the second after RA is terminated

No additional RA Broadcast Interrogation Messages are transmitted following the ‘final’ transmission with RAT = 1.

2.4.2.4.3 Coordination with Active CAS

2.4.2.4.3.1 Sense Selection and Communication (§2.2.3.9.3.1, §2.2.3.9.3.2)

The following test procedures verify that ACAS X correctly coordinates its resolution of the conflict with a threat that is also Active CAS-equipped.

Scenario A

This test verifies that the UUT correctly transmits a TCAS Resolution Message to the threat.

The UUT and an Active CAS-equipped threat are on a horizontal collision course. The UUT is the first to declare the conflict and selects a climb sense RA. All communications must be present and in the proper sequence.

Input:

UUT:	Z = 12,200 ft ZDOT = 0 fpm	at T = 0 from T = 0 to 90
Int1:	Equip = TCAS II R = 6.2 NM RDOT = -360 kt Z = 12,000 ft ZDOT = 0 fpm	at T = 0 from T = 0 to 90 from T = 0 to 90 from T = 0 to 90

DF=16 replies are sent with the following data:

DF=16 MV Field Contents Coordination Reply Message				
VDS	ARA	RAC	Timing	
48	Don't Care	Don't Care	In response to all UF=16 interrogations from the UUT	

Expected Output:

Display: “Climb” RA

UUT transmits UF=16 interrogations with following data:

UF=16 MU Field Contents TCAS Resolution Message									
UDS	MTB	CVC	VRC	CHC	HRC	HSB	VSB	MID	Timing
48	0	0	2	0	0	0	7	1	Every second during RA.

Scenario B

This test verifies that the UUT correctly selects the complementary vertical sense when an Active CAS-equipped threat has previously communicated its sense. In this case, the UUT

selects the complementary crossing sense because the threat has the lower ICAO 24-bit Aircraft Address.

The UUT and an Active CAS-equipped threat are on a horizontal collision course. The threat is level 200 feet below the UUT and selects a Climb advisory with respect to the UUT. Shortly thereafter, the UUT selects a descend sense advisory. All communications must be present and in the proper sequence.

Input:

UUT:	Z = 12,200 ft ZDOT = 0 fpm	at T = 0 from T = 0 to 90
------	-------------------------------	------------------------------

Int1:	Equip = TCAS II R = 6.2 NM RDOT = -360 kt Z = 12,000 ft ZDOT = 0 fpm	at T = 0 from T = 0 to 90 from T = 0 to 90 from T = 0 to 90
-------	--	--

Msgs:

Int1 transmits UF=16 interrogations with following data:

UF=16 MU Field Contents TCAS Resolution Message									
UDS	MTB	CVC	VRC	CHC	HRC	HSB	VSB	MID	Timing
48	0	0	2	0	0	0	7	1	Beginning at least one second prior to UUT RA.

Int1 transmits DF=16 replies with the following data:

DF=16 MV Field Contents Coordination Reply Message			
VDS	ARA	RAC	Timing
48	Don't Care	Don't Care	In response to all UF=16 interrogations from the UUT

Expected Output:

Display: Descend sense advisory (i.e. “Monitor Vertical Speed” with red arc above, or “Descend, Crossing Descend”)

UUT transmits UF=16 interrogations with following data:

UF=16 MU Field Contents TCAS Resolution Message									
UDS	MTB	CVC	VRC	CHC	HRC	HSB	VSB	MID	Timing
48	0	0	1	0	0	0	14	2	Every second during the UUT RA.

UUT transmits DF=16 replies with the following data:

DF=16 MV Field Contents Coordination Reply Message							
VDS	ARA	LDI	RMF	RAC	RAT	MTE	Sequence
48	0000000000	00	01	0000	0	0	In response to first UF=16 Interrogation from Int1
48	1110001000 Or 1110100000	00	01	0100	0	0	In response to subsequent UF-16 interrogations from Int1.*

*The ARA must match the aural advisory: "Monitor Vertical Speed" or "Descend, Crossing Descend".

Note: The UUT would normally select a Climb advisory based on the intruder surveillance information. The selection of a Descend advisory by the UUT is due to the coordination process.

Scenario C

This test verifies that the UUT selects the correct vertical sense when an Active CAS-equipped threat has previously communicated its sense. In this case, the UUT selects the (incompatible) non-crossing sense because the UUT has the lower ICAO 24-bit Aircraft Address and the threat has communicated its sense not more than three seconds prior to the UUT's selection.

The UUT and an Active CAS-equipped threat are on a horizontal collision course. The threat is level 200 feet below the UUT and selects a Climb advisory with respect to the UUT. Shortly thereafter, the UUT selects a Climb advisory. All communications must be present and in the proper sequence.

Input:

UUT: Z = 5,000 ft at T = 0
ZDOT = 0 fpm from T = 0 to 90

Int1: Equip = TCAS II
R = 6.2 NM at T = 0
RDOT = -360 kt from T = 0 to 90
Z = 4,800 ft from T = 0 to 90
ZDOT = 0 fpm from T = 0 to 90

Msgs:

Int1 transmits UF=16 interrogations with following data:

UF=16 MU Field Contents TCAS Resolution Message									
UDS	MTB	CVC	VRC	CHC	HRC	HSB	VSB	MID	Timing
48	0	0	2	0	0	0	7	2	At t=29s (1s before UUT RA.)
48	0	2	1	0	0	0	3	2	After UUT Climb RA has been issued. (Slave intruder reverses to the non-crossing downward sense)

Int1 transmits DF=16 replies with the following data:

DF=16 MV Field Contents Coordination Reply Message			
VDS	ARA	RAC	Timing
48	Don't Care	Don't Care	In response to all UF=16 interrogations from the UUT

Expected Output:

Display: "Climb" advisory at t=30s (1s after intruder UF16)

UUT transmits UF=16 interrogations with following data:

UF=16 MU Field Contents TCAS Resolution Message									
UDS	MTB	CVC	VRC	CHC	HRC	HSB	VSB	MID	Timing
48	0	1	2	0	0	0	12	1	Every second during the UUT climb RA.

UUT transmits DF=16 replies with the following data:

DF=16 MV Field Contents Coordination Reply Message							
VDS	ARA	LDI	RM F	RAC	RAT	MT E	Sequence
48	000000000	00	01	0000	0	0	In response to first UF=16 Interrogation from Int1
48	100010000	00	01	0100	0	0	In response to UF=16 interrogations from Int1 immediately after the UUT climb RA
48	100010000	00	01	1000	0	0	In response to subsequent UF=16 interrogations from Int1

Scenario D

This test verifies that the UUT correctly selects the complementary vertical sense when an Active CAS-equipped threat has previously communicated its sense. In this case, the UUT selects the complementary crossing sense because, although the UUT has the lower ICAO 24-bit Aircraft Address, the threat has communicated its sense more than three seconds prior to the UUT's selection.

The UUT and an Active CAS-equipped threat are on a horizontal collision course. The threat is level 200 feet below the UUT and selects a Climb advisory with respect to the

UUT. Shortly thereafter, the UUT selects a Descend advisory. All communications must be present and in the proper sequence.

Input:

UUT: Z = 5,000 ft	at T = 0
ZDOT = 0 fpm	from T = 0 to 90

Int1: Equip = TCAS II	
R = 6.2 NM	at T = 0
RDOT = -360 kt	from T = 0 to 90
Z = 4,800 ft	from T = 0 to 90
ZDOT = 0 fpm	from T = 0 to 90

Msgs:

Int1 transmits UF=16 interrogations with following data:

UF=16 MU Field Contents TCAS Resolution Message									
UDS	MTB	CVC	VRC	CHC	HRC	HSB	VSB	MID	Timing
48	0	0	2	0	0	0	7	2	At t=26s (4s before UUT RA)
48	0	0	2	0	0	0	7	2	After UUT Descend RA has been issued.

Int1 transmits DF=16 replies with the following data:

DF=16 MV Field Contents Coordination Reply Message			
VDS	ARA	RAC	Timing
48	Don't Care	Don't Care	In response to all UF=16 interrogations from the UUT

Expected Output:

Display: Monitor Vertical Speed or Crossing Descend at t=30s (4s after intruder UF16)

UUT transmits UF=16 interrogations with following data:

UF=16 MU Field Contents TCAS Resolution Message									
UDS	MTB	CVC	VRC	CHC	HRC	HSB	VSB	MID	Timing
48	0	0	1	0	0	0	14	1	Every second during the UUT Descend RA.

UUT transmits DF=16 replies with the following data:

DF=16 MV Field Contents Coordination Reply Message							
VDS	ARA	LDI	RMF	RAC	RAT	MTE	Sequence
48	0000000000	00	01	0000	0	0	In response to first UF=16 Interrogation from Int1
48	1110001000 Or 1110100000	00	01	0100	0	0	In response to subsequent UF-16 interrogations from Int1.*

*The ARA must match the aural advisory: "Monitor Vertical Speed" or "Descend, Crossing Descend".

Note: The UUT would normally select a Climb advisory based on the intruder surveillance information. The selection of a Descend advisory by the UUT is due to the coordination process.

Scenario E

This test verifies that the UUT correctly reverses its sense when it has the higher ICAO 24-bit Aircraft Address and discovers that an Active CAS-equipped threat has selected an incompatible sense.

The UUT and an Active CAS-equipped threat are on a horizontal collision course. The UUT is the first to declare the conflict and select a climb sense, sending a 'Do not pass above' to the threat. One second later, the threat also selects a climb sense. The UUT then reverses its sense and communicates a 'Do not pass below', 'Cancel Do not pass above' to the threat. The displayed advisory changes to a Descend.

Input:

UUT: Z = 12,200 ft at T = 0
ZDOT = 0 fpm from T = 0 to 90

Int1: Equip = TCAS II
R = 6.1 NM at T = 0
RDOT = -360 kt from T = 0 to 90
Z = 12,000 ft from T = 0 to 90
ZDOT = 0 fpm from T = 0 to 90

Msgs:

Int1 transmits UF=16 interrogations with following data:

UF=16 MU Field Contents TCAS Resolution Message									
UDS	MTB	CVC	VRC	CHC	HRC	HSB	VSB	MID	Timing
48	0	0	2	0	0	0	7	1	Beginning one second after the UUT declares an RA

Int1 transmits DF=16 replies with the following data:

DF=16 MV Field Contents Coordination Reply Message			
VDS	ARA	RAC	Timing
48	Don't Care	Don't Care	In response to all UF=16 interrogations from the UUT

Expected Output:

Display: "Climb" advisory that reverses to a "Descend, Descend Now" advisory

UUT transmits UF=16 interrogations with following data:

UF=16 MU Field Contents TCAS Resolution Message									
UDS	MTB	CVC	VRC	CHC	HRC	HSB	VSB	MID	Timing
48	0	0	2	0	0	0	7	2	Every second during the UUT Climb RA.
48	0	2	1	0	0	0	3	2	Every second during the UUT Descend RA.

UUT transmits DF=16 replies with the following data:

DF=16 MV Field Contents Coordination Reply Message			
VDS	ARA	RAC	Timing
48	1000100000	0000	In response to first UF=16 interrogation from Int1
48	1110101000	0100	In response to UF=16 interrogation from Int1 when UUT RA reverses to Descend advisory.
48	1110101000	0100	In response to subsequent UF=16 interrogations from Int1 while UUT Descend advisory is active

Subfields LDI, RAT, and MTE =0; RMF =01

Scenario F

This test verifies that the UUT does not reverse its sense when it has the lower ICAO 24-bit Aircraft Address and discovers that an Active CAS-equipped threat has selected an incompatible sense. The test also verifies that the UUT correctly updates its RAC field when the threat reverses its sense.

The UUT and an Active CAS-equipped threat are on a horizontal collision course. The UUT is the first to declare the conflict and select a climb sense. Shortly thereafter (within 500 milliseconds), the threat also selects a climb sense.

Input:

UUT: Z = 12,200 ft at T = 0
ZDOT = 0 fpm from T = 0 to 60

Int1: Equip = TCAS II
 R = 6.2 NM at T = 0
 RDOT = -360 kt from T = 0 to 60
 Z = 12,000 ft from T = 0 to 60
 ZDOT = 0 fpm from T = 0 to 60

Msgs:

Int1 transmits UF=16 interrogations with following data:

UF=16 MU Field Contents TCAS Resolution Message									
UDS	MTB	CVC	VRC	CHC	HRC	HSB	VSB	MID	Timing
48	0	0	2	0	0	0	7	2	First UF=16 transmission within 500 milliseconds after the UUT declares an RA
48	0	2	1	0	0	0	3	2	Subsequent UF=16 transmissions

Int1 transmits DF=16 replies with the following data:

DF=16 MV Field Contents Coordination Reply Message			
VDS	ARA	RAC	Timing
48	Don't Care	Don't Care	In response to all UF=16 interrogations from the UUT

Expected Output:

Display: "Climb" advisory

UUT transmits UF=16 interrogations with following data:

UF=16 MU Field Contents TCAS Resolution Message									
UDS	MTB	CVC	VRC	CHC	HRC	HSB	VSB	MID	Timing
48	0	0	2	0	0	0	7	1	Every second during the UUT Climb RA.

UUT transmits DF=16 replies with the following data:

DF=16 MV Field Content Coordination Reply Message			
VDS	ARA	RAC	Timing
48	1000100000	0000	In response to first UF=16 interrogation from Int1
48	1000100000	0100	In response to next t UF=16 from Int1
48	1000100000	1000	In response to subsequent UF=16 interrogations from Int1

Subfields LDI, RAT, and MTE =0; RMF =01.

Scenario G

This test verifies that the UUT is able to correctly handle the absence of a reply to its TCAS Resolution Message. The UUT has the lower ICAO 24-bit Aircraft Address.

Input:

UUT:	Z = 12,200 ft	at T = 0
	ZDOT = 0 fpm	from T = 0 to 31

Int1:	Equip = TCAS II	at T = 0
	R = 6.2 NM	from T = 0 to 60
	RDOT = -360 kt	from T = 0 to 60
	Z = 12,000 ft	from T = 0 to 60
	ZDOT = 0 fpm	from T = 0 to 60

Msgs:

Int1 transmits DF=16 replies with the following data:

DF=16 MV Field Contents Coordination Reply Message			
VDS	ARA	RAC	Timing
48	Don't Care	Don't Care	In response to the UF=16 interrogation from UUT during the second and subsequent cycles of the UUT RA. (Int1 does not reply to UF=16 interrogations during the first cycle of the RA.)

Expected Output:

Display: "Climb" advisory

UUT transmits UF=16 interrogations with following data:

UF=16 MU Field Contents TCAS Resolution Message									
UDS	MTB	CVC	VRC	CHC	HRC	HSB	VSB	MID	Timing
48	0	0	2	0	0	0	7	1	Transmitted between 6 and 12 times throughout a 100 millisecond period when a valid reply is not received (first cycle of RA) and a single transmission when a valid reply is received (second and subsequent cycles of RA).

Scenario H

This test verifies that the UUT correctly communicates cancellation of a vertical RA to an Active CAS-equipped threat. It also verifies that the UUT correctly processes a cancellation indication from the threat. The UUT has the higher ICAO 24-bit Aircraft Address.

In this scenario, the UUT removes the intruder from threat status twice. The first instance is due to the threat sending a “cancel vertical complement” to the UUT. The second instance is due to the disappearance of threat replies and subsequent dropping the track by surveillance.

Input:

UUT:	Z = 12,200 ft	at T = 0
	ZDOT = 0 fpm	from T = 0 to 64

Int1:	Equip = TCAS II	
	R = 6.2 NM	at T = 0
	RDOT = -360 kt	from T = 0 to 60
	Z = 12,000 ft	from T = 0 to 60
	ZDOT = 0 fpm	from T = 0 to 60

Other: Int1 stops sending surveillance updates (DF=0 replies after T=30 (i.e., no track reports sent to TRM after T=35)

Msgs:

Int1 transmits UF=16 interrogations with following data:

UF=16 MU Field Contents TCAS Resolution Message									
UDS	MTB	CVC	VRC	CHC	HRC	HSB	VSB	MID	Timing
48	0	0	1	0	0	0	14	1	UF=16 transmissions at T=10, 11.
48	0	1	0	0	0	0	11	1	UF=16 transmission at T=12.
48	0	0	0	0	0	0	0	1	UF=16 transmission at T=13
48	0	0	2	0	0	0	7	1	UF=16 transmission beginning same cycle as when UUT issues RA until T=30.
48	0	0	0	0	0	0	0	1	UF=16 transmission after UUT RA is terminated

Int1 transmits DF=16 replies with the following data:

DF=16 MV Field Contents Coordination Reply Message			
VDS	ARA	RAC	Timing
48	Don't Care	Don't Care	In response to all UF=16 interrogations from the UUT

Mode S Sensor transmits UF = 20 interrogations with the following data:

UF=20 Interrogation Contents							
UF(5)	PC(3)	RR(5)	DI(3)	SD(16)	MA(56)	AP (24)	Timing
10100	000	10011	7	All zeros	M/CH=5	Addressed to UUT (MID=2)	Once per second for duration of the scenario.

Expected Output:

Display: Descend sense advisory (i.e. “Monitor Vertical Speed” with red arc above)

UUT transmits UF=16 interrogations with following data:

UF=16 MU Field Contents TCAS Resolution Message									
UDS	MTB	CVC	VRC	CHC	HRC	HSB	VSB	MID	Timing
48	0	0	1	0	0	0	14	2	Every second during the UUT RA.*
48	0	1	0	0	0	0	11	2	The second after the UUT RA is terminated because int1 track is dropped. (T=36) *

**Note: The UUT UF=16 transmissions should be repeated a minimum of 6 times and a maximum of 12 times during a 100 millisecond period when Int1 does not reply.*

UUT transmits DF=16 replies with the following data:

DF=16 MV Field Contents Coordination Reply Message			
VDS	ARA	RAC	Timing
48	0000000000	0000	In response to first UF=16 interrogation from Int1 (T=10)
48	0000000000	1000	In response to second and third UF-16 interrogation from Int1 (T=11,12)
48	0000000000	0000	In response to fourth UF=16 interrogation from Int1 (T=13)
48	0000000000	0000	In response to UF-16 interrogation from Int1 same cycle as RA Issued
48	1110001000	0100	In response to subsequent UF-16 interrogation from Int1 while RA is active.
48	0000000000	0000	In response to the last UF-16 interrogation from Int1

Subfields LDI, RAT, and MTE =0; RMF =01.

DF=20 replies received every second with the following data:

DF=20 MB Field Contents RA Report								
BDS	ARA	RAC	RAT	MTE	TTI	TID		Sequence
48	0000000000	0000	0	0	0	0		Prior to Int1 VRC
48	0000000000	1000	0	0	1	0		While to Int1 VRC is valid (T=11 – 12)
48	0000000000	0000	0	0	1	0		After Int1 VRC is cancelled
48	1110001000	0100	0	0	1	4		While UUT RA is active
48	1110001000	0100	1	0	1	4		After RA is terminated and for 18 ± 1 seconds
48	0000000000	0000	0	0	0	0		Remaining replies

Subfields LDI, CNT, DSI, SPI =0; RMF =01; See full RA Report format in tests §2.4.2.3.1A and §2.4.2.4.1A above.

Scenario I

This test verifies that the UUT correctly times out/clears the RAC subfield when the intruder stops sending UF=16 Resolution Coordination Messages without sending a cancellation (CVC) message.

Input:

UUT: Z = 12,200 ft
ZDOT = 0 fpm

Int1: Equip = TCAS II
R = 4 NM
RDOT = 0 kt
Z = 12,000 ft
ZDOT = 0 fpm

Msgs:

Int1 transmits UF=16 interrogations with following data:

UF=16 MU Field Contents TCAS Resolution Message									
UDS	MTB	CVC	VRC	CHC	HRC	HSB	VSB	MID	Timing
48	0	0	1	0	0	0	14	1	UF=16 interrogations from Int1 once per second for 5 seconds starting at T= 10 seconds.
48	0	0	0	0	0	0	0	1	UF=16 transmission sent at T = 22 seconds.

Mode S Sensor transmits UF = 20 interrogations with the following data:

UF=20 Interrogation Contents							
UF(5)	PC(3)	RR(5)	DI(3)	SD(16)	MA(56)	AP (24)	Timing
10100	000	10011	7	IIS=0001 ADS=0x5 All other fields=All zeros	All zeros	Addressed to UUT (MID=1)	Once per second

Expected Output:

UUT transmits DF=16 replies with the following data:

DF=16 MV Field Contents Coordination Reply Message				
VDS	ARA	RAC	Timing	
48	0000000000	0000	In response to UF=16 interrogation from Int1 at T = 10 seconds	
48	0000000000	1000	In response to the four subsequent UF-16 interrogations from Int1 starting at T = 11 seconds.	
48	0000000000	0000	In response to final UF-16 interrogation sent at T = 22 seconds.	

Subfields LDI, RAT, and MTE =0; RMF =01.

DF=20 replies received every second with the following data:

DF=20 MB Field Contents RA Report								
BDS	ARA	RAC	RAT	MTE	TTI	TID		Sequence
48	0000000000	0000	0	0	0	0		Prior to UF=16 transmission by Int1
48	0000000000	1000	0	0	0	0		After first UF=16 transmission from Int1 has been sent

48	0000000000	0000	0	0	0	0			7 seconds after last UF=16 transmission from Int1.
----	------------	------	---	---	---	---	--	--	--

Subfields LDI, CNT, DSI, SPI =0; RMF =01; See full RA Report format in tests §2.4.2.3.1A and §2.4.2.4.1A above.

Scenario J

This test verifies that the UUT is able to correctly handle the absence of a reply to its TCAS Resolution Message during transmission of the cancellation of the RA and that it retransmits that message for six update cycles if a reply is not received.

Input:

UUT: Z = 12,200 ft at T = 0
ZDOT = 0 fpm from T = 0 to 70

Int1: Equip = TCAS II
R = 6.2 NM at T = 0
RDOT = -360 kt from T = 0 to 70
Z = 12,000 ft from T = 0 to 70
ZDOT = 0 fpm from T = 0 to 70

Msgs:

Int1 transmits DF=16 replies with the following data:

DF=16 MV Field Contents Coordination Reply Message				
VDS	ARA	RAC	Timing	
48	Don't Care	Don't Care	In response to the UF=16 interrogation from UUT for each second during which the RA is active. Int1 does not reply to UF=16 interrogations once the RA has concluded and the cancellation of the RA is transmitted.	

Expected Output:

Display: "Climb" advisory
UUT transmits UF=16 interrogations with following data:

UF=16 MU Field Contents TCAS Resolution Message									
UDS	MTB	CVC	VRC	CHC	HRC	HSB	VSB	MID	Timing
48	0	0	2	0	0	0	7	1	Transmitted each cycle during the RA when a valid reply is received.

48	0	2	0	0	0	D	1	Transmitted between 6 and 12 times throughout a 100 millisecond period during each of the six cycles after the RA has concluded. (Note: Intruder does not reply)
----	---	---	---	---	---	---	---	--

2.4.2.4.3.2 Validity Check for ACAS X Communications Link with Another Active CAS

This test verifies that an Active CAS-equipped intruder will be declared a threat even if the communication link with that threat is determined to be intermittent. As long as the intruder is tracked, ACAS X will declare a threat and attempt coordination.

Scenario A

This scenario is identical to that of §2.4.2.4.3.1, Scenario A, except that surveillance replies from the intruder are intentionally corrupted or missing so that the UUT receives replies to its surveillance interrogations only every fifth cycle, i.e., four misses, one reply, four misses, one reply, etc. In order for the intruder track to be properly established, the UUT should receive valid surveillance replies from the intruder for the first 15 seconds of the scenario. After 15 seconds, the pattern of four misses, one reply should be maintained.

The UUT and an Active CAS-equipped threat are on a horizontal collision course. The UUT should detect a conflict and select a climb sense RA.

Input:

UUT: Z = 12,200 ft at T = 0
ZDOT = 0 fpm from T = 0 to 38

Int1: Equip = TCAS II
R = 6.2 NM at T = 0
RDOT = -360 kt from T = 0 to 38
Z = 12,000 ft from T = 0 to 38
ZDOT = 0 fpm from T = 0 to 38

DF=0 replies are sent every second for T=0 through T=14
DF=0 replies are sent every fifth second for T=15 through T=38

DF=16 replies are sent with the following data:

DF=16 MV Field Contents Coordination Reply Message			
VDS	ARA	RAC	Timing
48	Don't Care	Don't Care	In response to all UF=16 interrogations from the UUT

Expected Output:

Display: "Climb" RA
UUT transmits UF=16 interrogations with following data:

UF=16 MU Field Contents										
UDS	MTB	CVC	VRC	CHC	HRC	HSB	VSB	MID	Timing	
48	0	0	2	0	0	0	7	1	Every second during RA.	

2.4.2.4.3.3 ACAS X in Multi-Aircraft Conflict (§2.2.3.8.3.2.3.1, §2.2.3.9.3.2)

This test procedure verifies that ACAS X correctly transmits its intent message when in a multi-aircraft conflict with another Active CAS-equipped aircraft and correctly identifies the most recently declared threat in RA Reports to a Mode S ground sensor.

Scenario A

This test verifies that the multi-threat bit is correctly set in the UUT intent messages when the UUT is in a multi-aircraft conflict. UUT's ICAO 24-bit Aircraft Address is lower than that of the threat.

Input:

UUT: Z = 12,200 ft at T = 0
ZDOT = 0 fpm from T = 0 to 50

Int1: Equip = MODE C
R = 9.6 NM at T = 0
RDOT = -720 kt from T = 0 to 50
Z = 12,000 ft from T = 0 to 50
ZDOT = 0 fpm from T = 0 to 50

Int2: Equip = TCAS II
R = 10.6 NM at T = 0
RDOT = -720 kt from T = 0 to 50
Z = 12,600 ft from T = 0 to 50
ZDOT = 0 fpm from T = 0 to 50

DF=16 replies are sent with the following data:

DF=16 MV Field Contents			
Coordination Reply Message			
VDS	ARA	RAC	Timing
48	Don't Care	Don't Care	In response to all UF=16 interrogations from the UUT

MSS: Msgs = UF=4, with RR=19 from T=0 to 50

Expected Output:

Display: "Monitor Vertical Speed" with red arc below, followed by a Multi-threat "Level-off" with red arc above and below and green arc in between

UUT transmits UF=16 interrogations with following data:

UF=16 MU Field Contents TCAS Resolution Message										
UDS	MTB	CVC	VRC	CHC	HRC	HSB	VSB	MID	Timing	
48	1	0	1	0	0	0	14	1	Every second during RA against TCAS II equipped intruder.	

UUT transmits DF=20 replies with the following data:

		DF=20 MB Field Contents RA Report														
BD S	ARA	LD I	RM F	RAC	RA T	MT E	CN T	TT I	TID	TID A	TID R	TID B	DS I	SP I	Sequenc e	
48	0000000000 0	00	01	000 0	0	0	0	0	NA	0	0	0	0	0	Prior to RA	
48	100000100 0	00	01	000 0	0	0	0	0	NA	alt*	rng*	brg*	0	0	While RA against only Mode C intruder is active	
48	000111000 0	00	01	000 0	0	1	0	1	ICA O addr of Int1	NA	NA	NA	0	0	While multi-threat RA is active (through end of encounter)	

For the 14 seconds that the RA is active against only the Mode C intruder, alt, rng, and brg take on the following values:

TIDA	TIDR	TIDB	Sequence
000010000011	1000101	000001	1st cycle of Mode C RA
000010000011	1000011	000001	2nd cycle of Mode C RA
000010000011	1000001	000001	3rd cycle of Mode C RA
000010000011	0111111	000001	4th cycle of Mode C RA
000010000011	0111101	000001	5th cycle of Mode C RA
000010000011	0111011	000001	6th cycle of Mode C RA
000010000011	0111001	000001	7th cycle of Mode C RA
000010000011	0110111	000001	8th cycle of Mode C RA
000010000011	0110101	000001	9th cycle of Mode C RA
000010000011	0110011	000001	10th cycle of Mode C RA
000010000011	0110001	000001	11th cycle of Mode C RA
000010000011	0101111	000001	12th cycle of Mode C RA
000010000011	0101101	000001	13th cycle of Mode C RA
000010000011	0101011	000001	14th cycle of Mode C RA

2.4.2.4.3.4 Coordination Monitor (§2.2.3.8.3.2.4.1, §2.2.7.2.2)

This test verifies proper operation of the coordination monitor in the ACAS X computer.

Scenario A

This test procedure verifies that if a TCAS Resolution Message received from a threat contains an illegal value or a parity error, the vertical resolution data in the message is not used by own ACAS X.

The Active CAS-equipped threat is 3 NM away and stationary with respect to the UUT. The threat declares a conflict and selects a climb sense. Subsequent UF=16 TCAS Resolution Messages are transmitted with invalid and valid intent values. The UUT's replies are monitored to determine if the invalid messages are ignored and the valid messages are processed. The UUT has the lower ICAO 24-bit Aircraft Address.

Input:

UUT:	Z = 12,200 ft	at T = 0
ZDOT =	0 fpm	from T = 0 to 50

Int1:	Equip = TCAS II	
R =	3 NM	at T = 0
RDOT =	0 kt	from T = 0 to 50
Z =	12,000 ft	from T = 0 to 50
ZDOT =	0 fpm	from T = 0 to 50

Msgs:

Int1 transmits DF=16 replies with the following data

DF=16 MV Field Contents Coordination Reply Message			
VDS	ARA	RAC	Timing
48	Don't Care	Don't Care	In response to all UF=16 interrogations from the UUT

Int1 transmits UF=16 interrogations with following data:

UF=16 MU Field Contents TCAS Resolution Message									
UDS	MTB	CVC	VRC	CHC	HRC	HSB	VSB	MID	Timing
48	0	0	1	0	0	0	14	2	T= 10 (Valid message)
48	0	3	2	0	0	0	1	2	T=11 (Invalid message)
48	0	0	2	0	0	0	14	2	T=12 (Invalid message)
48	0	1	0	0	0	0	11	2	T=13 (Valid message)
48	0	0	1	0	0	0	7	2	T=14 (Invalid message)
48	0	1	2	0	0	0	14	2	T=15 (Invalid message)
48	0	0	2	0	0	0	7	2	T=16 (Valid message)
48	0	1	2	0	0	0	7	2	T=17 (Invalid message)
48	0	1	0	0	0	0	7	2	T=18 (Invalid message)
48	0	2	0	0	0	0	13	2	T=19 (Valid message)
48	0	0	0	0	0	0	11	2	T=20 (Invalid message)
48	0	0	2	0	0	0	6	2	T=21 (Invalid message)
48	0	0	1	0	0	0	14	2	T=22 (Valid message)
48	0	1	0	0	0	0	11	2	T=23 (Valid message)
48	0	0	0	0	0	0	0	2	T=24 (Invalid message)
48	0	1	2	0	0	0	12	2	T=25 (Valid message)
48	0	1	2	0	0	0	11	2	T=26 (Invalid message)
48	0	0	1	0	0	0	0	2	T=27 (Invalid message)
48	0	2	1	0	0	0	3	2	T=28 (Valid message)
48	0	1	0	0	0	0	9	2	T=29 (Invalid message)
48	0	2	2	0	0	0	4	2	T=30 (Invalid message)

Expected Output:

Display: No RA.

UUT transmits DF=16 replies with the following data:

DF=16 MV Field Contents Coordination Reply Message			
VDS	ARA	RAC	Timing
48	00000000000	0000	T=10
48	00000000000	1000	T=11 - 13
48	00000000000	0000	T=14 - 16
48	00000000000	0100	T=17 - 19
48	00000000000	0000	T=20 - 22
48	00000000000	1000	T=23
48	00000000000	0000	T=24 - 25

48	0000000000	0100	T=26 - 28
48	0000000000	1000	T=28 - 30

Subfields LDI, RAT, and MTE =0; RMF =01.

2.4.2.4.3.5 Transponder to ACAS X Interface and Transponder/ACAS X Throughput (&2.2.3.12.2.1.4, &2.2.3.9.3.3)

This testing verifies correct operation of the transponder and ACAS X unit in high-density coordination situations. The following four scenarios test that, in the presence of significant numbers of 1030 MHz TCAS Broadcast Interrogation Messages, the transponder provides TCAS Resolution Messages (1030 MHz discretely-addressed coordination interrogations) to ACAS X in a timely manner and that ACAS X properly processes the received TCAS Resolution Messages.

When this testing is performed by ACAS X paired with a DO-181F or later transponder, the system under test is required to demonstrate that TCAS Resolution Messages have priority over TCAS Broadcast Interrogations Messages (1030 MHz broadcast transmissions used in Interference Limiting). When this testing is performed by ACAS X paired with a DO-181E or earlier transponder, such demonstration is not required.

Verification associated with these tests may be accomplished through internal data recording, monitoring of interface data if applicable, or other methods proposed by the manufacturer.

Scenario A

This test verifies that the transponder and ACAS X are able to receive and process incoming TCAS Resolution Messages and TCAS Broadcast Interrogation Messages under the most extreme conditions expected.

Maximum expected unique TCAS Resolution Messages and TCAS Broadcast Interrogation Messsages

Interrogate the Mode S transponder with the following:

50 Unique TCAS Broadcast Interrogation Messages, followed by
6 Unique TCAS Resolution Messages

All transmitted within a 100 msec burst every second for 10 seconds.

UF=16 MU Field Contents										
TCAS Resolution Message										
UDS	MTB	CVC	VRC	CHC	HRC	HSB	VSB	MID	Timing	
48	0	0	2	0	0	0	7	1	1 per second	
48	0	0	1	0	0	0	14	2	1 per second	
48	0	0	2	0	0	0	7	3	1 per second	
48	0	0	1	0	0	0	14	4	1 per second	
48	0	0	2	0	0	0	7	5	1 per second	
48	0	0	1	0	0	0	14	6	1 per second	

UF=16 MU Field Contents									
TCAS Broadcast Interrogation Message									
UDS	MID							Timing	
50	1							1 per second	
50	2							1 per second	
50	...							1 per second	
50	50							1 per second	

Verify that:

- a. The transponder replies to all 6 of the TCAS Resolution Messages and successfully transmits data to ACAS X for all 6 messages in the order input to the transponder.
- b. The first TCAS Resolution Message data is received by ACAS X within 10 ms of receipt by the transponder.
Item c is required only when testing with ED-73F/DO-181F or later transponders
- c. Data for the remaining 5 TCAS Resolution Messages are transmitted to ACAS X as quickly as the interface permits and that any waiting TCAS Resolution Message is given priority on the interface over any waiting TCAS Broadcast Interrogation Messages.
- d. All TCAS Resolution Messages are processed by ACAS X in the order in which they are received.
- e. All TCAS Broadcast Interrogation Message data is successfully transmitted to and processed by ACAS X.

Scenario B

This test verifies that the transponder and ACAS X are able to receive and process incoming TCAS Resolution Messages and TCAS Broadcast Interrogation Messages under the most extreme conditions expected.

Maximum Expected Re-Interrogation Rate and Transponder to ACAS X Transmission Rate

Interrogate the Mode S transponder with the following:

10 Unique TCAS Broadcast Interrogation Messages, followed by
1 TCAS Resolution Message

The 11-message block is transmitted 6 times (i.e., 66 messages) within a 100 msec burst every second for 10 seconds.

During every one-second interval, each TCAS Resolution Message and each TCAS Broadcast Interrogation Message have a unique MID subfield as shown below.

UF=16 MU Field Contents									
TCAS ResolutionMessage									
UDS	MTB	CVC	VRC	CHC	HRC	HSB	VSB	MID	Timing
48	0	0	2	0	0	0	7	1	1 per second
48	0	0	1	0	0	0	14	2	1 per second
48	0	0	2	0	0	0	7	3	1 per second

48	0	0	1	0	0	0	14	4	1 per second
48	0	0	2	0	0	0	7	5	1 per second
48	0	0	1	0	0	0	14	6	1 per second
UF=16 MU Field Contents									
TCAS Broadcast Interrogation Message									
UDS	MID						Timing		
50	1						1 per second		
50	2						1 per second		
50	...						1 per second		
50	60						1 per second		

Verify that:

- a. The transponder replies to all 6 of the TCAS Resolution Messages and successfully transmits data to ACAS X in the order input to the transponder within 10 msec of receipt by the transponder.
- b. All TCAS Resolution Messages are processed by ACAS X in the order in which they are received.
- c. All TCAS Broadcast Interrogation Message data is successfully transmitted to and processed by ACAS X.

Scenario C

This test verifies that the transponder and ACAS X are able to receive and process incoming TCAS Resolution Messages and TCAS Broadcast Interrogation Messages correctly when the transponder is not able to reply to all interrogations and while under the most extreme conditions expected.

Exceed Transponder Reply Capability

Interrogate the Mode S transponder with the following:

10 Unique TCAS Broadcast Interrogation Messages, followed by
3 Unique TCAS Resolution Messages

The 13-message block is transmitted 15 times (i.e., 195 messages) within a 100 msec burst every second for 10 seconds.

UF=16 MU Field Contents									
TCAS Resolution Message									
UDS	MTB	CVC	VRC	CHC	HRC	HSB	VSB	MID	Timing
48	0	0	2	0	0	0	7	1	15 per second
48	0	0	1	0	0	0	14	2	15 per second
48	0	0	2	0	0	0	7	3	15 per second
UF=16 MU Field Contents									
TCAS Broadcast Interrogation Message									
UDS	MID						Timing		
50	1						15 per second		
50	2						15 per second		
50	...						15 per second		
50	10						15 per second		

Verify that:

- a. The transponder replies to TCAS Resolution Messages according to reply rate limiting and successfully transmits data to ACAS X in the order input to the transponder for each interrogation for which a reply is made.
Item b is required only when testing with ED-73F/DO-181F or later transponders
- b. Data for any incoming TCAS Resolution Message is transmitted to ACAS X as quickly as the interface permits and that any waiting TCAS Resolution Message is given priority on the interface over any waiting TCAS Broadcast Interrogation Messages.
- c. All TCAS Resolution Messages received by ACAS X are processed by ACAS X in the order in which they are received.
- d. TCAS Broadcast Interrogation Message data is successfully transmitted to ACAS X as per bus throughput capabilities. It is not required that all broadcast data be transmitted to ACAS X.

If the ACAS X is not able to accommodate the TCAS Resolution Messages received from the transponder verify that the ACAS X:

- indicates to the flight crew that an abnormal condition exists,
- causes any Mode S transmissions that report ownship status to show that ownship has no on-board resolution capability,
- prevents interrogations by own ACAS X, and
- deactivates the normal ACAS X display functions.

Scenario D

This test verifies that the transponder and ACAS X are able to receive and process incoming TCAS Resolution Messages and TCAS Broadcast Interrogation Messages correctly when the transponder is not able to reply to all interrogations and while under the most extreme conditions expected.

Exceed Transponder to ACAS X transmission throughput

Interrogate the Mode S transponder with the following:

20 Unique TCAS Broadcast Interrogation Messages, followed by
 2 Unique Sensitivity Level Commands, followed by
 10 Unique TCAS Resolution Messages

The 32-message block is transmitted 6 times (i.e., 192 messages) within a 600 msec burst every second for 10 seconds
 Increase the transmission rate until the system cannot accommodate the data throughput.

UF=16 MU Field Contents TCAS Resolution Message									
UDS	MTB	CVC	VRC	CHC	HRC	HSB	VSB	MID	Timing
48	0	0	2	0	0	0	7	1	6 per second
48	0	0	1	0	0	0	14	2	6 per second
48	0	0	2	0	0	0	7	3	6 per second
48	0	0	1	0	0	0	14	4	6 per second
48	0	0	2	0	0	0	7	5	6 per second
48	0	0	1	0	0	0	14	6	6 per second
48	0	0	2	0	0	0	7	7	6 per second
48	0	0	1	0	0	0	14	8	6 per second
48	0	0	2	0	0	0	7	9	6 per second
48	0	0	1	0	0	0	14	10	6 per second

UF=16 MU Field Contents TCAS Broadcast Interrogation Message									
UDS	MID				Timing				
50	1				6 per second				
50	2				6 per second				
50	...				6 per second				
50	20				6 per second				

UF=20 MA Field Contents Sensitivity Level Command			
UDS	SLC	IIS	Timing
50	2	1	6 per second
50	2	2	6 per second

Verify that:

- a. The transponder replies to TCAS Resolution Messages according to reply rate limiting and successfully transmits data to ACAS X for each interrogation for which a reply is made.
Item b is required only when testing with DO-181F or later transponders
- b. Data for any incoming TCAS Resolution Message is received by ACAS X as quickly as the interface/ARINC 429 bus permits and that any waiting TCAS Resolution Message is given priority on the interface/ARINC 429 bus over any waiting TCAS Broadcast Interrogation Messages.
- c. All TCAS Resolution Messages received by ACAS X are processed by ACAS X in the order in which they are received.
- d. TCAS Broadcast Interrogation Message data is successfully transmitted to ACAS X as throughput permits. It is not required that all broadcast data be transmitted to ACAS X.

Verify the UF=20 Sensitivity Level Commands have no adverse effects on the system operation (no system failures or SL change to ‘TA-Only’).

If the ACAS X is not able to accommodate the TCAS Resolution Messages received from the transponder verify that the ACAS X:

- indicates to the flight crew that an abnormal condition exists,
- causes any Mode S transmissions that report ownship status to show that ownship has no on-board resolution capability,

- prevents interrogations by own ACAS X, and
- deactivates the normal ACAS X display functions.

2.4.2.4.3.6 Coordination Timing (§2.2.3.9.3.4)

This test verifies that the coordination timing requirements (§2.2.3.9.3.4) are met. The verifications may be performed through actual measurements, analysis of recorded timing data or through other analysis or method proposed by the manufacturer. In the case of analysis there must be sufficient evidence to support that the timing has been appropriately evaluated (i.e. timing analysis during development or with instrumented code).

The following two timing measurements are to be made while the ACAS X is subjected to a worst case execution timing scenario. This scenario at a minimum includes the combined surveillance target capacity defined in §2.4.2.1.8.1, transmission of the UF=16 messages defined in Scenario A of §2.4.2.4.3.5 and simulation of a single TCAS II/ACAS X intruder aircraft for which an RA is generated. The measurements should be taken over at least a 5 minute timeframe.

Measurement 1) Timing of transmission of UF=16 TCAS Resolution Messages

Measure the time from sense selection of the advisory upon completion of StateAndCostEstimation: DO-385 Volume II, Algorithm 144, Line 55 until the UF=16 resolution coordination message is transmitted. Verify the timing meets the requirements for “T1” (≤ 11 msec for 95% of transmissions, ≤ 25 msec for remainder) for the duration of the advisory.

Measurement 2) Timing of reception of VRC from transponder until TRM completes sense selection processing

Verify the timing requirements for T3, T4, and T5. The reception time of the VRC is defined as the time at which the complete intent message containing the VRC is received at unit level. For example, in a system that communicates the VRC over ARINC 429 messages then the VRC is considered received by the ACAS X unit when the last bit of the ARINC 429 message conveying the VRC is received by unit at its electrical interface. The T3, T4, T5 timing requirement must be met no matter what delays are introduced by operating system, interrupt priority structure, or other design choices implemented by the equipment manufacturer.

In order to verify the overall T3, T4, T5 timing requirements an equipment manufacturer must allocate the time budget to T3 and T5 so that the timing requirements of §2.2.3.9.3.4 are satisfied. (Note: T4 is defined as zero.) One way to verify T3 timing is to show that all intent messages are passed to the ADD STM routine receiveUF16UDS30 within the time allocated by the manufacturer to T3 before UpdateIntruderInputs in Volume II begins to execute. Another method to show that T3 timing is met is to demonstrate that at the time UpdateIntruderInputs in Volume II begins to execute that all VRC/intent messages received by the unit have been processed by the ADD STM routine receiveUF16UDS30.

Furthermore the manufacturer must also verify T5 timing.

Note: Timing of transponder reception of the UF=16 TCAS Resolution Message until transmission to ACAS X (“T2”) is verified as part of transponder testing. For ACAS X units that have an integrated transponder this timing should be taken in context of the overall delay between “T2” and “T5”.

2.4.2.4.3.7 Use of Received Collision Avoidance Coordination Capability Bits (CCCB) (§2.2.3.9.3.1, §2.2.3.9.3.6.1.2, §2.2.3.9.3.6.2.1)

This test verifies proper receipt and use of an intruder's CCCB to determine whether to coordinate with the intruder and if so, what type of coordination protocol to perform. The test checks that ACAS X properly determines ownship master/slave relationship and intruder equipage and that the proper coordination messages (TCAS Resolution Message or ADS-B Operational Coordination Message (OCM)), if any, are transmitted.

The test consists of twelve scenarios, A – L. Scenarios A - K correspond to the eleven intruder types shown immediately below. Scenario L tests OCM prioritization.

- A. Active CAS (TCAS II)
RI=3; CCCB=0000000; DAA=00
- B. Active CAS (not TCAS II)
RI=3; CCCB=0000100; DAA=00
- C. Active CAS (not TCAS II) with OCM transmit capability
RI=3; CCCB=0001000; DAA=00
- D. DAA/TCAS II Combination
RI=3; CCCB=0000000; DAA=01
- E. Responsive 1030/1090 CAS
RI=2; CCCB=0001100; DAA=00
- F. Passive CAS with 1030 TCAS Resolution Message receive capability
RI=0; CCCB=0010000; DAA=00
- G. DAA-only requesting TCAS Resolution Message
RI=0; CCCB=0000000; DAA=01
- H. Mode S but neither CA nor DAA
RI=0; CCCB=0000000; DAA=00
- I. Passive CAS with only OCM receive capability
RI=N/A; CCCB=0010100; DAA=00
- J. DAA-only requesting OCM
RI=N/A; CCCB=0000000; DAA=10
- K. Mode C but neither CA or DAA
RI=N/A; CCCB=0000000; DAA=00

Scenarios I, J, K, and L can have different results depending on the capability of the Mode S transponder with which the UUT is paired. More details are found at the beginning of scenario I.

In each scenario, an encounter is set up in which the intruder is broadcasting ADS-B Aircraft Operational Status Messages containing CA Operational, CCCB, and DAA fields corresponding to its intruder type. It may additionally transmit UF=16 TCAS Resolution Messages, DF=16 Coordination Replies, and DF=0 surveillance replies as appropriate.

Throughout each scenario, a second Active CAS-equipped aircraft interrogates the UUT to request a read-out of transponder register 33₁₆, to determine whether an OCM has been stored into transponder register 33₁₆.

Scenario A

The UUT and an Active CAS-equipped threat are on a horizontal collision course. The UUT is the first to declare the conflict and selects a climb sense RA. Five seconds later, the threat also selects a climb sense RA.

Run this scenario twice, alternating higher and lower ICAO 24-bit Aircraft Addresses. When the UUT has the higher ICAO 24-bit Aircraft Address, it is the slave and should reverse its sense from a climb to a descend. When the UUT has the lower ICAO 24-bit Aircraft Address, it is the master and should not reverse its sense. This scenario tests that the UUT correctly interprets received capability information from a TCAS II intruder and uses the ICAO 24-bit Aircraft Address to determine master/slave (i.e. performs Active Coordination).

All communications must be present and in the proper sequence.

Input:

UUT:	Z = 12,200 ft ZDOT = 0 fpm	at T = 0 from T = 0 to 90
------	-------------------------------	------------------------------

Int1:	Equip = TCAS II R = 6.2 NM RDOT = -360 kt Z = 12,000 ft ZDOT = 0 fpm	at T = 0 from T = 0 to 90 from T = 0 to 90 from T = 0 to 90
-------	--	--

Msgs:

A DF=0 surveillance reply is transmitted in response to each UUT UF=0 surveillance interrogation, with
RI = 3

An ADS-B Aircraft Operational Status Message (TYPE=31, Subtype=0) is broadcast every 2.4-2.6 seconds throughout the encounter with the following data:

ADS-B Aircraft Operational Status Message TYPE=31, Subtype=0			
ME Bit 11 (CA Operational)	ME Bits 21-22 (DAA)	ME Bits 33-39 (CCCB)	Timing
1	00	0000000	Every 2.4-2.6 seconds throughout the encounter

Int1 transmits UF=16 TCAS Resolution Messages with the following data:

UF=16 MU Field Contents TCAS Resolution Message									
UDS	MTB	CVC	VRC	CHC	HRC	HSB	VSB	MID	Timing
48	0	0	2	0	0	0	7	ICAO aircraft address of Int1	Beginning five seconds after the UUT declares an RA

A DF=16 Coordination Reply (VDS=48*) is transmitted in response to each UUT UF=16 TCAS Resolution Message
*Remainder of MV field is 'don't care'

Int2: Msgs: UF=0, RL=1, BDS=51 (33_{16}) once per second from T=0 to 90

Expected Output:

UUT does not transmit any OCM

UUT transmits DF=16 replies with the following data:

DF=16 MV Field Contents	
Data Content	Timing
All Zeros	In response to UF=0 interrogations with BDS=51 (i.e., asking for read-out of transponder register 33_{16})

When Int1 has a lower ICAO 24-bit Aircraft Address than UUT:

Display: "Climb" advisory that reverses to a "Descend, Descend Now" advisory

UUT transmits UF=16 interrogations with the following data:

UF=16 MU Field Contents									
TCAS Resolution Message									
UDS	MTB	CVC	VRC	CHC	HRC	HSB	VSB	MID	Timing
48	0	0	2	0	0	0	7	ICAO aircraft address of UUT	Every second during the UUT Climb RA.
48	0	2	1	0	0	0	3	ICAO aircraft address of UUT	Every second during the UUT Descend RA.

UUT transmits DF=16 replies with the following data:

DF=16 MV Field Contents			
Coordination Reply Message			
VDS	ARA	RAC	Timing
48	1000100000	0000	In response to first UF=16 interrogation from Int1
48	0110101000	0100	In response to UF=16 interrogation from Int1 when UUT RA reverses to Descend advisory.
48	0110101000	0100	In response to subsequent UF=16 interrogations from Int1 while UUT Descend advisory is active

Subfields LDI, RAT, and MTE =0; RMF =01.

When Int1 has a higher ICAO 24-bit Aircraft Address than UUT:

Display: "Climb" advisory

UUT transmits UF=16 interrogations with the following data:

UF=16 MU Field Contents TCAS Resolution Message									
UDS	MTB	CVC	VRC	CHC	HRC	HSB	VSB	MID	Timing
48	0	0	2	0	0	0	7	ICAO aircraft address of UUT	Every second during the UUT Climb RA.

UUT transmits DF=16 replies with the following data:

DF=16 MV Field Contents Coordination Reply Message			
VDS	ARA	RAC	Timing
48	1000100000	0000	In response to all UF=16 interrogation from Int1 during the UUT Climb RA.

Subfields LDI, RAT, and MTE =0; RMF =01.

Scenario B

Identical to Scenario A, except that the intruder is equipped with ACAS X and the intruder's ADS-B Aircraft Operational Status Message contains the following data:

ADS-B Aircraft Operational Status Message TYPE=31, Subtype=0			
ME Bit 11 (CA Operational)	ME Bits 21-22 (DAA)	ME Bits 33-39 (CCCB)	Timing
1	00	0000100	Every 2.4-2.6 seconds throughout the encounter

All other Inputs and Expected Outputs are identical to Scenario A.

Scenario C

Identical to Scenario A, except that the intruder is equipped with ACAS X and the intruder's ADS-B Aircraft Operational Status Message contains the following data:

ADS-B Aircraft Operational Status Message TYPE=31, Subtype=0			
ME Bit 11 (CA Operational)	ME Bits 21-22 (DAA)	ME Bits 33-39 (CCCB)	Timing
1	00	0001000	Every 2.4-2.6 seconds throughout the encounter

All other Inputs and Expected Outputs are identical to Scenario A.

Scenario D

Identical to Scenario A, except that the intruder is equipped with a DAA/TCAS II combination (Int1 Equip = TCAS II) and the intruder's ADS-B Aircraft Operational Status Message contains the following data:

ADS-B Aircraft Operational Status Message TYPE=31, Subtype=0			
ME Bit 11 (CA Operational)	ME Bits 21-22 (DAA)	ME Bits 33-39 (CCCB)	Timing
1	01	0000000	Every 2.4-2.6 seconds throughout the encounter

All other Inputs and Expected Outputs are identical to Scenario A.

Scenario E

The UUT and a Responsive 1030/1090 CAS-equipped threat are on a horizontal collision course. The UUT is the first to declare the conflict and selects a climb sense RA. Five seconds later, the threat also selects a climb sense RA.

Run this scenario twice, alternating higher and lower ICAO 24-bit Aircraft Addresses. Regardless of whether the UUT has a higher or lower ICAO 24-bit Aircraft Address than the threat, it is the master and should not reverse when it receives conflicting UF=16 TCAS Resolution Messages from the threat. This scenario tests that the UUT correctly interprets received capability information from a Responsive 1030/1090 CAS, does not use ICAO 24-bit Aircraft Address to determine master/slave, and transmits UF=16 TCAS Resolution Messages to the threat (i.e. performs Modified Active Coordination).

Input:

UUT:	Z = 12,200 ft	at T = 0
	ZDOT = 0 fpm	from T = 0 to 90

Int1:	Equip = TCAS II	
	R = 6.2 NM	at T = 0
	RDOT = -360 kt	from T = 0 to 90
	Z = 12,000 ft	from T = 0 to 90
	ZDOT = 0 fpm	from T = 0 to 90

Msgs:

A DF=0 surveillance reply is transmitted in response to each UUT UF=0 surveillance interrogation, with
RI = 2

An ADS-B Aircraft Operational Status Message (TYPE=31, Subtype=0) is broadcast every 2.4-2.6 seconds throughout the encounter with the following data:

ADS-B Aircraft Operational Status Message TYPE=31, Subtype=0			
ME Bit 11 (CA Operational)	ME Bits 21-22 (DAA)	ME Bits 33-39 (CCCB)	Timing
1	00	0001100	Every 2.4-2.6 seconds throughout the encounter

Int1 transmits UF=16 TCAS Resolution Messages with the following data:

UF=16 MU Field Contents TCAS Resolution Message									
UDS	MTB	CVC	VRC	CHC	HRC	HSB	VSB	MID	Timing
48	0	0	2	0	0	0	7	ICAO aircraft address of Int1	Beginning five seconds after the UUT declares an RA

A DF=16 Coordination Reply (VDS=48*) is transmitted in response to each UUT UF=16 TCAS Resolution Message

*Remainder of MV field is ‘don’t care’

Int2: Msgs: UF=0, RL=1, BDS=51 (33_{16}) once per second from T=0 to 90

Expected Output:

UUT does not transmit any OCM

UUT transmits DF=16 replies with the following data:

DF=16 MV Field Contents	
Data Content	Timing
All Zeros	In response to UF=0 interrogations with BDS=51 (i.e., asking for read-out of transponder register 33_{16})

For both runs, regardless of whether Int1 has a higher or lower ICAO aircraft address than UUT:

Display: “Climb” advisory

UUT transmits UF=16 interrogations with the following data:

UF=16 MU Field Contents TCAS Resolution Message									
UDS	MTB	CVC	VRC	CHC	HRC	HSB	VSB	MID	Timing
48	0	0	2	0	0	0	7	ICAO aircraft address of UUT	Every second during the UUT Climb RA.

UUT transmits DF=16 replies with the following data:

DF=16 MV Field Contents Coordination Reply Message			
VDS	ARA	RAC	Timing
48	1000100000	0000	In response to all UF=16 interrogation from Int1 during the UUT Climb RA.

Subfields LDI, RAT, and MTE =0; RMF =01.

Scenario F

The UUT and a Mode S aircraft carrying a Passive CAS are on a horizontal collision course. The UUT is the first to declare the conflict and selects a climb sense RA. Five seconds later, the threat also selects a climb sense RA.

Run this scenario twice, alternating higher and lower ICAO 24-bit Aircraft Addresses. Regardless of whether the UUT has a higher or lower ICAO 24-bit Aircraft Address than the threat, it is the master and should not reverse when it receives conflicting UF=16 TCAS Resolution Messages from the threat. All communications must be present and in the proper sequence.

Note: In the real world, Mode S aircraft carrying a Passive CAS would be able to receive UF=16 TCAS Resolution Messages, but not able to send them as they lack a 1030 MHz interrogator. Nonetheless, UF=16 TCAS Resolution Messages appearing to be from the threat are used to test that the UUT acts as the master regardless of ICAO 24-bit Aircraft Address.

Input:

UUT:	Z = 12,200 ft ZDOT = 0 fpm	at T = 0 from T = 0 to 90
Int1:	Equip = Mode S R = 6.2 NM RDOT = -360 kt Z = 12,000 ft ZDOT = 0 fpm	at T = 0 from T = 0 to 90 from T = 0 to 90 from T = 0 to 90

Msgs:

A DF=0 surveillance reply is transmitted in response to each UUT UF=0 surveillance interrogation, with
RI = 0

An ADS-B Aircraft Operational Status Message (TYPE=31, Subtype=0) is broadcast every 2.4-2.6 seconds throughout the encounter with the following data:

ADS-B Aircraft Operational Status Message TYPE=31, Subtype=0			
ME Bit 11 (CA Operational)	ME Bits 21-22 (DAA)	ME Bits 33-39 (CCCB)	Timing
1	00	0010000	Every 2.4-2.6 seconds throughout the encounter

Int1 transmits UF=16 TCAS Resolution Messages with the following data:

UF=16 MU Field Contents TCAS Resolution Message									
UDS	MTB	CVC	VRC	CHC	HRC	HSB	VSB	MID	Timing
48	0	0	2	0	0	0	7	ICAO aircraft address of Int1	Beginning five seconds after the UUT declares an RA

A DF=16 Coordination Reply (VDS=48*) is transmitted in response to each UUT UF=16 TCAS Resolution Message
 *Remainder of MV field is ‘don’t care’

Int2: Msgs: UF=0, RL=1, BDS=51 (33₁₆) once per second from T=0 to 90

Expected Output:

UUT does not transmit any OCM

UUT transmits DF=16 replies with the following data:

DF=16 MV Field Contents	
Data Content	Timing
All Zeros	In response to UF=0 interrogations with BDS=51 (i.e., asking for read-out of transponder register 33 ₁₆)

For both runs, regardless of whether Int1 has a higher or lower ICAO aircraft address than UUT:

Display: “Climb” advisory

UUT transmits UF=16 interrogations with the following data:

UF=16 MU Field Contents TCAS Resolution Message									
UDS	MTB	CVC	VRC	CHC	HRC	HSB	VSB	MID	Timing
48	0	0	2	0	0	0	7	ICAO aircraft address of UUT	Every second during the UUT Climb RA.

UUT transmits DF=16 replies with the following data:

DF=16 MV Field Contents Coordination Reply Message			
VDS	ARA	RAC	Timing
48	1000100000	0000	In response to all UF=16 interrogation from Int1 during the UUT Climb RA.

Subfields LDI, RAT, and MTE =0; RMF =01.

Scenario G

Identical to Scenario F, except that the intruder is a Mode S aircraft carrying a DAA-only system that is requesting UF=16 TCAS Resolution Messages. The intruder's ADS-B Aircraft Operational Status Message contains the following data:

ADS-B Aircraft Operational Status Message TYPE=31, Subtype=0			
ME Bit 11 (CA Operational)	ME Bits 21-22 (DAA)	ME Bits 33-39 (CCCB)	Timing
0	01	0000000	Every 2.4-2.6 seconds throughout the encounter

All other Inputs and Expected Outputs are identical to Scenario F.

Scenario H

Identical to Scenario F, except that the intruder is a Mode S aircraft carrying neither CA nor DAA. The UUT recognizes it as a normal Mode S intruder and does not transmit any UF=16 TCAS Resolution Messages. The intruder's ADS-B Aircraft Operational Status Message contains the following data:

ADS-B Aircraft Operational Status Message TYPE=31, Subtype=0			
ME Bit 11 (CA Operational)	ME Bits 21-22 (DAA)	ME Bits 33-39 (CCCB)	Timing
0	00	0000000	Every 2.4-2.6 seconds throughout the encounter

The UUT does not transmit any UF=16 TCAS Resolution Messages during the encounter.

All other Inputs and Expected Outputs are identical to Scenario F.

Transponder and ACAS X considerations for scenarios I, J, K, and L

The final four scenarios test that the UUT correctly determines its ACAS X/transponder system capability based on communication with the on-board Mode S transponder and then, if capable, stores OCM data in the correct transponder register and transmits OCMs to a Mode C intruder.

Manufacturers need only test against the Expected Outputs corresponding to the capability of the ACAS X and associated Mode S transponder under test. Per §2.2.3.12.2.1.3, ACAS X systems built to these MOPS do not have the capability to transmit OCMs. The capability to transmit OCMs may be incorporated into a future version of these MOPS, and in that event, the following scenarios would expand to include a test of correct OCM transmission.

Scenario I

The UUT and a Mode C aircraft carrying a Passive CAS are on a horizontal collision course. The Mode C aircraft is requesting OCMs from the UUT because it is not able to

receive UF=16 TCAS Resolution Messages, nor does it have a 1030 MHz transmitter to interrogate the UUT.

Input:

UUT: Z = 12,200 ft at T = 0
 ZDOT = 0 fpm from T = 0 to 90

Int1: Equip = Mode C at T = 0
 R = 6.2 NM from T = 0 to 90
 RDOT = -360 kt from T = 0 to 90
 Z = 12,000 ft from T = 0 to 90
 ZDOT = 0 fpm from T = 0 to 90

Msgs:

An ADS-B Aircraft Operational Status Message (TYPE=31, Subtype=0) is broadcast every 2.4-2.6 seconds throughout the encounter with the following data:

ADS-B Aircraft Operational Status Message TYPE=31, Subtype=0			
ME Bit 11 (CA Operational)	ME Bits 21-22 (DAA)	ME Bits 33-39 (CCCB)	Timing
1	00	0010100	Every 2.4-2.6 seconds throughout the encounter

Int2: Msgs: UF=0, RL=1, BDS=51 (33₁₆) once per second from T=0 to 90
 Msgs: UF=0, RL=1, BDS=52 (3416) once per second from T=0 to 90
 Msgs: UF=0, RL=1, BDS=53 (3516) once per second from T=0 to 90
 Msgs: UF=0, RL=1, BDS=54 (3616) once per second from T=0 to 90
 Msgs: UF=0, RL=1, BDS=55 (3716) once per second from T=0 to 90

Expected Output:

Display: “Climb” advisory

UUT transmits DF=16 replies with the following data:

DF=16 MV Field Contents	
Data Content	Timing
All Zeros	In response to all UF=0 interrogations with BDS=51, 52, 53, 54, 55 (i.e., asking for read-out of the transponder register)

UUT does not transmit any OCMs or TCAS Resolution Messages while UUT has declared an RA.

Scenario J

Identical to Scenario I, except that the intruder is a Mode C aircraft carrying a DAA-only system that is requesting OCMs. The intruder's ADS-B Aircraft Operational Status Message contains the following data:

ADS-B Aircraft Operational Status Message TYPE=31, Subtype=0			
ME Bit 11 (CA Operational)	ME Bits 21-22 (DAA)	ME Bits 33-39 (CCCB)	Timing
0	10	0000000	Every 2.4-2.6 seconds throughout the encounter

All other Inputs and Expected Outputs are identical to Scenario I.

Scenario K

Identical to Scenario I, except that the intruder is a Mode C aircraft carrying neither CA nor DAA. The UUT recognizes it as a normal Mode C intruder and does not transmit any OCMs. The intruder's ADS-B Aircraft Operational Status Message contains the following data:

ADS-B Aircraft Operational Status Message TYPE=31, Subtype=0			
ME Bit 11 (CA Operational)	ME Bits 21-22 (DAA)	ME Bits 33-39 (CCCB)	Timing
0	00	0000000	Every 2.4-2.6 seconds throughout the encounter

All other Inputs and Expected Outputs are identical to Scenario I.

Scenario L

This test ensures the OCM Message Prioritization (§2.2.3.12.2.2.5.1) is accomplished correctly. Since ACAS X systems built to these MOPS do not have the capability to transmit OCMs, manufacturers do not need to run this scenario until building to the version of MOPS that includes OCM transmission.

Input:

Simulate ten intruders that require passive coordination and for which simultaneous RAs are active. The advisories must include both a reversal and a crossing advisory.

Expected Output:

Verify that five OCMs are transmitted every update interval during which five or more RAs are active.

Verify the five OCMs are prioritized according to the following advisory criteria:

- 1) RA Reversal for which an OCM has not been broadcast
- 2) Initial crossing RA for which an OCM has not been broadcast;

- 3) Initial non-crossing RA for which an OCM has not been broadcast;
- 4) RA reversal for which an OCM has not been broadcast twice;
- 5) Initial crossing RA for which an OCM that has not been broadcast twice;
- 6) Initial non-crossing RA for which an OCM has not been broadcast twice;
- 7) RA for which an OCM has not been broadcast for the longest period of time.
- 8) When the number of OCMs with equal priority exceeds the number with that priority that can be selected, the OCMs corresponding to the intruders with the highest Track Display Score are transmitted.

Note: This test is not required to be performed at a system (black box) level, rather it may be performed as a software/module test or by other means as recommended by the equipment manufacturer.

2.4.2.4.3.8 Transmission of Own Collision Avoidance Coordination Capability Bits (CCCB) (§2.2.3.12.2.1.3)

This test requires *either* Scenario A *or* Scenario B to be run, depending upon the capability of the Mode S transponder with which ACAS X is paired.

Scenario A is required when the paired Mode S transponder is capable of transmitting ACAS X CCCB values in outgoing ADS-B Operational Status Messages, i.e., the Mode S transponder conforms to RTCA/DO-181F or later and RTCA/DO-260C or later. Otherwise, Scenario B is required.

Scenario A

The UUT and a Mode S-equipped intruder are on a horizontal collision course. The UUT declares the threat and selects a climb sense RA. All communications must be present and in the proper sequence.

Input:

UUT:	Z = 12,200 ft ZDOT = 0 fpm	at T = 0 from T = 0 to 90
Int1:	R = 6.2 NM RDOT = -360 kt Z = 12,000 ft ZDOT = 0 fpm	at T = 0 from T = 0 to 90 from T = 0 to 90 from T = 0 to 90

Expected Output:

Display: "Climb" RA

DF=17 ME Field Contents
ADS-B Airborne Aircraft Operational Status Message
transmitted by UUT every 2.4-2.6 seconds* from T=0 to 90

ME bit 11 (CA Operational) =1
ME bits 21-22 (DAA) =00
ME bit 27 (RA Active) =1 while UUT RA is active, =0 otherwise
ME bits 33-39 (CCCB) =0000100

Notes:

1. **If Target State and Status Messages are NOT being transmitted, then Airborne Aircraft Operational Status Messages are transmitted by the UUT every 0.7-0.9 seconds starting when the RA is first active and for 24 ±1 seconds after the RA has terminated.*
 2. *CCCB=0000100 has the meaning “Active CAS (not TCAS II)” and is silent on the ability of ACAS X to transmit OCMs. The initial release of DO-181F and DO-260C are not expected to contain provisions for transmitting OCMs. Safety studies and spectrum studies, which will determine the necessary OCM transmission rate, are required before OCM transmit capability will be included in these MOPS.*

Scenario B

The UUT and a Mode S-equipped intruder are on a horizontal collision course. The UUT declares the threat and selects a climb sense RA. All communications must be present and in the proper sequence.

Input:

UUT: Z = 12,200 ft at T = 0
 ZDOT = 0 fpm from T = 0 to 90

Int1: R = 6.2 NM at T = 0
 RDOT = -360 kt from T = 0 to 90
 Z = 12,000 ft from T = 0 to 90
 ZDOT = 0 fpm from T = 0 to 90

Expected Output:

Display: “Climb” RA

DF=17 ME Field Contents
ADS-B Airborne Aircraft Operational Status Message
transmitted by UUT every 2.4-2.6 seconds* from T=0 to 90

ME bit 11 (CA Operational) =1
ME bits 21-22 (DAA) =00
ME bit 27 (RA Active) =1 while UUT RA is active, =0 otherwise
ME bits 33-39 (CCCB) =0000000

Note: *If Target State and Status Messages are NOT being transmitted, then Airborne Aircraft Operational Status Messages are transmitted by the UUT every 0.7-0.9 seconds starting when the RA is first active and for 24 ±1 seconds after the RA has terminated.

2.4.2.5 ACAS X Capability and Part Number Reporting (§2.2.3.9.5.3, §2.2.3.12.2.2.4, §2.2.3.12.2.2.1, §2.2.5.5.5)

Note: See note on test operation in §2.4.2.3.

This test verifies that the ACAS X/transponder system properly communicates its ACAS X capability to a ground Mode S sensor and to other Active CAS aircraft. This includes passing ACAS X capability information from ACAS X to the transponder and transmitting this information in DF=0, DF=16, DF=20, and DF=21 replies.

This test also verifies that the ACAS X/transponder system properly communicates its part numbers to a ground Mode S sensor. This includes passing ACAS X part number information from ACAS X to the transponder and transmitting this information in DF=16 replies.

Scenario A

Use manual selection on the control panel to select among the modes “standby,” “TA-only”, and “TA/RA.” Allow each mode to persist for at least 20 seconds.

Input:

UUT: Z = 12,000 ft
ZDOT = 0 fpm

Int1: Equip = TCAS II
R = 4 NM
RDOT = 0 kt
Z = 10,500 ft
ZDOT = 0 fpm
Msgs = UF=0 with AQ =0 transmitted every second

MSS: Msgs = UF=4 with RR=0 transmitted every second
UF=5 with RR=0 transmitted every second
UF=4 with RR=17 transmitted every second
UF=5 with RR=17 transmitted every second
UF=20 with RR=17 transmitted every second
UF=21 with RR=17 transmitted every second

Expected Output:

Msgs:

Unless otherwise noted verify transition of data occurs within 1 second of ACAS X mode change.

DF=0 Reply every second with RI=0 when ACAS X is in ‘Standby’ mode
 DF=0 Reply every second with RI=2 when ACAS X is in ‘TA-Only’ mode
 DF=0 Reply every second with RI=3 when ACAS X is in ‘TA/RA’ mode

DF=4 replies to UF=4(RR=0) interrogations with DR=0 until each ACAS X capability change and then DR=4 or 5 for 18 +/- 1 seconds after the change (transitions within 4 seconds of the change and alternates between 4 and 5 for each change).

DF=5 replies to UF=5(RR=0) interrogations with DR=0 until the ACAS X capability change and then DR=4 or 5 for 18 +/- 1 seconds after the change (transitions within 4 seconds of the change and alternates between 4 and 5 for each change).

DF=20 replies are sent with the following data:

DF=20 Message Contents Data Link Capability Report						Timing	
MB Field Bits							
33- 40	43- 46	48	69	70	71,72		
16	0001	0	0	Don't Care	1,1	Reply to UF=4(RR=17), UF=20(RR=17) when ACAS X is in ‘Standby’ Mode	
16	0001	1	0	0	1,1	Reply to UF=4(RR=17), UF=20(RR=17) when ACAS X is in ‘TA-Only’ Mode	
16	0001	1	0	1	1,1	Reply to UF=4(RR=17), UF=20(RR=17) when ACAS X is in ‘TA/RA’ Mode	

DF=21 replies are sent with the following data:

DF=21 Message Contents Data Link Capability Report						Timing	
MB Field Bits							
33- 40	43- 46	48	69	70	71,72		
16	0001	0	0	Don't Care	1,1	Reply to UF=5(RR=17), UF=21(RR=17) when ACAS X is in ‘Standby’ Mode	
16	0001	1	0	0	1,1	Reply to UF=5(RR=17), UF=21(RR=17) when ACAS X is in ‘TA-Only’ Mode	
16	0001	1	0	1	1,1	Reply to UF=5(RR=17), UF=21(RR=17) when ACAS X is in ‘TA/RA’ Mode	

DF=17 Aircraft Operational Status Messages are sent with the following data:

DF=17 Aircraft Operational Status Message		
Bit 11 TCAS Operational	Bit 27 TCAS RA Active	Timing
0	0	While ACAS X is in ‘Standby’ Mode
0	0	While ACAS X is in ‘TA-Only’ Mode
1	0	While ACAS X is in ‘TA/RA’ Mode

Scenario B

Radio Altitude will be varied to select among the modes “standby,” “TA- Only”, and “TA/RA.” Allow each mode to persist for at least 20 seconds.

Input:

UUT:	Z	= 12,000 ft
	ZDOT	= 0 fpm
	Radio Altitude	= 2,000 ft for TCAS in ‘TA/RA’ Mode = 300 ft for TCAS in ‘TA-Only’ Mode = 0 ft for TCAS in ‘Standby’ Mode (on ground display mode must be configured to ‘off’)
Int1:	Equip	= TCAS II
	R	= 4 NM
	RDOT	= 0 kt
	Z	= 10,500 ft
	ZDOT	= 0 fpm
	Msgs	= UF=0 with AQ =0 transmitted every second
MSS:	Msgs	= UF=4 with RR=0 UF=5 with RR=0 UF=4 with RR=17 UF=5 with RR=17 UF=20 with RR=17 UF=21 with RR=17

Expected Output:

Msgs:

Unless otherwise noted verify transition of data occurs within 1 second of ACAS X mode change

DF=0 Reply every second with RI=0 when ACAS X is in ‘Standby’ mode
DF=0 Reply every second with RI=2 when ACAS X is in ‘TA-Only’ mode
DF=0 Reply every second with RI=3 when ACAS X is in ‘TA/RA’ mode

DF=4 replies to UF=4(RR=0) interrogations with DR=0 until each ACAS X capability change and then DR=4 or 5 for 18 +/- 1 seconds after the change (transitions within 4 seconds of the change and alternates between 4 and 5 for each change).

DF=5 replies to UF=5(RR=0) interrogations with DR=0 until the ACAS X capability change and then DR=4 or 5 for 18 +/- 1 seconds after the change (transitions within 4 seconds of the change and alternates between 4 and 5 for each change).

DF=20 replies are sent with the following data:

DF=20 Message Contents Data Link Capability Report						
MB Field Bits						Timing
33- 40	43- 46	48	69	70	71,72	
16	0001	0	0	Don't Care	1,1	Reply to UF=4(RR=17), UF=20(RR=17) when ACAS X is in 'Standby' Mode
16	0001	1	0	0	1,1	Reply to UF=4(RR=17), UF=20(RR=17) when ACAS X is in 'TA-Only' Mode
16	0001	1	0	1	1,1	Reply to UF=4(RR=17), UF=20(RR=17) when ACAS X is in 'TA/RA' Mode

DF=21 replies are sent with the following data:

	DF=21 Message Contents Data Link Capability Report						
	MB Field Bits						Timing
33- 40	43- 46	48	69	70	71,72		
16	0001	0	0	Don't Care	1,1	Reply to UF=5(RR=17), UF=21(RR=17) when ACAS X is in 'Standby' Mode	
16	0001	1	0	0	1,1	Reply to UF=5(RR=17), UF=21(RR=17) when ACAS X is in 'TA-Only' Mode	
16	0001	1	0	1	1,1	Reply to UF=5(RR=17), UF=21(RR=17) when ACAS X is in 'TA/RA' Mode	

DF=17 Aircraft Operational Status Messages are sent with the following data:

DF=17 Aircraft Operational Status Message			
Bit 11 TCAS Operational		Bit 27 TCAS RA Active	Timing
0		0	While ACAS X is in 'Standby' Mode
0		0	While ACAS X is in 'TA-Only' Mode
1		0	While ACAS X is in 'TA/RA' Mode

Scenario C

Power on the ACAS X system with the "Software Part Number Enable" discrete disabled.

Input:

UUT: Z = 12,000 ft
ZDOT = 0 fpm

INT1: Msgs = UF=0, RL=1, DS=229 (E5₁₆) once per second
UF=0, RL=1, DS=230 (E6₁₆) once per second

Expected Output:

Msgs:

DF=16 replies are sent with the following data:

DF=16 MV Field Contents	
Data Content	Timing
All Zeros	In response to UF=0 interrogations with DS=229
All Zeros	In response to UF=0 interrogations with DS=230

Scenario D

Power on the ACAS X system with the “Software Part Number Enable” discrete enabled.

Input:UUT: Z = 12,000 ft
ZDOT = 0 fpmINT1: Msgs = UF=0, RL=1, DS=229 (E5₁₆) once per second
UF=0, RL=1, DS=230 (E6₁₆) once per secondExpected Output:

Msgs:

DF=16 replies are sent with the following data:

DF=16 MV Field Contents	
Data Content	Timing
ACAS X Unit Part Number	In response to UF=0 interrogations with DS=229
ACAS X Software Part Number	In response to UF=0 interrogations with DS=230

ACAS X to XPDR Communication:

Unit Part Number (GICB=E5) is transmitted to the Mode S transponder once every 10 seconds.

Software Part Number (GICB=E6) is transmitted to the Mode S transponder once every 10 seconds.

2.4.2.6 Transmission of Low-level Descend Inhibit (LDI) Information in RF Messages (§2.2.3.8.3.2.3.1)

This test verifies that the ACAS X/transponder system correctly:

- a. Reads in altitude information from the onboard radar altimeter.
- b. Passes this information to the TRM, where hysteresis is applied according to whether ownship is climbing or descending, and LDI is computed and output for insertion into RF messages.
- c. Passes LDI from ACAS X to the transponder as a part of the RA Report.
- d. Correctly includes LDI in four RF messages: RA Report, ADS-B TCAS RA Broadcast, Coordination Reply Message, and RA Broadcast Interrogation Message.
- e. Transitions correctly through the four LDI values as ownship passes through the altitude thresholds.

This test consists of two scenarios, one in which ownship is climbing from the ground to 2000 feet AGL, and one in which ownship is descending from 2000 feet AGL to the ground. There is a nearby intruder continuously present throughout both scenarios in order to ensure that RAs are generated and thus all four types of RF messages are transmitted.

A Mode S ground sensor interrogates the UUT each second for a read-out of the RA Report. An Active CAS-equipped intruder sends coordination interrogations to the UUT each second in order to generate Coordination Reply Messages. The ADS-B TCAS RA Broadcast and the RA Broadcast Interrogation Message are automatically transmitted for the period that the RA is active and for 24 ± 1 seconds and 1 second, respectively, after the RA has terminated.

Notes:

1. *In the scenarios, both barometric altitude and radio altitude are specified for the UUT. The barometric altitude is constant throughout the scenarios for both the UUT and the intruder, while the UUT radio altitude increases (Scenario A) or decreases (Scenario B). Inclusion of both altitudes is intended to verify that the TRM correctly uses radio altitude to determine the LDI value.*
2. *Test Suite scenarios from test group 53 perform detailed LDI to check that the TRM properly inhibits increase rate descend RAs, positive descend RAs, and all RAs at the proper altitude thresholds and that the UUT RI values change appropriately.*

Scenario A

The scenario runs from T=0 to T=80 seconds

Input

UUT: Z	= 0 ft from T=0 to T=80
Radio alt	= 0 ft at T=0
Radio alt rate	= 1500 fpm

Int1: Equip	= TCAS II
R	= 0 NM (directly above)
RDOT	= 0 kt
Z	= 200 ft at T=0

ZDOT = 0 fpm
Msgs = UF16 with UDS=48, other subfields = 0, transmitted every second

MSS: Msgs = UF=4 with RR=19, transmitted every second

Expected Output:

Msgs: DF=16 every second, starting with LDI=3,
transitioning to LDI=2 as UUT passes through 1100 ft AGL,
transitioning to LDI=1 as UUT passes through 1200 ft AGL, and
transitioning to LDI=0 as UUT passes through 1650 ft AGL.

DF=20 every second, starting with LDI=3,
transitioning to LDI=2 as UUT passes through 1100 ft AGL,
transitioning to LDI=1 as UUT passes through 1200 ft AGL, and
transitioning to LDI=0 as UUT passes through 1650 ft AGL.

ADS-B TCAS RA Broadcast every 0.7-0.9 seconds
for the period that the RA is active,
starting with LDI=2 as UUT passes through 1100 ft AGL, and
transitioning to LDI=1 as UUT passes through 1200 ft AGL, and
transitioning to LDI=0 as UUT passes through 1650 ft AGL.

RA Broadcast Interrogation Message every 1 second
for the period that the RA is active,
starting with LDI=2 as UUT passes through 1100 ft AGL, and
transitioning to LDI=1 as UUT passes through 1200 ft AGL, and
transitioning to LDI=0 as UUT passes through 1650 ft AGL.

Scenario B

The scenario runs from T=0 to T=80 seconds

Input

UUT: Z = 2000 ft from T=0 to T=80
Radio alt = 2000 ft AGL at T=0
Radio alt rate = -1500 fpm

Int1: Equip = TCAS II
R = 0 NM (directly above)
RDOT = 0 kt
Z = 2200 ft at T=0
ZDOT = 0 fpm
Msgs = UF16 with UDS=48, other subfields = 0,
transmitted every second

MSS: Msgs = UF=4 with RR=19, transmitted every second

Expected Output:

Msgs: DF=16 every second, starting with LDI=0,

transitioning to LDI=1 as UUT passes through 1450 ft AGL,
 transitioning to LDI=2 as UUT passes through 1000 ft AGL, and
 transitioning to LDI=3 as UUT passes through 900 ft AGL.

DF=20 every second, starting with LDI=0,
 transitioning to LDI=1 as UUT passes through 1450 ft AGL,
 transitioning to LDI=2 as UUT passes through 1000 ft AGL, and
 transitioning to LDI=3 as UUT passes through 900 ft AGL.

ADS-B TCAS RA Broadcast every 0.7-0.9 seconds for the period that the RA is active and for 24 ± 1 seconds after the end of the RA
 starting with LDI=0,
 transitioning to LDI=1 as UUT passes through 1450 ft AGL, and
 transitioning to LDI=2 as UUT passes through 1000 ft AGL, and
 transitioning to LDI=3 as UUT passes through 900 ft AGL.

RA Broadcast Interrogation Message every second for the period that the RA is active and for 1 sec after the end of the RA
 starting with LDI=0,
 transitioning to LDI=1 as UUT passes through 1450 ft AGL, and
 transitioning to LDI=2 as UUT passes through 1000 ft AGL, and
 transitioning to LDI=3 as UUT passes through 900 ft AGL.

2.4.2.7 System Integration Tests

In addition to the Stand Alone Display and Control tests defined in §2.4.2.8, a set of system level integration tests will be required for those manufacturers that produce ACAS X systems. The definition and details of these tests are left to the discretion of the manufacturer and the certifying agency. While the development of the detailed test procedures is the responsibility of the manufacturer, there are certain guidelines that must be followed for the development of these procedures.

The system integration tests are not required to be completed on every combination of ACAS X processor unit, traffic display, RA display, and control panel that is produced by the manufacturer. Instead, the system integration tests are required to be completed for each type of interface between system components that is supported by a manufacturer. For example, if a manufacturer provides systems which use the ARINC 429 architecture, the MIL-STD-1553 architecture, and a company proprietary architecture, the manufacturer would be required to complete three sets of system integration tests.

The system integration tests are bench tests that are conducted under ambient conditions.

These tests require that the following system elements be interconnected:

- a. ACAS X processor unit which includes all software that will be present in the equipment's flight configuration, i.e., threat resolution, surveillance, and monitor.
- b. ACAS X traffic display
- c. ACAS X RA display
- d. ACAS X/Mode S control panel
- e. Speaker and headset for aural annunciations
- f. Mode S transponder

The tests also require that the required inputs from the ownship (data and discretes) and intruder aircraft be available to the ACAS X processor unit.

The completion of these integrated tests will demonstrate the following functions of the ACAS X displays and controls:

- a. All aural annunciations shown in §2.2.6.3.1.3 are annunciated through both the speaker and the headset. The aural annunciations for an RA reversal and an increase rate RA are spoken with a sense of urgency.
- b. The information shown on the RA display when an RA is issued is consistent with the aural annunciations issued by the ACAS X processor unit. This requires the issuance of all RAs shown in Table 2-49 (§2.2.6.4.1).
- c. ACAS X automatically switches to ‘TA-Only’ mode or ‘Standby’ as appropriate when an advisory inhibit condition is activated (i.e. when a higher priority warning such as GPWS or wind shear is active.) Verify when in the advisory inhibit ‘TA-Only’ mode a TA is issued properly without an aural annunciation. Verify that the ACAS X system remains in “TA-Only” mode for 10 seconds after the inhibit is cleared and during that 10 second period a TA is issued properly with the “Traffic, Traffic” aural annunciation.
- d. All ACAS X aural annunciations are inhibited when the ownship is below the aural inhibit altitude determined by the TRM.
- e. The ACAS X ‘Descend’ and ‘All’ Advisory Inhibit conditions are correctly applied based on the radio altitude interface. Verify that Aural annunciations are generated whenever an RA is initially issued and when the strength of an RA is either strengthened or weakened.
 1. “Increase Descend” advisories are inhibited below $1,550' \pm 100'$ hysteresis
 2. “Descend” advisories are inhibited below $1,100' \pm 100'$ hysteresis
 3. All RAs are inhibited (TA-Only Mode) below $1,000' \pm 100'$ hysteresis.
 4. All “Traffic” aurals are inhibited below $500' +/- 100'$ hysteresis.
- f. An aural annunciation is interrupted when the TRM logic determines a higher priority aural annunciation should be announced.
- g. Aural annunciations are generated whenever an RA is initially issued and when the strength of an RA is either strengthened or weakened.
- h. The aural annunciation “Clear of Conflict” is provided when an RA is removed from the display. A separate test is required to demonstrate that this aural is not issued when surveillance drops a track while an RA is displayed or when the ownship passes below the RA inhibit altitude.
- i. The aural annunciation “Traffic-Traffic” is provided when a TA is initially issued. A demonstration is also required to show that this aural is not issued when the RA reverts back to a TA or when a TA is issued and an RA against another intruder is active.
- j. The ACAS X processor unit operates in Standby, TA-Only, and TA/RA modes when these modes are selected on the ACAS X control panel for both states of the Ground Display Mode input.
- k. ACAS X automatically fails whenever altitude reporting is disabled or switches to Standby when the transponder-only mode of operation is selected on the control panel. ACAS X indicates (via outputs and aural annunciation), as required by §2.2.7.2.7, when own transponder mode is standby or failed when ownship is above 2500 ft.

- i. All levels of intruders (Proximate, TA, and RA) are displayed at the proper range, bearing, and relative altitude and with the proper threat level as determined by the TRM. If ACAS X is implemented with the capability to display Other Traffic, then “all levels of intruders” **shall** (2450) include Other Traffic for this test. If ACAS X is implemented with AOTO capability, then “all levels of intruders” should include AOTO for this test. Verify proper display of ADS-B Only/ADS-R Only intruders per §2.2.3.12.9 and §2.2.6.1.2.1.9.
- m. The proper mode annunciations are shown on the traffic and RA displays when ACAS X is operating in the TA-Only and the TA/RA mode. Proper annunciations are displayed for all required failure conditions of the ACAS X processor unit, traffic display, and RA display.
- n. The proper display and annunciation of multi-aircraft encounters.
- o. The RA display calls for maintaining a vertical speed of no more than 4400 fpm when a maintain rate RA is issued while the ownship is climbing or descending at a rate greater than 5,000 fpm.
- p. ACAS X correctly reports the hybrid capability of the system in bit 69 of the Data Link Capability report in compliance with the requirements of §2.2.3.8.3.2.3.2.1.
- q. Each surveillance update interval, surveillance provides the STM with the available ownship data in chronological order and intruder data (per intruder) in chronological order per the requirements of §2.2.5.5.
- r. ACAS X correctly determines and correctly provides the intruder data for each intruder type (Mode S, Mode C, ADS-B, ADS-R) and surveillance region (normal, reduced, hybrid). Verifying that the correct protection is provided and the correct data sources are sent to the display as enumerated in §2.2.5.1.2 is one method of performing this verification.
- s. The time of applicability error requirements (from time of receipt by the equipment) and update rate requirements of §2.2.5.5 are satisfied. It is acceptable to record data or perform a thorough analysis to demonstrate verification of these requirements.
- t. The requirements associated with precision and use of untracked data of §2.2.5.5 are satisfied. It is acceptable to record data or perform a thorough analysis to demonstrate verification of these requirements.
- u. The requirements associated with marking invalid data (including the use of *Nan*) of §2.2.5.5 are satisfied. It is acceptable to record data or perform a thorough analysis to demonstrate verification of these requirements.
- v. ACAS X announces an ACAS failure if the paired Mode S transponder indicated that it is TSO-C119A compatible, as opposed to RTCA/DO-185A,B compatible.

2.4.2.8

Display and Controls Stand Alone Tests

A series of stand alone tests are defined to verify the proper implementation of the display and control requirements. These tests are intended to be performed on each type of display and control panel implementation provided by a manufacturer. These tests do not require the presence of an ACAS X processor unit; instead the manufacturer is responsible for defining a set of test drivers which is consistent with the type of interface (i.e., hardware and software protocols) designed into the item under test. Likewise, the manufacturer is responsible for defining the types of instrumentation to be used to collect any quantitative data required to demonstrate compliance with the requirements.

It is expected that the display drivers and data collection equipment used in these tests will be approved by the appropriate certification authority prior to the initiation of these tests. Separate drivers may be used for different display implementations and separate drivers may be used to complete the testing defined for controls.

This subparagraph contains tests for each requirement contained in §2.2.6. It is recognized that all requirements contained in §2.2.6 will not be applicable to each display or control panel implementation. The display or control manufacturer, in conjunction with the appropriate certification authority, has the responsibility for determining which tests must be performed for any display or control implementation. In implementations having different pilot selectable display modes, the tests defined in this subparagraph that are appropriate for each available mode **shall** (2451) be completed.

Note: In the stand-alone tests, the test descriptions and the success criteria include a time period for which a symbol or other information should be displayed and for the time between steps in the procedure. All these times were set to 30 seconds to ensure that sufficient time is available to complete the required observations and to provide the test conductor sufficient time to complete any required configuration changes. If the tasks specified in the test procedures can be performed in less than 30 seconds without affecting the collection of data specified for the test, the 30 seconds time periods may be reduced.

2.4.2.8.1 Traffic Displays

The following tests verify the required functions of a traffic display are properly included in a manufacturer's implementation.

2.4.2.8.1.1 General Characteristics

2.4.2.8.1.1.1 Ownship Symbol (§2.2.6.1.2.1, §2.2.6.1.2.6.2, §2.2.6.1.2.7.1.2, §2.2.6.1.2.7.2.6.2, §2.2.6.1.2.7.2.7.3, §2.2.6.1.2.7.2.8.4, §2.2.6.1.2.7.2.9.3, §2.2.6.1.2.7.3.2, §2.2.6.1.2.7.4.2.2)

This test verifies that the traffic display information includes a reference symbol to represent the location of the ownship.

Conditions

This test requires that the traffic display be powered up and configured such that ACAS X information can be displayed. If the display provides multiple, pilot-selectable modes, this test **shall** (2452) be completed for each available display mode. During the test, at least one intruder classified as Proximate Traffic or Other Traffic **shall** (2453) be displayed.

Success

- a. The traffic display **shall** (2454) show a symbol representing the ownship in either cyan or white.
- b. The color of the ownship symbol **shall** (2455) be different than the color of the displayed Proximate or Other Traffic.
- c. If the selected display mode displays only ACAS X traffic information, the ownship symbol is centered horizontally on the display and approximately 1/3 of the height from the bottom of the display.
- d. Shared Weather Radar Displays

1. In the WX/ACAS X mode, the ownship symbol is centered horizontally on the display and approximately 1/3 of the height from the bottom of the display.
 2. In the WX-and-Traffic mode, a unique ownship symbol is not displayed and the ownship reference of the weather radar remains displayed.
 3. In the Traffic-Only mode, the ownship symbol is centered horizontally on the display and approximately 1/3 of the height from the bottom of the display.
 - e. If traffic is displayed in the HSI Mode on an ND or EHSI, a unique ownship symbol is not displayed and the HSI's ownship symbol remains displayed at the center of the display.
 - f. If traffic is displayed in the EXPANDED mode on an ND or EHSI, a unique ownship symbol is not displayed and the EXPANDED mode's aircraft reference symbol remains displayed.
 - g. If the traffic display is the MAP mode of an ND or EHSI, a unique ownship symbol is not displayed and the MAP's aircraft reference symbol remains displayed.
 - h. If traffic is displayed when a North-up mode is selected, the front of the ownship symbol is oriented in the direction of the aircraft's heading.
- Note: In accomplishing this test, multiple heading inputs should be used to ensure the ownship symbol and the displayed intruder are properly oriented.*
- i. When the traffic display is shown on a shared EICAS/SYSTEMS display, the ownship symbol is centered horizontally on the display and approximately 1/3 of the height from the bottom of the display.
 - j. When the traffic display is shown on a TA/RA/VSI, the ownship symbol is centered horizontally on the display and approximately 1/3 of the height from the bottom of the display.

2.4.2.8.1.1.2 Range Rings (§2.2.6.1.2.1.2, §2.2.6.1.2.2.2, §2.2.6.1.2.3.3, §2.2.6.1.2.7.2.7.2, §2.2.6.1.2.7.2.8.2, §2.2.6.1.2.7.2.9.2)

This test verifies that the traffic display information includes the required range rings or markings to aid the pilot in visually acquiring displayed traffic.

Conditions

This test requires that the traffic display be powered up and configured such that ACAS X information can be displayed. If the display provides multiple, pilot-selectable modes, this test **shall** (2456) be completed for each available display mode.

Success

- a. At least one range ring is shown on the display and has the same color as the ownship symbol.
- b. If multiple range rings are displayed, the inner most range ring is not a solid circle and consists of discrete markings at the 12 clock positions. In the EXPANDED mode of an ND and in the WX/ACAS X mode of a shared Weather Radar display, the inner range ring consists of markings only in the forward quadrants.
- c. The markings used for the range rings are of a size and shape that do not add unnecessary clutter to the display.
- d. On a fixed range display, a range ring is provided at two miles.

- e. On a variable range display, a range ring is provided at either two or three miles when a display range of 12 NM or less is selected. If the selected range is greater than 12 NM, at least one range ring is shown.

Note: All available display ranges should be selected during this test.

- f. In the EXPANDED, North-up, or MAP modes of an ND, a two or three mile range ring is shown unless the display already provides range reference marks.

2.4.2.8.1.1.3 Threat Symbology (§2.2.6.1.2.1.3, §2.2.6.1.2.1.12)

This test verifies that the displayed traffic is shown with the proper symbols and in the proper color.

Conditions

This test requires that the traffic display be powered up and configured such that ACAS X information can be displayed. In addition, during this test the display must show an intruder classified as an RA, TA, Proximate Traffic, and Other Traffic. If the display provides multiple, pilot-selectable modes, this test **shall** (2457) be completed for each available display mode.

Success

- a. The intruder causing an RA is displayed as a red filled square. There is no RA maneuver information shown on the traffic display.
- b. The intruder causing a TA is displayed as an amber or yellow filled circle.
- c. The Proximate Traffic is displayed as a white or cyan filled diamond and the color is different than that of the display's ownship symbol.
- d. The Other Traffic is displayed in the same color as the Proximate Traffic, i.e., as a white or cyan diamond (outline only), and the color is different than that of the display's ownship symbol.

2.4.2.8.1.1.4 Off-Scale Symbology (§2.2.6.1.2.1.4, §2.2.6.1.2.1.11, §2.2.6.1.2.1.12)

This test verifies that the traffic display properly displays traffic that is beyond the selected or available range of the display.

Conditions

This test requires that the traffic display be powered up and configured such that ACAS X information can be displayed. In addition, the display must be receiving data for intruders classified as an RA, TA, Proximate Traffic, and Other Traffic. Each of these intruders **shall** (2458) have a range, relative to the ownship, that is greater than the available range of a fixed range display or greater than the selected range of a variable range display. The information sent to the display for each intruder **shall** (2459) indicate there is valid bearing and relative altitude data for the intruder. Each intruder **shall** (2460) have a vertical speed greater than 500 fpm. If the display provides multiple, pilot-selectable modes, this test **shall** (2461) be completed for each available display mode.

Success

- a. The intruder causing an RA is displayed with one half of a red filled square at the edge of the display area and at the proper bearing. The intruder's relative altitude and vertical trend arrow are displayed or masked as appropriate for the intruder's bearing. There is no RA maneuver information shown on the traffic display.

- b. The intruder causing a TA is displayed with one half of an amber or yellow filled circle at the edge of the display area and at the proper bearing. The intruder's relative altitude and vertical trend arrow are displayed or masked as appropriate for the intruder's bearing.
- c. Neither the Proximate or Other Traffic is displayed.
- d. If the traffic display is implemented on a shared display which does not support the use of the half symbols for displaying off-scale RAs and TAs, the message "TRAFFIC OFF-SCALE", "OFFSCALE", or "OFF-SCALE" is shown in red when an intruder becomes an RA and in either amber or yellow when an intruder becomes a TA.

Note: For these implementations, it is necessary to conduct separate tests for the off-scale TA and RA intruders or have a dynamic intruder that progresses from a classification of Other, to Proximate, to TA, and then to RA.

- e. The pilot selected or fixed display range is shown on the traffic display.

2.4.2.8.1.1.5 Altitude Data Tag (§2.2.6.1.2.1.5, §2.2.6.1.2.1.7)

This test verifies that the traffic display properly displays the altitude of displayed intruders.

2.4.2.8.1.1.5.1 Display of Intruder's Relative Altitude

Conditions

This test requires that the traffic display be powered up and configured such that ACAS X information can be displayed. In addition, the display must be receiving data for intruders with relative altitudes of $\pm 2,000$ feet, $\pm 9,900$ feet, $\pm 10,200$ feet, and zero feet. An additional intruder that is not reporting altitude **shall** (2462) be included in the scenario. Each of these intruders **shall** (2463) have a range, relative to the ownship, that is within the available range of a fixed range display or the selected range of a variable range display. The information sent to the display for each intruder **shall** (2464) indicate there is valid bearing and relative altitude data for the intruder. If the display provides multiple, pilot-selectable modes, this test **shall** (2465) be completed for each available display mode.

Success

- a. The display shows at least six intruders. The display shows intruders which have relative altitudes of -20 and -99 shown below an intruder symbol; +20 and +99 shown above an intruder symbol; and 00 shown either above or below an intruder symbol. One intruder is shown with no relative altitude.

Note: The intruders with a relative altitude of $\pm 10,200$ feet are not required to be displayed. However, if they are displayed, they have a relative altitude of either ± 99 or ± 102 associated with the traffic symbol.

- b. The colors of the relative altitude data blocks are the same as the intruder symbol.

2.4.2.8.1.1.5.2 Display of Intruder's Reported Altitude

These tests verify the proper implementation of the optional display feature of displaying actual altitude for an intruder instead of relative altitude.

Test 1 Conditions

This test requires that the traffic display be powered up and configured such that ACAS X information can be displayed. The ownship's altitude is set to FL240 and the display of

actual altitude is selected. In addition, the display must be receiving data for intruders with relative altitudes of $\pm 2,000$ feet, $\pm 9,900$ feet, $\pm 10,200$ feet, and zero feet. Each of these intruders **shall** (2466) have a range, relative to the ownship, that is within the available range of a fixed range display or the selected range of a variable range display. The information sent to the display for each intruder **shall** (2467) indicate there is valid bearing and relative altitude data for the intruder. If the display provides multiple, pilot-selectable modes, this test **shall** (2468) be completed for each available display mode.

Test 1 Success

- a. The display shows at least five intruders. The altitude data blocks show reported altitudes of 220 and 141 below an intruder symbol; 260 and 339 above an intruder symbol; and 240 either above or below an intruder symbol.

Note: The intruders with a relative altitude of $\pm 10,200$ feet are not required to be displayed. However, if they are displayed, the altitude associated with the traffic symbol will be 342 shown above the traffic symbol and 138 shown below the traffic symbol.

- b. The colors of the relative altitude data blocks are the same as the intruder symbol.
- c. An annunciation is shown on the traffic display indicating that the display of actual altitude has been selected.

Test 2 Conditions

This test requires that the traffic display be powered up and configured such that ACAS X information can be displayed. The ownship's altitude is set to 15,000 feet and the display is configured so that it is not receiving the local barometric altitude correction. In addition, the display must be receiving data for intruders with relative altitudes of $\pm 2,000$ feet, $\pm 9,900$ feet, $\pm 10,200$ feet, and zero feet. Each of these intruders **shall** (2469) have a range, relative to the ownship, that is within the available range of a fixed range display or the selected range of a variable range display. The information sent to the display for each intruder **shall** (2470) indicate there is valid bearing and relative altitude data for the intruder. If the display provides multiple, pilot-selectable modes, this test **shall** (2471) be completed for each available display mode.

At t=0, the display of actual altitude is selected.

Test 2 Success

- a. The display shows at least five intruders. The altitude data blocks show reported altitudes of 130 and 051 below an intruder symbol; 170 and 249 above an intruder symbol; and 150 either above or below an intruder symbol.

Note: The intruders with a relative altitude of $\pm 10,200$ feet are not required to be displayed. However, if they are displayed, the altitude associated with the traffic symbol will be 252 shown above the traffic symbol and 048 shown below the traffic symbol.

- b. The colors of the relative altitude data blocks are the same as the intruder symbol.
- c. An annunciation is shown on the traffic display indicating that the display of actual altitude has been selected.
- d. Prior to t=30 seconds, the display automatically reverts back to the display of relative altitude and the annunciation shown in item 3 is removed. The intruders have relative altitudes of -20 and -99 shown below an intruder symbol; +20 and +99 shown above an intruder symbol; and 00 shown either above or below an intruder symbol. The 00

is shown on the same side of the intruder symbol as the 150 when the actual altitude was selected.

Test 3 Conditions

This test requires that the traffic display be powered up and configured such that ACAS X information can be displayed. The ownship's altitude is set to 15,000 feet and the display is configured so that it is receiving the local barometric altitude correction. In addition, the display must be receiving data for intruders with relative altitudes of $\pm 2,000$ feet, $\pm 9,900$ feet, $\pm 10,200$ feet, and zero feet. Each of these intruders **shall** (2472) have a range, relative to the ownship, that is within the available range of a fixed range display or the selected range of a variable range display. The information sent to the display for each intruder **shall** (2473) indicate there is valid bearing and relative altitude data for the intruder. If the display provides multiple, pilot-selectable modes, this test **shall** (2474) be completed for each available display mode.

At t=0, the display of actual altitude is selected.

Test 3 Success

- a. The display shows at least five intruders. The altitude data blocks show reported altitudes of 130 and 051 below an intruder symbol; 170 and 249 above an intruder symbol; and 150 either above or below an intruder symbol.

Note: The intruders with a relative altitude of $\pm 10,200$ feet are not required to be displayed. However, if they are displayed, the altitude associated with the traffic symbol will be 252 shown above the traffic symbol and 048 shown below the traffic symbol.

- b. The colors of the relative altitude data blocks are the same as the intruder symbol.
- c. An annunciation is shown on the traffic display indicating that the display of actual altitude has been selected.
- d. The display of actual altitude continues until the display of relative altitude is manually selected at t=90 seconds.
- e. After the display of relative altitude is manually selected, the display reverts back to the display of relative altitude. The intruders have relative altitudes of -20 and -99 shown below an intruder symbol; +20 and +99 shown above an intruder symbol; and 00 shown either above or below an intruder symbol. The 00 is shown on the same side of the intruder symbol as the 150 when the actual altitude was selected.

2.4.2.8.1.1.6 Intruder Vertical Speed Arrow (§2.2.4.6.5, §2.2.6.1.2.1.6)

This test verifies that the traffic display properly displays the symbol denoting the intruder has a vertical speed in excess of 500 fpm.

Conditions

This test requires that the traffic display be powered up and configured such that ACAS X information can be displayed. In addition, the display must be receiving data for two intruders with an indication that their vertical speeds are outside the range of ± 500 fpm (one climbing and one descending) and for two other intruders with indications that their vertical speeds are within the range of ± 500 fpm (one climbing and one descending). Each of these intruders **shall** (2475) have a range, relative to the ownship, that is within the available range of a fixed range display or the selected range of a variable range display. The information sent to the display for each intruder **shall** (2476) include valid bearing and relative altitude data for the intruder. An additional intruder that is non-altitude reporting,

but with a vertical speed of +1,000 fpm **shall** (2477) be included in the scenario. If the display provides multiple, pilot-selectable modes, this test **shall** (2478) be completed for each available display mode.

Success

- a. The display shows five intruders. One intruder has a vertical arrow pointed down (\downarrow) to the immediate right of the intruder symbol and one intruder has a vertical arrow pointed up (\uparrow) to the immediate right of the intruder symbol. Two intruders do not have a vertical arrow displayed. The non-altitude reporting intruder is displayed with or without a vertical trend arrow.
- b. The colors of the vertical arrows are the same as the intruder symbol.

2.4.2.8.1.1.7 No-Bearing Advisories (§2.2.4.6.6.1, §2.2.6.1.2.1.8, §2.2.6.1.2.1.12)

This test verifies that the traffic display properly displays information on the intruder when a TA or RA is issued against an aircraft with no bearing information displayed. It also verifies that intruders not causing a TA or RA to be issued and for which bearing data are not available are not displayed.

Conditions

This test requires that the traffic display be powered up and configured such that ACAS X information can be displayed. In addition, the display must be initially receiving data for four intruders that have a threat status of either a TA or RA and for which no bearing information is available. The second part of the test eliminates three of the four intruders and adds a no-bearing intruder that is proximate traffic. Finally, the traffic display is configured to display the intruder's actual altitude instead of relative altitude. For this portion of the scenario, the display is assumed to have the local barometric correction.

For the first four intruders, at least one of the intruders **shall** (2479) be non-altitude reporting and causing a TA. The scenario **shall** (2480) be implemented so that the non-altitude reporting intruder is displayed during the initial phase of the scenario. All other intruders **shall** (2481) be altitude reporting and have a vertical speed greater than 500 fpm. At least one intruder **shall** (2482) be classified as causing an RA.

There are no additional requirements for the Proximate Traffic intruder.

Success

- a. The display shows at least two of the four initial intruders using an alpha-numeric string for each intruder. In this string, the threat information is shown in the following order: threat level using the indication "TA" or "RA"; range in NM; relative altitude; and the vertical speed arrow. For the non-altitude reporting TA traffic, the alpha-numeric string contains only the letters "TA" and the intruder's range. There is no maneuver information shown on the traffic display.
- b. The alpha-numeric string is shown in amber or yellow for the intruders causing a TA and in red for those intruders causing an RA.
- c. If less than four intruders are shown, the ones that are not displayed have a lower priority (by order of delivery) than the intruders that are displayed.
- d. When three of the four initial intruders are removed from the scenario and the Proximate Traffic is added, only one no-bearing intruder is shown and it is either a TA or an RA. The no-bearing proximate traffic is not displayed.

- e. When the display is configured to display actual altitude instead of relative altitude, the alpha-numeric string for the no-bearing intruder shows the actual altitude, using three digits, instead of relative altitude.

2.4.2.8.1.1.8 Display of Traffic (§2.2.6.1.2.1.9)

This test verifies that the traffic display includes information on the proper number and types of intruders whenever a TA or an RA is issued.

Conditions

This test requires that the traffic display be powered up and configured such that ACAS X information can be displayed. The test requires that the display receive information on 10 intruders that are prioritized in the same manner used by the CAS logic and which have a range of five NM or less. The test set-up **shall** (2483) ensure that the relative altitude of all 10 intruders is within the available or selected altitude band of the display. One of the intruders **shall** (2484) be classified as a TA or an RA; eight **shall** (2485) be classified as Proximate Traffic; and one **shall** (2486) be classified as Other Traffic.

If the AOTO option is to be supported on a display that is not ED-194 / RTCA/DO-317 compliant then the test setup **shall** (2595) include one TA that is based on a track that has active and successfully validated ADS-B data and one TA that is based only on ADS-B or ADS-R data.

If the display is not a TCAS Only Traffic Display and is not ED-194 / RTCA/DO-317 compliant, then the test setup **shall** (2596) include one target (not in a TA or RA state) that is based on a track that has active and successfully validated ADS-B data, and one target that is based only on ADS-B or ADS-R data.

Success

The display shows at least eight of the ten intruders. These eight intruders **shall** (2487) include the intruder classified as a TA or RA and seven Proximate intruders. The Proximate traffic that is not displayed has the lowest priority of the eight intruders classified as Proximate traffic.

If the AOTO option is to be supported on a display that is not ED-194 / RTCA/DO-317 compliant then verify that the presentation of the TA differentiates the presentation of the TA of the aircraft that includes both ADS-B and TCAS data and is successfully validated – is discernable from the TA for the aircraft that is only tracked based ADS-B or ADS-R data.

If the display is not a TCAS Only Traffic Display and is not ED-194 / RTCA/DO-317 compliant, then verify that the presentation of targets includes display of the target with active and successfully validated ADS-B data, and does NOT include the display of the target based only on ADS-B or ADS-R data.

2.4.2.8.1.1.9 Altitude Band for the Display (§2.2.6.1.2.1.10, §2.2.6.5.2)

This test verifies that the traffic display is capable of displaying intruders within the specified relative altitude of the ownship.

Conditions

This test is performed in three parts and all require that the traffic display be powered up and configured such that ACAS X information can be displayed.

The first part of the test verifies that traffic within $\pm 2,700$ feet of the ownship are displayed and that any intruder causing a TA or an RA outside this altitude band is displayed. On traffic displays which have a capability of expanded altitude display, the display **shall**

(2488) be configured to only show traffic within $\pm 2,700$ feet. Five intruders are used in this part of the test; three have relative altitudes within a range between $\pm 2,700$ feet and two have relative altitudes outside the range of $\pm 2,700$ feet. One of the intruders with an initial relative altitude outside the range of $\pm 2,700$ feet has a vertical speed towards the ownship such that its relative altitude becomes less than 2,700 feet at $t=10$ seconds. At $t=20$ seconds, the remaining aircraft with a relative altitude greater than 2,700 feet causes a TA or RA to be issued.

The second and third parts of the test verify the proper implementation of the expanded altitude band function. Six intruders are used in the second part of the test; three have relative altitudes between +9,900 and -2,700 feet; one has a relative altitude of -3,000 feet; and two have relative altitudes greater than +9,900 feet. The display is configured in the “Above” mode. One of the intruders with an initial relative altitude greater than +9,900 feet has a vertical speed towards the ownship such that its relative altitude becomes less than 9,900 feet at $t=10$ seconds.

Six intruders are used in the third part of the test; three have relative altitudes between -9,900 and +2,700 feet; one has a relative altitude of +3,000 feet; and two have relative altitudes in excess of +9,900 feet. The display is configured in the “Below” mode. One of the intruders with an initial relative altitude in excess of -9,900 feet has a vertical speed towards the ownship such that its relative altitude becomes less than 9,900 feet at $t=10$ seconds.

Success

Part 1:

- a. At $t=0$, three intruders are displayed and all have relative altitudes within a range between $\pm 2,700$ feet.
- b. At $t=10$ seconds, a fourth intruder is displayed and it also has a relative altitude within a range between $\pm 2,700$ feet.
- c. At $t=20$ seconds, the fifth intruder is shown as a TA or RA with a relative altitude outside the range of $\pm 2,700$ feet.

Part 2:

- a. At $t=0$, three intruders are displayed and all have relative altitudes between -2,700 and +9,900 feet.
- b. At $t=10$ seconds, a fourth intruder is displayed and it also has a relative altitude between -2,700 and +9,900 feet.
- c. The sixth intruder is never displayed.
- d. The selection of the “Above” mode is annunciated on the display.

Part 3:

- a. At $t=0$, three intruders are displayed and all have relative altitudes between +2,700 and -9,900 feet.
- b. At $t=10$ seconds, a fourth intruder is displayed and it also has a relative altitude between +2,700 and -9,900 feet.
- c. The sixth intruder is never displayed.
- d. The selection of the “Below” mode is annunciated on the display.

2.4.2.8.1.1.10 Display of ACAS X Operating Mode (§2.2.6.1.2.1.11, §2.2.6.6.2, §2.2.6.1.2.1.14)

This test verifies that the traffic display properly annunciates the current operating mode of ACAS X.

Conditions

This test requires that the traffic display be powered up and configured such that ACAS X information can be displayed. Sequential signals are provided to indicate that ACAS X is in Standby, TA-Only, and TA/RA mode. Sequential signals **shall** (2489) also be provided to indicate that ACAS X has failed, that the traffic display is unable to display traffic, and that a pilot-initiated Self Test is in progress.

Success

- a. When the display receives an indication that ACAS X is in Standby, this state is clearly annunciated on the traffic display. This annunciation is shown in a color other than yellow, amber, or red.
- b. When the display receives an indication that ACAS X is in the TA-Only mode, this state is clearly annunciated on the traffic display. This annunciation is shown in a color other than yellow, amber, or red when no TA is present.
- c. When the display receives an indication that ACAS X is in the TA/RA mode, this state is clearly annunciated on the traffic display. This annunciation is shown in a color other than yellow, amber, or red.

Note: It is acceptable for the traffic display to not provide an annunciation indicating the system is operating in the TA/RA mode. The lack of an annunciation on some displays of the TA/RA mode, along with annunciations showing other operating modes provides the pilot with a unique indication of the system's operating status.

- d. When the display receives an indication that ACAS X has failed, this state is clearly annunciated on the traffic display. This annunciation is shown in either yellow or amber.
- e. When the display receives an indication that the traffic display has failed, this state is clearly annunciated on the traffic display. This annunciation is shown in either yellow or amber.
- f. When the display receives an indication that a pilot initiated Self Test is in progress, this state is clearly annunciated on the traffic display. This annunciation is shown in a color other than yellow, amber, or red.
- g. All ACAS X-related annunciations have a single meaning which is consistent in all available display modes.
- h. On shared displays, the ACAS X annunciations are consistent with all other annunciations shown on the display.

Note: The completion of this step may require additional inputs to the display to generate the other annunciations.

2.4.2.8.1.2 Fixed Range Displays

There are no unique tests for fixed range displays.

Note: The unique requirements for the fixed range display are tested using the relevant test procedures shown in §2.4.2.8.1.1.

2.4.2.8.1.3 Variable Range Displays

2.4.2.8.1.3.1 Display Range (§2.2.6.1.2.3.1)

This test verifies that a variable range traffic display provides a pilot selectable range that is appropriate for use in the immediate vicinity of an airport.

Conditions

This test requires that the traffic display be powered up and configured such that ACAS X information can be displayed. The display is configured so that the lowest selectable display range is shown.

Note: If the traffic display provides a display range less than five miles, the five mile range (or closest available range) shall (2490) be selected for this test. A simulated intruder that is classified as Proximate or Other traffic begins the scenario at the 12 o'clock position and at a range of eight miles. The intruder moves toward the ownship and ends the scenario at the six o'clock position at a range of four miles.

Success

- a. At the beginning of the scenario, the intruder is not displayed. The traffic display has an annunciation of the range selected.
- b. The intruder is initially shown on the display at a maximum range of six NM and remains displayed until it is at least 2.5 NM to the rear of the ownship.

Note: On shared displays, the intruder is not required to be displayed after it passes the ownship reference symbol.

2.4.2.8.1.3.2 Range Selection (§2.2.6.1.2.3.2)

This test verifies that all available display ranges can be selected and that an intruder is shown at the proper location, relative to the ownship, in all display ranges.

Conditions

This test requires that the traffic display be powered up and configured such that ACAS X information can be displayed. The display is initially configured in the lowest available display range. All higher ranges are subsequently selected. An intruder is required to be shown at a range of nine NM at the 12 o'clock position. The test also requires a means of selecting all available display ranges. This selection may be done using controls located on the traffic display, with a connected control panel, or with simulated inputs.

Success

- a. At the beginning of the test, the intruder is not displayed and the selected range is annunciated on the display.
- b. For selected ranges less than nine NM, no intruder is displayed. The selected range is annunciated on the display.
- c. For selected ranges greater than nine NM, the intruder is displayed at the proper range and bearing. The selected range is annunciated on the display.

2.4.2.8.1.4 Part Time Displays

2.4.2.8.1.4.1 Activation and Information Displayed (§2.2.6.1.2.4.1, §2.2.6.1.2.4.2)

This test verifies that a part-time display activates at the appropriate time, displays the proper information when activated, and continues to display traffic information for the appropriate amount of time.

Conditions

This test requires that the traffic display be powered up and configured such that ACAS X information is displayed on a part time basis. It requires the presence of an intruder that is classified as Proximate Traffic and has a range of four NM; an intruder that is classified as a TA; and an intruder that is classified as an RA, but for which no TA was issued. It also requires the capability to select various display modes on a shared display. This selection may be done via controls located on the traffic display, with a connected control panel, or using simulated inputs.

The Proximate Traffic information is sent to the display throughout the scenario. Thirty seconds ($t=30$) after the start of the scenario, information for the TA is sent to the display. This information is sent for 30 seconds before being removed at $t=60$ seconds. If a written text message is shown on the display instead of the TA symbology, the display **shall** (2491) be reconfigured to show the traffic symbology after the TA is issued. After a 30 second delay (at $t=90$), information on the intruder causing an RA is sent to the display. This information is sent for 30 seconds before being removed at $t=120$ seconds. If a written text message is shown on the display instead of the RA symbology, the display **shall** (2492) be reconfigured to show the traffic symbology after the RA is issued. After a delay of 30 seconds (at $t=150$), the information on the intruder causing the TA and the intruder causing the RA (with no preceding TA) is sent to the display. This information is sent for 30 seconds before being removed at $t=180$ seconds. If a written text message is shown on the display instead of the RA symbology, the display **shall** (2493) be reconfigured to show the traffic symbology after the RA is issued. If the display supports another mode which supports the full time display of traffic, it is selected while the RA is displayed.

Success

- a. At the start of the scenario, the display shows no traffic information. If the display includes an annunciation which indicates that ACAS X is operating in a part time mode, it is permissible for this annunciation to be displayed.
- b. When the information on the intruder causing a TA is sent to the display, the intruder is shown as either a yellow or amber filled circle at the proper range and bearing. The display includes the appropriate ownship reference symbol, required range markings, and range annunciation. The Proximate Traffic is also shown at the proper range and bearing. As an alternative, the written text “TRAFFIC”, “TFC”, or “TCAS” is shown on the display in amber or yellow. If this written text is shown, the display configuration **shall** (2494) be changed to display the TA symbology. When this change is made, the display shows the appropriate ownship reference symbol, required range markings, range annunciation, the TA, and the Proximate Traffic. Either the TA symbology or the written text is shown within 0.5 seconds of the information being sent to the display.
- c. After 30 seconds ($t=60$), all traffic information is removed from the display. If the display includes an annunciation which indicates that ACAS X is operating in a part time mode, it is permissible for this annunciation to be displayed.
- d. When the information on the intruder causing a RA is sent to the display, the intruder is shown as a red filled square at the proper range and bearing. The display includes

the appropriate ownship reference symbol, required range markings, and range annunciation. The Proximate Traffic is also shown at the proper range and bearing. As an alternative, the written text “TRAFFIC”, “TFC”, or “TCAS” is shown on the display in red. If this written text is shown, the display configuration **shall** (2495) be changed to display the RA symbology. When this change is made, the display shows the appropriate ownship reference symbol, required range markings, range annunciation, the RA, and the Proximate Traffic. Either the RA symbology or the written text is shown within 0.5 seconds of the information being sent to the display.

- e. After 30 seconds (at t=120), all traffic information is removed from the display. If the display includes an annunciation which indicates that ACAS X is operating in a part time mode, it is permissible for this annunciation to be displayed.
- f. When the information on the intruders causing an RA and a TA is simultaneously sent to the display, the RA intruder is shown as a red filled square and the TA intruder is shown as either a yellow or amber filled circle. Both are shown at the proper range and bearing. The display includes the appropriate ownship reference symbol, required range markings, and range annunciation. The Proximate Traffic is also shown at the proper range and bearing. As an alternative, the written text “TRAFFIC”, “TFC”, or “TCAS” is shown on the display in red. If this written text is shown, the display configuration **shall** (2496) be changed to display the traffic symbology. When this change is made, the display shows the appropriate ownship reference symbol, required range markings, range annunciation, the RA, the TA, and the Proximate Traffic. Either the RA and TA symbology or the written text is shown within 0.5 seconds of the information being sent to the display.
- g. After 30 seconds (at t=180), all traffic information is removed from the display unless the traffic display includes a non-part-time mode which is selected while the TA and RA are displayed. If such a mode was selected, the Proximate Traffic remains displayed after the TA and RA information is removed from the display.

2.4.2.8.1.5 Full Time Displays

There are no unique tests for full time displays.

Note: Full time displays are tested using the relevant procedures shown in §2.4.2.8.1.1.

2.4.2.8.1.6 Dedicated Displays

There are no unique tests for dedicated displays.

Note: The unique requirements for dedicated displays are tested using the relevant procedures shown in §2.4.2.8.1.1.

2.4.2.8.1.7 Shared Displays

2.4.2.8.1.7.1 Shared Weather Radar Displays

2.4.2.8.1.7.1.1 Available Display Modes (§2.2.6.1.2.7.1.1)

This test verifies that the required and optional display modes are available when the ACAS X traffic display uses the same display as the aircraft’s weather radar.

Conditions

This test requires that the shared weather radar and traffic display be powered up and the capability to select various display modes is available. This capability may be provided via controls located on the display, a connected control panel, or simulated inputs. All available display modes are selected.

Success

- a. During the selection of the display modes, either a WX/ACAS X or a WX-and-Traffic is available.
- b. During the selection of the display modes, the optional WX-Only or Traffic-Only modes may be displayed.

2.4.2.8.1.7.1.2 Display Mode Characteristics (§2.2.6.1.2.7.1.1)

This test verifies that the required features are properly implemented in the required and optional display modes when the ACAS X traffic display uses the same display as the aircraft's weather radar.

Conditions

This test requires that the shared weather radar and traffic display be powered up and the capability to select various display modes is available. This capability may be provided via controls located on the display, a connected control panel, or simulated inputs. Simulated weather radar returns are provided to the display throughout the scenario. Separate scenario descriptions are provided for each display mode tested.

2.4.2.8.1.7.1.2.1WX/ACAS X Mode

Description

This test requires the presence of an intruder that is classified as Proximate Traffic and has a range of four NM; an intruder that is classified as a TA; and an intruder that is classified as an RA, but for which no TA was issued. It also requires the capability to select various display ranges on a variable range display. This selection may be done via controls located on the traffic display, with a connected control panel, or using simulated inputs.

The Proximate Traffic information is sent to the display throughout the scenario. Thirty seconds after the start of the scenario ($t=30$), information for the TA is sent to the display. This information is sent for 30 seconds before being removed at $t=60$ seconds. After a 30 second delay (at $t=90$), information on the intruder causing an RA is sent to the display. This information is sent for 30 seconds before being removed at $t=120$ seconds. If the display is a variable range display, the selected display range is increased while the RA is displayed.

Success

- a. At the start of the scenario, the display shows weather, but no traffic information. If the display includes an annunciation which indicates that ACAS X is operating in a part time mode, it is permissible for this annunciation to be displayed.
- b. When the information on the intruder causing a TA is sent to the display at $t=30$, the weather information is removed and the intruder is automatically shown as either a yellow or amber filled circle at the proper range and bearing. The display includes the appropriate ownship reference symbol, required range markings, and range annunciation. The Proximate Traffic is also shown at the proper range and bearing. If the ACAS X display is a variable range display, the display range is the same as the

- range selected for the display of the weather information. If the ACAS X display is a fixed range display, the display range is approximately six NM. The traffic information is shown within 0.5 seconds of the information being sent to the display.
- c. After 30 seconds (at t=60), all traffic information is removed from the display and the weather information is automatically redisplayed. The display range is the same as that selected at the start of the scenario. If the display includes an annunciation which indicates that ACAS X is operating in a part time mode, it is permissible for this annunciation to be displayed.
 - d. When the information on the intruder causing a RA is sent to the display at t=90, the weather information is removed and the intruder is automatically shown as a red filled square at the proper range and bearing. The display includes the appropriate ownship reference symbol, required range markings, and range annunciation. The Proximate Traffic is also shown at the proper range and bearing. If the ACAS X display is a variable range display, the display range is the same as the range selected for the display of the weather information. If the ACAS X display is a fixed range display, the display range is approximately six NM. The traffic information is shown within 0.5 seconds of the information being sent to the display. When the display range is increased and the ACAS X information is shown on a variable range display, the ACAS X information **shall** (2497) remain displayed at the proper range and bearing. When the display range is increased and the ACAS X information is shown on a fixed range display, there is no effect on the traffic display.
 - e. After 30 seconds (at t=120), all traffic information is removed from the display and the weather information is automatically redisplayed. The weather information is shown referenced to the increased range selected while the RA was displayed. If the display includes an annunciation which indicates that ACAS X is operating in a part time mode, it is permissible for this annunciation to be displayed.

2.4.2.8.1.7.1.2.2WX-and-Traffic Mode

Description

This test requires the presence of an intruder that is classified as Proximate Traffic and has a range of four NM; an intruder that is classified as a TA; and an intruder that is classified as an RA, but for which no TA was issued. It also requires the capability to select various display ranges. This selection may be done via controls located on the traffic display, with a connected control panel, or using simulated inputs. It also requires the presence of an intruder causing a TA at the six o'clock position and an intruder causing an RA at the six o'clock position.

The Proximate Traffic information is sent to the display throughout the scenario. Thirty seconds after the start of the scenario (t=30), information for the TA is sent to the display. The intruder causing the TA is located at the same position as a portion of the displayed weather. This information is sent for 30 seconds before being removed at t=60 seconds. After a 30 second delay (at t=90), information on the intruder causing an RA is sent to the display. The intruder causing the RA is located at the same position as a portion of the displayed weather. This information is sent for 30 seconds before being removed at t=120 seconds. The selected display range is increased while the RA is displayed. After waiting for 30 seconds (at t=150), information on the intruder causing the TA at six o'clock is sent to the display for a 30 second period (until t=180). Thirty seconds after the TA at six o'clock is removed (at t=210), information on the RA located at the six o'clock position is sent to the display for a 30 second period (until t=240).

Success

- a. At the start of the scenario, the display shows weather and the Proximate Traffic if the ACAS X information is shown on a full time basis. If the ACAS X information is shown on a part time basis, no traffic information is displayed. If a part time display includes an annunciation which indicates that ACAS X is operating in a part time mode, it is permissible for this annunciation to be displayed.
- b. When the information on the intruder causing a TA is sent to the display (at t=30) and the ACAS X information is shown on a part time basis, both the TA and the Proximate Traffic are shown on the display at the proper range and bearing. The traffic information is referenced to the weather radar's ownship symbol and shown at the range selected for the initial display of the weather information. If the ACAS X information is shown on a full time basis, the Proximate Traffic remains displayed and the TA symbol is added to the display at the proper range and bearing. The TA information is readily discernible and readable. Any information from other aircraft systems, with the exception of navigation data, sharing the weather radar display is automatically removed when the TA is issued.
- c. After 30 seconds (at t=60), all traffic information is removed from the display if the ACAS X display is implemented on a part time basis, but the weather information remains displayed. The display range is the same as that selected at the start of the scenario. If the display includes an annunciation which indicates that ACAS X is operating in a part time mode, it is permissible for this annunciation to be displayed. If the ACAS X display is implemented on a full time basis, the Proximate Traffic remains displayed at the range selected at the start of the scenario. Information from other aircraft systems sharing the weather radar display is automatically restored.
- d. When the information on the intruder causing an RA is sent to the display at t=90 seconds, and the ACAS X information is shown on a part time basis, both the RA and the Proximate Traffic are shown on the display at the proper range and bearing. The traffic information is referenced to the weather radar's ownship symbol and shown at the range selected for the initial display of the weather information. If the ACAS X information is shown on a full time basis, the Proximate Traffic remains displayed and the RA symbol is added to the display at the proper range and bearing. The RA information is readily discernible and readable. When the display range is increased, both the weather and traffic information are displayed relative to the increased range. Any information from other aircraft systems, with the exception of navigation data, sharing the weather radar display is automatically removed when the RA is issued.
- e. After 30 seconds (t=120), all traffic information is removed from the display if the ACAS X display is implemented on a part time basis, but the weather information remains displayed. The display range is the increased range selected while the RA was displayed. If the display includes an annunciation which indicates that ACAS X is operating in a part time mode, it is permissible for this annunciation to be displayed. If the ACAS X display is implemented on a full time basis, the Proximate Traffic remains displayed at the increased range selected while the RA was displayed. Information from other aircraft systems sharing the weather radar display is automatically restored.
- f. When the information on the intruder at six o'clock causing a TA is sent to the display at t=150 seconds, a written message is shown in either amber or yellow which indicates the TA is behind the ownship. The message may be "TA BEHIND", "TRAFFIC", or "TFC". When the display mode is changed to permit the TA traffic to be displayed, it is shown at the proper range and bearing. Once the TA symbol is displayed, the display mode is returned to the WX-and-Traffic mode.

- g. After 30 seconds ($t=180$), the written message for the TA is removed from the display, but the weather information remains displayed. The display range is the same as that selected at the start of the scenario. If the ACAS X display is implemented on a part time basis, all traffic information is removed from the display. If the display includes an annunciation which indicates that ACAS X is operating in a part time mode, it is permissible for this annunciation to be displayed. If the ACAS X display is implemented on a full time basis, the Proximate Traffic remains displayed at the range selected at the start of the scenario.
- h. When the information on the intruder at six o'clock causing a RA is sent to the display at $t=210$ seconds, a written message is shown in red which indicates the RA is behind the ownship. The message may be "RA BEHIND", "TRAFFIC", or "TFC". When the display mode is changed to permit the RA traffic to be displayed, it is shown at the proper range and bearing. Once the RA symbol is displayed, the display mode is returned to the WX-and-Traffic mode.
- i. After 30 seconds ($t=240$), the written message for the RA is removed from the display, but the weather information remains displayed. The display range is the same as that selected at the start of the scenario. If the ACAS X display is implemented on a part time basis, all traffic information is removed from the display. If the display includes an annunciation which indicates that ACAS X is operating in a part time mode, it is permissible for this annunciation to be displayed. If the ACAS X display is implemented on a full time basis, the Proximate Traffic remains displayed at the range selected at the start of the scenario.

2.4.2.8.1.7.1.2.3WX-Only Mode

Description

This test requires the presence of an intruder that is classified as Proximate Traffic and has a range of four NM; an intruder that is classified as a TA; and an intruder that is classified as an RA, but for which no TA was issued.

The Proximate Traffic information is sent to the display throughout the scenario. Thirty seconds after the start of the scenario ($t=30$), information for the TA is sent to the display. This information is sent for 30 seconds before being removed at $t=60$ seconds. After a 30 second delay ($t=90$), information on the intruder causing an RA is sent to the display. This information is sent for 30 seconds before being removed at $t=120$ seconds.

Success

- a. At the start of the scenario, the display shows only weather.
- b. When the information on the intruder causing a TA is sent to the display (at $t=30$), the displayed information does not change and no traffic information is shown. During the 30 seconds the TA information is sent to the display (until $t=60$), the display mode is manually changed and the traffic symbology is shown at the proper location relative to the ownship.
- c. After 30 seconds ($t=60$), all traffic information is removed from the display and the display automatically reverts back to the WX-Only mode.
- d. When the information on the intruder causing an RA is sent to the display at $t=90$ seconds, the displayed information does not change and no traffic information is shown. During the 30 seconds the RA information is sent to the display (until $t=120$), the display mode is manually changed and the traffic symbology is shown at the proper location relative to the ownship.

- e. After 30 seconds ($t=120$), all traffic information is removed from the display and the display automatically reverts back to the WX-Only mode.

2.4.2.8.1.7.1.2.4 Traffic-Only Mode

Description

This test requires the presence of an intruder that is classified as Proximate Traffic and has a range of four NM; an intruder that is classified as a TA; and an intruder that is classified as an RA, but for which no TA was issued.

The Proximate Traffic information is sent to the display throughout the scenario. Thirty seconds ($t=30$) after the start of the scenario, information for the TA is sent to the display. This information is sent for 30 seconds before being removed at $t=60$ seconds. After a 30 second delay ($t=90$), information on the intruder causing an RA is sent to the display. This information is sent for 30 seconds before being removed at $t=120$ seconds.

Success

- a. At the start of the scenario, no weather information is shown and the Proximate Traffic is displayed at the proper range and bearing.
- b. When the information on the intruder causing a TA is sent to the display at $t=30$ seconds, the TA is shown at the proper range and bearing, while the Proximate Traffic remains displayed. No weather information is displayed.
- c. After 30 seconds ($t=60$), the TA is removed from the display and the Proximate Traffic remains displayed. No weather information is displayed.
- d. When the information on the intruder causing an RA is sent to the display at $t=90$ seconds, the RA is shown at the proper range and bearing, while the Proximate Traffic remains displayed. No weather information is displayed.
- e. After 30 seconds ($t=120$), the RA is removed from the display and the Proximate Traffic remains displayed. No weather information is displayed.

2.4.2.8.1.7.2 Navigation Displays or Electronic Horizontal Situation Indicators

The tests in this subparagraph verify that traffic information displayed on a Navigation Display or EHSI meets the appropriate display requirements.

2.4.2.8.1.7.2.1 Failure Annunciations (§2.2.6.1.2.7.2.2)

This test verifies that the ACAS X failure annunciations are shown on the ND or EHSI.

Conditions

This test requires that the ND or EHSI be powered up and configured to display ACAS X information. The test setup **shall** (2498) include means of simulating failure modes and selecting various ACAS X operating modes. The conditions to be introduced include operating ACAS X in Standby or having ACAS X turned off; TA-Only mode; ACAS X failed; a condition in which the ND or EHSI is unable to display traffic; and selecting a pilot initiated Self Test. These conditions **shall** (2499) be simulated for each available display mode of the ND or EHSI. The mode selections may be made via controls located on the display, a control panel connected to the display, or using simulated inputs.

Success

- a. When ACAS X is operating in Standby or turned off, an annunciation is shown on the ND or EHSI in a color other than yellow or amber.

- b. When ACAS X is operating in the TA-Only mode, an annunciation is shown on the ND or EHSI in a color other than either yellow or amber when no TA is present.
- c. When ACAS X is failed, an annunciation is shown on the ND or EHSI in either yellow or amber.
- d. When the ND or EHSI is unable to display ACAS X information, an annunciation is displayed in either yellow or amber.
- e. When a pilot-initiated Self Test is in progress, an annunciation is shown on the ND or EHSI in a color other than yellow or amber.

2.4.2.8.1.7.2.2 Display of Traffic (§2.2.6.1.2.7.2.3)

This test verifies that traffic information is displayed in the required modes of the ND or EHSI. It also verifies that when a mode not supporting the display of traffic is selected, a suitable message is displayed when a TA or RA is issued.

Conditions

This test requires that the ND or EHSI be powered up, the MAP mode selected, and configured to display ACAS X information. The test setup **shall** (2500) include means of selecting all available display modes of the ND or ND or EHSI. The mode selections may be made via controls located on the display, a control panel connected to the display, or using simulated inputs.

The test scenario requires an intruder causing a TA to be displayed for a 30 second period which then becomes an RA for an additional 30 seconds. The display of the TA and RA **shall** (2501) be verified in all display modes of the ND or EHSI which supports the display of ACAS X information.

Success

- a. When the TA is issued ($t=0$), a yellow or amber filled circle is displayed at the proper bearing and range relative to the ownship. When the RA is issued at $t=30$ seconds, a red filled square is displayed. This information is displayed in all display modes which support the display of traffic.
- b. When a display mode does not support the display of traffic, a written message of "TRAFFIC", "TFC", or "TCAS" is displayed on the ND or EHSI in either amber or yellow. Immediately after this message is shown, the display **shall** (2502) be switched to a mode in which traffic can be displayed. An amber or yellow filled circle is shown at the proper range and bearing. While the TA is still displayed, the display is returned to a mode which does not display traffic. When the RA is issued at $t=30$ seconds, the color of the written message changes to red.

2.4.2.8.1.7.2.3 ACAS X and Weather Information (§2.2.6.1.2.7.2.4)

This test verifies that ACAS X information is discernible when shown on the same display as weather radar information.

Conditions

This test requires that the ND or EHSI be powered up and configured to display both ACAS X and weather radar information. The test requires the presence of intruders classified as Other Traffic, Proximate Traffic, a TA, and an RA. The test also requires that weather radar returns be shown on the display. All traffic **shall** (2503) be located at the same place as the weather radar returns.

Success

All types of traffic are readily discernible from the displayed weather information.

2.4.2.8.1.7.2.4 ACAS X and Navigation Information (§2.2.6.1.2.7.2.5)

This test verifies that ACAS X information is not hidden from view by navigation information.

Conditions

This test requires that the ND or EHSI be powered up and configured to display ACAS X. The test requires the presence of intruders classified as Other Traffic, Proximate Traffic, a TA, and an RA. The test also requires that all types of available navigation information supported by the ND or EHSI (routes, waypoints, airports, etc.,) be shown on the display. All traffic **shall** (2504) be located at the same place as the various types of navigation data. This test **shall** (2505) be completed for all ND or EHSI display modes which provide for the display of traffic.

Success

The displayed traffic information remains readable.

2.4.2.8.1.7.2.5 HSI Display Mode (§2.2.6.1.2.7.2.6.1)

This test verifies that the display of ACAS X information is properly implemented in the HSI mode.

Conditions

This test requires that the ND or EHSI be powered up, in the HSI display mode, and configured to display ACAS X information. The test requires the presence of intruders classified as Proximate Traffic, a TA, and an RA. The Proximate Traffic is displayed throughout the encounter. The TA is initially displayed 30 seconds after the start of the test ($t=30$) and is displayed for 30 seconds before being removed at $t=60$ seconds. While the TA is displayed, the display range is increased if the HSI mode provides for this capability. After 30 seconds ($t=90$), an intruder causing the RA is displayed (without a preceding TA) for 30 seconds.

If the HSI display mode is implemented as a variable range display, the test setup **shall** (2506) include means of modifying the display range. The range selections may be made via controls located on the display, a control panel connected to the display, or using simulated inputs.

Success

- a. At the beginning of the test scenario, no traffic information is displayed if the ACAS X information is shown on a part time basis. If the part time display implementation includes an annunciation indicating that ACAS X is operating in a part time mode, it is permissible for this annunciation to be displayed. If the ACAS X information is shown on a full time basis, the Proximate Traffic is shown at the proper range and bearing.
- b. When the TA is issued at $t=30$ seconds, an amber or yellow filled circle **shall** (2507) be shown at the proper range and bearing. The Proximate Traffic **shall** (2508) also be shown at the proper range and bearing.
- c. When the display range is increased on a variable range display, the TA and Proximate Traffic remain displayed at the correct range and bearing, relative to the ownship symbol, and the display range increases.

- d. When the TA is removed after 30 seconds ($t=60$), both the TA and Proximate Traffic symbol are removed and no traffic information is displayed when a part time display is used. If the display implementation includes an annunciation indicating that ACAS X is operating in a part time mode, it is permissible for this annunciation to be displayed. On a full time display, the Proximate Traffic will remain displayed at the proper range and bearing.
- e. When the RA is issued at $t=90$ seconds, a red filled square **shall** (2509) be shown at the proper range and bearing. The Proximate Traffic **shall** (2510) also be shown at the proper range and bearing.
- f. When the RA is removed after 30 seconds ($t=120$), both the RA and Proximate Traffic symbol are removed and no traffic information is displayed if the display is implemented on a part time basis. If the display implementation includes an annunciation indicating that ACAS X is operating in a part time mode, it is permissible for this annunciation to be displayed. On a full time display, the Proximate Traffic will remain displayed.

2.4.2.8.1.7.2.6 EXPANDED Mode (§2.2.6.1.2.7.2.7.1)

This test verifies that the display of ACAS X information is properly implemented in the EXPANDED mode.

Conditions

This test requires that the ND or EHSI be powered up, in the EXPANDED display mode, and configured to display ACAS X information. The test requires the presence of intruders classified as Proximate Traffic, a TA, and an RA. The Proximate Traffic is displayed throughout the encounter. The TA is initially displayed 30 seconds after the start of the test ($t=30$) and is displayed for 30 seconds before being removed at $t=60$ seconds. While the TA is displayed, the display range is increased. After 30 seconds ($t=90$), an intruder causing the RA is displayed (without a preceding TA) for 30 seconds.

The test setup **shall** (2511) include means of modifying the display range of the EXPANDED mode. The mode selections may be made via controls located on the display, a control panel connected to the display, or using simulated inputs.

Success

- a. At the beginning of the test scenario, no traffic information is displayed if the ACAS X information is shown on a part time mode. If the part time display implementation includes an annunciation indicating that ACAS X is operating in a part time mode, it is permissible for this annunciation to be displayed. If the ACAS X information is shown on a full time basis, the Proximate Traffic is shown at the proper range and bearing.
- b. When the TA is issued at $t=30$ seconds, an amber or yellow filled circle **shall** (2512) be shown at the proper range and bearing relative to the ownship symbol. The Proximate Traffic **shall** (2513) also be shown at the proper range and bearing.
- c. When the display range is increased, the TA and Proximate Traffic remain displayed at the correct range and bearing, relative to the ownship symbol, and the display range increases.
- d. When the TA is removed after 30 seconds ($t=60$), both the TA and Proximate Traffic symbol are removed and no traffic information is displayed when a part time display is used. If the display implementation includes an annunciation indicating that ACAS X is operating in a part time mode, it is permissible for this annunciation to be displayed.

displayed. On a full time display, the Proximate Traffic will remain displayed at the proper range and bearing.

- e. When the RA is issued at t=90 seconds, a red filled square **shall** (2514) be shown at the proper range and bearing. The Proximate Traffic **shall** (2515) also be shown at the proper range and bearing relative to the ownship symbol.
- f. When the RA is removed after 30 seconds (t=120), both the RA and Proximate Traffic symbol are removed and no traffic information is displayed if the display is implemented on a part time basis. If the display implementation includes an annunciation indicating that ACAS X is operating in a part time mode, it is permissible for this annunciation to be displayed. On a full time display, the Proximate Traffic will remain displayed.

2.4.2.8.1.7.2.7 MAP Mode

2.4.2.8.1.7.2.7.1 Display Characteristics (§2.2.6.1.2.7.2.8.1)

This test verifies that the display of ACAS X information is properly implemented in the MAP mode.

Conditions

This test requires that the ND or EHSI be powered up, in the MAP display mode, and configured to display ACAS X information. The test requires the presence of intruders classified as Proximate Traffic, a TA, and an RA. The Proximate Traffic is displayed throughout the encounter. The TA is initially displayed 30 seconds after the start of the test (t=30) and is displayed for 30 seconds before being removed at t=60 seconds. While the TA is displayed, the display range is increased. After 30 seconds (t=90), an intruder causing the RA is displayed (without a preceding TA) for 30 seconds.

The test setup **shall** (2516) include means of modifying the display range of the MAP mode. The mode selections may be made via the mode control panel for the ND or EHSI connected to the display, or using simulated inputs of this control panel.

Success

- a. At the beginning of the test scenario, no traffic information is displayed if the ACAS X information is shown on a part time mode. If the part time display implementation includes an annunciation indicating that ACAS X is operating in a part time mode, it is permissible for this annunciation to be displayed. If the ACAS X information is shown on a full time basis, the Proximate Traffic is shown at the proper range and bearing.
- b. When the TA is issued at t=30 seconds, an amber or yellow filled circle **shall** (2517) be shown at the proper range and bearing relative to the ownship symbol. The Proximate Traffic **shall** (2518) also be shown at the proper range and bearing.
- c. When the display range is increased, the TA and Proximate Traffic remain displayed at the correct range and bearing, relative to the ownship symbol, and the display range increases.
- d. When the TA is removed after 30 seconds (t=60), both the TA and Proximate Traffic symbol are removed and no traffic information is displayed when a part time display is used. If the display implementation includes an annunciation indicating that ACAS X is operating in a part time mode, it is permissible for this annunciation to be displayed. On a full time display, the Proximate Traffic will remain displayed at the proper range and bearing.

- e. When the RA is issued at t=90 seconds, a red filled square **shall** (2519) be shown at the proper range and bearing relative to the ownship symbol. The Proximate Traffic **shall** (2520) also be shown at the proper range and bearing.
- f. When the RA is removed after 30 seconds (t=120), both the RA and Proximate Traffic symbol are removed and no traffic information is displayed if the display is implemented on a part time basis. If the display implementation includes an annunciation indicating that ACAS X is operating in a part time mode, it is permissible for this annunciation to be displayed. On a full time display, the Proximate Traffic will remain displayed.

2.4.2.8.1.7.2.7.2Display Range and Orientation (§2.2.6.1.2.7.2.8.2; §2.2.6.1.2.7.2.8.3)

This test verifies that the MAP mode does not include an automatic range reversion feature and that a heading up orientation is available when a TA or an RA is issued.

Conditions

This test requires that the ND or EHSI be powered up, in the MAP display mode with the maximum available display range selected, and configured to display ACAS X information. The test requires the presence of intruders classified as Proximate Traffic, a TA, and an RA. The Proximate Traffic is displayed throughout the encounter. The TA is initially displayed 30 seconds after the start of the test (t=30) and is displayed for 30 seconds before being removed at t=60 seconds. After 30 seconds (t=90), an intruder causing the RA is displayed (without a preceding TA) for 30 seconds.

The test setup requires that the track shown on the MAP is at least 30 degrees different than the heading shown on the display. The test setup **shall** (2521) provide a means of modifying the display mode if the MAP mode does not display traffic information in a heading up orientation. The mode selections may be made via the mode control panel for the ND or EHSI connected to the display, or using simulated inputs of this control panel.

Success

- a. At the beginning of the test scenario, no traffic information is displayed if the ACAS X information is shown on a part time mode. If the part time display implementation includes an annunciation indicating that ACAS X is operating in a part time mode, it is permissible for this annunciation to be displayed. If the ACAS X information is shown on a full time basis, the Proximate Traffic is shown at the proper range and bearing.
- b. When the TA is issued at t=30 seconds, an amber or yellow filled circle **shall** (2522) be shown at the proper range and bearing relative to the ownship symbol. The Proximate Traffic **shall** (2523) also be shown at the proper range and bearing. The selected display range of the MAP display does not change. The traffic information is shown in a heading up orientation on the MAP or a display in a heading up orientation is available for selection via the display mode control panel.
- c. When the TA is removed after 30 seconds (t=60), both the TA and Proximate Traffic symbol are removed and no traffic information is displayed when a part time display is used. If the display implementation includes an annunciation indicating that ACAS X is operating in a part time mode, it is permissible for this annunciation to be displayed. On a full time display, the Proximate Traffic will remain displayed at the proper range and bearing relative to the ownship symbol. If the display was changed from the MAP mode, the MAP mode is reselected.

- d. When the RA is issued ($t=90$), a red filled square **shall** (2524) be shown at the proper range and bearing. The Proximate Traffic **shall** (2525) also be shown at the proper range and bearing relative to the ownship symbol. The selected display range of the MAP display does not change. The traffic information is shown in a heading up orientation on the MAP or a display in a heading up orientation is available for selection via the display mode control panel.
- e. When the RA is removed after 30 seconds ($t=120$), both the RA and Proximate Traffic symbol are removed and no traffic information is displayed if the display is implemented on a part time basis. If the display implementation includes an annunciation indicating that ACAS X is operating in a part time mode, it is permissible for this annunciation to be displayed. On a full time display, the Proximate Traffic will remain displayed.

2.4.2.8.1.7.2.8 North-up Displays

2.4.2.8.1.7.2.8.1 Display Characteristics (§2.2.6.1.2.7.2.9.1, §2.2.6.1.2.7.2.9.3, §2.2.6.1.2.7.2.9.4, §2.2.6.1.2.7.2.9.5)

This test verifies that the display of ACAS X information is properly implemented in the North-up mode.

Conditions

This test requires that the ND or EHSI be powered up, in the North-up display mode, and configured to display ACAS X. The test requires the presence of intruders classified as Proximate Traffic, a TA, and an RA. The Proximate Traffic is displayed throughout the encounter. The TA is initially displayed 30 seconds after the start of the test ($t=30$) and is displayed for 30 seconds before being removed at $t=60$ seconds. While the TA is displayed, the display range is increased. After 30 seconds ($t=90$), an intruder causing the RA is displayed (without a preceding TA) for 30 seconds.

The test setup **shall** (2526) include means of modifying the display range of the North-up mode, as well as the display mode of the ND or EHSI. The mode selections may be made via the mode control panel for the ND or EHSI connected to the display, or using simulated inputs of this control panel.

Success

- a. At the beginning of the test scenario, no traffic information is displayed if the ACAS X information is shown on a part time mode. If the part time display implementation includes an annunciation indicating that ACAS X is operating in a part time mode, it is permissible for this annunciation to be displayed. If the ACAS X information is shown on a full time basis, the Proximate Traffic is shown at the proper range and bearing relative to the ownship symbol. If the North-up presentation does not provide an ownship symbol, the Proximate Traffic is not displayed.
- b. When the TA is issued at $t=30$ seconds, an amber or yellow filled circle **shall** (2527) be shown at the proper range and bearing, referenced to the heading depicted by the ownship symbol. In addition, a written message of “TRAFFIC”, “TFC”, or “TCAS” **shall** (2528) be annunciated in amber or yellow. The Proximate Traffic **shall** (2529) also be shown at the proper range and bearing. If the North-up presentation does not provide an ownship symbol, no traffic symbology is shown.
- c. When the display range is increased, the TA and Proximate Traffic remain displayed at the correct range and bearing, relative to the heading depicted by the ownship symbol, and the display range increases.

- d. When the TA is removed after 30 seconds ($t=60$), both the TA and Proximate Traffic symbol are removed and no traffic information is displayed when a part time display is used. If the display implementation includes an annunciation indicating that ACAS X is operating in a part time mode, it is permissible for this annunciation to be displayed. On a full time display, the Proximate Traffic will remain displayed at the proper range and bearing. If the North-up presentation does not provide an ownship symbol, the Proximate Traffic is not displayed.
- e. When the RA is issued ($t=90$), a red filled square **shall** (2530) be shown at the proper range and bearing, referenced to the heading depicted by the ownship symbol. In addition, a written message of “TRAFFIC”, “TFC”, or “TCAS” **shall** (2531) be annunciated in red. The Proximate Traffic **shall** (2532) also be shown at the proper range and bearing relative to the ownship symbol. If the North-up presentation does not provide an ownship symbol, no traffic symbology is shown.
- f. When the RA is removed after 30 seconds ($t=120$), both the RA and Proximate Traffic symbol are removed and no traffic information is displayed if the display is implemented on a part time basis. If the display implementation includes an annunciation indicating that ACAS X is operating in a part time mode, it is permissible for this annunciation to be displayed. On a full time display, the Proximate Traffic will remain displayed. If the North-up presentation does not provide an ownship symbol, the Proximate Traffic is not displayed.

2.4.2.8.1.7.2.8.2 Display Orientation (§2.2.6.1.2.7.2.9.6)

This test verifies that a heading up orientation is available with a single action when a TA or an RA is issued.

Conditions

This test requires that the ND or EHSI be powered up, in the North-up display mode, and configured to display ACAS X. The test requires the presence of intruders classified as Proximate Traffic, a TA, and an RA. The Proximate Traffic is displayed throughout the encounter. The TA is initially displayed 30 seconds after the start of the test ($t=30$) and is displayed for 30 seconds before being removed. While the TA is displayed, the ND or EHSI is reconfigured to show the ACAS X information in a heading up orientation. Prior to the RA being issued, the display is returned to the North-up display mode. After 30 seconds ($t=90$), an intruder causing the RA is displayed (without a preceding TA) for 30 seconds. While the RA is displayed, the ND or EHSI is reconfigured to show the ACAS X information in a heading up orientation.

The test setup **shall** (2533) provide a means of modifying the display mode of the ND or EHSI. The mode selections may be made via the mode control panel for the ND or EHSI connected to the display, or using simulated inputs of this control panel.

Success

- a. At the beginning of the test scenario, no traffic information is displayed if the ACAS X information is shown on a part time mode. If the part time display implementation includes an annunciation indicating that ACAS X is operating in a part time mode, it is permissible for this annunciation to be displayed. If the ACAS X information is shown on a full time basis, the Proximate Traffic is shown at the proper range and bearing.
- b. When the TA is issued at $t=30$ seconds, an amber or yellow filled circle **shall** (2534) be shown at the proper range and bearing, referenced to the heading depicted by the ownship symbol. In addition, a written message of “TRAFFIC”, “TFC”, or “TCAS”

shall (2535) be annunciated in amber or yellow. The Proximate Traffic **shall** (2536) also be shown at the proper range and bearing. If the North-up presentation does not provide an ownship symbol, no traffic symbology is shown.

- c. A heading up orientation is available with a single action.
- d. When the TA is removed after 30 seconds ($t=60$), both the TA and Proximate Traffic symbol are removed and no traffic information is displayed when a part time display is used. If the display implementation includes an annunciation indicating that ACAS X is operating in a part time mode, it is permissible for this annunciation to be displayed. On a full time display, the Proximate Traffic will remain displayed at the proper range and bearing. If the North-up presentation does not provide an ownship symbol, the Proximate Traffic is not displayed.
- e. When the RA is issued at $t=90$ seconds, a red filled square **shall** (2537) be shown at the proper range and bearing, referenced to the heading depicted by the ownship symbol. In addition, a written message of “TRAFFIC”, “TFC”, or “TCAS” **shall** (2538) be annunciated in red. The Proximate Traffic **shall** (2539) also be shown at the proper range and bearing. If the North-up presentation does not provide an ownship symbol, no traffic symbology is shown.
- f. A heading up orientation is available with a single action.

2.4.2.8.1.7.3 EICAS or Systems Displays (§2.2.6.1.2.7.3.3)

This test verifies that when ACAS X information is shown on an EICAS or SYSTEMS display, a capability exists to inhibit the display of the ACAS X information.

Conditions

This test requires that the EICAS/SYSTEMS display be powered up and configured to display ACAS X. The test requires the presence of intruders classified as Proximate Traffic, a TA, and an RA. The Proximate Traffic is displayed throughout the encounter. Prior to the TA being issued, the display is reconfigured so that no traffic information is shown. Once the display of traffic is inhibited, the display of traffic is again enabled. The TA is initially displayed 30 seconds after the start of the test ($t=30$) and is displayed for 30 seconds before being removed at $t=60$ seconds. While the TA is displayed, the display is reconfigured so that no traffic information is shown. Once the display of traffic is inhibited, the display of traffic is again enabled. After 30 seconds ($t=90$), an intruder causing the RA is displayed (without a preceding TA) for 30 seconds. While the RA is displayed, the display is reconfigured so that no traffic information is shown. Once the display of traffic is inhibited, the display of traffic is again enabled.

The test setup **shall** (2540) provide a means of modifying the display mode. The mode selections may be made via the mode control panel for the EICAS/SYSTEMS display connected to the display, or using simulated inputs of this control panel.

Success

- a. At the beginning of the test scenario, no traffic information is displayed if the ACAS X information is shown on a part time mode. If the part time display implementation includes an annunciation indicating that ACAS X is operating in a part time mode, it is permissible for this annunciation to be displayed. If the ACAS X information is shown on a full time basis, the Proximate Traffic is shown at the proper range and bearing.
- b. When the display mode is changed to inhibit the display of traffic information, no ACAS X information is displayed. Once the display of traffic is re-enabled, the

Proximate Traffic is displayed at the proper range and bearing on a full time display implementation.

- c. When the TA is issued at t=30 seconds, an amber or yellow filled circle **shall** (2541) be shown at the proper range and bearing, relative to the ownship symbol. The Proximate Traffic **shall** (2542) also be shown at the proper range and bearing. When the display mode is changed to inhibit the display of traffic information, no ACAS X information is displayed. Once the display of traffic is re-enabled, the TA and Proximate Traffic are displayed at the proper range and bearing.
- d. When the TA is removed after 30 seconds (t=60), both the TA and Proximate Traffic symbol are removed and no traffic information is displayed when a part time display is used. If the display implementation includes an annunciation indicating that ACAS X is operating in a part time mode, it is permissible for this annunciation to be displayed. On a full time display, the Proximate Traffic will remain displayed at the proper range and bearing relative to the ownship symbol.
- e. When the RA is issued at t=90 seconds, a red filled square **shall** (2543) be shown at the proper range and bearing, relative to the ownship symbol. The Proximate Traffic **shall** (2544) also be shown at the proper range and bearing. When the display mode is changed to inhibit the display of traffic information, no ACAS X information is displayed. Once the display of traffic is re-enabled, the RA and Proximate Traffic are displayed at the proper range and bearing.

2.4.2.8.1.7.4 TA/RA/VSI

These tests verify the proper implementation of the vertical speed indicator and the ACAS X traffic display on a single instrument.

2.4.2.8.1.7.4.1 VSI Requirements (§2.2.6.1.2.7.4.1.2)

This test verifies that the VSI needle supports the requirements for use as a vertical speed indicator.

Note: This test is not intended to be a complete qualification of the VSI function of this type of display. Additional testing may be required to demonstrate that the VSI meets all the applicable standards for such an instrument.

Conditions

This test requires that the TA/RA/VSI be powered up and that the display of traffic be enabled. At least one intruder must be displayed in a location which falls under the stem of the VSI needle. The test requires that instrumentation be provided which permits various vertical speeds to be displayed on the TA/RA/VSI. This instrumentation **shall** (2545) also permit the vertical speed to be varied using accelerations that can be expected during normal aircraft operations.

Success

Note: The assessment of the vertical speed function is a subjective assessment and should be performed by Flight Test personnel from the Aircraft Certification Office.

- a. Using peripheral vision only, the actual vertical speed depicted and the vertical speed trend is accurately interpreted.
- b. The stem of the VSI needle overlaps the displayed traffic, but the information on the displayed traffic (range, bearing, threat type, and relative altitude information) remains readable.

2.4.2.8.1.7.4.2 Interference with Vertical Speed Scale (§2.2.6.1.2.7.4.2.3)

This test verifies that none of the displayed traffic information overwrites any portion of the vertical speed scale.

Conditions

This test requires that the TA/RA/VSI be powered up and that the display of traffic be enabled. Six intruders must be evenly spaced around the display at the maximum range of the display at each location. Each intruder must be altitude reporting and have a vertical trend arrow.

Success

None of the traffic information overwrites or infringes on any portion of the vertical speed scale. This includes the traffic symbol, altitude data, and vertical trend arrow.

2.4.2.8.2 RA Displays

These tests verify the proper implementation of ACAS X RA display characteristics.

2.4.2.8.2.1 Round Dial VSI (§2.2.6.2.2, §2.2.6.2.2.1.1, §2.2.6.2.2.1.2, §2.2.6.2.2.1.3, §2.2.6.2.2.1.4)

The tests in this subparagraph verify the proper depiction of corrective and preventive RAs when the RAs are shown on a round dial VSI.

Conditions

These tests require that the round dial VSI be powered up. The test will verify the proper depiction of the RAs shown in Table 2-86.

Success

The success criteria for these tests are shown in Table 2-86. During the display of all RAs, both the red and green arcs depicted on the display **shall** (2546) be readily discernible and distinguishable from each other and from the portions of the VSI with no illuminated arcs.

Notes:

1. *The assessment of the distinction between the various illuminated arcs is a subjective assessment and should be performed by Flight Test personnel from the Aircraft Certification Office.*
2. *The length of the green arc in terms of vertical speed shown in Table 2-86's success criteria is approximate and should not be considered absolute. The length of the green arc should be the nominal length, as defined in §2.2.6.2.2.1.2 for all RAs except the multi-aircraft encounter (Maintain Existing Vertical Speed).*

Table 2-86: Round Dial VSI RA Tests

Advisory	RA Success Criteria
Climb	The VSI has a red arc from -6,000 fpm around to +1,500 fpm and a green arc between +1,500 and +2,000 fpm. The green arc is either wider than the red arc, or is offset from the red arc. The remaining scale of the VSI (from +2,000 to +6,000 fpm) has no illuminated arcs and only the background color of the VSI is displayed.
Descend	The VSI has a red arc from +6,000 fpm around to -1,500 fpm and a green arc between -1,500 and -2,000 fpm. The green arc is either wider than the red arc, or is offset from the red arc. The remaining scale of the VSI (from -2,000 to -6,000 fpm) has no illuminated arcs and only the background color of the VSI is displayed.
Altitude Crossing Climb	The VSI has a red arc from -6,000 fpm around to +1,500 fpm and a green arc between +1,500 and +2,000 fpm. The green arc is either wider than the red arc, or is offset from the red arc. The remaining scale of the VSI (from +2,000 to +6,000 fpm) has no illuminated arcs and only the background color of the VSI is displayed.
Altitude Crossing Descend	The VSI has a red arc from +6,000 fpm around to -1,500 fpm and a green arc between -1,500 and -2,000 fpm. The green arc is either wider than the red arc, or is offset from the red arc. The remaining scale of the VSI (from -2,000 to -6,000 fpm) has no illuminated arcs and only the background color of the VSI is displayed.
Reduce Climb (Do Not Climb)	The VSI has a red arc from +6,000 fpm around to 0 fpm and a green arc between 0 and approximately -250 fpm. The green arc is either wider than the red arc, or is offset from the red arc. The remaining scale of the VSI (from -250 to -6,000 fpm) has no illuminated arcs and only the background color of the VSI is displayed.
Reduce Descent (Do Not Descend)	The VSI has a red arc from -6,000 fpm around to 0 fpm and a green arc between 0 and approximately +250 fpm. The green arc is either wider than the red arc, or is offset from the red arc. The remaining scale of the VSI (from +250 to +6,000 fpm) has no illuminated arcs and only the background color of the VSI is displayed.
RA Reversal (Descend to Climb)	The VSI initially has a red arc from +6,000 fpm around to -1,500 fpm and a green arc between -1,500 and -2,000 fpm. The green arc is either wider than the red arc, or is offset from the red arc. The remaining scale of the VSI (from -2,000 to -6,000 fpm) has no illuminated arcs and only the background color of the VSI is displayed. When the RA reverses to a Climb RA, the VSI has a red arc from -6,000 fpm around to +1,500 fpm and a green arc between +1,500 and +2,000 fpm. The green arc is either wider than the red arc, or is offset from the red arc. The remaining scale of the VSI (from +2,000 to +6,000 fpm) has no illuminated arcs and only the background color of the VSI is displayed.

Table 2-86 Round Dial VSI RA Tests (cont.)

Advisory	RA Success Criteria
RA Reversal (Climb to Descend)	The VSI initially has a red arc from -6,000 fpm around to +1,500 fpm and a green arc between +1,500 and +2,000 fpm. The green arc is either wider than the red arc, or is offset from the red arc. The remaining scale of the VSI (from +2,000 to +6,000 fpm) has no illuminated arcs and only the background color of the VSI is displayed. When the RA reverses to a Descend RA, the VSI has a red arc from +6,000 fpm around to -1,500 fpm and a green arc between -1,500 and -2,000 fpm. The green arc is either wider than the red arc, or is offset from the red arc. The remaining scale of the VSI (from -2,000 to -6,000 fpm) has no illuminated arcs and only the background color of the VSI is displayed.
Increase Climb	The VSI has a red arc from -6,000 fpm around to +2,500 fpm and a green arc between +2,500 fpm and approximately +3,000 fpm. The remaining scale of the VSI (from +3,000 to +6,000 fpm) has no illuminated arcs and only the background color of the VSI is displayed.
Increase Descent	The VSI has a red arc from +6,000 fpm around to -2,500 fpm and a green arc between -2,500 fpm and approximately -3,000 fpm. The remaining scale of the VSI (from -3,000 to -6,000 fpm) has no illuminated arcs and only the background color of the VSI is displayed.
Maintain Rate RA (Maintain Climb Rate of 4400 fpm)	The VSI has a red arc from -6,000 fpm around to +4,400 fpm and a green arc between +4,400 and approximately +6,000 fpm. The green arc is either wider than the red arc, or is offset from the red arc. Any remaining scale of the VSI beyond the upper end of the green arc has no illuminated arcs and only the background color of the VSI is displayed.
Maintain Rate RA (Maintain Descent Rate of 4400 fpm)	The VSI has a red arc from +6,000 fpm around to -4,400 fpm and a green arc between -4,400 and approximately -6,000 fpm. The green arc is either wider than the red arc, or is offset from the red arc. Any remaining scale of the VSI beyond the upper end of the green arc has no illuminated arcs and only the background color of the VSI is displayed.
Altitude Crossing Maintain Rate (Maintain Climb Rate of 3200 fpm)	The VSI has a red arc from -6,000 fpm around to +3,200 fpm and a green arc between +3,200 and +4,000 fpm. The green arc is either wider than the red arc, or is offset from the red arc. The remaining scale of the VSI (from +4,000 to +6,000 fpm) has no illuminated arcs and only the background color of the VSI is displayed.
Altitude Crossing Maintain Rate (Maintain Descent Rate of 2600)	The VSI has a red arc from +6,000 fpm around to -2,600 fpm and a green arc between -2,600 and -3,300 fpm. The green arc is either wider than the red arc, or is offset from the red arc. The remaining scale of the VSI (from -3,300 to -6,000 fpm) has no illuminated arcs and only the background color of the VSI is displayed. <i>Note:</i> When a mechanical VSI uses a series of LED's to provide the green arc, it is permissible for the green arc to extend to -3,500 fpm.

Table 2-86 Round Dial VSI RA Tests (cont.)

Advisory	RA Success Criteria
Weakening of Positive RAs (Climb RA Weakens to a Level-Off RA)	The VSI initially has a red arc from -6,000 fpm around to +1,500 fpm and a green arc between +1,500 and +2,000 fpm. The green arc is either wider than the red arc, or is offset from the red arc. The remaining scale of the VSI (from +2,000 to +6,000 fpm) has no illuminated arcs and only the background color of the VSI is displayed. When the RA weakens, the VSI has a red arc from -6,000 fpm around to 0 fpm and a green arc between 0 and +250 fpm. The green arc is either wider than the red arc, or is offset from the red arc. The remaining scale of the VSI (from +250 to +6,000 fpm) has no illuminated arcs and only the background color of the VSI is displayed.
Weakening of Positive RAs (Descend RA Weakens to a Level-Off 0)	The VSI initially has a red arc from +6,000 fpm around to -1,500 fpm and a green arc between -1,500 and -2,000 fpm. The green arc is either wider than the red arc, or is offset from the red arc. The remaining scale of the VSI (from -2,000 to -6,000 fpm) has no illuminated arcs and only the background color of the VSI is displayed. When the RA weakens, the VSI has a red arc from +6,000 fpm around to 0 fpm and a green arc between 0 and -250 fpm. The green arc is either wider than the red arc, or is offset from the red arc. The remaining scale of the VSI (from -250 to -6,000 fpm) has no illuminated arcs and only the background color of the VSI is displayed.
Preventive Limit Climb (Do Not Climb)	The VSI has a red arc from +6,000 fpm around to 0 fpm and no green arc. The remaining scale of the VSI (from 0 to -6,000 fpm) has no illuminated arcs and only the background color of the VSI is displayed.
Preventive Limit Descent (Do Not Descend)	The VSI has a red arc from -6,000 fpm around to 0 fpm and no green arc. The remaining scale of the VSI (from 0 to +6,000 fpm) has no illuminated arcs and only the background color of the VSI is displayed.
Multi-Aircraft Encounter (Maintain Existing V/S)	The VSI has a red arc from -6,000 fpm around to -250 fpm, a second red arc from +6,000 fpm around to +250 fpm, and a green arc between -250 and +250 fpm. The green arc is either wider than the red arc, or is offset from the red arc.
Multi-Aircraft Encounter (Second Intruder Becomes a Threat While Responding to an Initial Climb RA)	The VSI initially has a red arc from -6,000 fpm around to +1,500 fpm and a green arc between +1,500 and +2,000 fpm. The green arc is either wider than the red arc, or is offset from the red arc. The remaining scale of the VSI (from +2,000 to +6,000 fpm) has no illuminated arcs and only the background color of the VSI is displayed. When the second intruder causes an RA, the VSI has a red arc from -6,000 fpm around to -250 fpm, a second red arc from +6,000 fpm around to +250 fpm, and a green arc between -250 and +250 fpm. The green arc is either wider than the red arc, or is offset from the red arc. It is also acceptable to display a nominal length green arc beginning at 0 fpm and extending around to approximately -250 fpm.
Multi-Aircraft Encounter (Preventive RA Limiting Climb and Descent Rate to ±1,000 fpm)	The VSI has a red arc from -6,000 fpm around to -1,000 fpm, a second red arc from +6,000 fpm around to +1,000 fpm. The remaining scale of the VSI (from -1,000 to +1,000 fpm) has no illuminated arcs and only the background color of the VSI is displayed.

2.4.2.8.2.1.1 VSI Range (§2.2.6.2.2)

This test verifies the RA display has sufficient range on the vertical speed scale to display all RAs that can be issued by the CAS logic.

Conditions

This test requires that the RA display be powered up and configured such that RA can be displayed.

Success

The scale of the vertical speed indicator can display vertical speeds between ± 6000 fpm.

Note: If the VSI implementation does not provide a scale between $\pm 6,000$ fpm, alternative test procedures must be defined by the display manufacturer. These alternative tests must demonstrate that each of the RAs shown in Table 2-86 are displayed in a manner which permits a pilot to make the desired response to the displayed RA.

2.4.2.8.2.2 Vertical Speed Tape (§2.2.6.2.3, §2.2.6.2.3.1.1, §2.2.6.2.3.1.2, §2.2.6.2.3.1.3, §2.2.6.2.3.1.4)

The tests in this subparagraph verify the proper depiction of corrective and preventive RAs when the RAs are shown on a vertical speed tape within the PFD.

Conditions

These tests require that the PFD be powered up. The test will verify the proper depiction of the RAs shown in Table 2-87.

Success

The success criteria for these tests are shown in Table 2-87. During the display of all RAs, both the red and green zones depicted on the display **shall** (2547) be readily discernible and distinguishable from each other and from the portions of the vertical speed tape with no illuminated arcs.

Notes:

1. *The assessment of the distinction between the various illuminated zones is a subjective assessment and should be performed by Flight Test personnel from the Aircraft Certification Office.*
2. *If the vertical speed tape implementation does not provide a scale between $\pm 6,000$ fpm, alternative test procedures must be defined by the display manufacturer. These alternative tests must demonstrate that each of the RAs shown in Table 2-87 are displayed in a manner which permits a pilot to make the desired response to the displayed RA.*
3. *The length of the green zone in terms of vertical speed shown in Table 2-87's success criteria is approximate and should not be considered absolute. The length of the green zone should be the nominal length as defined in §2.2.6.2.3.1.2 for all RAs except the multi-aircraft encounter (Maintain Existing Vertical Speed)*

Table 2-87: Vertical Speed Tape RA Tests

Advisory	RA Success Criteria
Climb	The vertical speed tape has a red zone from -6,000 fpm to +1,500 fpm and a green zone between +1,500 and +2,000 fpm. The green zone is either wider than the red zone, or is offset from the red zone. The remaining scale of the vertical speed tape (from +2,000 to +6,000 fpm) has no illuminated zones and only the background color of the vertical speed tape is displayed.
Descend	The vertical speed tape has a red zone from +6,000 fpm to -1,500 fpm and a green zone between -1,500 and -2,000 fpm. The green zone is either wider than the red zone, or is offset from the red zone. The remaining scale of the vertical speed tape (from -2,000 to -6,000 fpm) has no illuminated zones and only the background color of the vertical speed tape is displayed.
Altitude Crossing Climb	The vertical speed tape has a red zone from -6,000 fpm to +1,500 fpm and a green zone between +1,500 and +2,000 fpm. The green zone is either wider than the red zone, or is offset from the red zone. The remaining scale of the vertical speed tape (from +2,000 to +6,000 fpm) has no illuminated zones and only the background color of the vertical speed tape is displayed.
Altitude Crossing Descend	The vertical speed tape has a red zone from +6,000 fpm to -1,500 fpm and a green zone between -1,500 and -2,000 fpm. The green zone is either wider than the red zone, or is offset from the red zone. The remaining scale of the vertical speed tape (from -2,000 to -6,000 fpm) has no illuminated zones and only the background color of the vertical speed tape is displayed.
Reduce Climb (Do Not Climb)	The vertical speed tape has a red zone from +6,000 fpm to 0 fpm and a green zone between 0 and approximately -250 fpm. The green zone is either wider than the red zone, or is offset from the red zone. The remaining scale of the vertical speed tape (from -250 to -6,000 fpm) has no illuminated zones and only the background color of the vertical speed tape is displayed.
Reduce Descent (Do Not Descend)	The vertical speed tape has a red zone from -6,000 fpm to 0 fpm and a green zone between 0 and approximately +250 fpm. The green zone is either wider than the red zone, or is offset from the red zone. The remaining scale of the vertical speed tape (from +250 to +6,000 fpm) has no illuminated zones and only the background color of the vertical speed tape is displayed.

Advisory	RA Success Criteria
RA Reversal (Descend to Climb)	The vertical speed tape initially has a red zone from +6,000 fpm to -1,500 fpm and a green zone between -1,500 and -2,000 fpm. The green zone is either wider than the red zone, or is offset from the red zone. The remaining scale of the vertical speed tape (from -2,000 to -6,000 fpm) has no illuminated zones and only the background color of the vertical speed tape is displayed. When the RA reverses to a Climb RA, the vertical speed tape has a red zone from -6,000 fpm to +1,500 fpm and a green zone between +1,500 and +2,000 fpm. The green zone is either wider than the red zone, or is offset from the red zone. The remaining scale of the vertical speed tape (from +2,000 to +6,000 fpm) has no illuminated zones and only the background color of the vertical speed tape is displayed.
RA Reversal (Climb to Descend)	The vertical speed tape initially has a red zone from -6,000 fpm to +1,500 fpm and a green zone between +1,500 and +2,000 fpm. The green zone is either wider than the red zone, or is offset from the red zone. The remaining scale of the vertical speed tape (from +2,000 to +6,000 fpm) has no illuminated zones and only the background color of the vertical speed tape is displayed. When the RA reverses to a Descend RA, the vertical speed tape has a red zone from +6,000 fpm to -1,500 fpm and a green zone between -1,500 and -2,000 fpm. The green zone is either wider than the red zone, or is offset from the red zone. The remaining scale of the vertical speed tape (from -2,000 to -6,000 fpm) has no illuminated zones and only the background color of the vertical speed tape is displayed.
Increase Climb	The vertical speed tape has a red zone from -6,000 fpm to +2,500 fpm and a green zone between +2,500 fpm and approximately +3,000 fpm. The remaining scale of the vertical speed tape (from +3,000 to +6,000 fpm) has no illuminated zones and only the background color of the vertical speed tape is displayed.
Increase Descent	The vertical speed tape has a red zone from +6,000 fpm to -2,500 fpm and a green zone between -2,500 fpm and approximately -3,000 fpm. The remaining scale of the vertical speed tape (from -3,000 to -6,000 fpm) has no illuminated zones and only the background color of the vertical speed tape is displayed.
Maintain Rate RA (Maintain Climb Rate of 4400 fpm)	The vertical speed tape has a red zone from -6,000 fpm to +4,400 fpm and a green zone between +4,400 and approximately +6,000 fpm. The green zone is either wider than the red zone, or is offset from the red zone. Any remaining scale of the vertical speed tape beyond the upper end of the green zone has no illuminated zones and only the background color of the vertical speed tape is displayed.
Maintain Rate RA (Maintain Descent Rate of 4400 fpm)	The vertical speed tape has a red zone from +6,000 fpm to -4,400 fpm and a green zone between -4,400 and approximately -6,000 fpm. The green zone is either wider than the red zone, or is offset from the red zone. Any remaining scale of the vertical speed tape beyond the upper end of the green zone has no illuminated zones and only the background color of the vertical speed tape is displayed.

Advisory	RA Success Criteria
Altitude Crossing Maintain Rate (Maintain Climb Rate of 3200 fpm)	The vertical speed tape has a red zone from -6,000 fpm to +3,200 fpm and a green zone between +3,200 and +4,000 fpm. The green zone is either wider than the red zone, or is offset from the red zone. The remaining scale of the vertical speed tape (from +4,000 to +6,000 fpm) has no illuminated zones and only the background color of the vertical speed tape is displayed.
Altitude Crossing Maintain Rate (Maintain Descent Rate of 2600)	The vertical speed tape has a red zone from +6,000 fpm to -2,600 fpm and a green zone between -2,600 and -3,300 fpm. The green zone is either wider than the red zone, or is offset from the red zone. The remaining scale of the vertical speed tape (from -3,300 to -6,000 fpm) has no illuminated zones and only the background color of the vertical speed tape is displayed.
Weakening of Positive RAs (Climb RA Weakens to a Level-Off RA)	The vertical speed tape initially has a red zone from -6,000 fpm to +1,500 fpm and a green zone between +1,500 and +2,000 fpm. The green zone is either wider than the red zone, or is offset from the red zone. The remaining scale of the vertical speed tape (from +2,000 to +6,000 fpm) has no illuminated zones and only the background color of the vertical speed tape is displayed. When the RA weakens, the vertical speed tape has a red zone from -6,000 fpm to 0 fpm and a green zone between 0 and +250 fpm. The green zone is either wider than the red zone, or is offset from the red zone. The remaining scale of the vertical speed tape (from +250 to +6,000 fpm) has no illuminated zones and only the background color of the vertical speed tape is displayed.
Weakening of Positive RAs (Descend RA Weakens to a Level-Off RA)	The vertical speed tape initially has a red zone from +6,000 fpm to -1,500 fpm and a green zone between -1,500 and -2,000 fpm. The green zone is either wider than the red zone, or is offset from the red zone. The remaining scale of the vertical speed tape (from -2,000 to -6,000 fpm) has no illuminated zones and only the background color of the vertical speed tape is displayed. When the RA weakens, the vertical speed tape has a red zone from +6,000 fpm to 0 fpm and a green zone between 0 and -250 fpm. The green zone is either wider than the red zone, or is offset from the red zone. The remaining scale of the vertical speed tape (from -250 to -6,000 fpm) has no illuminated zones and only the background color of the vertical speed tape is displayed.
Preventive Limit Climb (Do Not Climb)	The vertical speed tape has a red zone from +6,000 fpm to 0 fpm and no green zone. The remaining scale of the vertical speed tape (from 0 to -6,000 fpm) has no illuminated zones and only the background color of the vertical speed tape is displayed.
Preventive Limit Descent (Do Not Descend)	The vertical speed tape has a red zone from -6,000 fpm to 0 fpm and no green zone. The remaining scale of the vertical speed tape (from 0 to +6,000 fpm) has no illuminated zones and only the background color of the vertical speed tape is displayed.

Advisory	RA Success Criteria
Multi-Aircraft Encounter (Maintain Existing V/S)	The vertical speed tape has a red zone from -6,000 fpm to -250 fpm, a second red zone from +6,000 fpm to +250 fpm, and a green zone between -250 and +250 fpm. The green zone is either wider than the red zone, or is offset from the red zone.
Multi-Aircraft Encounter (Second Intruder Becomes a Threat While Responding to an Initial Climb RA)	The vertical speed tape initially has a red zone from -6,000 fpm to +1,500 fpm and a green zone between +1,500 and +2,000 fpm. The green zone is either wider than the red zone, or is offset from the red zone. The remaining scale of the vertical speed tape (from +2,000 to +6,000 fpm) has no illuminated zones and only the background color of the vertical speed tape is displayed. When the second intruder causes an RA, the vertical speed tape has a red zone from -6,000 fpm to -250 fpm, a second red zone from +6,000 fpm to +250 fpm, and a green zone between -250 and +250 fpm. The green zone is either wider than the red zone, or is offset from the red zone. It is also acceptable to display a nominal length green zone beginning at 0 fpm and extending downwards to approximately -250 fpm.
Multi-Aircraft Encounter (Preventive RA Limiting Climb and Descent Rate to $\pm 1,000$ fpm)	The vertical speed tape has a red zone from -6,000 fpm to -1,000 fpm, a second red zone from +6,000 fpm to +1,000 fpm. The remaining scale of the vertical speed tape (from -1,000 to +1,000 fpm) has no illuminated zones and only the background color of the vertical speed tape is displayed.

2.4.2.8.2.2.1 Vertical Speed Tape Range (§2.2.6.2.3)

This test verifies the vertical speed tape has sufficient range on the vertical speed scale to display all RAs that can be issued by the CAS logic.

Conditions

This test requires that the PFD be powered up and configured such that RA can be displayed.

Success

The scale of the vertical speed indicator can display vertical speeds between ± 6000 fpm.

Note: If the vertical speed tape implementation does not provide a scale between $\pm 6,000$ fpm, alternative test procedures must be defined by the display manufacturer. These alternative tests must demonstrate that each of the RAs shown in Table 2-87 are displayed in a manner which permits a pilot to make the desired response to the displayed RA.

2.4.2.8.2.3 Pitch Cues on the PFD (§2.2.6.2.4.1, §2.2.6.2.4.2, §2.2.6.2.4.3)

The tests in this subparagraph verify the proper depiction of corrective and preventive RAs when the RA guidance is shown using pitch cues displayed on the PFD.

Conditions

These tests require that the PFD be powered up. The test will verify the proper depiction of the RAs shown in Table 2-88.

Success

The success criteria for these tests are shown in Table 2-88. During the display of all RAs, the ACAS X trapezoid, or other geometric shape, depicting the pitch angle to maintain or attain **shall** (2548) be readily discernible and distinguishable from all other information shown on the PFD.

Note: This assessment is a subjective one and should be performed by Flight Test personnel from the Aircraft Certification Office.

Table 2-88: RA Tests for Implementations Using Pitch Cues on a PFD

Advisory	RA Success Criteria
Climb	A red trapezoid is displayed, beginning at the bottom of the PFD and extending upwards to the pitch angle required to obtain a 1,500 fpm rate of climb. The trapezoid does not occlude any other information shown on the PFD. If the implementation includes visual alerts on the PFD, “CLIMB” is written in red on the PFD. This visual alert flashes or is otherwise highlighted.
Descend	A red trapezoid is displayed, beginning at the top of the PFD and extending downwards to the pitch angle required to obtain a 1,500 fpm rate of descent. The trapezoid does not occlude any other information shown on the PFD. If the implementation includes visual alerts on the PFD, “DESCEND” is written in red on the PFD. This visual alert either flashes or is otherwise highlighted.
Altitude Crossing Climb	A red trapezoid is displayed, beginning at the bottom of the PFD and extending upwards to the pitch angle required to obtain a 1,500 fpm rate of climb. The trapezoid does not occlude any other information shown on the PFD. If the implementation includes visual alerts on the PFD, “CROSSING CLIMB” is written in red on the PFD. This visual alert either flashes or is otherwise highlighted.
Altitude Crossing Descend	A red trapezoid is displayed, beginning at the top of the PFD and extending downwards to the pitch angle required to obtain a 1,500 fpm rate of descent. The trapezoid does not occlude any other information shown on the PFD. If the implementation includes visual alerts on the PFD, “CROSSING DESCEND” is written in red on the PFD. This visual alert either flashes or is otherwise highlighted.
Reduce Climb (Do Not Climb)	A red trapezoid is displayed, beginning at the top of the PFD and extending downwards to the pitch angle required to obtain level flight. The trapezoid does not occlude any other information shown on the PFD. If the implementation includes visual alerts on the PFD, “LEVEL-OFF” is written in red on the PFD. This visual alert either flashes or is otherwise highlighted.
Reduce Descent (Do Not Descend)	A red trapezoid is displayed, beginning at the bottom of the PFD and extending upwards to the pitch angle required to obtain level flight. The trapezoid does not occlude any other information shown on the PFD. If the implementation includes visual alerts on the PFD, “LEVEL-OFF” is written in red on the PFD. This visual alert either flashes or is otherwise highlighted.

Advisory	RA Success Criteria
RA Reversal (Descend to Climb)	A red trapezoid is initially displayed, beginning at the top of the PFD and extending downwards to the pitch angle required to obtain a 1,500 fpm rate of descent. The trapezoid does not occlude any other information shown on the PFD. If the implementation includes visual alerts on the PFD, “DESCEND” is written in red on the PFD. This visual alert either flashes or is otherwise highlighted. When the RA reverses to a Climb RA, the red trapezoid begins at the bottom of the PFD and extends upwards to the pitch angle required to obtain a 1,500 fpm rate of climb. If the implementation includes visual alerts on the PFD, “CLIMB NOW” is written in red on the PFD after the RA reverses. This visual alert either flashes or is otherwise highlighted.
RA Reversal (Climb to Descend)	A red trapezoid is initially displayed, beginning at the bottom of the PFD and extending upwards to the pitch angle required to obtain a 1,500 fpm rate of climb. The trapezoid does not occlude any other information shown on the PFD. If the implementation includes visual alerts on the PFD, “CLIMB” is written in red on the PFD. This visual alert either flashes or is otherwise highlighted. When the RA reverses to a Descend RA, the red trapezoid begins at the top of the PFD and extends downwards to the pitch angle required to obtain a 1,500 fpm rate of descent. If the implementation includes visual alerts on the PFD, “DESCEND NOW” is written in red on the PFD after the RA reverses. This visual alert either flashes or is otherwise highlighted.
Increase Climb	A red trapezoid is displayed, beginning at the bottom of the PFD and extending upwards to the pitch angle required to obtain a 2,500 fpm rate of descent. The trapezoid does not occlude any other information shown on the PFD. If the implementation includes visual alerts on the PFD, “INCREASE CLIMB” is written in red on the PFD. This visual alert either flashes or is otherwise highlighted.
Increase Descent	A red trapezoid is displayed, beginning at the top of the PFD and extending downwards to the pitch angle required to obtain a 2,500 fpm rate of descent. The trapezoid does not occlude any other information shown on the PFD. If the implementation includes visual alerts on the PFD, “INCREASE DESCENT” is written in red on the PFD. This visual alert either flashes or is otherwise highlighted.
Maintain Rate RA (Maintain Climb Rate of 4000 fpm)	A red trapezoid is displayed, beginning at the bottom of the PFD and extending upwards to the pitch angle required to obtain a 4,000 fpm rate of climb. The trapezoid does not occlude any other information shown on the PFD. If the implementation includes visual alerts on the PFD, “MAINTAIN V/S” is written in red on the PFD. This visual alert either flashes or is otherwise highlighted.
Maintain Rate RA (Maintain Descent Rate of 4000 fpm)	A red trapezoid is displayed, beginning at the top of the PFD and extending downwards to the pitch angle required to obtain a 4,000 fpm rate of descent. The trapezoid does not occlude any other information shown on the PFD. If the implementation includes visual alerts on the PFD, “MAINTAIN V/S” is written in red on the PFD. This visual alert either flashes or is otherwise highlighted.

Advisory	RA Success Criteria
Altitude Crossing Maintain Rate (Maintain Climb Rate of 3200 fpm)	A red trapezoid is displayed, beginning at the bottom of the PFD and extending upwards to the pitch angle required to obtain a 3,200 fpm rate of descent. The trapezoid does not occlude any other information shown on the PFD. If the implementation includes visual alerts on the PFD, “MAINTAIN V/S CROSSING” is written in red on the PFD. This visual alert either flashes or is otherwise highlighted.
Altitude Crossing Maintain Rate (Maintain Descent Rate of 2600)	A red trapezoid is displayed, beginning at the top of the PFD and extending downwards to the pitch angle required to obtain a 2,600 fpm rate of descent. The trapezoid does not occlude any other information shown on the PFD. If the implementation includes visual alerts on the PFD, “MAINTAIN V/S CROSSING” is written in red on the PFD. This visual alert either flashes or is otherwise highlighted.
Weakening of Positive RAs (Climb RA Weakens to a Level-Off RA)	A red trapezoid is initially displayed, beginning at the bottom of the PFD and extending upwards to the pitch angle required to obtain a 1,500 fpm rate of climb. The trapezoid does not occlude any other information shown on the PFD. If the implementation includes visual alerts on the PFD, “CLIMB” is written in red on the PFD. This visual alert flashes or is otherwise highlighted. When the RA weakens, the red trapezoid is displayed, beginning at the bottom of the PFD and extending upwards to the pitch angle required to obtain level flight. The trapezoid does not occlude any other information shown on the PFD. If the implementation includes visual alerts on the PFD, “LEVEL-OFF” is written in red on the PFD. This visual alert either flashes or is otherwise highlighted.
Weakening of Positive RAs (Descend RA Weakens to a Level-Off RA)	A red trapezoid is initially displayed, beginning at the top of the PFD and extending downwards to the pitch angle required to obtain a 1,500 fpm rate of descent. The trapezoid does not occlude any other information shown on the PFD. If the implementation includes visual alerts on the PFD, “DESCEND” is written in red on the PFD. This visual alert flashes or is otherwise highlighted. When the RA weakens, the red trapezoid is displayed, beginning at the top of the PFD and extending downwards to the pitch angle required to obtain level flight. The trapezoid does not occlude any other information shown on the PFD. If the implementation includes visual alerts on the PFD, “LEVEL-OFF” is written in red on the PFD. This visual alert either flashes or is otherwise highlighted.
Preventive Limit Climb (Do Not Climb)	A red trapezoid is displayed, beginning at the top of the PFD and extending downwards to the pitch angle required to obtain level flight. The trapezoid does not occlude any other information shown on the PFD. If the implementation includes visual alerts on the PFD, “MONITOR V/S” is written in red on the PFD. This visual alert either flashes or is otherwise highlighted.
Preventive Limit Descent (Do Not Descend)	A red trapezoid is displayed, beginning at the bottom of the PFD and extending upwards to the pitch angle required to obtain level flight. The trapezoid does not occlude any other information shown on the PFD. If the implementation includes visual alerts on the PFD, “MONITOR V/S” is written in red on the PFD. This visual alert either flashes or is otherwise highlighted.

Advisory	RA Success Criteria
Multi-Aircraft Encounter (Maintain Existing V/S)	A red trapezoid is displayed beginning at the top of the PFD and extending downwards to the pitch angle representing a 250 fpm rate of climb. A second red trapezoid is also displayed beginning at the bottom of the PFD and extending upwards to the pitch angle representing a 250 fpm rate of descent. There is sufficient room between the closed ends of the two trapezoids for the ownship reference symbol of the PFD to be displayed. If the implementation includes visual alerts on the PFD, “MAINTAIN V/S” is written in red on the PFD. This visual alert either flashes or is otherwise highlighted.
Multi-Aircraft Encounter (Second Intruder Becomes a Threat While Responding to an Initial Climb RA)	A red trapezoid is displayed beginning at the bottom of the PFD and extending upwards to the pitch angle representing a 1,500 fpm rate of climb. If the implementation includes visual alerts on the PFD, “CLIMB” is written in red on the PFD. This visual alert either flashes or is otherwise highlighted. When the second intruder becomes a threat, the trapezoid for the initial RA begins at the bottom of the PFD and extends upwards to the pitch angle representing a 250 fpm rate of descent. A second red trapezoid is also displayed beginning at the top of the PFD and extending downwards to the pitch angle representing a 250 fpm rate of climb. There is sufficient room between the closed ends of the two trapezoids for the ownship reference symbol of the PFD to be displayed. If the implementation includes visual alerts on the PFD, “LEVEL-OFF” is written in red on the PFD. This visual alert either flashes or is otherwise highlighted.
Multi-Aircraft Encounter (Preventive RA Limiting Climb and Descent Rate to ±1,000 fpm)	A red trapezoid is displayed beginning at the top of the PFD and extending downwards to the pitch angle representing a 1,000 fpm rate of climb. A second red trapezoid is also displayed beginning at the bottom of the PFD and extending upwards to the pitch angle representing a 1,000 fpm rate of descent. There is sufficient room between the closed ends of the two trapezoids for the ownship reference symbol of the PFD to be displayed. If the implementation includes visual alerts on the PFD, “MONITOR V/S” is written in red on the PFD. This visual alert either flashes or is otherwise highlighted.

2.4.2.8.2.4 Flight Director Guidance (§3.2.13.2)

The tests in this subparagraph verify the proper depiction of corrective and preventive RAs when the RA guidance is shown using the flight director.

Conditions

These tests require that the ADI or PFD be powered up. The test will verify the proper depiction of the RAs shown in Table 2-89.

Success

The success criteria for these tests are shown in Table 2-89. The success criteria refer to both a single cue flight director and the vertical bar of a split cue flight director. The lateral guidance shown by the flight director is not constrained by these tests.

Table 2-89: RA Depiction Using Flight Director Guidance

Advisory	RA Success Criteria
Climb	When the RA is issued, a mode annunciation is provided and the vertical guidance mode is changed to display the pitch angle required to provide a 1,500 fpm rate of climb.
Descend	When the RA is issued, a mode annunciation is provided and the vertical guidance mode is changed to display the pitch angle required to provide a 1,500 fpm rate of descent.
Altitude Crossing Climb	When the RA is issued, a mode annunciation is provided and the vertical guidance mode is changed to display the pitch angle required to provide a 1,500 fpm rate of climb.
Altitude Crossing Descend	When the RA is issued, a mode annunciation is provided and the vertical guidance mode is changed to display the pitch angle required to provide a 1,500 fpm rate of descent.
Reduce Climb (Do Not Climb)	When the RA is issued, a mode annunciation is provided and the vertical guidance mode is changed to display the pitch angle required to provide level flight.
Reduce Descent (Do Not Descend)	When the RA is issued, a mode annunciation is provided and the vertical guidance mode is changed to display the pitch angle required to provide level flight.
RA Reversal (Descend to Climb)	When the RA is initially issued, a mode annunciation is provided and the vertical guidance mode is changed to display the pitch angle required to provide a 1,500 fpm rate of descent. When the RA reverses, the flight director calls for the pitch angle required to provide a 1,500 fpm rate of climb.
RA Reversal (Climb to Descend)	When the RA is initially issued, a mode annunciation is provided and the vertical guidance mode is changed to display the pitch angle required to provide a 1,500 fpm rate of climb. When the RA reverses, the flight director calls for the pitch angle required to provide a 1,500 fpm rate of descent.
Increase Climb	When the RA is initially issued, a mode annunciation is provided and the vertical guidance mode is changed to display the pitch angle required to provide a 1,500 fpm rate of climb. When the RA strengthens, the flight director calls for the pitch angle required to provide a 2,500 fpm rate of climb.
Increase Descent	When the RA is initially issued, a mode annunciation is provided and the vertical guidance mode is changed to display the pitch angle required to provide a 1,500 fpm rate of descent. When the RA strengthens, the flight director calls for the pitch angle required to provide a 2,500 fpm rate of descent.
Maintain Rate RA (Maintain Climb Rate of 4000 fpm)	When the RA is issued, a mode annunciation is provided and the vertical guidance mode is changed to display the pitch angle required to provide a 4,000 fpm rate of climb.

Advisory	RA Success Criteria
Maintain Rate RA (Maintain Descent Rate of 4000 fpm)	When the RA is issued, a mode annunciation is provided and the vertical guidance mode is changed to display the pitch angle required to provide a 4,000 fpm rate of descent.
Altitude Crossing Maintain Rate (Maintain Climb Rate of 3200 fpm)	When the RA is issued, a mode annunciation is provided and the vertical guidance mode is changed to display the pitch angle required to provide a 3,200 fpm rate of climb.
Altitude Crossing Maintain Rate (Maintain Descent Rate of 2600)	When the RA is issued, a mode annunciation is provided and the vertical guidance mode is changed to display the pitch angle required to provide a 2,600 fpm rate of descent.
Weakening of Positive RAs (Climb RA Weakens to a Level-Off RA)	When the RA is initially issued, a mode annunciation is provided and the vertical guidance mode is changed to display the pitch angle required to provide a 1,500 fpm rate of climb. When the RA weakens, the flight director calls for the pitch angle required to provide level flight.
Weakening of Positive RAs (Descend RA Weakens to a Level-Off RA)	When the RA is initially issued, a mode annunciation is provided and the vertical guidance mode is changed to display the pitch angle required to provide a 1,500 fpm rate of descent. When the RA reverses, the flight director calls for the pitch angle required to provide level flight.
Preventive Limit Climb (Do Not Climb)	When the RA is issued, a mode annunciation is provided and the vertical guidance mode is changed to display the pitch angle required to provide level flight.
Preventive Limit Descent (Do Not Descend)	When the RA is issued, a mode annunciation is provided and the vertical guidance mode is changed to display the pitch angle required to provide level flight.
Multi-Aircraft Encounter (Maintain Existing V/S)	When the RA is issued, a mode annunciation is provided and the vertical guidance mode is changed to display the pitch angle required to provide level flight.
Multi-Aircraft Encounter (Second Intruder Becomes a Threat While Responding to an Initial Climb RA)	When the RA is initially issued, a mode annunciation is provided and the vertical guidance mode is changed to display the pitch angle required to provide a 1,500 fpm rate of climb. When the second intruder becomes a threat, the flight director displays the pitch angle required to provide level flight.
Multi-Aircraft Encounter (Preventive RA Limiting Climb and Descent Rate to $\pm 1,000$ fpm)	When the RA is issued, a mode annunciation is provided and the vertical guidance mode is changed to display the pitch angle required to provide a vertical speed between $\pm 1,000$ fpm.

2.4.2.8.2.5 HUD (§2.2.6.2.6.1; §2.2.6.2.6.2; §2.2.6.2.6.3; §2.2.6.2.6.4; §2.2.6.2.6.5; §2.2.6.2.6.6)

The tests in this subparagraph verify the proper depiction of corrective and preventive RAs when the RA guidance is shown on a HUD.

Conditions

These tests require that the HUD be powered up and that navigation data be displayed. The test will verify the proper depiction of the RAs shown in Table 2-90. The tests **shall** (2549) be completed for each available display mode.

Success

The success criteria for these tests are shown in Table 2-90. RA guidance information **shall** (2550) be displayed in all available display modes. During the display of all corrective RAs, a flight path target consisting of a box with lines twice the width of the trapezoid comprising the “no fly” zone **shall** (2551) be displayed. The trapezoid representing the flight path to be attained or maintained and the flight path target **shall** (2552) be readily discernible and distinguishable from all other information shown on the HUD.

Note: This assessment is a subjective one and should be performed by Flight Test personnel from the Aircraft Certification Office.

The underlying navigation data and HUD guidance cue remain displayed at all times. Other information may be removed to declutter the display as long as this does not interfere with the pilot’s ability to comply with the RA or operate the aircraft in compliance with the appropriate regulations and requirements.

Note: This assessment of which information can be removed during an RA is a subjective one and should be performed by Flight Test personnel from the Aircraft Certification Office.

Table 2-90: Tests for Implementations Displaying RAs on a HUD

Advisory	RA Success Criteria
Climb	A trapezoid is displayed, beginning at the bottom of the HUD and extending upwards to the flight path angle required to obtain a 1,500 fpm rate of climb. The trapezoid does not occlude any other information shown on the HUD. A flight path target is shown at the closed end of the trapezoid. While the RA is displayed, either the message “TCAS” or “TRAFFIC” is written on the HUD.
Descend	A trapezoid is displayed, beginning at the top of the HUD and extending downwards to the flight path angle required to obtain a 1,500 fpm rate of descent. The trapezoid does not occlude any other information shown on the HUD. A flight path target is shown at the closed end of the trapezoid. While the RA is displayed, either the message “TCAS” or “TRAFFIC” is written on the HUD.
Altitude Crossing Climb	A trapezoid is displayed, beginning at the bottom of the HUD and extending upwards to the flight path angle required to obtain a 1,500 fpm rate of climb. The trapezoid does not occlude any other information shown on the HUD. A flight path target is shown at the closed end of the trapezoid. While the RA is displayed, either the message “TCAS” or “TRAFFIC” is written on the HUD.
Altitude Crossing Descend	A trapezoid is displayed, beginning at the top of the HUD and extending downwards to the flight path angle required to obtain a 1,500 fpm rate of descent. The trapezoid does not occlude any other information shown on the HUD. A flight path target is shown at the closed end of the trapezoid. While the RA is displayed, either the message “TCAS” or “TRAFFIC” is written on the HUD.
Reduce Climb (Do Not Climb)	A trapezoid is displayed, beginning at the top of the HUD and extending downwards to the flight path angle required to obtain level flight. The trapezoid does not occlude any other information shown on the HUD. A flight path target is shown at the closed end of the trapezoid. While the RA is displayed, either the message “TCAS” or “TRAFFIC” is written on the HUD.
Reduce Descent (Do Not Descend)	A trapezoid is displayed, beginning at the bottom of the HUD and extending upwards to the flight path angle required to obtain level flight. The trapezoid does not occlude any other information shown on the HUD. A flight path target is shown at the closed end of the trapezoid. While the RA is displayed, either the message “TCAS” or “TRAFFIC” is written on the HUD.

Advisory	RA Success Criteria
RA Reversal (Descend to Climb)	A trapezoid is initially displayed, beginning at the top of the HUD and extending downwards to the flight path angle required to obtain a 1,500 fpm rate of descent. The trapezoid does not occlude any other information shown on the HUD. A flight path target is shown at the closed end of the trapezoid. While the RA is displayed, either the message "TCAS" or "TRAFFIC" is written on the HUD. When the RA reverses to a Climb RA, the trapezoid begins at the bottom of the HUD and extends upwards to the flight path angle required to obtain a 1,500 fpm rate of climb. A flight path target is shown at the closed end of the trapezoid. The message "TCAS" or "TRAFFIC" remains written on the HUD.
RA Reversal (Climb to Descend)	A trapezoid is initially displayed, beginning at the bottom of the HUD and extending upwards to the flight path angle required to obtain a 1,500 fpm rate of climb. The trapezoid does not occlude any other information shown on the HUD. A flight path target is shown at the closed end of the trapezoid. While the RA is displayed, either the message "TCAS" or "TRAFFIC" is written on the HUD. When the RA reverses to a Descend RA, the trapezoid begins at the top of the HUD and extends downwards to the flight path angle required to obtain a 1,500 fpm rate of descent. A flight path target is shown at the closed end of the trapezoid. The message "TCAS" or "TRAFFIC" remains written on the HUD.
Increase Climb	A trapezoid is displayed, beginning at the bottom of the HUD and extending upwards to the flight path angle required to obtain a 2,500 fpm rate of climb. The trapezoid does not occlude any other information shown on the HUD. A flight path target is shown at the closed end of the trapezoid. While the RA is displayed, either the message "TCAS" or "TRAFFIC" is written on the HUD.
Increase Descent	A trapezoid is displayed, beginning at the top of the HUD and extending downwards to the flight path angle required to obtain a 2,500 fpm rate of descent. The trapezoid does not occlude any other information shown on the HUD. A flight path target is shown at the closed end of the trapezoid. While the RA is displayed, either the message "TCAS" or "TRAFFIC" is written on the HUD.
Maintain Rate RA (Maintain Climb Rate of 4000 fpm)	A trapezoid is displayed, beginning at the bottom of the HUD and extending upwards to the flight path angle required to obtain a 4,000 fpm rate of climb. The trapezoid does not occlude any other information shown on the HUD. A flight path target is shown at the closed end of the trapezoid. While the RA is displayed, either the message "TCAS" or "TRAFFIC" is written on the HUD.
Maintain Rate RA (Maintain Descent Rate of 4000 fpm)	A trapezoid is displayed, beginning at the top of the HUD and extending downwards to the flight path angle required to obtain a 4,000 fpm rate of descent. The trapezoid does not occlude any other information shown on the HUD. A flight path target is shown at the closed end of the trapezoid. While the RA is displayed, either the message "TCAS" or "TRAFFIC" is written on the HUD.

Advisory	RA Success Criteria
Altitude Crossing Maintain Rate (Maintain Climb Rate of 3200 fpm)	A trapezoid is displayed, beginning at the bottom of the HUD and extending upwards to the flight path angle required to obtain a 3,200 fpm rate of climb. The trapezoid does not occlude any other information shown on the HUD. A flight path target is shown at the closed end of the trapezoid. While the RA is displayed, either the message "TCAS" or "TRAFFIC" is written on the HUD.
Altitude Crossing Maintain Rate (Maintain Descent Rate of 2600)	A trapezoid is displayed, beginning at the top of the HUD and extending downwards to the flight path angle required to obtain a 2,600 fpm rate of descent. The trapezoid does not occlude any other information shown on the HUD. A flight path target is shown at the closed end of the trapezoid. While the RA is displayed, either the message "TCAS" or "TRAFFIC" is written on the HUD.
Weakening of Positive RAs (Climb RA Weakens to a Level-Off RA)	A trapezoid is initially displayed, beginning at the bottom of the HUD and extending upwards to the flight path angle required to obtain a 1,500 fpm rate of climb. The trapezoid does not occlude any other information shown on the HUD. A flight path target is shown at the closed end of the trapezoid. While the RA is displayed, either the message "TCAS" or "TRAFFIC" is written on the HUD. When the RA weakens, the trapezoid is displayed, beginning at the bottom of the HUD and extending upwards to the flight path angle required to obtain level flight. The flight path target is shown at the closed end of the trapezoid. The trapezoid does not occlude any other information shown on the HUD. While the RA is displayed, either the message "TCAS" or "TRAFFIC" remains written on the HUD.
Weakening of Positive RAs (Descend RA Weakens to a Level-Off RA)	A trapezoid is initially displayed, beginning at the top of the HUD and extending downwards to the flight path angle required to obtain a 1,500 fpm rate of descent. The trapezoid does not occlude any other information shown on the HUD. A flight path target is shown at the closed end of the trapezoid. While the RA is displayed, either the message "TCAS" or "TRAFFIC" is written on the HUD. When the RA weakens, the trapezoid is displayed, beginning at the top of the HUD and extending downwards to the flight path angle required to obtain level flight. The trapezoid does not occlude any other information shown on the HUD. While the RA is displayed, either the message "TCAS" or "TRAFFIC" remains written on the HUD.
Preventive Limit Climb (Do Not Climb)	A trapezoid is displayed, beginning at the top of the HUD and extending downwards to the flight path angle required to obtain level flight. The trapezoid does not occlude any other information shown on the HUD. While the RA is displayed, either the message "TCAS" or "TRAFFIC" is written on the HUD.
Preventive Limit Descent (Do Not Descend)	A trapezoid is displayed, beginning at the bottom of the HUD and extending upwards to the flight path angle required to obtain level flight. The trapezoid does not occlude any other information shown on the HUD. While the RA is displayed, either the message "TCAS" or "TRAFFIC" is written on the HUD.

Advisory	RA Success Criteria
Multi-Aircraft Encounter (Maintain Existing V/S)	A trapezoid is displayed beginning at the top of the HUD and extending downwards to the flight path angle representing a 250 fpm rate of climb. A second trapezoid is also displayed beginning at the bottom of the HUD and extending upwards to the flight path angle representing a 250 fpm rate of descent. There is sufficient room between the closed ends of the two trapezoids for the flight path target and the ownship reference symbol of the HUD to be displayed. While the RA is displayed, either the message "TCAS" or "TRAFFIC" is written on the HUD.
Multi-Aircraft Encounter (Second Intruder Becomes a Threat While Responding to an Initial Climb RA)	A trapezoid is displayed beginning at the bottom of the HUD and extending upwards to the flight path angle representing a 1,500 fpm rate of climb. A flight path target is shown at the closed end of the trapezoid. While the RA is displayed, either the message "TCAS" or "TRAFFIC" is written on the HUD. When the second intruder becomes a threat, a trapezoid begins at the bottom of the HUD and extends upwards to the flight path angle representing a 250 fpm rate of descent. A second trapezoid is also displayed beginning at the top of the HUD and extending downwards to the flight path angle representing a 250 fpm rate of climb. There is sufficient room between the closed ends of the two trapezoids for the flight path target and the ownship reference symbol of the HUD to be displayed. While the composite RA is displayed, either the message "TCAS" or "TRAFFIC" remains written on the HUD.
Multi-Aircraft Encounter (Preventive RA Limiting Climb and Descent Rate to $\pm 1,000$ fpm)	A trapezoid is displayed beginning at the top of the HUD and extending downwards to the flight path angle representing a 1,000 fpm rate of climb. A second trapezoid is also displayed beginning at the bottom of the HUD and extending upwards to the flight path angle representing a 1,000 fpm rate of descent. While the RA is displayed, either the message "TCAS" or "TRAFFIC" is written on the HUD.

2.4.2.8.3 Controls

The tests defined in the following sections verify that the ACAS X and transponder controls perform the functions required in §2.2.6.5. These tests are to be performed using the actual controls or control panels. The controls are not required to be connected to the ACAS X processor unit or to the display which they are intended to control. If the controls are not actually connected to the ACAS X processor unit and associated displays, appropriate instrumentation must be provided to demonstrate the required inputs and outputs are available at the controls or control panel.

2.4.2.8.3.1 ACAS X/Mode S Controls (§2.2.6.5.1)

These tests verify that the ACAS X and Mode S control panel(s) provide the means to select the required operating modes of ACAS X and the Mode S transponder.

Conditions

The ACAS X/Mode S control panel is powered up and connected to either: (1) the ACAS X processor unit and the Mode S transponder; or (2) appropriate test instrumentation to

demonstrate the control panel provides the correct outputs. If a design implementation uses separate control panels for ACAS X and the Mode S transponder, both must be present for the completion of these tests.

The test requires that each of the four required ACAS X/transponder operating modes be selected and that the proper output of the control panel be verified. The test also requires that all required control functions of the control panel be exercised.

Success

- a. When the operation of the Mode S transponder only is selected, the control panel provides a unique signal which indicates the transponder is operative, but that ACAS X must be placed in Standby.
- b. When the TA/RA mode is selected, the control panel provides a unique signal which indicates the transponder is operative and that ACAS X is capable of issuing RAs.
- c. When the TA-Only mode is selected, the control panel provides a unique signal which indicates the transponder is operative and the pilot has selected an ACAS X operating mode which inhibits RAs.
- d. When the ACAS X Self Test is selected, the control panel provides a unique signal which initiates the ACAS X Self Test.
- e. The control panel provides the capability to select all 4096 Mode A transponder codes. When a code is selected, the correct information is provided as an output from the control panel to the transponder.
- f. When altitude reporting is disabled, a unique code is provided to indicate that Mode C should not be included in the transponder replies and that ACAS X should be failed.
- g. The position of all controls is readily discernible by inspection.

Note: This assessment is a subjective one and should be performed by Flight Test personnel from the Aircraft Certification Office.

2.4.2.8.3.2 Traffic Display Controls (§2.2.6.5.2)

These tests verify the proper implementations of the optional traffic display controls.

Conditions

The test set-up is dependent on the actual implementation of the traffic display controls. If the controls are integrated into the traffic display, the actual traffic display must be used in these tests. If the controls are included as part of the ACAS X/Mode S control panel, or as part of another type of control panel, only that control panel is required.

Success

- a. Altitude Range Selector. A switch is provided to permit the pilot to display traffic that is outside a relative altitude of $\pm 2,700$ feet. The switch provides three selections indicated by the words “Above”, “Normal”, or “Below” or a suitable abbreviation of these words. When each of the positions is selected, a unique signal is provided by the control to the traffic display which indicates the relative altitudes to be displayed.
- b. Range Selection. If the traffic display is a variable range display, a switch is provided to select the available display ranges. When each of the positions is selected, a unique signal is provided by the control to the traffic display which indicates the selected display range.

- c. Actual Altitude. If the traffic display provides for the display of actual altitude, a switch is provided to select between the display of actual and relative altitude. When each of the positions is selected, a unique signal is provided by the control to the traffic display which indicates the format of altitude to be displayed.
- d. Traffic Override. On a part time display, a switch is provided to override the display of ACAS X information and return the display to its other function. When traffic override is selected, a unique signal is provided by the control to the traffic display which indicates that the display of traffic is no longer desired.
- e. Display Mode. On displays with both a part time and a full time display capability, a switch is provided to enable the pilot to configure the display to either operating mode. When either switch position is selected, a unique signal is provided by the control to the traffic display which indicates the desired display configuration.
- f. The position of all switches is readily discernible by inspection.

Note: This assessment is a subjective one and should be performed by Flight Test personnel from the Aircraft Certification Office.

2.4.2.8.3.3 Shared Weather Radar/Traffic Display Controls (§2.2.6.5.3, §2.2.6.5.3.1, §2.2.6.5.3.2, §2.2.6.5.3.3)

These tests verify the proper implementations of the controls when the ACAS X traffic display is shared with the weather radar display.

Conditions

The test set-up is dependent on the actual implementation of the traffic display and weather radar controls. If the controls are integrated into the traffic display, the actual traffic display must be used in these tests. If the controls are included as part of the ACAS X/Mode S control panel, or as part of another type of control panel, only that control panel is required.

Success

- a. A switch is provided to select either the WX/ACAS X mode or the WX-and-Traffic Mode.
- b. If the display supports a WX-Only and a ACAS X-Only mode, a switch is provided to select these modes.
- c. The position of all switches is readily discernible by inspection.

Note: This assessment is a subjective one and should be performed by Flight Test personnel from the Aircraft Certification Office.

2.4.2.8.4 Status and Failure Annunciations (§2.2.6.6)

These tests verify that the required status and failure annunciations are properly displayed on the ACAS X displays.

2.4.2.8.4.1 Traffic Display Annunciations (§2.2.6.6.1)

The required annunciations on the traffic display are contained in §2.4.2.8.1.1.10.

2.4.2.8.4.2 RA Display Annunciations (§2.2.6.6.2)

These tests verify that the RA display properly annunciates the required ACAS X operating modes and failure conditions.

Conditions

This test requires that the RA display be powered up and configured such that ACAS X information can be displayed. Signals are provided to indicate that ACAS X is in Standby, TA-Only, and TA/RA mode. Signals **shall** (2553) also be provided to indicate that ACAS X has failed, that the RA display is unable to display RAs, and that a pilot-initiated Self Test is in progress.

Success

- a. When the display receives an indication that ACAS X is in Standby, this state is clearly annunciated on the RA display. This annunciation is shown in a color other than yellow, amber, or red.
- b. When the display receives an indication that ACAS X is in the TA-Only mode, this state is clearly annunciated on the RA display. This annunciation is shown in a color other than yellow, amber, or red when there is no TA present.
- c. When the display receives an indication that ACAS X is in the TA/RA mode, this state is clearly annunciated on the RA display. This annunciation is shown in a color other than yellow, amber, or red.
- d. When the display receives an indication that ACAS X has failed, this state is clearly annunciated on the RA display. This annunciation is shown in either yellow or amber.
- e. When the display receives an indication that the RA display has failed, this state is clearly annunciated on the display. This annunciation is shown in either yellow or amber.
- f. When the display receives an indication that a pilot initiated Self Test is in progress, this state is clearly annunciated on the RA display. This annunciation is shown in a color other than yellow, amber, or red.
- g. All ACAS X-related annunciations have a single meaning which is consistent in all available display modes.
- h. On shared displays, the ACAS X annunciations are consistent with all other annunciations shown on the display.

Note: The completion of this step may require additional inputs to the display to generate the other annunciations.

2.4.2.9 Automatic Performance Monitoring and Self Test (§2.2.7)

2.4.2.9.1 Automatic Performance Monitoring (§2.2.7.1.1)

The manufacturer **shall** (2554) provide a test plan to verify that the failure response specified in §2.2.7.1.1 is actuated within the time specified in §2.2.7.1.1 under each of the fault conditions that would be detected if the Monitor is operating as specified in §2.2.7.1, §2.2.7.2, and §2.2.7.3. The plan **shall** (2555) include a procedure to verify that the message interface to the Mode S transponder indicates a non-operative ACAS X whenever the failure response is actuated.

These tests may, at the manufacturer's option, be performed in an open box mode. The manufacturer **shall** (2556) provide means of his own design (e.g., the manual disconnecting of cables or cutting of wires, a temporary test harness, a permanent test connector or any combination of these) for inducing each of the malfunctions whose presence the Monitor is required to detect. In any instance where the relation between the malfunction and the

means for inducing the malfunction is not physically obvious, the manufacturer **shall** (2557) provide an engineering analysis that establishes this relation.

Engineering analysis may be substituted for any test in which the manufacturer's design makes the artificial stimulation of a malfunction impractical.

2.4.2.9.2 Self-Test (§2.2.7.1.3)

The manufacturer **shall** (2558) provide a test plan to verify that the self-test function operates as specified in §2.2.7.1.3. The general considerations stated in §2.4.2.9.1 apply. Procedures for self-test verification may, whenever appropriate, use the results of tests performed under §2.4.2.9.1.

2.4.2.9.3 Own Transponder ICAO 24-Bit Aircraft Address (§2.2.7.2.6)

This test verifies that the Monitor recognizes an own ICAO 24-bit Aircraft Address of either all 0s or all 1s and upon recognition, declares an ACAS X failure.

The manufacturer must simulate ownership ICAO 24-bit Aircraft Address conditions of all 0s and all 1s and demonstrate for each condition that within one second the Monitor:

- a. indicates to the flight crew that an abnormal condition exists;
- b. causes any Mode S transmissions that report ownership status to show that ownership has no on-board resolution capability;
- c. prevents interrogations by own ACAS X; and
- d. deactivates the normal ACAS X display functions.

2.4.2.10 Performance Compatibility with Ownership's Mode S Transponder

The following tests verify that under normal operating conditions the Mode S transponder and the ACAS X functions do not interfere with one another.

2.4.2.10.1 ACAS X to Mode S Transponder Interference (§2.1.10.1)

This test verifies that the ACAS X functions do not couple spurious signals into the Mode S transponder and thereby degrade its minimum threshold (sensitivity) level.

Test Scenario: A sensitivity measurement is to be performed on the Mode S receiver immediately following completion of the ACAS X transmitter active state (as defined in §2.1.10.1) during which an ACAS X interrogation occurred at its maximum output power level value.

If separate antenna ports are provided for the Mode S transponder and the ACAS X, the external coupling paths between the transponder receiver input and the ACAS X transmitter output **shall** (2559) be simulated to represent the estimated maximum coupling with the equipment installed in an aircraft.

A separate transponder sensitivity measurement **shall** (2560) be conducted for the top and bottom antenna ports in accordance with Ref. B, §2.2.2.3 and §2.4.2.1.

2.4.2.10.2 Mode S Transponder to ACAS X Interference (§2.2.3.10)

This test verifies that the Mode S functions do not couple spurious signals into the ACAS X receiver and thereby degrade its minimum threshold (sensitivity) level.

Test Scenario: A sensitivity measurement is to be performed on the ACAS X receiver immediately following completion of the Mode S transmitter active state as defined in §2.2.3.10.

If separate antenna ports are provided for the Mode S transponder and ACAS X, the external coupling paths between the ACAS X receiver input and the Mode S transponder transmitter output **shall** (2561) simulate the estimated maximum coupling with the equipment installed in an aircraft.

2.4.2.11 Tests Related to Passive Surveillance

2.4.2.11.1 General Scenario Description

This section defines common scenario parameters that are applicable to all tests in this section unless otherwise stated.

2.4.2.11.1.1 ACAS X (Own) Aircraft

Altitude = 12,000 ft

Position always available

Position: The following positions will be used for ownship in the tests. These were selected to check the Airborne Position Message decoding for different geographic areas and areas of high density.

- London lat/lon: 51° 32' N / 0° 5' W
- Frankfurt lat/lon: 50° 7' N / 8° 41' E
- New York lat/lon: 40° 47' N / 73° 58' W
- Dallas lat/lon: 32° 51' N / 96° 51' W
- Sydney lat/lon: 33° 52' S / 151° 12' E
- São Paulo lat/lon: 23° 33' S / 46° 38' W

Velocity = 0 kt

Heading = 0 degrees

Sensitivity level Selection = Automatic

Horizontal Position Uncertainty = 0.05 NM

Horizontal Position Integrity Bound = 0.5 NM

2.4.2.11.1.2 Intruder Aircraft

All passive data used in all tests are assumed to use DF=17 ADS-B messages unless otherwise stated.

All the tests specify a range and relative velocity of intruder aircraft with respect to ownship. The equipment manufacturer may select what relative azimuth the intruder is with respect to ownship.

Set up a nominal difference between all passive and active data so that the correct data source (active or passive) is output or provided to the STM logic to be verified. The suggested difference is:

- in altitude: 25 ft when quantization is 25 ft
- in range: 100 ft

Altitude Q	= 25 ft
CC	= true
Reply/Squitter Power	= -50 dBm
Reply/Squitter Power	= -70 dBm (for version ≥ 2 ADS-B)

Note: It is acceptable to set the MTL of version ≥ 2 ADS-B messages below the Active Interrogation Reply (DF=0) MTL (-74 dBm) for up to 10 seconds before the start of the test ($T=0$) to allow the track to be acquired passively before being acquired actively with DF=0 replies.

Active Interrogation Reply (DF=0)

Always available per the intruder set-up in each test scenario. Formatted per Ref. B.

Active Squitter (DF=11)

Always available per the intruder set-up in each test scenario. Formatted per Ref. B.

Long Active Interrogation Reply (DF=16)

Always available per the intruder set-up in each test scenario. Formatted per Ref. B.
Used to support the crosslink interrogation.

Airborne Position Message (DF=17, 18)

All intruders, unless otherwise stated, should transmit their Airborne Position Message as required by Ref. G and per the specific field settings defined.

The intruder should transmit DF=17, 18 airborne position squitters at a nominal rate of 2/sec when ACAS X is in squitter listening mode. These squitters alternate their CPR Format 0,1,0,1, etc. with respective even, odd position encoding.

The following provide the standard data content for Airborne Position Messages.

DF=17 CA (6-8)	= 5 (ADS-B airborne)
DF=18 CF (6-8)	= value determined in test, (CF=0 ADS-B ICAO 24-bit Aircraft Address, CF=1 ADS-B Anonymous Address, or CF=6 ADS-R)
AA (9-32)	= intruder's address
ME	
- Type Code (33-37)	= 9
- Surv Status (38-39)	= 0
- Single Ant (40)	= 0
- Altitude (41-52)	= value specified in test, encoding per Ref. G
- Time (53)	= 0
- CPR Format (54)	= 0/1 alternating with even/odd encoding
- Encoded Latitude (55-71)	= Value determined from test own lat, lon, and range to intruder
- Encoded Long (72-88)	= Value determined from test own lat, lon, and range to intruder
- IMF (ADS-R)	= value determined in test, (ADS-R ICAO 24-bit Aircraft Address or Anonymous Address)

Airborne Velocity Message

All intruders, unless otherwise stated, should transmit their Airborne Velocity Message per the required encoding and rate as required by Ref. G. This section defines example settings for subfields settings defined in this section. All values are encoded per Ref. G.

DF=17 CA (6-8)	= 5 (airborne)
DF=18 CF (6-8)	= value determined in test, (CF=0 ADS-B ICAO 24-bit
Aircraft Address, CF=1 ADS-B Anonymous Address, or CF=6 ADS-R)	
AA (9-32)	= intruder's address
ME	
- Type Code (33-37)	= 19
- Subtype (38-40)	= 1
- Intent Change Flag (41)	= 0
- IFR Capability Flag (42)	= 1
- NAC _v (43-45)	= 4
- E/W Direction Bit (46)	= value specified in test
- E/W Velocity (47-56)	= value specified in test
- N/S Direction Bit (57)	= value specified in test
- N/S Velocity (58-67)	= value specified in test
- Vert Rate Source (68)	= 1
- Vert Rate Sign (69)	= value specified in test
- Vert Rate (70-78)	= value specified in test
- Reserved (79-80)	= 0
- Difference from Barometric Altitude Sign (81)	= 0
- Difference from Barometric Altitude (82-88)	= 0

Flight Identification Message

All intruders, unless otherwise stated, should transmit their Flight Identification Message per the required encoding and rate as required by Ref. G and per the subfield settings defined in this section. The equipment manufacturer may specify Ident Chars (41-88) that allow the test aircraft to be individually identified. All values are encoded per Ref. G.

DF=17 CA (6-8)	= 5 (airborne)
DF=18 CF (6-8)	= value determined in test, (CF=0 ADS-B ICAO 24-bit
Aircraft Address, CF=1 ADS-B Anonymous Address, or CF=6 ADS-R)	
AA (9-32)	= intruder's address
ME	
- Type Code (33-37)	= 4
- ADS-B Emitter Category (38-40)	= 0
- Ident Char #1 (41-46)	= manufacturer specified
- Ident Char #2 (47-52)	= manufacturer specified
- Ident Char #3 (53-58)	= manufacturer specified
- Ident Char #4 (59-64)	= manufacturer specified
- Ident Char #5 (65-70)	= manufacturer specified
- Ident Char #6 (71-76)	= manufacturer specified
- Ident Char #7 (77-82)	= manufacturer specified
- Ident Char #8 (83-88)	= manufacturer specified

Note: Squitter Bit field numbering. All the bit field numbering identified in this tests is referenced to the entire DF=17, 18 message and not the ME field numbering.

For intruders that are identified in the test sections to be ADS-B Version Number ≥ 2 targets the Target State and Status Message and the Aircraft Operational Status Message should be transmitted, otherwise these messages should not be transmitted (this identifies them as version=0 intruders).

Target State and Status Message

When specified in the test, ADS-B version 2 intruders should transmit their Target State and Status Message per the required encoding and rate as required by Ref. G and per the subfield settings defined in this section. All values are encoded per Ref. G. Only the subfields applicable to passive surveillance are defined for this message.

DF=17 CA (6-8)	= 5 (airborne)
DF=18 CF (6-8)	= value determined in test, (CF=0 ADS-B ICAO 24-bit Aircraft Address, CF=1 ADS-B Anonymous Address, or CF=6 ADS-R)
AA (9-32)	= intruder's address
ME	
- NACp (72-75)	= 7
- SIL (77-79)	= 3

Aircraft Operational Status Message

When specified in the test, ADS-B version 2 intruders should transmit their Aircraft Operational Status Message per the required encoding and rate as required per Ref. G and per the subfield settings defined in this section. All values are encoded per Ref. G. Only the subfields applicable to passive surveillance are defined for this message.

DF=17 CA (6-8)	= 5 (airborne)
DF=18 CF (6-8)	= value determined in test, (CF=0 ADS-B ICAO 24-bit Aircraft Address, CF=1 ADS-B Anonymous Address, or CF=6 ADS-R)
AA (9-32)	= intruder's address
ME	
- ADS-B Version Number (73-75)	= 2
- NACp (77-80)	= 7
- SIL (83-84)	= 3
- SDA (63-64)	= 2

2.4.2.11.1.3 Test Success Criteria

The success criteria specified for each test includes margin to account for timing differences between different implementations. Some timing success criteria have as much as 10 sec of allowed variation. These timing variations account for the low interrogation rates (0.1 Hz or 0.016 Hz) required for aircraft being tracked with passive surveillance. The tests, even with the allowed variation in timing, verify correct implementation of the requirements.

Many of the tests specify verifying interrogation rates or intervals. For example tests that indicate that an interrogation interval should be verified from T=A to B are not requiring an interrogation at T=A or B but that the correct interrogation rate is maintained within the interval. Also, the interrogation interval requirements should be checked after that corresponding intruder has become an established track.

Results specified to occur at or within X sec may occur at or within X surveillance update intervals.

2.4.2.11.2 Verification of Shared Use of 1090 MHz Receiver with an ADS-B Receiving Subsystem (§2.1.13)

Receivers shared with a ACAS X unit **shall** (2562) be tested to comply with the test requirements of Ref. G, §2.4.4.2.

Note: The only requirement in this section is for shared receiver systems to meet the requirements of RTCA/DO-260B §2.2.4.2. The above Hybrid test requirement references the RTCA/DO-260B test section defined for §2.2.4.2.

2.4.2.11.3 Verification of Acquisition and Maintenance of Established Tracks Using Active Surveillance (§2.2.4.6.4.2.3.2)

Test 1 Active Surveillance

Verify that the requirements for maintaining a track under Active Surveillance are met (§2.2.4.6.4.2.3.2.1).

Note: Although this test has multiple intruders, the purpose of this test can be met by running each intruder as an individual test.

This test verifies that active tracking is maintained for the following reasons: non-correlating ICAO 24-bit Aircraft Address, lack of DF=17 data, non-crosslink capable transponders, failure to validate intruders, and for being outside of the passive surveillance region.

Scenario Description

- Intruder 1 tests §2.2.4.6.4.2.3.2.2 by having mismatched active and passive ICAO 24-bit Aircraft Addresses.
- Intruder 2 tests §2.2.4.6.4.2.3.2.2, §2.2.3.8.3.2.8.4 and §2.2.3.8.3.2.8.5 by having invalid intruder DF=17 position data.
- Intruder 3 tests §2.2.4.6.4.2.3.2.2 and §2.2.3.8.3.2.8.2 by having invalid intruder altitude data.
- Intruder 4 tests §2.2.4.6.4.2.3.2.7 by having the intruder fail the range validation criteria in the 10 sec interrogation area.
- Intruder 5 tests §2.2.4.6.4.2.3.2.7 by having the intruder fail the altitude validation criteria in the 10 sec interrogation area.
- Intruder 6 tests §2.2.4.6.4.2.3.2.7 by having the intruder fail the bearing validation criteria in the 10 sec interrogation area.
- Intruder 7 tests that when an initiating track does not meet the passive surveillance region requirements (§2.2.4.6.4.2.3.1.6) the intruder is tracked actively.
- Intruder 8 tests the condition where validation fails, but subsequent validation passes (§2.2.4.6.4.2.3.2.2).
- Intruder 9 tests the condition where validation passes, but subsequent revalidation fails and then passes again (§2.2.4.6.4.2.3.1.7).

ACAS X Aircraft

Altitude	= 12,000 ft
Altitude Rate	= 0 FPM
Position	= London

Note: For easier identification of intruders 1-6, different altitudes and ranges may be used as long as they are within 9,000 ft and 25 NM of ownship; but, ranges less than those defined for intruders 1-6 should not be used.

Intruder Aircraft #1

Altitude = 17,000 ft
 Altitude Rate = 0 FPM
 Range = 10 NM at T=0 sec
 Relative Speed = -144 kt (-0.04 NM/sec)
 DF=17 extended squitters contain a different ICAO 24-bit Aircraft Address than the Active Replies and DF=11 squitters.
 At T=30 the intruder is terminated.

Intruder Aircraft #2

Altitude = 17,000 ft
 Altitude Rate = 0 FPM
 Range = 10 NM at T=0 sec
 Relative Speed = -144 kt (-0.04 NM/sec)
 DF=17 extended squitters contain invalid position data (format type code is set to 0).
 At T=30 the intruder is terminated.

Intruder Aircraft #3

Altitude = 17,000 ft
 Altitude Rate = 0 FPM
 Range = 10 NM at T=0 sec
 Relative Speed = -144 kt (-0.04 NM/sec)
 DF=17 extended squitters contain invalid altitude (AC field is all zeroes).
 At T=30 the intruder is terminated.

Intruder Aircraft #4

Altitude = 16,000 ft
 Altitude Rate = 0 FPM
 Range = 12 NM at T=0 sec
 Relative Speed = -360 kt (-0.1 NM/sec)
 The Range Difference is 305 m.
 At T=30 the intruder is terminated.

Intruder Aircraft #5

Altitude = 16,000 ft
 Altitude Rate = 0 FPM
 Range = 12 NM at T=0 sec
 Relative Speed = -360 kt (-0.1 NM/sec)
 The Altitude Difference is 150 ft.
 At T=30 the intruder is terminated.

Intruder Aircraft #6

Altitude = 16,000 ft
 Altitude Rate = 0 FPM
 Range = 12 NM at T=0 sec
 Relative Speed = -360 kt (-0.1 NM/sec)
 The Bearing Difference is 50 degrees.
 At T=30 the intruder is terminated.

Intruder Aircraft #7

Altitude = 7400 ft
Altitude Rate = 0 FPM
Range = 3.1 NM at T=0 sec
Relative Speed = 0 kt
At T=30 the intruder is terminated.

Intruder Aircraft #8

Altitude = 17500 ft
Altitude Rate = 0 FPM
Range = 10 NM at T=0 sec
Relative Speed = -144 kt (-0.04 NM/sec)
From T=0 to T=30, DF=17 extended squitters result in a range difference that is 390 m.
From T=31 to T=60, DF=17 extended squitters result in a range difference that is 150 m.
At T=60 the intruder is terminated.

Intruder Aircraft #9

Altitude = 17,500 ft
Altitude Rate = 0 FPM
Range = 2.9 NM at T=0 sec
Relative Speed = 0 kt
From T=60 to T=79, DF=17 extended squitters result in a range difference that is 390 m.
From T=80 to T=90, DF=17 extended squitters result in a range difference that is 150 m.
At T=90 the intruder is terminated.

Success Criteria

Intruder 1, 2, & 3

UF=0 and RL=0 for all interrogations.

Interrogation interval is 5 sec.

Verify DF=0 replies are provided to the STM with surv_mode = 1/Red.

Verify that the intruder is tracked actively by the STM and provided in the STM display output (Intruder 1 will also be tracked passively by the STM and provided to the display due to the DF=17 extended squitters containing a different ICAO 24-bit Aircraft Address than the Active Replies and DF=11 squitters).

Intruder 4, 5, & 6

Each interrogation after T=5 is a UF=0, RL=0 interrogation. By T=5, the Mode S track will have had a chance to be acquired and established.

Interrogation interval is no greater than 5 sec.

Verify DF=0 replies are provided to the STM with surv_mode = 0/Norm or 1/Red.

Verify that the intruder is provided in the STM display output for the full duration of the test.

Intruder 7

UF=0 and RL=0 for all interrogations.

Interrogation interval is no greater than 5 sec.

Verify DF=0 replies are provided to the STM with surv_mode = 0/Norm or 1/Red.

Verify that the intruder is tracked passively by the STM and provided in the STM display output.

Intruder 8

Each interrogation after T=5 is a UF=0, RL=0 interrogation. By T=5, the Mode S track will have had a chance to be acquired and established.

Interrogation interval is 5 sec.

Verify that at T<30, DF=0 replies are provided to the STM with surv_mode = 1/Red.

Verify that at T≥40, DF=0 replies are provided to the STM with surv_mode = 2/HS.

Verify that the intruder is provided in the STM display output for the full duration of the test.

Intruder 9

Interrogation rate after T=10 sec is 60 sec (Hybrid).

Interrogation rate after T=60 should be 1 Hz (Active).

Interrogation rate should change to 60 sec after T=82 – this is to verify that only after two successful validation attempts at T=61 and T=62 that the track is transitioned to passive.

Verify that after T=10, DF=0 replies are provided to the STM with surv_mode = 2/HS.

Verify that after T=60, DF=0 replies are provided to the STM with surv_mode = 0/Norm.

Verify that after T=82, DF=0 replies are provided to the STM with surv_mode = 2/HS.

Verify that after T=10, the intruder is provided in the STM display output for the remainder of the test.

Test 2a– Ownship on-ground with Extended Hybrid

Verify that the requirements for maintaining a track under Extended Hybrid Surveillance are met (§2.2.4.6.4.2.3.1.3).

This test verifies that passive tracking is maintained for the following reasons: ownship is on-ground with good ownship and target traffic quality.

Scenario Description

- Intruder 1 tests §2.2.4.6.4.2.3.1.3 by having a target that meets the Extended Hybrid Surveillance criteria. (§2.2.4.6.4.2.2.2).

ACAS X Aircraft

Altitude = 0 ft (Ground Level)

Altitude Rate = 0 FPM

Position = London

Radio altitude input = 0 ft

Ground Speed is valid and at 0 knots and ACAS X Air/Ground (own.on_ground) indicates on-ground.

Intruder Aircraft #1

Altitude = 2,800 ft

Altitude Rate = 0 FPM

Range = 3 NM at T=0 sec

Relative Speed = -144 kt (-0.04 NM/sec)

At T=30 the intruder is terminated.

Success Criteria

Intruder 1

Verify that there are no interrogations after T=5 since the target is tracked by Extended Hybrid Surveillance. By T=5, the Mode S track will have had a chance to be acquired and established.

Verify that after T=5, DF=0 replies are not provided to the STM.

Verify that the intruder is provided in the STM display output for the full duration of the test.

Test 2b – Ownship transitions to on-ground with Extended Hybrid

Verify that the requirements for maintaining a track under Extended Hybrid Surveillance are met (§2.2.4.6.4.2.3.1.3).

This test verifies that passive tracking is maintained for the following reasons: ownship transitions to on-ground with good ownship and target traffic quality.

Scenario Description

- Intruder 1 tests §2.2.4.6.4.2.3.1.3 by having a target that meets the Extended Hybrid Surveillance criteria. (§2.2.4.6.4.2.2.2).

ACAS X Aircraft

Altitude = 0 ft (Ground Level)

Altitude Rate = 0 FPM

Position = London

Ownship horizontal position uncertainty (95%) is 0.05 NM

Ownship horizontal position integrity bounds is < 0.5 NM with an integrity level of $1e^{-7}$

Ground Speed is valid and at 0 knots

Radio altitude input > 100 ft prior to T=15, at T=15 it transitions to < 50 ft (this forces ACAS X Air/Ground, own.on_ground, to transition from airborne to ground after T=15).

Intruder Aircraft #1

Altitude = 2,800 ft

Altitude Rate = 0 FPM

Range = 3 NM at T=0 sec

Relative Speed = -144 kt (-0.04 NM/sec)

Reply/Squitter Power = -50dBm

At T=30 the intruder is terminated.

Success Criteria

Intruder 1

Each interrogation from T=5 to T=15 is a UF=0, RL=0 interrogation. By T=5, the Mode S track will have had a chance to be acquired and established.

Interrogation interval is 1 sec.

Verify that there are no interrogations after T=20 since the target is tracked by Extended Hybrid Surveillance.

Verify that from T=5 to T=15, DF=0 replies are provided to the STM with surv_mode = 0/Norm.

Verify that after T=20, DF=0 replies are not provided to the STM.

Verify that the intruder is provided in the STM display output for the full duration of the test.

2.4.2.11.4 Verification of Maintenance of Established Tracks Using Passive Surveillance (&2.2.4.6.4.2.3.1)

Note: Although this test has multiple intruders, the purpose of this test can be met by running each intruder as an individual test.

Test 1

This test will verify that tracks that do not meet the range or altitude criteria of §2.2.4.6.4.2.3.1.4 are tracked using passive surveillance while being revalidated with ACAS X active range measurements at the correct rate per §2.2.4.6.4.2.3.1.10. The test also verifies that the Surveillance Mode is set correctly per §2.2.4.6.4.2.3.1.11. Additional requirement coverage is identified in the tests scenarios descriptions below.

Scenario Description

- Intruders 1 & 2 start out far enough beyond the criteria taus to be revalidated at a 60 sec interval. Intruder 2 will then meet the criteria to switch to a 10 sec interrogation interval.
- Intruder 1 will meet the active range criteria (§2.2.4.6.4.2.3.2.3), but it will never meet the altitude criteria and therefore is tracked passively with a 60 second revalidation rate. This test also demonstrates correct decoding of the TYPE subfield (§2.2.3.8.3.2.8.1).
- Intruder 2 will meet the active altitude criteria (§2.2.4.6.4.2.3.2.3), but it will never meet the range criteria and therefore is tracked passively with a 60 second revalidation rate when the altitude criteria is false and a 10 second revalidation rate when the altitude criteria is true.
- Intruder 3 meets the criteria to be passively tracked with a 10 sec interrogation interval even though the bearing is unavailable (§2.2.4.6.4.2.3.2.7).

ACAS X Aircraft

Altitude	= 12,000 ft
Altitude Rate	= 0 FPM
Position	= Sydney

Intruder Aircraft #1

Altitude	= 17,000 ft
Altitude Rate	= 0 FPM
Range	= 10 NM at T=0 sec
Relative Speed	= -144 kt (-0.04 NM/sec)
At T=200 the intruder is terminated.	

Intruder Aircraft #2

Altitude	= 18,100 ft at T=0
Altitude Rate	= -600 FPM
Range	= 3.5 NM
Relative Speed	= 0 kt
At T=200 the intruder is terminated.	

Note: In a realistic simulation it is not possible to maintain a constant slant range while changing the relative altitude. Some small relative speed is permitted.

Intruder Aircraft #3

Altitude	= 8000 ft
Altitude Rate	= 0 FPM
Range	= 9 NM at T=0 sec
Relative Speed	= -360 kt (-0.1 NM/sec)

At T=15 the Active Bearing becomes unavailable. One method for achieving this is to have the target tracked only on a bottom OMNI antenna.

At T=45 the intruder is terminated.

Success Criteria**Intruder 1**

Verify the following:

- 1) Acquisition interrogations have RL=0.
- 2) From T=10 to T=200 validation interrogations, spaced by 60 seconds, are transmitted to this intruder.
- 3) All validation interrogations (after T=10) are standard ACAS X surveillance interrogations (UF=0 RL=0).
- 4) Verify that after T=10, DF=0 replies are provided to the STM with surv_mode = 2/HS.
- 5) Verify that the intruder is provided in the STM display output for the full duration of the test.

Intruder 2

Verify the following:

- 1) From T=10 to T=90 validation interrogations, spaced by 60 seconds, are transmitted to this intruder.
- 2) All validation interrogations (after T=10) are standard ACAS X surveillance interrogations (UF=0, RL=0).
- 3) From T=110 to T=200 a validation interrogation is transmitted once every 10 sec to the intruder aircraft.
- 4) Verify that after T=10, DF=0 replies are provided to the STM with surv_mode = 2/HS.
- 5) Verify that the intruder is provided in the STM display output for the full duration of the test.

Intruder 3

Verify the following:

- 1) From T=10 to T=45, a validation interrogation is transmitted to the intruder aircraft once every 10 seconds.
- 2) Verify that after T=10, DF=0 replies are provided to the STM with surv_mode = 2/HS.
- 3) Verify that the intruder is provided in the STM display output for the full duration of the test.

Test 2

This test verifies basic tracking when updating the track with Airborne Position Messages.

Scenario Description

- Intruder 1 shows the implementation of range correlation windows (§2.2.4.6.4.2.3.1.8) and correct CPR decoding (§2.2.3.8.3.2.8.3).
- Intruder 2 shows the implementation of altitude correlation windows for passive tracks (§2.2.4.6.4.2.3.1.8).
- Intruder 3 shows that the passive track will be coasted when a position squitter is not received for a short period of time (§2.2.4.6.4.2.3.1.9).
- Intruder 4 shows that the passive track will be coasted when an invalid position squitter is received for a short period of time (§2.2.4.6.4.2.3.1.9).

ACAS X Aircraft

Altitude = 12,000 ft
 Altitude Rate = 0 FPM
 Position = New York

Intruder Aircraft #1

Altitude = 17,000 ft
 Altitude Rate = 0 FPM
 Range = 10 NM at T=0 sec
 Relative Speed = -144 kt (-0.04 NM/sec)

At T=15 the reported position data jumps so that the calculated slant range value jumps ≥ 0.25 and < 0.6 NM for 1 sec. The range jump is timed so as not to occur during validation.

At T=45 the intruder is terminated.

Intruder Aircraft #2

Altitude = 18,100 ft at T=0
 Altitude Rate = -600 FPM
 Range = 3.5 NM
 Relative Speed = 0 kt

At T=15 the reported altitude jumps so that the altitude is reported as 18,550 ft for 1 sec. This should produce a jump in altitude of 600 ft, resulting in miscorrelation, and therefore coasting of the altitude. The altitude jump is timed so as not to occur during validation.

At T=45 the intruder is terminated.

Intruder Aircraft #3

Altitude = 7000 ft
 Altitude Rate = 0 FPM
 Range = 10 NM at T=0 sec
 Relative Speed = -144 kt (-0.04 NM/sec)

At T=30 – 31 No Position Squitters.

At T=45 the intruder is terminated.

Intruder Aircraft #4

Altitude = 7000 ft
 Altitude Rate = 0 FPM
 Range = 10 NM, at T=0 sec
 Relative Speed = -144 kt (-0.04 NM/sec)

At T=30 – 31 Invalid Position Squitters (format type code set to 0).

At T=45 the intruder is terminated.

Success Criteria

Intruder 1

Verify that the intruder is provided in the STM display output for the full duration of the test.

No interrogations of the intruder aircraft from T = 10 to T = 45.

Verify that after T=10, DF=0 replies are not provided to the STM.

Intruder 2

Verify that the intruder is provided in the STM display output for the full duration of the test.

No interrogations of the intruder aircraft from T = 10 to T = 45.

Verify that after T=10, DF=0 replies are not provided to the STM.

Intruder 3

Verify that the intruder is provided in the STM display output for the full duration of the test.

No interrogations of the intruder aircraft from T = 10 to T = 45.

Verify that after T=10, DF=0 replies are not provided to the STM.

Intruder 4

Verify that the intruder is provided in the STM display output for the full duration of the test.

No interrogations of the intruder aircraft from T = 10 to T = 45.

Verify that after T=10, DF=0 replies are not provided to the STM.

Test 3 (range tau=false after active to passive transition §2.2.4.6.4.2.3.2.3)

This test verifies that when a track initially transitions to passive surveillance that the 2nd criteria (range tau) of §2.2.4.6.4.2.3.2.3 is assumed to be false until a passive range rate estimate is available or until the track is dropped or transitions back to active surveillance.

The requirement being tested is a corner case, and designing a test which is design independent is difficult. Therefore the equipment manufacturer must design a test that shows coverage of the requirement identified above.

Test 4 (Establish an Extended Hybrid Surveillance Track §2.2.4.6.4.2.2.1.2)

This test verifies that when qualifying ADS-B reports are received an Extended Hybrid Surveillance Track is established. It also verifies that a track is not established if the reports do not meet the required criteria of §2.2.4.6.4.2.2.1.2.

(The following tests may be performed using ADS-B reports or directly decoded ADS-B messages. TIS-B and ADS-R data is not permitted.)

Scenario Description

- Intruder 1 will provide ADS-B reports at T = 1 and 6 seconds which is just outside the limit (2 within 5 seconds) for establishing an Extended Hybrid Surveillance track. (§2.2.4.6.4.2.2.1.2).

- Intruder 2 will provide ADS-B reports at T= 1, 2 and 3 which meets the criteria for Extended Hybrid Surveillance, however, the reported altitude in the reports is not within 500 feet. (§2.2.4.6.4.2.2.1.2).
- Intruder 3 will provide ADS-B reports at T= 1 and 2 which meets the criteria for Extended Hybrid Surveillance, however, the Q-bit values in the reports are different. (§2.2.4.6.4.2.2.1.2).
- Intruder 4 will provide ADS-B reports at T= 1, 2, 3, and 4 which meet the criteria for Extended Hybrid Surveillance, however, the ICAO 24-bit Aircraft Address in the reports are different (1st and 2nd are different valid addresses, 3rd is all ones, 4th is all zeros.) (§2.2.4.6.4.2.2.1.2).
- Intruder 5 will meet all criteria for Extended Hybrid Surveillance (ADS-B reports at T= 1 and 5 that have same altitude, same Q-bit value and same valid ICAO 24-bit Aircraft Address. (§2.2.4.6.4.2.2.1.2).

ACAS X Aircraft

Altitude	= 12,000 ft
Altitude Rate	= 0 FPM
Position	= Sydney

Intruder Aircraft #1

Altitude	= 17,000 ft
Altitude Rate	= 0 FPM
Range	= 3.2 NM at T=0 sec
Relative Speed	= 0 kt
ADS-B Version Number	= 2
ADS-B report at T= 1 and 6 seconds	
At T=10 the intruder is terminated.	

Intruder Aircraft #2

Altitude	= 17,200 ft
Altitude Rate	= 0 FPM
Range	= 3.4 NM at T=0 sec
Relative Speed	= 0 kt
ADS-B Version Number	= 2
ADS-B report at T= 1 (Altitude = 17,200), 2 (Altitude = 17725) and 3 (Altitude = 16,675)	
At T=10 the intruder is terminated.	

Intruder Aircraft #3

Altitude	= 17,400 ft
Altitude Rate	= 0 FPM
Range	= 3.6 NM at T=0 sec
Relative Speed	= 0 kt
ADS-B Version Number	= 2
ADS-B report at T= 1 (Q-bit = 0), and 2 (Q-bit = 1)	
At T=10 the intruder is terminated.	

Intruder Aircraft #4

Altitude	= 17,600 ft
Altitude Rate	= 0 FPM
Range	= 3.8 NM at T=0 sec
Relative Speed	= 0 kt
ADS-B Version Number	= 2

ADS-B report at T= 1, 2, 3, 4 and 5 seconds with ICAO 24-bit Aircraft Address = assigned address at T=1, Assigned Address + 1 at T=2, all zeros at T=3, all ones at T=4. At T=10 the intruder is terminated.

Intruder Aircraft #5

Altitude = 17,800 ft
Altitude Rate = 0 FPM
Range = 4.0 NM at T=0 sec
Relative Speed = 0 kt
ADS-B Version Number=2
ADS-B report at T= 1 and 5 seconds.
At T=10 the intruder is terminated.

Success Criteria

Intruders 1, 2, 3 and 4

The intruder is not established as an Extended Hybrid Surveillance track.

Intruder 5

The Intruder is established as an Extended Hybrid Surveillance track after reception of the 2nd ADS-B report.

Test 5 (track using extended Hybrid Surveillance based on range/altitude criteria §2.2.4.6.4.2.3.2.3)

This test verifies that when a track meets the range/altitude conditions for Extended Hybrid Surveillance it will be passively tracked without validation. Two intruder aircraft will be simulated. One will qualify for Extended Hybrid Surveillance based on altitude and the other will qualify based on range.

(The following tests may be performed using ADS-B reports or directly decoded ADS-B messages. TIS-B and ADS-R data is not permitted.)

Scenario Description

- Intruder 1 shows that when an intruder meets all conditions for Extended Hybrid Surveillance and is within the range but not the altitude criteria of the active surveillance region it is tracked passively without validation. (§2.2.4.6.4.2.2.1; §2.2.4.6.4.2.2.1.1; §2.2.4.6.4.2.2.1.2, §2.2.4.6.4.2.2.1.3, §2.2.4.6.4.2.3.2.3).
- Intruder 2 shows that when an intruder meets all conditions for Extended Hybrid Surveillance and is within the altitude but not the range criteria of the active surveillance region it is tracked passively without validation. (§2.2.4.6.4.2.2.1; §2.2.4.6.4.2.2.1.1; §2.2.4.6.4.2.2.1.2, §2.2.4.6.4.2.2.1.3, §2.2.4.6.4.2.3.2.3).

ACAS X Aircraft

Altitude = 12,000 ft
Altitude Rate = 0 FPM
Position = Sydney

Intruder Aircraft #1

Altitude = 16,600 ft
Altitude Rate = 0 FPM
Range = 2.8 NM at T=0 sec
Relative Speed = 0 kt

ADS-B Version Number=2
At T=100 the intruder is terminated.

Intruder Aircraft #2

Altitude = 12,000 ft
Altitude Rate = 0 FPM
Range = 3.2 NM at T=0 sec
Relative Speed = 0 kt
ADS-B Version Number=2
At T=100 the intruder is terminated.

Success Criteria

Intruder 1 and 2

Verify that the intruder is provided in the STM display output for the full duration of the test.

No validation/revalidation interrogations are transmitted to the intruders.

Test 6 (Validation required due to Ownship Horizontal Position Uncertainty)

§2.2.4.6.4.2.2.1, §2.2.4.6.4.2.2.1.1)

This test verifies that an intruder that would otherwise qualify for Extended Hybrid Surveillance will require validation when the ownship traffic quality requirement (Ownship Horizontal Position Uncertainty (95%) is < 0.1 NM) is not met. (§2.2.4.6.4.2.2.1; §2.2.4.6.4.2.2.1.1).

(The following tests may be performed using ADS-B reports or directly decoded ADS-B messages. TIS-B and ADS-R data is not permitted.)

ACAS X Aircraft

Altitude = 12,000 ft
Altitude Rate = 0 FPM
Position = Sydney
Ownship Horizontal Position Uncertainty (95%) is \geq 0.1 NM

Intruder Aircraft #1

Altitude = 13,000 ft
Altitude Rate = 0 FPM
Range = 3.2 NM at T=0 sec
Relative Speed = 0 kt
ADS-B Version Number = 2
At T=40 the intruder is terminated.

Success Criteria

Intruder 1

Verify that the intruder is provided in the STM display output for the full duration of the test.

Verify that validation/revalidation interrogations are transmitted to the intruder every 10 seconds.

Test 7 (Validation required due to Ownship Horizontal Position Integrity)**§2.2.4.6.4.2.2.1, §2.2.4.6.4.2.2.1.1)**

This test verifies that an intruder that would otherwise qualify for Extended Hybrid Surveillance will require validation when the ownship traffic quality requirement (Ownship Horizontal Position Integrity bounds is < 0.6 with an integrity level of $1e^{-7}$) is not met. (§2.2.4.6.4.2.2.1; §2.2.4.6.4.2.2.1.1).

(The following tests may be performed using ADS-B reports or directly decoded ADS-B messages. TIS-B and ADS-R data is not permitted.)

ACAS X Aircraft

Altitude = 12,000 ft

Altitude Rate = 0 FPM

Position = Sydney

Ownship Horizontal Position Integrity bounds is ≥ 0.6 with an integrity level of $1e^{-7}$

Intruder Aircraft #1

Altitude = 13,000 ft

Altitude Rate = 0 FPM

Range = 3.2 NM at T=0 sec

Relative Speed = 0 kt

ADS-B Version Number = 2

At T=40 the intruder is terminated.

Success Criteria**Intruder 1**

Verify that the intruder is provided in the STM display output for the full duration of the test.

Verify that validation/revalidation interrogations are transmitted to the intruder every 10 seconds.

Test 8 (Validation required due to Intruder Aircraft Traffic Quality or Signal Level Requirements §2.2.4.6.4.2.2.1, §2.2.4.6.4.2.2.1.1, §2.2.4.6.4.2.2.1.3)

This test verifies that when one of the Intruder Aircraft Traffic Quality requirements for Extended Hybrid Surveillance is not met the Hybrid Surveillance track will require validation.

(The following tests may be performed using ADS-B reports or directly decoded ADS-B messages. TIS-B and ADS-R data is not permitted.)

Scenario Description

- Intruder 1 shows that when an intruder meets all conditions for Extended Hybrid Surveillance except for Traffic Quality Requirements (ADS-B Version Number) it will be tracked passively with validation (§2.2.4.6.4.2.2.1.1).

- Intruder 2 shows that when an intruder meets all conditions for Extended Hybrid Surveillance except for Traffic Quality Requirements (Reported NIC) it will be tracked passively with validation (§2.2.4.6.4.2.2.1.1).
- Intruder 3 shows that when an intruder meets all conditions for Extended Hybrid Surveillance except for Traffic Quality Requirements (Reported NACp) it will be tracked passively with validation (§2.2.4.6.4.2.2.1.1).
- Intruder 4 shows that when an intruder meets all conditions for Extended Hybrid Surveillance except for Traffic Quality Requirements (Reported SIL≠3) it will be tracked passively with validation (§2.2.4.6.4.2.2.1.1).
- Intruder 5 shows what when an intruder meets all conditions for Extended Hybrid Surveillance except for Traffic Quality Requirements (Reported SDA ≠2 or 3) it will be tracked passively with validation (§2.2.4.6.4.2.2.1.1).
- Intruder 6 shows that when an intruder meets all conditions for Extended Hybrid Surveillance except for Traffic Quality Requirements (Invalid Barometric Altitude) it will be tracked passively with validation (§2.2.4.6.4.2.2.1.1).
- Intruder 7 shows that when an intruder meets all conditions for Extended Hybrid Surveillance (Extended Hybrid Surveillance MTL) it will be tracked passively with validation (§2.2.4.6.4.2.2.1.3).

ACAS X Aircraft

Altitude	= 12,000 ft
Altitude Rate	= 0 FPM
Position	= Sydney

Intruder Aircraft #1

Altitude	= 13,000 ft
Altitude Rate	= 0 FPM
Range	= 3.2 NM at T=0 sec
Relative Speed	= 0 kt
ADS-B Version Number	= 2
At T=40 the intruder is terminated.	

Intruder Aircraft #2

Altitude	= 13,200 ft
Altitude Rate	= 0 FPM
Range	= 3.4 NM at T=0 sec
Relative Speed	= 0 kt
ADS-B Version Number	= 2
Reported NIC	< 6
At T=40 the intruder is terminated.	

Intruder Aircraft #3

Altitude	= 13,400 ft
Altitude Rate	= 0 FPM
Range	= 3.6 NM at T=0 sec
Relative Speed	= 0 kt
ADS-B Version Number	= 2
Reported NACp	< 7
At T=40 the intruder is terminated.	

Intruder Aircraft #4

Altitude	= 13,600 ft
Altitude Rate	= 0 FPM

Range = 3.8 NM at T=0 sec
Relative Speed = 0 kt
ADS-B Version Number=2
Reported SIL ≠ 3
At T=40 the intruder is terminated.

Intruder Aircraft #5

Altitude = 13,800 ft
Altitude Rate = 0 FPM
Range = 4.0 NM at T=0 sec
Relative Speed = 0 kt
ADS-B Version Number=2
Reported SDA ≠ 2 or 3
At T=40 the intruder is terminated.

Intruder Aircraft #6

Altitude = 14,000 ft
Altitude Rate = 0 FPM
Range = 4.2 NM at T=0 sec
Relative Speed = 0 kt
ADS-B Version Number=2
Invalid Barometric Altitude
At T=100 the intruder is terminated.

Intruder Aircraft #7

Altitude = 14,200 ft
Altitude Rate = 0 FPM
Range = 4.4 NM at T=0 sec
Relative Speed = 0 kt
ADS-B Version Number=2
At T=40 the intruder is terminated.

Success Criteria

Intruder 1 through 7

Verify that the intruder is provided in the STM display output for the full duration of the test.

Verify that validation/revalidation interrogations are transmitted to each of the intruders every 10 seconds.

Test 9 (Validation required due to Signal Level Requirements based Interference Limiting MTL §2.2.4.6.4.2.2.1, §2.2.4.6.4.2.2.1.1, §2.2.4.6.4.2.2.1.3)

This test verifies that when interference limiting is invoked the system correctly uses the interference limiting MTL to determine whether an intruder aircraft should be tracked using Extended Hybrid Surveillance.

(The manufacturer will devise a test that demonstrates that when interference limiting is in effect and the interference limiting MTL is greater than the Extended Hybrid Surveillance MTL the system uses the interference limiting MTL to determine whether validation/revalidation should be performed.)

Scenario Description

Intruder Simulation begins once the interference limiting criteria are established.

- Intruder 1 shows that when an intruder meets all conditions for Extended Hybrid Surveillance except for the Intruder Signal Strength > IL MTL (extended Interference Limiting MTL) it will be tracked passively with validation (§2.2.4.6.4.2.2.1.3).
- Intruder 2 shows that when an intruder meets all conditions for Extended Hybrid Surveillance and the Intruder Signal Strength < IL MTL (extended Interference Limiting MTL) it will be tracked passively without validation (§2.2.4.6.4.2.2.1.3).

ACAS X Aircraft

Altitude = 12,000 ft

Altitude Rate = 0 FPM

Position = Sydney

Interference Limiting in effect such that MTL is raised to \geq -64 dBm

Intruder Aircraft #1

Altitude = 13,000 ft

Altitude Rate = 0 FPM

Range = 3.2 NM at T=0 sec

Relative Speed = 0 kt

Reply Power = -62 dBm

ADS-B Version Number = 2

At T=40 the intruder is terminated.

Intruder Aircraft #2

Altitude = 13,200 ft

Altitude Rate = 0 FPM

Range = 3.4 NM at T=0 sec

Relative Speed = 0 kt

Reply Power = -66 dBm

ADS-B Version Number = 2

At T=40 the intruder is terminated.

Success Criteria

Intruder 1

Verify that the intruder is provided in the STM display output for the full duration of the test.

Verify that validation/revalidation interrogations are transmitted to the intruder every 10 seconds.

Intruder 2

Verify that the intruder is provided in the STM display output for the full duration of the test.

Verify that validation/revalidation interrogations are not transmitted to the intruder.

Test 10 (Track Using Extended Hybrid Surveillance when on Surface)

§2.2.4.6.4.2.2.1, §2.2.4.6.4.2.3.1.5, §2.2.4.6.3)

This test verifies that an intruder that would otherwise be tracked using hybrid or active surveillance will be tracked using Extended Hybrid Surveillance (no validation) when ownship is on the surface. (§2.2.4.6.4.2.2.1, §2.2.4.6.4.2.3.1.5, §2.2.4.6.3).

(The following tests may be performed using ADS-B reports or directly decoded ADS-B messages. TIS-B and ADS-R data is not permitted.)

- Intruder 1 shows that when an intruder meets the conditions for Extended Hybrid Surveillance and is within both the range and altitude criteria of the active surveillance region it is tracked passively without validation as long as ownship is on the surface. (§2.2.4.6.4.2.2.1; §2.2.4.6.4.2.3.1.5; §2.2.4.6.3).

ACAS X Aircraft

Altitude = 12,000 ft
Altitude Rate = 0 FPM
Position = Sydney
From T= 0 – 30
 On Surface – Radio Altitude = 40 feet
 AND Ground Speed = 20 knots
From T= 31 – 60
 Airborne/Taking Off- Radio Altitude = 0 feet
 AND Ground Speed = 40 knots
From T= 61 – 90
 On Surface – Radio Altitude = 40 feet
 AND Ground Speed = 20 knots
From T= 91 – 120
 Airborne/Taking Off- Radio Altitude = 55 feet
 AND Ground Speed = 0 knots

Intruder Aircraft #1

Altitude = 12,000 ft
Altitude Rate = 0 FPM
Range = 0.2 NM at T=0 sec
Relative Speed = 0 kt (-0.03 NM/s)
ADS-B Version Number=2
At T=120 the intruder is terminated.

Success Criteria

Intruder 1

Verify that the intruder is provided in the STM display output for the full duration of the test.

Verify that no validation/revalidation interrogations are transmitted to the intruder from T=0 to 30 and T = 66 to 90 seconds.

Verify that from T=36 to T=60, and from T=96 to T=120, DF=0 replies are provided to the STM with surv_mode = 0/Norm.

Test 11 (Intruder Revalidation Rate §2.2.4.6.4.2.3.1.10)

This test verifies the revalidation rate based on intruder range and range rate (§2.2.4.6.4.2.3.1.10).

(The following tests may be performed using ADS-B reports or directly decoded ADS-B messages. TIS-B and ADS-R data is not permitted.)

Scenario Description

- Intruder 1 - 10 show that when an intruder is within the altitude but not the range criteria for active surveillance it will be tracked using Hybrid Surveillance with a variable revalidation rate according to the requirements in (§2.2.4.6.4.2.3.1.10).
- Intruder 11 shows that when an intruder does not meet the altitude criteria for active surveillance but is within the range criteria it will be tracked using Hybrid Surveillance with a 60 second revalidation rate (§2.2.4.6.4.2.3.1.10).

ACAS X Aircraft

Altitude = 12,000 ft
 Altitude Rate = 0 FPM
 Position = Sydney

Intruder Aircraft #1

Altitude = 12,000 ft
 Altitude Rate = 0 FPM
 Range = 13 NM
 Relative Speed = 0 kt
 At T=100 the intruder is terminated.

Intruder Aircraft #2

Altitude = 12,000 ft
 Altitude Rate = 0 FPM
 Range = 12 NM
 Relative Speed = 0 kt
 At T=100 the intruder is terminated.

Intruder Aircraft #3

Altitude = 12,000 ft
 Altitude Rate = 0 FPM
 Range = 9 NM
 Relative Speed = 0 kt
 At T=100 the intruder is terminated.

Intruder Aircraft #4

Altitude = 12,000 ft
 Altitude Rate = 0 FPM
 Range = 8 NM
 Relative Speed = 0 kt
 At T=100 the intruder is terminated.

Intruder Aircraft #5

Altitude = 12,000 ft
 Altitude Rate = 0 FPM
 Range = 7 NM
 Relative Speed = 0 kt
 At T=100 the intruder is terminated.

Intruder Aircraft #6

Altitude = 12,000 ft
Altitude Rate = 0 FPM
Range = 6 NM
Relative Speed = 0 kt
At T=100 the intruder is terminated.

Intruder Aircraft #7

Altitude = 12,000 ft
Altitude Rate = 0 FPM
Range = 5 NM
Relative Speed = 0 kt
At T=100 the intruder is terminated.

Intruder Aircraft #8

Altitude = 12,000 ft
Altitude Rate = 0 FPM
Range = 34 NM
Relative Speed = -1200 kt
At T=100 the intruder is terminated.

Intruder Aircraft #9

Altitude = 12,000 ft
Altitude Rate = 0 FPM
Range = 30 NM
Relative Speed = -600 kt
At T=100 the intruder is terminated.

Intruder Aircraft #10

Altitude = 12,000 ft
Altitude Rate = 0 FPM
Range = 3.4 NM
Relative Speed = 300 kt
At T=100 the intruder is terminated.

Intruder Aircraft #11

Altitude = 17,000 ft
Altitude Rate = 0 FPM
Range = 2.8 NM
Relative Speed = 0 kt
At T=100 the intruder is terminated.

Success Criteria

For the tests in this section the revalidation rate for each applicable success criteria was identified using the table in §2.2.4.6.4.2.3.1.10. If the implementation uses the equation method then the revalidation interval can be longer by 10 to 20 seconds. Care should be taken to verify that the success criteria matches the value expected based on the implementation.

For each intruder:

Verify that DF=0 replies are provided to the STM with surv_mode = 2/HS.

Verify that the intruder is provided in the STM display output for the full duration of the test.

Intruder 1

Verify that revalidation interrogations are transmitted every 60 seconds.

Intruder 2

Verify that revalidation interrogations are transmitted every 40 seconds.

Intruder 3

Verify that revalidation interrogations are transmitted every 40 seconds.

Intruder 4

Verify that revalidation interrogations are transmitted every 30 seconds.

Intruder 5

Verify that revalidation interrogations are transmitted every 20 seconds.

Intruder 6

Verify that revalidation interrogations are transmitted every 20 seconds.

Intruder 7

Verify that revalidation interrogations are transmitted every 10 seconds.

Intruder 8

Verify that the first revalidation interrogation is transmitted 20 seconds after initial validation and then every 10 seconds until the intruder transitions to active tracking at 20 NM.

Intruder 9

Verify that the first revalidation interrogation is transmitted 40 seconds after initial validation, the 2nd at 30 seconds after the 1st revalidation and the 3rd at 20 seconds after the 2nd revalidation.

Intruder 10

Verify that revalidation interrogations are transmitted every 60 seconds.

Intruder 11

Verify that revalidation interrogations are transmitted every 60 seconds.

Test 11a (Intruder Revalidation Rate when ownship is operating on the surface
§2.2.4.6.4.2.3.1.10

This test verifies the revalidation rate when ownship is operating on the surface based on the altitude and range criteria for active tracking (§2.2.4.6.4.2.3.1.10).

(The following tests may be performed using ADS-B reports or directly decoded ADS-B messages. TIS-B and ADS-R data is not permitted.)

Scenario Description

- Intruder 1 shows that when ownship is operating on the airport surface and an intruder is within the altitude and range criteria for active surveillance it will be tracked using Hybrid Surveillance with a 10 second revalidation rate (§2.2.4.6.4.2.3.1.10).
- Intruder 2 shows that when ownship is operating on the airport surface and an intruder is within the altitude but not the range criteria for active surveillance it will be tracked using Hybrid Surveillance with a variable revalidation rate according to the requirements in (§2.2.4.6.4.2.3.1.10).

ACAS X Aircraft

Altitude = 0 ft (Ground Level)
 Altitude Rate = 0 FPM
 Position = Sydney
 Radio altitude input = 0 ft

Ground Speed is valid and at 0 knots and ACAS X Air/Ground (own.on_ground) indicates on-ground.

Intruder Aircraft #1

Altitude = 2,000 ft
 Altitude Rate = 0 FPM
 Range = 2 NM
 Relative Speed = 0 kt

At T=100 the intruder is terminated.

Intruder Aircraft #2

Altitude = 2,000 ft
 Altitude Rate = 0 FPM
 Range = 8 NM
 Relative Speed = 0 kt

At T=100 the intruder is terminated.

Success Criteria

For the tests in this section the revalidation rate for each applicable success criteria was identified using the table in §2.2.4.6.4.2.3.1.10. If the implementation uses the equation method then the revalidation interval can be longer by 10 to 20 seconds. Care should be taken to verify that the success criteria matches the value expected based on the implementation.

For each intruder:

Verify that DF=0 replies are provided to the STM with surv_mode = 2/HS.
 Verify that the intruder is provided in the STM display output for the full duration of the test.

Intruder 1

Verify that revalidation interrogations are transmitted every 10 seconds.

Intruder 2

Verify that revalidation interrogations are transmitted every 30 seconds.

The revalidation rate for each applicable success criteria was identified using the table in §2.2.4.6.4.2.3.1.10. If the implementation uses the equation method then the revalidation interval can be longer by 10 to 20 seconds. Care should be taken to verify that the success criteria matches the value expected based on the implementation.

Test 12 (Signal Strength Estimation (§2.2.4.6.4.2.2.1.4))

This test verifies the signal strength is estimated from DF=11 or DF=17 squitters every surveillance processing cycle and is set to the maximum signal strength of the squitters received since the last estimate (§2.2.4.6.4.2.2.1.4).

(The following tests may be performed using ADS-B reports or directly decoded ADS-B messages. TIS-B and ADS-R data is not permitted.)

Scenario Description

- Intruder 1 will simulate DF=11 and DF=17 squitters at signal strength < -68 dBm. While all of the squitters are < -68 dBm the track is maintained in Extended Hybrid Surveillance. When the power of one of the DF=17 squitters is raised to > -68 dBm the track will transition to Hybrid Surveillance (§2.2.4.6.4.2.2.1.4).
- Intruder 2 will simulate DF=11 and DF=17 squitters at signal strength < -68 dBm. While all of the squitters are < -68 dBm the track is maintained in Extended Hybrid Surveillance. When the power of the DF=11 squitter is raised to > -68 dBm the track will transition to Hybrid Surveillance (§2.2.4.6.4.2.2.1.4).

ACAS X Aircraft

Altitude	= 12,000 ft
Altitude Rate	= 0 FPM
Position	= Sydney

Intruder Aircraft #1

Altitude	= 17,000 ft
Altitude Rate	= 0 FPM
Range	= 3.2 NM
Relative Speed	= 0 kt

ADS-B Version Number=2

At T=1-20 the intruder sends at least 1 DF=11 and 2 DF=17 airborne position squitters per second all with power of -70 dBm. At T=21 the transmission power of one of the DF=17 squitters will be raised to -66 dBm.

At T=25 the intruder is terminated.

Intruder Aircraft #2

Altitude	= 17,200 ft
Altitude Rate	= 0 FPM
Range	= 4.0 NM

Relative Speed = 0 kt

At T=1-20 the intruder sends at least 1 DF=11 and 2 DF=17 airborne position squitters per second all with power of -70 dBm.

At T=21 the transmission power of one of the DF=17 squitters will be raised to -66 dBm.
At T=25 the intruder is terminated.

Success Criteria

Intruder 1 and 2

Verify that the intruder is provided in the STM display output for the full duration of the test.

Verify that from T = 0 to T = 20, DF=0 replies are not provided to the STM.

Verify that a validation interrogation is transmitted after T=20 seconds.

Verify that after T = 20, DF=0 replies are provided to the STM with surv_mode = 2/HS.

Test 13 (Validation of Airborne Position §2.2.4.6.4.2.3.2.7)

This test verifies that ADS-B airborne position is correctly validated (§2.2.4.6.4.2.3.2.7).

(The following tests may be performed using ADS-B reports or directly decoded ADS-B messages. TIS-B and ADS-R data is not permitted.)

Scenario Description

- Intruders 1 and 2 will qualify for Hybrid Surveillance and require validation.
- Intruder 1 will not reply to validation interrogations.
- Intruder 2 will pass the validation even though bearing is not available (§2.2.4.6.4.2.3.2.7).

ACAS X Aircraft

Altitude = 12,000 ft

Altitude Rate = 0 FPM

Position = Sydney

Intruder Aircraft #1

Altitude = 12,400 ft

Altitude Rate = 0 FPM

Range = 3.6 NM

Relative Speed = 0 kt

Bearing = 0 degrees

DF=0 replies are not transmitted.

At T=10 the intruder is terminated.

Intruder Aircraft #2

Altitude = 12,600 ft
 Altitude Rate = 0 FPM
 Range = 3.8 NM
 Relative Speed = 0 kt
 Bearing = 0 degrees
 DF=0 replies are transmitted with:
 slant range = 3.94 NM
 altitude = 12,700 feet
 bearing = none available.

At T=10 the intruder is terminated.

Success Criteria

Intruder 1

Verify that validation/acquisition interrogations are transmitted to the intruder according to the interrogation rate limits implemented per requirements of §2.2.4.6.4.2.2 (Acquisition)

Verify that a track is not initiated for the intruder.

Intruder 2

Verify that a single validation interrogation is transmitted to the intruder.

Verify that the intruder is provided in the STM display output for the full duration of the test.

Test 14 (Revalidation of Airborne Position §2.2.4.6.4.2.3.1.10)

This test verifies that ADS-B airborne position is correctly revalidated (§2.2.4.6.4.2.3.1.10).

(The following tests may be performed using ADS-B reports or directly decoded ADS-B messages. TIS-B and ADS-R data is not permitted.)

Scenario Description

- Intruders 1 and 2 will qualify for Hybrid Surveillance and pass initial validation. Revalidation will be attempted 10 seconds later.
- Intruder 1 will not reply to revalidation interrogations.
- Intruder 2 will pass the revalidation even though bearing is not available (§2.2.4.6.4.2.3.1.10).

ACAS X Aircraft

Altitude = 12,000 ft
 Altitude Rate = 0 FPM
 Position = Sydney

Intruder Aircraft #1

Altitude = 12,400 ft
 Altitude Rate = 0 FPM
 Range = 3.6 NM
 Relative Speed = 0 kt
 Bearing = 0 degrees

DF=0 replies (after initial validation) are not transmitted.
At T=20 the intruder is terminated.

Intruder Aircraft #2

Altitude	= 12,600 ft
Altitude Rate	= 0 FPM
Range	= 3.8 NM
Relative Speed	= 0 kt
Bearing	= 0 degrees

DF=0 replies (after initial validation) are transmitted with:

slant range = 3.97 NM (note the range was selected to be just inside the range revalidation window of 340 m ... 3.8 +0.185)
altitude = 12,700 feet
bearing = None available.

At T=20 the intruder is terminated.

Success Criteria

Intruder 1

Verify that a single validation interrogation is transmitted to the intruder.
Verify that the intruder is provided in the STM display output for the full duration of the test.
Verify that, beginning at 10 seconds after validation, revalidation interrogations are transmitted to the intruder according to the interrogation rate limits implemented per requirements of §2.2.4.6.4.2.2 (Acquisition Using Interrogations).
Verify that, beginning at 17 seconds after validation, no further revalidation interrogations are transmitted to the intruder.

Intruder 2

Verify that a single validation interrogation is transmitted to the intruder.
Verify that, at 10 seconds after validation, a single revalidation interrogation is transmitted to the intruder.
Verify that the intruder is provided in the STM display output for the full duration of the test.
Verify that the DF=0 replies are provided to the STM with surv_mode = 2/HS.

Test 15 (Maintain Extended Hybrid Surveillance Track §2.2.4.6.4.2.3.1.8,

§2.2.4.6.4.2.3.1.9)

This test verifies that a track under Extended Hybrid Surveillance will be tracked providing ADS-B position updates are received according to the requirement in (§2.2.4.6.4.2.3.1.8, §2.2.4.6.4.2.3.1.9).

(The following tests may be performed using ADS-B reports or directly decoded ADS-B messages. TIS-B and ADS-R data is not permitted.)

Scenario Description

- Intruders 1 - 3 will demonstrate that a track is maintained in Extended Hybrid Surveillance when valid ADS-B position updates are received.
- Intruder 1 demonstrates range criteria.

- Intruder 2 demonstrates altitude criteria (§2.2.4.6.4.2.3.1.8).
- Intruder 3 will demonstrate coast capability when some reports are missing (§2.2.4.6.4.2.3.1.9).

ACAS X Aircraft

Altitude = 12,000 ft
 Altitude Rate = 0 FPM
 Position = Sydney

Intruder Aircraft #1

Altitude = 17,000 ft
 Altitude Rate = 0 FPM

ADS-B Version Number=2

The range and relative speed of the intruder will vary to demonstrate that the ADS-B reports are used to update the track when they fall within the appropriate range window. The range will be maintained such that the intruder remains in Extended Hybrid Surveillance.

At T=60 the intruder is terminated.

Intruder Aircraft #2

The Altitude and Altitude Rate of the intruder will vary to demonstrate that the ADS-B reports are used to update the track when they fall within the appropriate altitude window. The altitude will be maintained such that the intruder remains in Extended Hybrid.

Range = 3.4 NM

Relative Speed = 0 kt

ADS-B Version Number=2

At T=60 the intruder is terminated.

Intruder Aircraft #3

Altitude = 17,000 ft

Altitude Rate = 0 FPM

Range = 3.4 NM

Relative Speed = 0 kt

ADS-B Version Number=2

The intruder will provide a valid ADS-B position report every 5 seconds from T=0 through T=45.

The intruder will provide an ADS-B report with altitude outside of the altitude window twice every 5 seconds.

The intruder will provide an ADS-B report with range outside of the range window twice every 5 seconds.

At T=60 the intruder is terminated.

Success Criteria

Intruder 1 and 2

Verify that the intruder is provided in the STM display output for the full duration of the test.

Verify that no validation/revalidation interrogations are transmitted.

Verify the intruder position is updated every second (no coasting).

Intruder 3

Verify that the intruder is provided in the STM display output for the full duration of the test.

Verify that no validation/revalidation interrogations are transmitted.

Verify that after T=45 the track is coasted and dropped.

2.4.2.11.5 Verification of Requirements Related to Transitions Between Passive and Active Surveillance

Test 1 – Hybrid to Active

This test shows correct transitions to active surveillance based on the criteria of §2.2.4.6.4.2.3.2.3.

Also the scenario tests the 4500 ft and 3 NM altitude and range offsets in the tau equations.

Scenario Description

- Intruder 1 verifies the range criteria of §2.2.4.6.4.2.3.2.3, and verifies the requirement of §2.2.4.6.4.2.3.2.4 that a track not be dropped for a transition from passive to active surveillance when the passive data correlates well with active data.
- Intruder 2 verifies the altitude criteria of §2.2.4.6.4.2.3.2.3 when the aircraft is above and descending. Also, verifies the 60 second revalidation interval for intruders, §2.2.4.6.4.2.3.1.10.
- Intruder 3 verifies the altitude criteria of §2.2.4.6.4.2.3.2.3 when the aircraft is below and climbing. Also, verifies the 60 second revalidation interval for intruders, §2.2.4.6.4.2.3.1.10.
- Intruder 4 verifies that when tracking the intruder with passive reports and the intruder is accelerating rapidly towards ownship in altitude the transition to active occurs in a timely manner (§2.2.4.6.4.2.3.2.3).

ACAS X Aircraft

Altitude = 12,000 ft
Altitude Rate = 0 FPM
Position = Frankfurt

Intruder Aircraft #1

Altitude = 7600 ft
Altitude Rate = 0 FPM
Range = 11 NM at T=0 sec
Range Error of 330 m inserted after T=10 through end of test.
Relative Speed = -360 kt (-0.1 NM/sec)
At T=40 the intruder is terminated.

Intruder Aircraft #2

Altitude = 20,250 ft at T=0
Altitude Rate = -1,500 FPM
Range = 2.9 NM
Relative Speed = 0 kt
At T=105 the intruder is terminated.

Intruder Aircraft #3

Altitude = 3,750 ft at T=0
 Altitude Rate = 1,500 FPM
 Range = 2.9 NM
 Relative Speed = 0 kt
 At T=105 the intruder is terminated.

Intruder Aircraft #4

Altitude = 17,500 ft at T=0
 Altitude Rate = 0 FPM at T=0
 Altitude Rate = at T=15 descend rate goes to -6000 FPM at 0.5 g.
 Range = 2.9 NM
 Relative Speed = 0 kt
 At T=40 the intruder is terminated.

Success Criteria

Intruder 1

From T=10 to T=19, verify that DF=0 replies are provided to the STM with surv_mode = 2/HS.

From T=10 to T=19, verify that no more than 1 UF=0 interrogation every 10 sec (after the track has transitioned to passive).

Interrogation interval is no greater than 5 sec from T=21 to T=40.

From T=21 to the end of the test, verify that DF=0 replies are provided to the STM with surv_mode = 0/Norm or 1/Red.

Verify that the intruder is provided in the STM display output for the full duration of the test.

Intruder 2

From T=10 to T=85, verify that DF=0 replies are provided to the STM with surv_mode = 2/HS.

From T=10 to T=85, verify that no more than 1 UF=0 interrogation every 60 sec (after the track has transitioned to Hybrid Surveillance.)

One interrogation every sec of intruder aircraft from T=95 to T=105.

From T=95 to the end of the test, verify that DF=0 replies are provided to the STM with surv_mode = 0/Norm.

Verify that the intruder is provided in the STM display output for the full duration of the test.

Intruder 3

From T=10 to T=85, verify that DF=0 replies are provided to the STM with surv_mode = 2/HS.

From T=10 to T=85, verify that no more than 1 UF=0 interrogation every 60 sec (after the track has transitioned to passive).

One interrogation every sec of intruder aircraft from T=95 to T=105.

From T=95 to the end of the test, verify that DF=0 replies are provided to the STM with surv_mode = 0/Norm.

Verify that the intruder is provided in the STM display output for the full duration of the test.

Intruder 4

From T=10 to T=15, verify that DF=0 replies are provided to the STM with surv_mode = 2/HS.

From T=20 to the end of the test, verify that DF=0 replies are provided to the STM with surv_mode = 0/Norm or 1/Red.

Test 1A – Passive to Active with Accelerating Range

This test shows correct transitions to active surveillance based on the criteria of §2.2.4.6.4.2.3.2.3. Specifically tests this when there is a range acceleration while the aircraft is still being tracked passively.

Perform test §2.4.2.1.8.2.1) Range Tracking Accuracy as modified below:

This test should be modified so that the intruder is Mode S and squitters Airborne Position Messages.

Success Criteria

Verify that the track transitions to active surveillance based on the range tau.

Verify the success criteria as specified in the Range Tracking Accuracy test.

Test 2 – Active to Hybrid

This test shows correct transitions to passive surveillance based on meeting criteria of §2.2.4.6.4.2.3.1.6.

The test verifies the 4900 ft and 3.2 NM altitude and range offsets for switching to passive as well as the 4500 ft and 3 NM altitude and range offsets of §2.2.4.6.4.2.3.2.3. It also verifies the validation windows of §2.2.4.6.4.2.3.2.7 when errors are just within the validation window.

This test verifies that even after initially failing validation an intruder continues to be revalidated and will transition to Hybrid Surveillance if the validation criteria are met as required by §2.2.4.6.4.2.3.1.6. It also shows that two validation attempts are required to transition to Hybrid Surveillance if a track has failed validation as required by §2.2.4.6.4.2.3.1.7.

Scenario Description

- Intruder 1 verifies the range criteria of §2.2.4.6.4.2.3.1.6 when the intruder is outbound.
- Intruder 2 verifies the altitude criteria of §2.2.4.6.4.2.3.1.6 when the intruder is climbing.
- Intruder 3 verifies that an intruder which initially fails revalidation – can still transition to Hybrid Surveillance if it subsequently passes the validation window. It also verifies that two validations are required since the initial validation failed.

ACAS X Aircraft

Altitude = 12,000 ft
Altitude Rate = 0 FPM
Position = Dallas

Intruder Aircraft #1

Altitude = 16,400 ft at T=0
Altitude Rate = 0 FPM
Range = 1.2 NM
Range Error is 280 m.

The equipment manufacturer should set this error as close to the 290m as possible based on the capabilities of its test equipment.

Relative Speed = +360 kt (+0.1 NM/sec)

At T=40 the intruder is terminated.

Intruder Aircraft #2

Altitude = 16,400 ft at T=0

Altitude Rate = 1800 FPM

Range = 2.9 NM

Relative Speed = 0 kt

At T=30 the intruder is terminated.

Intruder Aircraft #3

Altitude = 16,000 ft

Altitude Rate = 0 FPM

Range = 20 NM at T=0 sec

Relative Speed = -360 kt (-0.1 NM/sec)

The Range Difference is 310 m.

At T=20, the Range difference is reduced to 250 m

At T=100 the intruder is terminated.

Success Criteria

Intruder 1

DF=0 replies are provided to the STM with surv_mode = 0/Norm until at least T=18.

DF=0 replies are provided to the STM with surv_mode = 2/HS as soon as T=21, but at least by T=28 and until the end of the scenario.

Verify that the intruder is provided in the STM display output for the full duration of the test.

Intruder 2

DF=0 replies are provided to the STM with surv_mode = 0/Norm until at least T=18.

DF=0 replies are provided to the STM with surv_mode = 2/HS starting no later than T=21 until the end of the scenario.

Intruder 3

DF=0 replies are provided to the STM with surv_mode = 0/Norm or 1/Red until at least after T=24. This can be accomplished by showing that interrogations are made to the intruder at least once every 5 seconds.

Verify that by T=32, DF=0 replies are provided to the STM with surv_mode = 2/HS and that the interrogation are made to the intruder no more than once every 60 seconds.

Note: A transition to Hybrid Surveillance is not permitted at T=20 because two (2) successful validation are required once a track has failed validation.

Test 3 – Hybrid to Active Abnormal Conditions

This test primarily verifies abnormal conditions which result in a transition from using passive reports to update a track to using active data.

Scenario Description

- Intruder 1 verifies the handling of momentary and permanent transitions to non-altitude reporting intruders (§2.2.4.6.4.2.3.1.9).
- Intruder 2 verifies the transition from passive to active when the range revalidation check fails (§2.2.4.6.4.2.3.2.7, §2.2.4.6.4.2.3.1.10).
- Intruder 3 verifies the transition from passive to active when the bearing revalidation check fails (§2.2.4.6.4.2.3.2.7, §2.2.4.6.4.2.3.1.10).
- Intruder 4 verifies the transition from passive to active when the altitude revalidation check fails (§2.2.4.6.4.2.3.2.7, §2.2.4.6.4.2.3.1.10).
- Intruder 5 verifies the transition from passive to active when the intruder stops reporting position (§2.2.4.6.4.2.3.1.9).
- Intruder 6 verifies the revalidation requirements (§2.2.4.6.4.2.3.1.10) and that passive data is used to determine that a track should transition to active surveillance (§2.2.4.6.4.2.3.2.3).

ACAS X Aircraft

Altitude = 12,000 ft
 Altitude Rate = 0 FPM
 Position = São Paulo

Intruder Aircraft #1

Altitude = 17,000 ft
 Altitude Rate = 0 FPM
 Range = 10 NM at T=0 sec
 Relative Speed = -144 kt (-0.04 NM/sec)
 At T=15 the aircraft stops reporting altitude for 1 sec.
 At T=20 the aircraft stops reporting altitude.
 At T=60 the intruder is terminated.

Intruder Aircraft #2

Altitude = 17,000 ft
 Altitude Rate = 0 FPM
 Range = 12 NM at T=0 sec
 Relative Speed = -360 kt (-0.1 NM/sec)
 At T=15 the Range Difference is increased to 390 m.

Note: 390m was selected instead of an error closer to 340m to account for test equipment limitations during black box testing. It is expected that software testing will verify that exact implementation of the 340m window.

At T=60 the intruder is terminated.

Intruder Aircraft #3

Altitude = 17,000 ft
 Altitude Rate = 0 FPM
 Range = 12 NM at T=0 sec
 Relative Speed = -360 kt (-0.1 NM/sec)
 At T=15 the Bearing Difference is increased to 50 degrees.
 At T=60 the intruder is terminated.

Intruder Aircraft #4

Altitude = 6500 ft
 Altitude Rate = 0 FPM

Range = 12 NM at T=0 sec
 Relative Speed = -360 kt (-0.1 NM/sec)
 At T=15 the Altitude Difference is increased to 150 ft.
 At T=60 the intruder is terminated.

Intruder Aircraft #5

Altitude = 17,000 ft
 Altitude Rate = 0 FPM
 Range = 10 NM at T=0 sec
 Relative Speed = -144 kt (-0.04 NM/sec)
 At T=15 the aircraft starts reporting invalid position, but continues to provide extended squitters.
 At T=60 the intruder is terminated.

Intruder Aircraft #6

Altitude = 16,400 ft at T=0
 Altitude Rate = 0 FPM at T=0
 Range = 10.0 NM
 Relative Speed = -144 kt at T=0
 At T=10 the active range jumps to 13 NM (moving the active range causes the active data to fail the revalidation).
 At T=120 the intruder is terminated.

Success Criteria

Intruder 1

Verify that from T=5 to T=20, DF=0 replies are provided to the STM with surv_mode = 2/HS.

Verify that from T=5 to T=20 the intruder is provided to the display with continuous valid altitude.

Verify that the intruder is provided in the STM display output for the full duration of the test.

The intruder provided to the display changed to non-altitude reporting some time between T=20 and T=33 and remains non-altitude reporting until the end of the scenario.

After T=26, DF=0 replies are provided to the STM with surv_mode = 0/Norm or 1/Red.

Intruder 2

Verify that DF=0 replies are provided to the STM with surv_mode = 2/HS.

By T=26 the intruder should be interrogated once per surveillance interval (1 sec).

By T=26 DF=0 replies are provided to the STM with surv_mode = 0/Norm.

Note: The 1 sec interrogation rate could start as early as T=16.

Intruder 3

Verify that DF=0 replies are provided to the STM with surv_mode = 2/HS.

By T=26 the intruder should be interrogated once per surveillance interval (1 sec).

By T=26 DF=0 replies are provided to the STM with surv_mode = 0/Norm.

Note: The 1 sec interrogation rate could start as early as T=16.

Intruder 4

Verify that DF=0 replies are provided to the STM with surv_mode = 2/HS.

By T=26 the intruder should be interrogated once per surveillance interval (1 sec).

By T=26 DF=0 replies are provided to the STM with surv_mode = 0/Norm.

Note: The 1 sec interrogation rate could start as early as T=16.

Intruder 5

Verify that DF=0 replies are provided to the STM with surv_mode = 2/HS.

By T=24 DF=0 replies are provided to the STM with surv_mode = 0/Norm or 1/Red.

Intruder 6

Verify that DF=0 replies are provided to the STM with surv_mode = 2/HS from T=10 to T=20.

Approximately 20 to 26 seconds after the initial validation, verify that a validation interrogation is performed. The revalidation attempt fails because the active range position does not follow the passive range position. Verify that as a result of the failed revalidation, the track transitions to active surveillance within 3 sec of the failed revalidation attempt

Note: Based on the change in active range at T=10, the revalidation time would be greater if active data was being used.

Test 3A –Passive (Extended Hybrid) to Active Abnormal Conditions

This test primarily verifies abnormal conditions which result in a transition from using passive reports to update a track to using active data. It also tests other abnormal conditions.

Scenario Description

- Intruder 1 verifies the handling of momentary and permanent transitions to non-altitude reporting intruders (§2.2.4.6.4.2.3.1.9).
- Intruder 2 verifies the requirement to transition from Extended Hybrid to Hybrid Surveillance if the intruder's reported quality does not meet the SIL requirements of §2.2.4.6.4.2.2.1.1. Also verifies variable revalidation rate of §2.2.4.6.4.2.3.1.10.
- Intruder 3 verifies the requirement to transition from Extended Hybrid to Hybrid Surveillance if the intruder's reported quality does not meet the SDA requirements of §2.2.4.6.4.2.2.1.1. Also verifies variable revalidation rate of §2.2.4.6.4.2.3.1.10.
- Intruder 4 verifies the transition from passive to active when the intruder stops reporting valid position (§2.2.4.6.4.2.3.1.9).

ACAS X Aircraft

Altitude	= 12,000 ft
Altitude Rate	= 0 FPM
Position	= São Paulo

Intruder Aircraft #1

Version	= 3
Altitude	= 17,000 ft
Altitude Rate	= 0 FPM
Range	= 10 NM at T=0 sec
Relative Speed	= -144 kt (-0.04 NM/sec)
At T=15 the aircraft stops reporting altitude for 1 sec.	
At T=20 the aircraft stops reporting altitude.	

At T=60 the intruder is terminated.

Intruder Aircraft #2

Version = 2

Altitude = 7600 ft

Altitude Rate = 0 FPM

Range = 25 NM at T=0 sec

Relative Speed = -420 kt (-0.1167 NM/sec)

At T=20 the SIL from the intruder is degraded to 1.

At T=80 the intruder is terminated.

Intruder Aircraft #3

Version = 2

Altitude = 7600 ft

Altitude Rate = 0 FPM

Range = 15 NM at T=0 sec

Relative Speed = -220 kt (-0.06 NM/sec)

At T=20 the SDA from the intruder is degraded to 1.

At T=80 the intruder is terminated.

Intruder Aircraft #4

Version = 2

Altitude = 17,000 ft

Altitude Rate = 0 FPM

Range = 10 NM at T=0 sec

Relative Speed = -144 kt (-0.04 NM/sec)

At T=15 the aircraft starts reporting invalid horizontal position (i.e. latitude and longitude fields encoded with zeros) but continues to provide extended squitters.

At T=60 the intruder is terminated.

Success Criteria

Intruder 1

From T=0 to T=20, verify that DF=0 replies are not provided to the STM.

Verify that no interrogations are addressed to this intruder from T=0 to T=20.

Verify that the intruder is provided in the STM display output for the full duration of the test.

Verify that the intruder provided to the display changed to non-altitude reporting some time between T=20 and T=33 and remains non-altitude reporting until the end of the scenario.

Verify after T=26, DF=0 replies are provided to the STM.

Intruder 2

From T=1 to T=19, verify that DF=0 replies are not provided to the STM.

From T=20 to T=80, verify that DF=0 replies are provided to the STM with surv_mode = 2/HS.

From T=1 to T=19, verify that no UF=0 interrogation are made to this intruder.

From T=20 to 25, verify that only one UF=0 interrogation is made to this intruder because of the transition to Hybrid Surveillance.

From T=25 to 80, verify that only one more additional interrogation is made to the intruder between T=49 and T=59 (nominally T=54).

Verify that the intruder is provided in the STM display output for the full duration of the test.

Intruder 3

From T=1 to T=19, verify that DF=0 replies are not provided to the STM.

From T=20 to T=80, verify that DF=0 replies are provided to the STM with surv_mode = 2/HS.

From T=1 to T=19, verify that no UF=0 interrogation are made to this intruder.

From T=20 to 25, verify that only one UF=0 interrogation is made to this intruder because of the transition to Hybrid Surveillance.

From T=25 to 80, verify that only one more additional interrogation is made to the intruder between T=43 and T=53 (nominally T=48).

Verify that the intruder is provided in the STM display output for the full duration of the test.

Intruder 4

From T=1 to T=19, verify that DF=0 replies are not provided to the STM.

By T=24, verify that DF=0 replies are provided to the STM with surv_mode = 0/Norm or 1/Red.

Test 4 – Passive (Extended Hybrid) to Active or Hybrid Surveillance

This test shows correct transitions to active surveillance as required by §2.2.4.6.4.2.3.1.2 based on the criteria of §2.2.4.6.4.2.3.2.3.

Also the scenario tests the 4500 ft and 3 NM altitude and range offsets in the tau equations. The test shows correct transitions from Hybrid to Extended Hybrid Surveillance as required by §2.2.4.6.4.2.3.1.3.

The test shows correct validation / revalidation rates for the case where an intruder transitions between Extended Hybrid Surveillance and Hybrid Surveillance more than once per §2.2.4.6.4.2.3.1.2.

Scenario Description

- Intruder 1 verifies the range criteria of §2.2.4.6.4.2.3.2.3, and verifies the requirement of §2.2.4.6.4.2.3.2.4 that a track not be dropped for a transition from passive to active surveillance.
- Intruder 2 verifies the altitude criteria of §2.2.4.6.4.2.3.2.3 when the aircraft is above and descending.
- Intruder 3 verifies the altitude criteria of §2.2.4.6.4.2.3.2.3 when the aircraft is below and climbing.
- Intruder 4 verifies that when tracking the intruder with passive reports and the intruder is accelerating rapidly towards ownship in altitude the transition to active occurs in a timely manner (§2.2.4.6.4.2.3.2.3).
- Intruder 5 verifies the requirement to transition from Extended Hybrid to Hybrid Surveillance if the intruder's reported quality does not meet the NACp requirements of §2.2.4.6.4.2.2.1.1. Also verifies variable revalidation rate of §2.2.4.6.4.2.3.1.10.
- Intruder 6 verifies the requirement to transition from Extended Hybrid to Hybrid Surveillance if the intruder's signal strength does not meet the requirements for Extended Hybrid Surveillance per §2.2.4.6.4.2.2.1 and §2.2.4.6.4.2.3.1.3.
- Intruder 7 verifies the requirement to transition from Extended Hybrid to Hybrid Surveillance and to active surveillance if the intruder's reported quality does not meet the NACp requirement of §2.2.4.6.4.2.2.1.1 and subsequently fails validation.
- Intruder 8 verifies the requirement of §2.2.4.6.4.2.3.1.3 to transition from Hybrid to Extended Hybrid Surveillance.

- Intruder 9 verifies the requirement of §2.2.4.6.4.2.3.1.7 associated with validation and initial transitions from Extended Hybrid Surveillance to Hybrid Surveillance. It also verifies the requirements associated with revalidation for subsequent transitions from Extended Hybrid to Hybrid Surveillance.

ACAS X Aircraft

Altitude	= 12,000 ft
Altitude Rate	= 0 FPM
Position	= Frankfurt

Intruder Aircraft #1

Version	= 2
Altitude	= 7600 ft
Altitude Rate	= 0 FPM
Range	= 11 NM at T=0 sec
Relative Speed	= -360 kt (-0.1 NM/sec)

At T=40 the intruder is terminated.

Intruder Aircraft #2

Version	= 2
Altitude	= 20,250 ft at T=0
Altitude Rate	= -1,500 FPM
Range	= 2.9 NM
Relative Speed	= 0 kt

At T=105 the intruder is terminated.

Intruder Aircraft #3

Version	= 2
Altitude	= 3,750 ft at T=0
Altitude Rate	= 1,500 FPM
Range	= 2.9 NM
Relative Speed	= 0 kt

At T=105 the intruder is terminated.

Intruder Aircraft #4

Version	= 2
Altitude	= 17,500 ft at T=0
Altitude Rate	= 0 FPM at T=0
Altitude Rate	= at T=15 descend rate goes to -6000 FPM at 0.5g.
Range	= 2.9 NM
Relative Speed	= 0 kt

At T=40 the intruder is terminated.

Intruder Aircraft #5

Version	= 2
Altitude	= 7600 ft
Altitude Rate	= 0 FPM
Range	= 20 NM at T=0 sec
Relative Speed	= -360 kt (-0.1 NM/sec)

At T=20 the NACp from the intruder is degraded to 5.
At T=60 the intruder is terminated.

Success Criteria

Intruder 1

From T=1 to T=19, verify that DF=0 replies are not provided to the STM.

From T=1 to T=19, verify that no interrogation are address to this intruder.

Verify that the interrogation interval is no greater than 5 sec from T=21 to T=40.

From T=21 to the end of the test, verify that DF=0 replies are provided to the STM with surv_mode = 0/Norm or 1/Red.

Verify that the STM track is not dropped and the intruder is provided in the STM display output for the full duration of the test.

Intruder 2

From T=1 to T=85, verify that DF=0 replies are not provided to the STM.

From T=1 to T=85, verify that no UF=0 interrogations are made to this intruder

Verify one interrogation every sec of intruder aircraft from T=95 to T=105.

From T=95 to the end of the test, verify that DF=0 replies are provided to the STM with surv_mode = 0/Norm or 1/Red.

Verify that the STM track is not dropped and the intruder is provided in the STM display output for the full duration of the test.

Intruder 3

From T=1 to T=85, verify that DF=0 replies are not provided to the STM.

From T=1 to T=85, verify that no UF=0 interrogations are made to this intruder.

Verify one interrogation every sec of intruder aircraft from T=95 to T=105.

From T=95 to the end of the test, verify that DF=0 replies are provided to the STM with surv_mode = 0/Norm or 1/Red.

Verify that the STM track is not dropped and the intruder is provided in the STM display output for the full duration of the test.

Intruder 4

From T=1 to T=15, verify DF=0 replies are not provided to the STM.

From T=20 to the end of the test, verify that DF=0 replies are provided to the STM with surv_mode = 0/Norm or 1/Red.

Intruder 5

From T=1 to T=19, verify that DF=0 replies are not provided to the STM.

From T=1 to T=19, verify that no UF=0 interrogation are made to this intruder.

From T=20 to 25, verify that only one UF=0 interrogation is made to this intruder because of the transition to Hybrid Surveillance.

From T=25 to 60, verify that the next validation occurs within 30 to 40 seconds of the initial validation.

Verify that the STM track is not dropped and the intruder is provided in the STM display output for the full duration of the test.

From T=20 to T=60, verify that DF=0 replies are provided to the STM with surv_mode = 2/HSd.

Intruder 6

From T=1 to T=19, verify that DF=0 replies are not provided to the STM.

From T=1 to T=19, verify that no UF=0 interrogation are made to this intruder.

From T=20 to 40, verify that only one UF=0 interrogation is made to this intruder.

Verify that the STM track is not dropped and the intruder is provided in the STM display output for the full duration of the test.

From T=20 to T=40, verify that DF=0 replies are provided to the STM with surv_mode = 2/HS.

Intruder 7

From T=1 to T=19, verify DF=0 replies are not provided to the STM.

From T=25 to T=40, verify that DF=0 replies are provided to the STM with surv_mode = 0/Norm or 1/Red.

From T=1 to T=19, verify that no UF=0 interrogation are made to this intruder.

From T=25 to 40, verify that the maximum update interval between UF=0 interrogations to this intruder is 5 seconds.

Verify that the STM track is not dropped and the intruder is provided in the STM display output for the full duration of the test.

Intruder 8

From T=5 to T=30, verify that DF=0 replies are provided to the STM with surv_mode = 2/HS

From T=5 to T=30, verify that UF=0 interrogation are made no more frequently than once every 10 seconds.

From T=35 to the end of the test, verify that DF=0 replies are not provided to the STM.

From T=35 to the end of the test, verify that no interrogations are made to this intruder.

Intruder 9

From T=1 to T=19, verify that DF=0 replies are not provided to the STM.

After T=19, verify that DF=0 replies are provided to the STM with surv_mode = 2/HS.

From T=1 to T=19, verify that no UF=0 interrogation are to the intruder.

Verify that the first UF=0 interrogation to ownship occurs after T=19.

Verify that the interrogation interval to the intruder from T=20 to the end of the test is never less than 60 seconds and no more than two interrogations are made to the intruder.

Test 5 Air / Ground Transitions and Transitions between Active and Passive Surveillance

This test shows correct transitions between active and passive (Hybrid and Extended Hybrid) surveillance for air ground transitions per sections §2.2.4.6.4.2.3.2.3, §2.2.4.6.4.2.3.1.6, and §2.2.4.6.4.2.3.1.3 and provides requirement coverage related to section §2.2.4.6.4.2.2.1.

This test also shows correct air to ground transitions per section §2.2.4.6.3 and that invalid ground speed is monitored as per §2.2.7.2.9.

Scenario Description

Two ADS-B equipped intruders are defined – one is a version = 0 intruder and the other is a version = 2 intruder. A third intruder is Mode S equipped (but not ADS-B) and the fourth intruder is ATCRBS/Mode C equipped.

There are 5 runs for the test where the behavior of ownship is modified:

- In runs 1 and 2 Ownship is airborne and lands based on different triggering criteria.
- In runs 3 and 4 Ownship is on the ground and takes off based on different triggering criteria.
- In run 5 Ownship is on the ground but does not have valid ground speed.
- Intruder 1 does not qualify for being tracked with Hybrid Surveillance while ownship is airborne, but will qualify for Hybrid Surveillance when ownship is operating on the surface.

- Intruder 2 does not qualify for being tracked with Extended Hybrid Surveillance while ownship is airborne, but will qualify for Extended Hybrid Surveillance when ownship is operating on the surface.
- Intruder 3 is a Mode S non-ADS-B equipped traffic.
- Intruder 4 is a Mode C / ATCRBS transponder-equipped aircraft.

Intruder Aircraft #1

Version	= 0
Altitude	= 2,000 ft
Altitude Rate	= 0 FPM
Range	= 2.0 NM
Range Rate	= 0 kt

Intruder Aircraft #2

Version	= 2
Altitude	= 2,500 ft
Altitude Rate	= 0 FPM
Range	= 0 NM
Range Rate	= 0 kt

Intruder Aircraft #3

Altitude	= 1,000 ft
Altitude Rate	= 0 FPM
Range	= 1.0 NM
Range Rate	= 0 kt

Mode S transponder-equipped

No ADS-B Data

Intruder Aircraft #4

Altitude	= 1,000 ft
Altitude Rate	= 0 FPM
Range	= 7.0 NM
Range Rate	= 0 kt

ATCRBS/Mode C transponder-equipped

No ADS-B Data

All the tests have duration of 60 seconds.

Test Run 1

ACAS X Aircraft

Barometric Altitude	= 340 ft
Radio Altitude	= 340 ft
Altitude Rate	= -600 FPM (descending)

Altitude rate is for both barometric and radio altitudes.

Ground Speed = 25 kt

Note: Radio Altitude is only sent while its value is ≥ 0 .

Test Run 2

ACAS X Aircraft

Barometric Altitude	= 0 ft
Radio Altitude	= 0 ft

Altitude Rate = 0 FPM
 Ground Speed = 40 kt
 At T=30 Ground Speed is < 25 kt

Test Run 3

ACAS X Aircraft

Barometric Altitude = 0 ft
 Radio Altitude = 0 ft
 Altitude Rate = 0 FPM
 At T=14, Altitude Rate = +600 FPM (climbing)
 Altitude rate is for both barometric and radio altitudes.
 Goal is to transition 60 ft at T=30
 Ground Speed = 0 kt

Test Run 4

ACAS X Aircraft

Barometric Altitude = 0 ft
 Radio Altitude = 0 ft
 Altitude Rate = 0 FPM
 Ground Speed = 0 kt
 Ground Speed increased so at T=30 seconds, Ground Speed > 35 kt

Test Run 5

ACAS X Aircraft

Barometric Altitude = 0 ft
 Radio Altitude = 0 ft
 Altitude Rate = 0 FPM
 Ground Speed is invalid or unavailable for the duration of the test.

Success Criteria

Test Run	Intruder 1	Intruder 2	Intruder 3	Intruder 4
1 and 2 Ownship Landing	Active T < 25 Hybrid T > 35	Active T < 25 Extended Hybrid T > 35	Active for whole test	Active for whole test
3 and 4 Ownship Taking Off	Hybrid T < 25 Active > 35	Extended Hybrid T < 25 Active > 35	Active for whole test	Active for whole test
5 Ownship on- ground but ground speed invalid	Active for the whole test.	Active for the whole test.	Active for the whole test.	Active for the whole test.

The first two runs have ownship transitioning from Airborne to Operating On the Surface. Intruder 1 and Intruder 2 transition from active surveillance to Hybrid and Extended Hybrid Surveillance respectively. Intruders 3 and 4 are not ADS-B equipped intruders and should remain active tracked for the duration of the entire test.

In test runs 3 and 4 own takes off. Therefore Intruders 1 and 2 are initially tracked using Hybrid and Extended Hybrid Surveillance respectively. Then they transition to active surveillance. Intruders 3 and 4 are tracked using active surveillance the entire time.

The 5th test run verifies that when ground speed is unavailable ownship assumes that it is taking off or airborne per section §2.2.4.6.3. Therefore all the intruders should be tracked actively.

Test Run 1 and 2

Intruder 1

- Verify that the intruder is interrogated at a rate of 1 Hz from T=5 to T=25.
- Verify that the intruder is interrogated at a rate of 0.1 Hz from T = 35 to the end of the test.
- Verify that DF=0 replies are provided to the STM with surv_mode = 0/Norm while T ≤ 25 and with surv_mode =2/HS from T ≥ 35.

Intruder 2

- Verify that the intruder is interrogated at a rate of 1 Hz from T=5 to T=25.
- Verify that no interrogations are addressed to the intruder from T = 35 to the end of the test.
- Verify that DF=0 replies are provided to the STM with surv_mode = 0/Norm while T ≤ 25 and that DF=0 replies are not provided to the STM from T ≥ 35.

Intruder 3

- Verify that the intruder is interrogated at a rate of 1 Hz from T=5 to the end of the test.
- Verify that DF=0 replies are provided to the STM with surv_mode =0/Norm from T=5 to the end of the test.
-

Intruder 4

- Verify that the Mode C intruder is interrogated and updated every second for the duration of the test.

For Test runs 3 and 4

Intruder 1

- Verify that the intruder is interrogated at a rate from 0.1 Hz from T = 5 to T = 25.
- Verify that the intruder is interrogated at a rate of 1 Hz from T=35 to the end of the test.
- Verify that DF=0 replies are provided to the STM with surv_mode = 2/HS while T ≤ 25 and with surv_mode = 1/Norm from T ≥ 35.

Intruder 2

- Verify that no interrogations are addressed to the intruder from while T ≤ 25.
- Verify that the intruder is interrogated at a rate of 1 Hz from T ≥ 35 to the end of the test.
- Verify that DF=0 replies are not provided to the STM while T ≤ 25 and that DF=0 replies are provided to the STM with surv_mode = 0/Norm from T ≥ 35.

Intruder 3

- Verify that the intruder is interrogated at a rate of 1 Hz from T=5 to the end of the test.
- Verify that DF=0 replies are provided to the STM with surv_mode =0/Norm from T=5 to the end of the test.

Intruder 4

- Verify that the Mode C intruder is interrogated and updated every second for the duration of the test.

For Test Run 5**Intruder 1, 2 3**

- Verify that the intruder is interrogated at a rate of 1 Hz from T=5 to the end of the test.
- Verify that DF=0 replies are provided to the STM with surv_mode =0/Norm from T = 5 to the end of test.

Intruder 4

- Verify that the Mode C intruder is interrogated and updated every second for the duration of the test.
- Verify that an output is present indicating that Hybrid Surveillance capability has been degraded.

**2.4.2.11.6 Verification of Error Budget in Computing Slant Range from Passive Data
(§2.2.4.6.4.2.3.1.11)**

The following test only applies to the Front End Surveillance (i.e. non-ADD surveillance).

Test 1

The equipment manufacturer must define a test or document an analysis that demonstrates that the error budget in computing slant range is met by their surveillance design.

If the test method is used to demonstrate compliance to the requirement then this paragraph describes one potential scenario. Ownship and intruder aircraft are traveling towards each other at 600 kt at high latitude. If the error between the passive range estimate and active range measurement is less than 145 meters then the intent of the requirement is met. The error in range computation of tests at slower closure rates can be used to extrapolate or predict errors at the 1200 kt closure rate.

If an analysis method is used to demonstrate compliance to the requirement then the analysis must identify and/or describe the following:

- Ownship latitude / longitude sampling rate and time stamping accuracy / granularity.
- Ownship latitude / longitude extrapolation.
- Intruder latitude / longitude time stamping accuracy / granularity.
- Intruder latitude / longitude extrapolation.
- Identify the maximum error in slant range if ownship was traveling at 600 kt.
- Identify the maximum error in slant range if intruder aircraft was traveling at 600 kt.
- Identify errors resulting from latitude / longitude to slant range computations.

2.4.2.11.7 Surveillance Overload and Capacity Tests (§2.2.4.6.4.2.3.2.5)**Test 1**

Replicate Mode S Capacity Test (§2.4.2.1.7.5). All 150 intruders squitter both DF=17 and DF=11. Each of the 150 intruders **shall** (2563) squitter 2 airborne position squitters per sec, 2 velocity squitters per sec, and one Flight ID squitter every 5 sec and one DF=11 squitter per sec. Intruders 25 to 52, inclusive, squitter Airborne Position Messages that successfully validate. It is not necessary for the other intruders to validate.

Success Criteria

Verify success criteria per Mode S Capacity Test.

Verify that Mode S tracks which are in the passive surveillance region per §2.2.4.6.4.2.3.1.6 and have been established for 15 seconds are tracked using Hybrid Surveillance.

Test 2

Replicate Mode C, Mode S Test (§2.4.2.1.8.1). All Mode S intruders squitter both DF=17 and DF=11. Each of the Mode S intruders **shall** (2564) squitter 2 airborne position squitters per sec, 2 velocity squitters per sec, and one Flight ID squitter every 5 sec and one DF=11 squitter per sec. Even numbered intruders 26 to 80, inclusive, squitter Airborne Position Messages which successfully validate. It is not necessary for the other intruders to validate.

Success Criteria

Verify success criteria per Mode C, Mode S Test.

Verify that Mode S tracks which are in the passive surveillance region per §2.2.4.6.4.2.3.1.6 and have been established for 15 seconds are tracked using Hybrid Surveillance.

Test 3

This test verifies the overload requirements of §2.2.4.6.1.1.

ACAS X Aircraft

Per §2.4.2.11.1

Intruder Aircraft

Replicate Mode S Capacity Test (§2.4.2.1.7.5) with the following changes:

- All 150 intruders squitter both DF=17 and DF=11 as version=2 intruders per §2.4.2.11.1.
 - All intruders will have a squitter and reply probability of 1.
 - As per §2.4.2.11.1 all intruders have a power signal level of -70 dBm.
 - Intruder 1-24 Range Rate = 0
 - Intruder 1-24 Range = $3.5 + (\text{Intruder Number} * 0.5)$ NM
 - Intruder 1-24 Altitude relative to own = $-3000 + (\text{Intruder Number} * 200)$ ft
 - Intruder 1-24 start time is $T=10 + (5 * \text{Intruder Number})$
 - The test ends at $T=160$ seconds

Success Criteria

Verify that after T=10 there should be no active interrogations to any intruder.

Verify that within 30 seconds of Intruder 1 to 24 being enabled that the surveillance overload function has maintained the closer track and discarded the farthest track.

2.4.2.11.8

Verification of DF17 Decoding (§2.2.3.8.3.2.8.1, §2.2.3.8.3.2.8.3, §2.2.3.8.3.2.8.4, §2.2.3.8.3.2.8.5)

The following test only applies to the Front End Surveillance (i.e. non-ADD surveillance).

Test 1

This test shows the correct decode of Intruder Latitudes and Longitudes as well as the calculated Range and Bearing from ownship. Ref. G, §2.4.3.2.3.7.1 (Verification of Airborne Latitude and Longitude Data Encoding) identifies the Intruder Latitude and Longitude combinations that can robustly test the decoding function.

Scenario Description

Table 2-91 defines intruders that verify that the Airborne Position Messages are correctly decoded and that the ranges and bearings from ownship to the intruders are correctly calculated. Each intruder is static and the scenario needs to only be as long as it takes to calculate the range and bearing using the Airborne Position Message.

ACAS X (Own) Aircraft

Altitude	= 12,000 ft
Altitude Rate	= 0 FPM
Position	= Defined in Table 2-91 for each intruder
Heading	= 0 degrees
Velocity	= 0 kt

Intruder Aircraft

Altitude	= Defined in Table 2-91
Altitude Rate	= 0 FPM
Position	= Defined in Table 2-91
Velocity	= 0 kt

Table 2-91: Airborne Position Decoding Values With Range and Bearing

Intruder Aircraft No.	Own Latitude (deg)	Own Longitude (deg)	Intruder Latitude (deg)	Intruder Longitude (deg)	Intruder Altitude (ft)	Planar Range (NM)	Slant Range (NM)	Bearing (deg)
1	-89.5	-178	-90	175.5	16000	30.1776	30.1821	-180.0000
2	-89.5	-178	-89.95	-165	16000	27.2457	27.2512	178.5723
3	-89.5	-178	-89.5	-171.5	11000	3.4209	3.4249	93.2499
4	-89.5	-178	-89	-172.5	10000	30.4454	30.4489	10.9495
5	-87.4	62	-87.5	65.75	14000	11.7357	11.7398	122.8010
6	-87.4	62	-86.5	60	15000	54.6840	54.6840	-7.7428
7	-86	-60	-85.85	-60	12000	9.0512	9.0512	0.0000
8	-84.5	136	-85	120	11000	92.9738	92.9846	-116.7081
9	-84.5	136	-83.55	144	14000	76.0987	76.1004	45.2644
10	-83.2	-132	-84.25	-120	10000	101.0281	101.0443	134.5384
11	-83.2	-132	-82.68	-144	13000	94.1512	94.1579	-76.6016
12	-83.2	-132	-81.75	-121	15000	122.9425	122.9542	50.3582
13	-80	121.25	-80.25	121	12000	15.3018	15.3018	-170.3897
14	-80	-154	-79.75	-154.28	11000	15.3702	15.3715	-11.2787
15	-78	-121.1	-78.4	-121	16000	24.1635	24.1703	177.1215
16	-78	121.1	-77.4	121	16000	36.2191	36.2221	-2.0829
17	-76.55	-154.58	-76.55	-154.28	11000	4.2098	4.2131	90.1459
18	-75.6	154.58	-75.6	154.28	10000	4.5007	4.5130	-90.1453
19	-75.6	154.58	-74.75	157.5	14000	68.2458	68.2467	42.7452
20	-73.5	-157.4	-73.65	-157.5	13000	9.2042	9.2055	-169.3670
21	-72.35	-120.76	-72.75	-120	15000	27.7628	27.7654	150.6744
22	-72.35	120.76	-71.55	120	12000	50.2745	50.2759	-16.7737

Intruder Aircraft No.	Own Latitude (deg)	Own Longitude (deg)	Intruder Latitude (deg)	Intruder Longitude (deg)	Intruder Altitude (ft)	Planar Range (NM)	Slant Range (NM)	Bearing (deg)
23	-70	-144.05	-70.65	-144	11000	39.1894	39.1913	178.5388
24	-70	144.05	-69.55	144	10000	27.1394	27.1429	-2.2250
25	-86.7	-143	-86.75	-142.75	13000	3.1379	3.1422	164.1829
26	-69	-114	-68.75	-114.55	16000	19.2375	19.2470	-38.6916
27	-68	115	-67.75	114.55	16000	18.2087	18.2189	-34.3686
28	-66.25	-67.75	-66.55	-67.5	11000	19.0544	19.0556	161.6460
29	-65.75	67.5	-65.55	67.5	10000	12.0470	12.0521	0.0000
30	-64.3	-83	-64.45	-83.08	14000	9.2733	9.2787	-167.0318
31	-64.3	83	-63.25	83.08	13000	63.2722	63.2736	1.9668
32	-62.5	-64.25	-62.25	-64.29	15000	15.0970	15.1040	-4.2668
33	-60.75	64.3	-61.25	64.29	12000	30.1017	30.1020	-179.4480
34	-60.75	-71.75	-60.25	-72	11000	30.9982	30.9997	-13.9624
35	-60.75	-120.25	-59.96	-120.5	10000	48.1292	48.1337	-9.0193
36	-60.75	120	-59.955	120	14000	47.8585	47.8585	0.0000
37	-60.75	-120.25	-59.93	-119.5	13000	54.1954	54.1960	24.7138
38	-58.1	-79	-58	-78.75	15000	9.9925	10.0040	53.0728
39	-58.1	78.5	-58.5	78.75	12000	25.3398	25.3400	161.8935
40	-58.1	-22.3	-57.95	-22.5	11000	11.0556	11.0571	-35.3604
41	-57	22.55	-56.85	22.5	16000	9.1749	9.1976	-10.3512
42	-54.7	-53.05	-55.5	-52.94	16000	48.2732	48.2743	175.5370
43	-54.7	53.05	-54.62	52.94	11000	6.1525	6.1549	-38.6043
44	-53	-29.95	-53.25	-30	10000	15.1383	15.1426	-173.1600
45	-53	29.95	-51.95	30	14000	63.1540	63.1545	1.6854
46	-49.9	-75.8	-50.75	-75.79	13000	51.0842	51.0847	179.5723
47	-49.9	75.8	-49.65	75.79	15000	15.0299	15.0370	-1.4877
48	-47.7	-27.1	-48.25	-27	12000	33.2845	33.2849	173.0765
49	-47.7	27.1	-47	27	11000	42.2405	42.2427	-5.5821
50	-46	-68.6	-45.6	-68.57	10000	24.0502	24.0538	3.0137
51	-43	179	-44.25	180	16000	86.7887	86.7898	150.1813
52	-43	179	-42.85	179.5	16000	23.8248	23.8318	67.9639
53	-43	179	-41.4	178.5	11000	98.5573	98.5699	-13.2494
54	-40.1	175.4	-40	175.5	10000	7.5650	7.5725	37.5746
55	-38	-160.5	-38.5	-160.9	14000	35.4561	35.4564	-147.8773
56	-36	-171.9	-36.9	-171.5	13000	57.3295	57.3303	160.3675
57	-36	-171.9	-35.25	-172.5	15000	53.7036	53.7037	-33.3266
58	-33.65	65.85	-33.6	65.75	12000	5.8402	5.8402	-59.1618
59	-31.14	-142.76	-31.8	-142.75	11000	39.5369	39.5389	179.2586
60	-30	60	-30	59.5	10000	26.0614	26.0649	-90.1250
61	-28.3	-60	-28	-60	14000	17.9637	17.9659	0.0000
62	-25.9	-66.11	-25.9	-66.7	13000	31.9412	31.9412	-90.1289
63	-23.55	-120.05	-23.6	-120	15000	4.0694	4.0989	137.3425
64	-21	41	-21.1	41.5	12000	28.7063	28.7065	102.1168
65	-18.4	-144.2	-18.2	-144	11000	16.5382	16.5395	43.7185
66	-15	-121.22	-14.9	-121	16000	14.1174	14.1314	64.9708
67	-10.2	121.22	-10.5	121	16000	22.1594	22.1671	-144.0362
68	-5	6.25	-5.1	6.25	11000	5.9742	5.9766	180.000
69	-2.11	154.35	-2.5	154.28	10000	23.6731	23.6767	-169.7670
70	0.02	59.76	0	60	14000	14.4848	14.4879	94.7319

Intruder Aircraft No.	Own Latitude (deg)	Own Longitude (deg)	Intruder Latitude (deg)	Intruder Longitude (deg)	Intruder Altitude (ft)	Planar Range (NM)	Slant Range (NM)	Bearing (deg)
71	-0.02	-157.14	0	-157.5	13000	21.6850	21.6852	-86.8414
72	89.9	-179.6	90	-180	15000	6.0353	6.0550	0.0000
73	89.9	-179.6	89.95	179.5	12000	3.0180	3.0180	-0.8998
74	89.9	-179.6	89.5	178.5	11000	24.1406	24.1418	-177.6252
75	89.9	-179.6	89	175.5	10000	54.3270	54.3322	-174.5565
76	87	-167.5	87.5	-165	16000	31.0259	31.0303	12.2435
77	87	-167.5	86.75	-171.5	16000	20.0131	20.0221	-136.8949
78	87	-167.5	86.5	-172.5	11000	34.6529	34.6546	-147.9377
79	85.1	60	85.85	65.75	10000	52.8289	52.8340	28.3243
80	85.2	-143	85	-142.75	14000	12.1377	12.1416	173.7807
81	85.1	60	84.25	60	13000	51.2888	51.2893	180.0000
82	83.25	-60.11	83.55	-60	15000	18.1195	18.1250	2.3588
83	83.25	120.2	82.68	120	12000	34.4222	34.4226	-177.4398
84	81.9	-120.2	81.75	-120	11000	9.2106	9.2123	169.1639
85	80.23	144.15	80.25	144	10000	1.9517	1.9793	-51.7455
86	80.23	-144.15	79.75	-144	14000	29.0016	29.0023	176.8163
87	78.2	-121.3	78.4	-121	13000	12.6105	12.6113	16.7763
88	78.2	121.3	77.4	121	15000	48.4103	48.4105	-175.3209
89	76.65	-154.28	76.55	-154.28	12000	6.0312	6.0312	-180.0000
90	75.35	154.3	75.6	154.28	11000	15.0788	15.0801	-1.1402
91	74	157.65	74.75	157.5	16000	45.2981	45.2995	-3.0127
92	74	-157.65	73.65	-157.5	16000	21.2579	21.2662	173.1162
93	72	-120.2	72.75	-120	11000	45.3602	45.3626	4.5238
94	72	120.2	71.55	120	10000	27.3864	27.3899	-171.9864
95	70.05	-144.11	70.65	-144	14000	36.2368	36.2371	3.4787
96	70.05	144.11	69.55	144	13000	30.2239	30.2239	-175.6007
97	68.2	-114.35	68.75	-114.55	15000	33.4428	33.4444	-7.5134
98	68.2	114.35	67.75	114.55	12000	27.4909	27.4911	170.4359
99	66.1	-67.45	66.55	-67.5	11000	27.1363	27.1376	-2.5345
100	66.1	67.45	65.55	67.5	10000	33.1526	33.1562	177.8427
101	63.99	-82.99	64.45	-83.08	14000	27.8082	27.8090	-4.8294
102	63.99	83.14	63.25	83.08	13000	44.5958	44.5960	-177.9071
103	61.3	-64.45	62.25	-64.29	15000	57.3851	57.3851	4.4905
104	61.3	64.45	61.25	64.29	12000	5.5281	5.5281	-122.9219
105	60.11	-72.14	60.25	-72	11000	9.4142	9.4159	26.4174
106	60.11	-120.25	59.96	-120.5	16000	11.7582	11.7755	-140.0686
107	60.11	120.25	59.955	120	16000	11.9913	12.0082	-140.9877
108	60.11	-120.25	59.93	-119.5	11000	25.0581	25.0594	115.2918
109	58.01	-78.5	58	-78.75	10000	8.0076	8.0148	-94.2031
110	58.01	79	58.5	78.75	14000	30.5364	30.5371	-14.9467
111	58.01	-22.55	57.95	-22.5	13000	3.9485	3.9519	156.1007
112	58.01	22.55	56.85	22.5	15000	69.8191	69.8194	-178.6471
113	55.09	-53.04	55.5	-52.94	12000	24.8978	24.8980	7.8818
114	54.39	53.04	54.62	52.94	11000	14.2672	14.2686	-14.1584
115	53.4	-29.76	53.25	-30	10000	12.4884	12.4933	-136.1344
116	52.4	30.34	51.95	30	14000	29.8307	29.8314	-154.9498
117	50	-75.65	50.75	-75.79	13000	45.3938	45.3940	-6.7536
118	50	75.65	49.65	75.79	15000	21.7278	21.7319	165.4381

Intruder Aircraft No.	Own Latitude (deg)	Own Longitude (deg)	Intruder Latitude (deg)	Intruder Longitude (deg)	Intruder Altitude (ft)	Planar Range (NM)	Slant Range (NM)	Bearing (deg)
119	47.82	-27.22	48.25	-27	12000	27.3095	27.3097	18.8576
120	47.82	27.22	47	27	11000	50.0603	50.0631	-169.5970
121	45.3	-69	45.6	-68.57	16000	25.5929	25.5991	45.0997
122	42.9	179	44.25	180	16000	92.0437	92.0454	27.9609
123	42.9	179	42.85	179.5	11000	22.2733	22.2746	97.5725
124	42.9	179	41.4	178.5	10000	92.7234	92.7369	-165.9010
125	39.23	176.02	40	175.5	14000	52.1115	52.1116	-27.4132
126	38.2	-160.05	38.5	-160.9	13000	43.9906	43.9907	-65.5953
127	35.5	-172.04	36.9	-171.5	15000	87.9376	87.9399	17.2014
128	35.5	-172.04	35.25	-172.5	12000	27.1024	27.1026	-123.4334
129	33.4	65.8	33.6	65.75	11000	12.2440	12.2454	-11.8157
130	31.75	-142.95	31.8	-142.75	10000	10.6634	10.6690	73.6352
131	29.8	60.02	30	59.5	14000	29.6624	29.6631	-66.0522
132	28.2	-60.02	28	-60	13000	12.0222	12.0230	174.9278
133	25.32	-67.1	25.9	-66.7	15000	40.9471	40.9478	31.9318
134	23.55	-120.2	23.6	-120	12000	11.4289	11.4289	74.7847
135	21	41	21.1	41.5	11000	28.7049	28.7063	77.8832
136	18.35	-144.22	18.2	-144	16000	15.4436	15.4561	125.4809
137	15.35	-120.75	14.9	-121	16000	30.5742	30.5787	-151.6126
138	10.6	121.2	10.5	121	11000	13.2499	13.2512	-116.7895
139	5.11	6.43	5.1	6.25	10000	10.7984	10.8040	-93.1633
140	2.8	155.54	2.5	154.28	14000	77.7921	77.7940	-103.2903
141	-0.03	60.11	0	60	10000	6.8534	6.8617	-74.8423
142	-0.03	-157.65	0	-157.5	9000	9.1963	9.2102	78.7638

Success Criteria

All Intruders.

For all of the Intruders verify that the range for each intruder is within 145 m of the calculated range identified in Table 2-91.

For all of the Intruders verify that the bearing for each intruder is within 3 deg of the calculated bearing identified in Table 2-91.

Verify that the error in range from the calculated range does not use more of the error budget allowed for range based on the completion of Test §2.4.2.11.6 (Verification of Error Budget in Computing Slant Range from Passive Data) Test 1.

2.4.2.11.9 Verification of Monitoring Requirements (§2.2.7.2)

Test 1

This test verifies that passive surveillance is not performed in the absence of valid own latitude and longitude information (§2.2.3.8.3.2.8).

Scenario Description

- Intruder 1 will meet the requirements for being tracked passively.

ACAS X Aircraft

Altitude = 12,000 ft

Altitude Rate = 0 FPM

Position = Sydney

At T=60 own latitude and longitude input is made unavailable.

Intruder Aircraft #1

Altitude = 17,000 ft

Altitude Rate = 0 FPM

Range = 10 NM at T=0 sec

Relative Speed = 0 kt

At T=90 the intruder is terminated.

Note: The equipment manufacturer should repeat this test for each ownship latitude and longitude source.

Success Criteria**Intruder 1**

Verify that the acquisition interrogations have RL=0.

No more than one interrogation of intruder aircraft from T=10 to T=59.

At least by T=65 the intruder is interrogated once every 5 sec indicating a transition to active surveillance.

Verify that by at least T=70 that an output is provided indicating that the capabilities of passive surveillance have been degraded.

Verify that the track is not dropped.

**2.4.2.11.10 Verification of On-Ground ACAS X Range Determination Using ADS-B
(§2.2.4.6.4.2.4)****Test 1**

This test will verify that when own should include on-ground ACAS X in its NTA3 and NTA6 estimates, it does not interrogate on-ground ACAS X-equipped aircraft that provide sufficient quality ADS-B OUT information.

Scenario Description

- Ownship is at 1500 ft AGL.
- One TCAS-equipped intruder is on the ground and at a range of 2 NM.
- One TCAS-equipped intruder is on the ground and at a range of 4 NM.
- One TCAS-equipped intruder is on the ground and at a range of 5 NM.

ACAS X Aircraft

Altitude = 1500 ft

Radio Altitude = 1500 ft

Altitude Rate = 0 FPM

Position = Sydney

Intruder Aircraft #1

Altitude = 500 ft
 Altitude Rate = 0 FPM
 Range = 2 NM at T=0 sec
 Relative Speed = 0 kt

Intruder is on the ground.

CA field of all squitters indicates on the ground.

Intruder generates DF=17 Surface Position Messages at a rate of once every 5 seconds per §2.2.3.8.3.2.9 with values determined to validate tests objective.

DF=17 Format Type Code = 5.

Intruder generates TCAS broadcasts.

At T=40 the intruder is terminated.

Intruder Aircraft #2

Altitude = 700 ft
 Altitude Rate = 0 FPM
 Range = 5 NM at T=0 sec
 Relative Speed = 0 kt

Intruder is on the ground.

CA field of all squitters indicates on the ground.

Intruder generates DF=17 Surface Position Messages per §2.2.3.8.3.2.9 with values determined to validate test objective.

DF=17 Format Type Code = 7.

Intruder generates TCAS broadcasts.

At T=40 the intruder is terminated.

Intruder Aircraft #3

Altitude = 300 ft
 Altitude Rate = 0 FPM
 Range = 4 NM at T=0 sec
 Relative Speed = 0 kt

Intruder is on the ground.

CA field of all squitters indicates on the ground.

Intruder generates DF=17 Surface Position Messages per §2.2.3.8.3.2.9 with values determined to validate test objective.

DF=17 Format Type Code = 8.

Intruder generates TCAS broadcasts.

At T=40 the intruder is terminated.

Success Criteria

Verify the following:

- 1) No interrogations to Intruder #1 or Intruder #2 are performed.
- 2) That the System sets NTA3 to 1 from T=10 to T=40
- 3) That the System sets NTA6 to 3 from T=10 to T=40
- 4) That the System sets NTA3 and NTA6 to zero by T=70

If legacy TCAS equipment (originally certified to TSO-C119c or earlier) has limitations which prevent it from implementing the requirements of §2.2.4.6.4.2.4 then the following will be documented in the test results:

- Identify the TSO to which the unit was previously certified.

- Identify and document the rationale which makes the implementation of the requirement in §2.2.4.6.4.2.4 not feasible.

2.4.2.11.11 Verification of DF=18 ADS-B and ADS-R Only Passive Surveillance (§2.2.3.8.3.2.8, §2.2.5.5.4, §2.2.5.5.9, §2.2.5.5.10)

The following tests may be performed using DF=18 ADS-B/ADS-R reports or directly decoded DF=18 ADS-B/ADS-R messages.

Test 1

This test will verify the decoding of the Extended Squitter ME field for DF=18 “ADS-B” messages per §2.2.3.8.3.2.8 and that DF=18 passive measurements are continuously provided by Surveillance to the STM per §2.2.5.5.9, §2.2.5.5.10, and §2.2.5.5.11.

Scenario Description:

- Intruder 1 shows that DF=18 ADS-B (ICAO 24-bit Aircraft Address) passive measurements are provided to the STM and meets all the minimum ADS-B data quality parameters (intruder will be tracked passively by the STM). Intruder 1 will also show that only TAs are issued with DF=18 ADS-B Only intruders (i.e. no RAs).

Note: The equipment manufacturer must ensure that the optional AOTO capability is enabled for TAs to be issued on this intruder for this test.

- Intruder 2 shows that DF=18 ADS-B (Anonymous Address) passive measurements are provided to the STM and meets all the minimum ADS-B data quality parameters, except for Traffic ADS-B Version (after T=10, intruder will not be tracked passively by the STM).
- Intruder 3 shows that DF=18 ADS-B (ICAO 24-bit Aircraft Address) passive measurements are provided to the STM and meets all the minimum ADS-B data quality parameters, except for Traffic NACp (after T=15, intruder will not be tracked passively by the STM).
- Intruder 4 shows that DF=18 ADS-B (Anonymous Address) passive measurements are provided to the STM and meets all the minimum ADS-B data quality parameters, except for Traffic NACv (after T=20, intruder will not be tracked passively by the STM).
- Intruder 5 shows that DF=18 ADS-B (ICAO 24-bit Aircraft Address) passive measurements are provided to the STM and meets all the minimum ADS-B data quality parameters, except for Traffic NIC (after T=25, intruder will not be tracked passively by the STM).
- Intruder 6 shows that DF=18 ADS-B (Anonymous Address) passive measurements are provided to the STM and meets all the minimum ADS-B data quality parameters, except for Traffic SIL (after T=30, intruder will not be tracked passively by the STM).

Note: The minimum ADS-B data quality parameters for tracking passively by the STM are as follows:

- ADS-B Version: 2
- NACp: 7
- NACv: 1
- NIC: 6
- SIL: 3

ACAS X Aircraft

Altitude	= 12,000 ft
Altitude Rate	= 0 FPM
Position	= Sydney

The following intruder aircraft conditions should use the same initial conditions defined in Section §2.4.2.6.1.2, except that all active data should not be provided (i.e. DF=0, DF=11, DF=16) and as otherwise stated in the intruder aircraft settings below:

Intruder Aircraft #1

Altitude	= 12,000 ft
Altitude Rate	= 0 FPM
Range	= 4.0 NM at T=0 sec
Relative Speed	= -360 kt (-0.1 NM/sec)
Relative Bearing	= 0 degrees
STM auto_on discrete input	= true
DF=18, CF=0 (ADS-B ICAO 24-bit Aircraft Address)	
At T=40 the intruder is terminated.	

Intruder Aircraft #2

Altitude	= 13,200 ft
Altitude Rate	= 0 FPM
Range	= 3.4 NM at T=0 sec
Relative Speed	= 0 kt
DF=18, CF=1 (ADS-B Anonymous Address)	
At T=10 the ADS-B Version from the intruder is degraded to 1.	
At T=40 the intruder is terminated.	

Intruder Aircraft #3

Altitude	= 13,400 ft
Altitude Rate	= 0 FPM
Range	= 3.6 NM at T=0 sec
Relative Speed	= 0 kt
DF=18, CF=0 (ADS-B ICAO 24-bit Aircraft Address)	
At T=15 the NACp from the intruder is degraded to 6.	
At T=40 the intruder is terminated.	

Intruder Aircraft #4

Altitude	= 13,600 ft
Altitude Rate	= 0 FPM
Range	= 3.8 NM at T=0 sec
Relative Speed	= 0 kt
DF=18, CF=1 (ADS-B Anonymous Address)	

At T=20 the NACv from the intruder is degraded to 0.
At T=40 the intruder is terminated.

Intruder Aircraft #5

Altitude = 13,800 ft
Altitude Rate = 0 FPM
Range = 4.0 NM at T=0 sec
Relative Speed = 0 kt
DF=18, CF=0 (ADS-B ICAO 24-bit Aircraft Address)
At T=25 the NIC from the intruder is degraded to 5.
At T=40 the intruder is terminated.

Intruder Aircraft #6

Altitude = 14,000 ft
Altitude Rate = 0 FPM
Range = 4.2 NM at T=0 sec
Relative Speed = 0 kt
DF=18, CF=1 (ADS-B Anonymous Address)
At T=30 the SIL from the intruder is degraded to 2.
At T=40 the intruder is terminated.

Success Criteria

Intruder 1

Verify that the STM track is not dropped and the intruder is provided in the STM display output for the full duration of the test.

Verify that only TAs are issued with this intruder (i.e. no RAs)

Intruder 2

From T=1 to T=9, verify that the STM track is not dropped and the intruder is provided in the STM display output.

After T=15, verify that the STM track is dropped and the intruder is no longer provided in the STM display output for the remainder of the test.

Intruder 3

From T=1 to T=14, verify that the STM track is not dropped and the intruder is provided in the STM display output.

After T=20, verify that the STM track is dropped and the intruder is no longer provided in the STM display output for the remainder of the test.

Intruder 4

From T=1 to T=19, verify that the STM track is not dropped and the intruder is provided in the STM display output.

After T=25, verify that the STM track is dropped and the intruder is no longer provided in the STM display output for the remainder of the test.

Intruder 5

From T=1 to T=24, verify that the STM track is not dropped and the intruder is provided in the STM display output.

After T=30, verify that the STM track is dropped and the intruder is no longer provided in the STM display output for the remainder of the test.

Intruder 6

From T=1 to T=29, verify that the STM track is not dropped and the intruder is provided in the STM display output.

After T=35, verify that the STM track is dropped and the intruder is no longer provided in the STM display output for the remainder of the test.

Test 2

This test will verify the decoding of the Extended Squitter ME field for DF=18 “ADS-R” messages per §2.2.3.8.3.2.8 and that DF=18 passive measurements are continuously provided by Surveillance to the STM per §2.2.5.5.9, §2.2.5.5.10, and §2.2.5.5.11.

Scenario Description:

- Intruder 1 shows that DF=18 ADS-R (ICAO 24-bit Aircraft Address) passive measurements are provided to the STM and meets all the minimum ADS-B data quality parameters (intruder will be tracked passively by the STM). Intruder 1 will also show that only TAs are issued with DF=18 ADS-R Only intruders (i.e. no RAs).

Note: The equipment manufacturer must ensure that the optional AOTO capability is enabled for TAs to be issued on this intruder for this test.

- Intruder 2 shows that DF=18 ADS-R (Anonymous Address) passive measurements are provided to the STM and meets all the minimum ADS-B data quality parameters, except for Traffic ADS-B Version (after T=10, intruder will not be tracked passively by the STM).
- Intruder 3 shows that DF=18 ADS-R (ICAO 24-bit Aircraft Address) passive measurements are provided to the STM and meets all the minimum ADS-B data quality parameters, except for Traffic NACp (after T=15, intruder will not be tracked passively by the STM).
- Intruder 4 shows that DF=18 ADS-R (Anonymous Address) passive measurements are provided to the STM and meets all the minimum ADS-B data quality parameters, except for Traffic NACv (after T=20, intruder will not be tracked passively by the STM).
- Intruder 5 shows that DF=18 ADS-R (ICAO 24-bit Aircraft Address) passive measurements are provided to the STM and meets all the minimum ADS-B data quality parameters, except for Traffic NIC (after T=25, intruder will not be tracked passively by the STM).
- Intruder 6 shows that DF=18 ADS-R (Anonymous Address) passive measurements are provided to the STM and meets all the minimum ADS-B data quality parameters, except for Traffic SIL (after T=30, intruder will not be tracked passively by the STM).

Note: The minimum ADS-B data quality parameters for tracking passively by the STM are as follows:

- ADS-B Version: 2
- NACp: 7
- NACv: 1
- NIC: 6
- SIL: 3

ACAS X Aircraft

Altitude = 12,000 ft
Altitude Rate = 0 FPM
Position = Sydney

The following intruder aircraft conditions should use the same initial conditions defined in Section §2.4.2.6.1.2, except that all active data should not be provided (i.e. DF=0, DF=11, DF=16) and as otherwise stated in the intruder aircraft settings below:

Intruder Aircraft #1

Altitude = 12,000 ft
Altitude Rate = 0 FPM
Range = 4.0 NM at T=0 sec
Relative Speed = -360 kt (-0.1 NM/sec)
Relative Bearing = 0 degrees
DF=18, CF=6, IMF=0 (ADS-R ICAO 24-bit Aircraft Address)
At T=40 the intruder is terminated.

Intruder Aircraft #2

Altitude = 13,200 ft
Altitude Rate = 0 FPM
Range = 3.4 NM at T=0 sec
Relative Speed = 0 kt
DF=18, CF=6, IMF=1 (ADS-R Anonymous Address)
At T=10 the ADS-B Version from the intruder is degraded to 1.
At T=40 the intruder is terminated.

Intruder Aircraft #3

Altitude = 13,400 ft
Altitude Rate = 0 FPM
Range = 3.6 NM at T=0 sec
Relative Speed = 0 kt
DF=18, CF=6, IMF=0 (ADS-R ICAO 24-bit Aircraft Address)
At T=15 the NACp from the intruder is degraded to 6.
At T=40 the intruder is terminated.

Intruder Aircraft #4

Altitude = 13,600 ft
Altitude Rate = 0 FPM
Range = 3.8 NM at T=0 sec
Relative Speed = 0 kt
DF=18, CF=6, IMF=1 (ADS-R Anonymous Address)
At T=20 the NACv from the intruder is degraded to 0.
At T=40 the intruder is terminated.

Intruder Aircraft #5

Altitude = 13,800 ft
Altitude Rate = 0 FPM
Range = 4.0 NM at T=0 sec
Relative Speed = 0 kt
DF=18, CF=6, IMF=0 (ADS-R ICAO 24-bit Aircraft Address)
At T=25 the NIC from the intruder is degraded to 5.

At T=40 the intruder is terminated.

Intruder Aircraft #6

Altitude	= 14,000 ft
Altitude Rate	= 0 FPM
Range	= 4.2 NM at T=0 sec
Relative Speed	= 0 kt

DF=18, CF=6, IMF=1 (ADS-R Anonymous Address)
At T=30 the SIL from the intruder is degraded to 2.
At T=40 the intruder is terminated.

Success Criteria

Intruder 1

Verify that the STM track is not dropped and the intruder is provided in the STM display output for the full duration of the test.

Verify that only TAs are issued with this intruder (i.e. no RAs)

Intruder 2

From T=1 to T=9, verify that the STM track is not dropped and the intruder is provided in the STM display output.

After T=15, verify that the STM track is dropped and the intruder is no longer provided in the STM display output for the remainder of the test.

Intruder 3

From T=1 to T=14, verify that the STM track is not dropped and the intruder is provided in the STM display output.

After T=20, verify that the STM track is dropped and the intruder is no longer provided in the STM display output for the remainder of the test.

Intruder 4

From T=1 to T=19, verify that the STM track is not dropped and the intruder is provided in the STM display output.

After T=25, verify that the STM track is dropped and the intruder is no longer provided in the STM display output for the remainder of the test.

Intruder 5

From T=1 to T=24, verify that the STM track is not dropped and the intruder is provided in the STM display output.

After T=30, verify that the STM track is dropped and the intruder is no longer provided in the STM display output for the remainder of the test.

Intruder 6

From T=1 to T=29, verify that the STM track is not dropped and the intruder is provided in the STM display output.

After T=35, verify that the STM track is dropped and the intruder is no longer provided in the STM display output for the remainder of the test.

2.4.2.11.12

Verification of Intruder ADS-B Reports to STM

This test will verify that surveillance will only provide the STM with Intruder ADS-B reports that have an ICAO 24-bit Aircraft Address not equal to ownship, not equal to either

0 (Hex) or FFFFFF (Hex), and are determined to be airborne per §2.2.5.5.9, §2.2.5.5.10, and §2.2.5.5.11.

Scenario Description:

- Intruder 1 shows that ADS-B State Vector Position Reports, State Vector Velocity Reports, and Mode Status Reports are provided to the STM that have an ICAO 24-bit Aircraft Address not equal to ownship, not equal to either 0 (Hex) or FFFFFF (Hex), and are determined to be airborne.
- Intruder 2 shows that ADS-B State Vector Position Reports, State Vector Velocity Reports, and Mode Status Reports are not provided to the STM that have an ICAO 24-bit Aircraft Address equal to ownship.
- Intruder 3 shows that ADS-B State Vector Position Reports, State Vector Velocity Reports, and Mode Status Reports are not provided to the STM that have an ICAO 24-bit Aircraft Address equal to 0 (Hex).
- Intruder 4 shows that ADS-B State Vector Position Reports, State Vector Velocity Reports, and Mode Status Reports are not provided to the STM that have an ICAO 24-bit Aircraft Address equal to FFFFFF (Hex).
- Intruder 5 shows that ADS-B State Vector Position Reports, State Vector Velocity Reports, and Mode Status Reports are not provided to the STM that are determined to be on the ground.

ACAS X Aircraft

Altitude	= 12,000 ft
Altitude Rate	= 0 FPM
Position	= Sydney

The following intruder aircraft conditions should use the same initial conditions defined in Section §2.4.2.11.1.2, except that all active data should not be provided (i.e. DF=0, DF=11, and DF=16 replies are not provided).

Intruder Aircraft #1

Altitude	= 12,000 ft
Altitude Rate	= 0 FPM
Range	= 4.0 NM at T=0 sec
Relative Speed	= 0 kt

DF=17 extended squitters contain a different ICAO 24-bit Aircraft Address than ownship, not equal to either 0 (Hex) or FFFFFF (Hex), and indicate airborne. At T=40 the intruder is terminated.

Intruder Aircraft #2

Altitude	= 13,200 ft
Altitude Rate	= 0 FPM
Range	= 3.4 NM at T=0 sec
Relative Speed	= 0 kt

DF=17 extended squitters contain the same ICAO 24-bit Aircraft Address as ownship and indicate airborne.

At T=40 the intruder is terminated.

Intruder Aircraft #3

Altitude = 13,400 ft

Altitude Rate = 0 FPM

Range = 3.6 NM at T=0 sec

Relative Speed = 0 kt

DF=17 extended squitters contain a ICAO 24-bit Aircraft Address equal to 0 (Hex) and indicate airborne.

At T=40 the intruder is terminated.

Intruder Aircraft #4

Altitude = 13,600 ft

Altitude Rate = 0 FPM

Range = 3.8 NM at T=0 sec

Relative Speed = 0 kt

DF=17 extended squitters contain an ICAO 24-bit Aircraft Address equal to FFFFFF (Hex) and indicate airborne.

At T=40 the intruder is terminated.

Intruder Aircraft #5

Range = 4.2 NM at T=0 sec

Relative Speed = 0 kt

DF=17 extended squitters contain a different ICAO 24-bit Aircraft Address than ownship, not equal to either 0 (Hex) or FFFFFF (Hex), and indicate on the ground.

At T=40 the intruder is terminated.

Success Criteria

Intruder 1

Verify that the intruder is provided in the STM display output for the full duration of the test.

Intruder 2

Verify that the intruder is not provided in the STM display output for the full duration of the test.

Intruder 3

Verify that the intruder is not provided in the STM display output for the full duration of the test.

Intruder 4

Verify that the intruder is not provided in the STM display output for the full duration of the test.

Intruder 5

Verify that the intruder is not provided in the STM display output for the full duration of the test.

2.4.2.12 ACAS Xo

The manufacturer **shall** (2444) provide a test plan that verifies the ACAS X system output of the following information elements:

- a. DNA Validity
- b. CSPO-3000 validity
- c. Xo Mode Availability for each mode
- d. Designated Xo mode
- e. Actual Xo mode

The implementer **shall** (2445) provide a test plan that verifies the integrated ACAS X/ASA System (or equivalent) allows a user to designate an Xo mode for an intruder.

The implementer **shall** (2446) provide a test plan that verifies the integrated ACAS X/ASA System (or equivalent) allows a user to undesignate an intruder.

The implementer **shall** (2447) provide a test plan that verifies AIRB.

Verification AIRB: Using test setups #1, and #2 as specified in §3.3.3.1, and §3.3.3.2 of RTCA/DO-317B (Ref. T), verify compliance to RTCA/DO-317B §3.3.2.17.

Note: This verifies that the ASA System (or equivalent) provides the designation tools.

The manufacturer **shall** (2448) provide a test plan that verifies the ACAS X system input of the following information elements for designated traffic:

- a. ICAO 24-bit Aircraft Address
- b. Xo Mode Request

The manufacturer **shall** (2449) provide a test plan that verifies the display/control system will not allow designation of another aircraft to ACAS Xo while a tracked aircraft is designated to an Xo mode.

2.4.3 Computer Performance Verification

Note: The details of these tests should be coordinated with the appropriate airworthiness authorities.

2.4.3.1 RAM Pattern Tests

The manufacturer **shall** (2565) verify the RAM by writing unique patterns of 1s and 0s into RAM and then reading these values back out of RAM to check for bad bits.

2.4.3.2 CPU Instruction Tests

The manufacturer **shall** (2566) check that CPU instructions are functional.

2.4.3.3 Program Memory Tests

The manufacturer **shall** (2567) check that the software program contained in memory is correct.

2.4.3.4 CPU Input/Output Tests

The manufacturer **shall** (2568) verify the CPU input/output functions.

2.4.3.5 CPU Timing Tests

The manufacturer **shall** (2569) verify that the computer operates within its allowable timing constraints as established by the manufacturer.

2.4.4 Cross-Reference of Requirements and Associated Tests

Notes:

1. *Table 2-92 and Table 2-93 relate the surveillance tests in §2.4 to the corresponding surveillance requirements in §2.2. Table 2-92 is ordered by test section. Table 2-93 is ordered by requirement section.*
2. *For additional tests of controls and displays, see §2-10 and §2-11 of AC 20-151C.*
3. *Additional verifications of the requirements listed in Table 2-93, including the tests in §2.4, are captured in the Verification Requirements Traceability Matrix (VRTM) tab of the RMJM workbook.*

Table 2-92: Cross-Reference of Requirements to Associated Tests

Tests		Requirements	
2.4.2.1.1.1	Transmit Frequency	2.2.3.5	Transmit Frequency and Tolerance
2.4.2.1.1.2	Radiated Output Power	2.2.3.1	Radiated Output Power
2.4.2.1.1.3	Control of Synchronous Interference by Transmitter Power	2.2.4.5.4.1	Control of Synchronous Garble by Transmitter Power
2.4.2.1.1.3.1	Control of Synchronous Garble by Whisper-Shout	2.2.4.5.4 2.2.4.5.4.1 2.2.4.5.4.1.1 2.2.4.5.4.1.2	Control of ATCRBS Synchronous Garble Control of Synchronous Garble by Transmitter Power Minimum Basic Whisper-Shout Sequence Higher Capability Whisper-Shout Sequences for Improved Degarbling Performance
2.4.2.1.1.3.2	Determination of Whisper-Shout Sequence	2.2.4.5.4 2.2.4.5.4.1.3 2.2.4.5.4.1.4 2.2.4.5.4.1.5	Control of ATCRBS Synchronous Garble Determination of Whisper-Shout Based on Synchronous Garble Criteria for Selection of a Specific Whisper-Shout Sequence Surveillance in Areas of No ATCRBS Intruder Aircraft
2.4.2.1.1.4	ACAS X Transmit Pulse Characteristics	2.2.3.7 2.2.3.7.1 2.2.3.7.2 2.2.4.5.4.2.1	Transmit Pulse Characteristics Mode C Transmissions Mode S Transmissions Directional Interrogation Beamwidth Control
2.4.2.1.1.5	Interrogation Spectrum	2.2.3.3	Interrogation Spectrum
2.4.2.1.1.6	Unwanted Output Power	2.2.3.2	Unwanted Output Power
2.4.2.1.1.7.1	Suppression Pulse Supplied by ACAS X	2.2.3.11	Aircraft Suppression Bus
2.4.2.1.1.7.2	Suppression Pulse Supplied by Other Avionics		
2.4.2.1.1.8	Interrogation Repetition Interval and Jitter	2.2.4.1 2.2.3.4	Surveillance Update Rate Interrogation Jitter
2.4.2.1.2	Receiver Characteristics	2.2.4.4.1	Receiver Sensitivity and Bandwidth
2.4.2.1.2.1	In-Band Acceptance	2.2.4.4.1.1	In-Band Acceptance
2.4.2.1.2.1.1	Ability to Operate Over the Frequency Band 1087 to 1093 MHz for ATCRBS and 1089 to 1091 MHz for Mode S Signals	2.2.4.4.1.1	In-Band Acceptance

Tests		Requirements	
2.4.2.1.2.1.2	Sensitivity and Dynamic Range at 1090 MHz	2.2.4.4.1.1	In-Band Acceptance
2.4.2.1.2.2	Out-of-Band Rejection	2.2.4.4.1.2	Out-of-Band Rejection
2.4.2.1.3	Reply Link Interference	2.2.4.5.1.2	Reply Link Interference
2.4.2.1.3.1	Mode C Reply Reception	2.2.4.4.2	Reply Detection and Decoding
		2.2.4.4.2.1	Mode C Reply Reception
	Narrow Pulse Discrimination	2.2.4.5.1.2.1	Narrow Pulse Discrimination
	TACAN and DME Discrimination	2.2.4.5.1.2.2	TACAN and DME Discrimination
2.4.2.1.3.2	Mode S Squitter and Reply Reception	2.2.4.4.2.2	Mode S Squitter and Reply Reception
	Narrow Pulse Discrimination	2.2.4.5.1.2.1	Narrow Pulse Discrimination
	TACAN and DME Discrimination	2.2.4.5.1.2.2	TACAN and DME Discrimination
2.4.2.1.4	Mode C Reply Reception	2.2.4.4.2.1	Mode C Reply Reception
2.4.2.1.4.1	Bracket Detection and Reply Decoding	2.2.4.4.2	Reply Detecton and Decoding
		2.2.4.4.2.1	Mode C Reply Reception
2.4.2.1.4.2	Wide Pulse Detection and Pulse Position Discrimination	2.2.4.4.2	Reply Detection and Decoding
		2.2.4.4.2.1	Mode C Reply Reception
		2.2.4.4.2.1c	Criteria for Acceptance of Garbled Mode C Replies
2.4.2.1.4.3	Narrow Pulse Rejection	2.2.4.4.2.1b	Criteria for Mode C Pulse Detection
		2.2.4.5.2	Narrow Pulse Rejection
2.4.2.1.4.4	Detection of Garbled Replies	2.2.4.4.2.1c	Criteria for Acceptance of Garbled Mode C Replies
2.4.2.1.4.5	Detection of Interleaved Replies	2.2.4.4.2.1c	Criteria for Acceptance of Garbled Mode C Replies
2.4.2.1.4.6	Phantom Rejection	2.2.4.4.2.1d	Phantom Rejection
2.4.2.1.4.7	TACAN and DME Pulse Rejection	2.2.4.5.3	TACAN and DME Signal Rejection
		2.2.4.5.1.2.2	TACAN and DME Discrimination
2.4.2.1.5	Mode S Squitter and Reply Reception	2.2.4.4.2.2	Mode S Squitter and Reply Reception
2.4.2.1.5.1	Mode S Preamble Reception	2.2.4.4.2.2b	Criteria for Preamble Acceptance
2.4.2.1.5.2	Mode S Squitter and Fruit Reply Reception	2.2.4.4.2.2c	Criteria for Block Acceptance in Squitter and Asynchronous Transmissions
2.4.2.1.5.3	Mode S Extended Squitter Reception (opt.)	2.2.3.9.8	Extended Squitter with Aircraft ID Message
		2.2.3.9.2.1	Detection
		2.2.4.6.4.2.1	Squitter Processing
		2.2.4.4.2.2c	Criteria for Block Acceptance in Squitter and Asynchronous Transmissions

Tests		Requirements	
2.4.2.1.5.4	Mode S Error Correction	2.2.4.4.2.2d	Additional Criterion for Data Block Acceptance in Discrete Transmissions
2.4.2.1.6	Mode C Target Surveillance Performance	2.2.4.6.4	Range and Altitude Estimation
2.4.2.1.6.1	Mode C Surveillance Initiation	2.2.4.6.4.1.2	Track Initiation
		2.2.4.5	Interference Rejection and Control
		2.2.4.6	Surveillance Tracking Requirements
2.4.2.1.6.2	Mode C Surveillance Extension	2.2.4.6.4.1.3	Maintenance of Established Tracks
		2.2.4.6.4.1.4	Multipath False Tracks
2.4.2.1.6.3	Missing Mode C Replies	2.2.4.6.4.1.3	Maintenance of Established Tracks
2.4.2.1.6.4	Surveillance Target Capacity, Mode C	2.2.4.6.1	Surveillance Target Track Capacity
2.4.2.1.6.5	Surveillance Overload	2.2.4.6.1.1	Surveillance Overload
2.4.2.1.7	Mode S Target Surveillance Performance	2.2.4.6.4.2	Mode S Targets
		2.2.4.4.2.2	Mode S Squitter and Reply Reception
2.4.2.1.7.1	Mode S Surveillance Initiation	2.2.4.6.2.2.2	Determination for Intruders Equipped with Mode S Transponders
		2.2.4.6.4.2.1	Squitter Processing
2.4.2.1.7.2	Mode S Range Acquisition	2.2.4.6.2.2.2	Determination for Intruders Equipped with Mode S Transponders
		2.2.4.6.4.2.2	Acquisition
		2.2.4.6.4.2.2.2	Acquisition Using Interrogations
		2.2.4.2	System Delay
		2.2.4.7	Antenna System
		2.2.4.7.4.1	Squitter Listening
2.4.2.1.7.3	Maintenance of Established Mode S Tracks	2.2.4.6.4.2.3	Maintenance of Established Tracks
		2.2.4.7	Antenna System
		2.2.5.5.6	ReceiveDF0
2.4.2.1.7.4	Interference Limiting	2.2.3.6	Interference Limiting
2.4.2.1.7.4.1	Interrogation Control of Airborne ACAS X	2.2.3.6.1	Interference Limiting Formulas
		2.2.3.6.2	Interference Limiting Procedures
		2.2.3.6.4	Interrogations From ACAS X Above 18,000 Ft Barometric Altitude
2.4.2.1.7.4.2	Interrogation Control of ACAS X On-The-Ground	2.2.3.6.3	Interrogations from ACAS X On-The-Ground

Tests		Requirements	
2.4.2.1.7.4.3	Correct Content of Transmitted TCAS Broadcast Interrogation Messages	2.2.3.8.3.2.4.2	Subfields in MU for a TCAS Broadcast Interrogation Message
		2.2.3.9.2.4	TCAS Broadcast Interrogations
2.4.2.1.7.4.4	Mode S Surveillance Special Functionality Test	2.2.3.6.1	Interference Limiting Formulas
		2.2.3.6.2	Interference Limiting Procedures
		2.2.3.6.3	Interrogations From ACAS X On The Ground
		2.2.4.6.4.2.1	Squitter Processing
		2.2.4.6.4.2.2	Acquisition
		2.2.4.6.4.2.2.2	Acquisition Using Interrogations
2.4.2.1.7.4.5	Interference Limiting and Required Interrogations	2.2.3.6.2	Interference Limiting Procedures
2.4.2.1.7.5	Surveillance Target Capacity and Overload, Mode S	2.2.3.10	Compatibility with Own Mode S Transponder
		2.2.4.6.1	Surveillance Target Track Capacity
		2.2.4.6.1.1	Mode S Overload
2.4.2.1.7.6	Mode S Power Programming	2.2.4.6.4.2.3.2.6	Power Programming
		2.2.4.6.4.2.4	NTA3/NTA6 Range Determination for Active CAS On-Ground Intruders
2.4.2.1.8.1	Surveillance Target Capacity	2.2.4.6.1	Surveillance Target Track Capacity
2.4.2.1.8.2	Altitude and Range Tracking of Mode C and Mode S	2.2.4.6	Surveillance Tracking Requirements
2.4.2.1.9.1	Bearing Accuracy with Standard Ground Plane	2.2.4.6.6.1	Bearing Estimation – General Requirements
		2.2.4.6.6.2	Bearing Accuracy with Standard Ground Plane
		2.2.4.6.6.2.1	Accuracy, -10 Deg. to +10 Deg. Elevation
		2.2.4.6.6.2.2	Accuracy, Greater Than 10 Deg. To +20 Deg. Elevation
2.4.2.1.9.2	Reply Processing	2.2.4.6.6.3	Bearing Accuracy in the Presence of Interference
2.4.2.1.9.2.1	Mode C Interleaved Replies	2.2.4.6.6.3.1	Mode C Interleaved Replies
2.4.2.1.9.2.2	Mode C Overlapped Replies	2.2.4.6.6.3.2	Mode C Overlapped Replies
2.4.2.1.9.2.3	Mode S Overlapped Replies	2.2.4.6.6.3.3	Mode S Overlapped Replies
2.4.2.1.9.3	Bearing Filter Performance	2.2.4.6.6.4	Bearing Filter Performance
2.4.2.1.9.3.1	Bearing Track and Coast	2.2.4.6.6.4.1	Bearing Filter
2.4.2.1.9.3.2	Filter Lag	2.2.4.6.6.4.1	Bearing Filter
2.4.2.1.9.4	Radiation Pattern	2.2.4.7.2.1	Transmit Radiation Pattern
		2.2.4.7.2.1	Transmit Radiation Pattern
2.4.2.1.9.5	Mode C Azimuth Filtering	2.2.4.6.4.1.2	Track Initiation

Tests		Requirements	
		2.2.4.6.4.1.3	Maintenance of Established Tracks
2.4.2.1.10.1	Use of Directional Interrogations for Mode C Surveillance and Bearing Receive Radiation Pattern	2.2.4.5.4.2.1	Directional Interrogation Beamwidth Control
		2.2.4.5.4.2.2	Directional Interrogation Radiated Power
		2.2.4.7.2.1	Transmit Radiation Pattern
		2.2.4.7.2.2	Receive Radiation Pattern
2.4.2.1.10.2	Use of Directional Antenna for TCAS Broadcast Interrogations	2.2.3.9.2.4	TCAS Broadcast Interrogations
		2.2.4.7.3	Use of Directional Antenna for Mode S Interrogations
2.4.2.3	External Parameter Selection	2.2.3.9.5.3	ACAS X Unit Part Number and ACAS X Software Part Number
		2.2.3.12.2.2.1	Data Provided to Own Transponder for Use in Special Surveillance Replies (DF=0, 16)
		2.2.3.12.2.2.4	Data Provided to Own Transponder for Use in Altitude and Identity Comm-B Replies (DF=20, 21)
2.4.2.3.1	Ground Control of Sensitivity Level	2.2.3.8.3.2.6	SL TCAS II Sensitivity Level Report
		2.2.3.12.2.3.2	Data Received in Altitude and Identity Comm-A Interrogations (UF=20, 21) from Mode S Ground Stations via Own Mode S Transponder
2.4.2.3.2	TA-Only Mode Selection	2.1.7.2	Minimum Flight Crew Control Instructions
		2.2.3.12.3	ACAS X Operating Mode Control
2.4.2.4	Coordination	2.2.5.5.11	ReceiveModeStatus
2.4.2.4.1	Transition of RA Report to Mode S Sensor	2.2.3.9.5.1	RA Report
		2.2.3.12.2.2.4	Data Provided to Own Transponder for Use in Altitude and Identity Comm-B Replies (DF=20, 21)
		2.2.5.5.5	ReceiveDiscretes
2.4.2.4.2	Transmission of RA Broadcast Interrogation	2.2.3.9.6.1	RA Broadcast Interrogations
2.4.2.4.3.1	Sense Selection and Communication	2.2.3.9.3.1	Determining Own Coordination Protocol
		2.2.3.9.3.2	Transmission of Coordination Interrogations to Other Active CAS Aircraft
		2.2.3.9.3.3	Capacity and Handling Requirements for Incoming Coordination Interrogations

Tests		Requirements	
2.4.2.4.3.3	ACAS X in Multi-Aircraft Conflict	2.2.3.8.3.2.3.1	Subfields in MB for RA Report
		2.2.3.9.3.2	Transmission of Coordination Interrogations to Other Active CAS Aircraft
2.4.2.4.3.4	Coordinaton Monitor	2.2.3.8.3.2.4.1	Subfields in MU for a TCAS Resolution Message
		2.2.7.2.2	Coordination Monitoring
2.4.2.4.3.5	Transponder to ACAS X Interface	2.2.3.9.3.3	Capacity and Handling Requirements for Incoming Coordination Interrogations
2.4.2.4.3.6	Coordination Timing	2.2.3.9.3.4	Coordination Delivery Delay Requirements
2.4.2.4.3.7	Use of Received Collision Avoidance Coordination Capability Bits (CCCB)	2.2.3.9.3.1	Determining Own Coordination Protocol
		2.2.3.9.3.6.1.2	Aircraft Operational Status Message
		2.2.3.9.3.6.2.1	Operational Coordination Message
2.4.2.4.3.8	Transmission of Own Collision Avoidance Coordination Capability Bits (CCCB)	2.2.3.12.2.1.3	Determination of Transponder OCM Transmit Capability and Setting of CCCB Value
2.4.2.5	ACAS X Capability and Part Number Reporting	2.2.3.9.5.3	ACAS X Unit Part Number and ACAS X Software Part Number
		2.2.3.12.2.2.1	Data Provided to Own Transponder for Use in Special Surveillance Replies (DF=0, 16)
		2.2.3.12.2.2.4	Data Provided to Own Transponder for Use in Altitude and Identity Comm-B Replies (DF=20, 21)
		2.2.5.5.5	ReceiveDiscretes
2.4.2.6	Transmission of Low-level Descend Inhibit (LDI) Information in RF Messages	2.2.3.8.3.2.3.1	Subfields in MB for RA Report
2.4.2.7	System Integration Tests	2.2.5.5	Combined STM/TRM Input Interfaces
		2.2.5.5.1	ReceiveBaroAltObservation
		2.2.5.5.2	ReceiveRadAltObservation
		2.2.5.5.3	ReceiveHeadingObservation
		2.2.5.5.4	ReceiveWgs84Observation
		2.2.5.5.5	ReceiveDiscretes
		2.2.5.5.6	ReceiveDF0
		2.2.5.5.7	ReceiveModeCReply
		2.2.5.5.8	ReceiveModeCReplies
		2.2.5.5.9	ReceiveStateVectorPosition
		2.2.5.5.10	ReceiveStateVectorVelocity

Tests		Requirements	
2.4.2.8.1.1.1	Ownship Symbol	2.2.5.5.11	ReceiveModeStatus
		2.2.5.5.12	ReceiveTargetDesignation
		2.2.6.1.2.1.1	Ownship Symbol
		2.2.6.1.2.6.2	Ownship Symbol and Location
		2.2.6.1.2.7.1.2	Ownship Symbol and Location
		2.2.6.1.2.7.2.6.2	Ownship Symbol and Location
		2.2.6.1.2.7.2.7.3	Ownship Symbol and Location
		2.2.6.1.2.7.2.8.4	Ownship Symbol and Location
		2.2.6.1.2.7.2.9.3	Ownship Symbol
		2.2.6.1.2.7.3.2	Ownship Symbol and Location
2.4.2.8.1.1.2	Range Rings	2.2.6.1.2.1.2	Range Rings
		2.2.6.1.2.2.2	Range Ring
		2.2.6.1.2.3.3	Range Rings
		2.2.6.1.2.7.2.7.2	Display Range and Range Ring
		2.2.6.1.2.7.2.8.2	Display Range and Range Rings
		2.2.6.1.2.7.2.9.2	Display Range and Range Rings
2.4.2.8.1.1.3	Threat Symbology	2.2.6.1.2.1.3	Traffic Symbology
		2.2.6.1.2.1.12	RA Annunciations
2.4.2.8.1.1.4	Off-Scale Symbology	2.2.6.1.2.1.4	Off-Scale Symbology
		2.2.6.1.2.1.11	ACAS X Operating Mode and Selected Display Range Annunciation
		2.2.6.1.2.1.12	RA Annunciations
2.4.2.8.1.1.5	Altitude Data Tag	2.2.6.1.2.1.5	Altitude Data Tag
		2.2.6.1.2.1.7	Non-Altitude Reporting Intruders
2.4.2.8.1.1.6	Intruder Vertical Speed Arrow	2.2.6.1.2.1.6	Intruder Vertical Speed Arrow
		2.2.4.6.5	Non-Altitude-Reporting Aircraft
2.4.2.8.1.1.7	No-Bearing Advisories	2.2.4.6.6.1	Bearing Estimation – General Requirements
		2.2.6.1.2.1.8	Intruder Bearing
		2.2.6.1.2.1.12	RA Annunciations
2.4.2.8.1.1.8	Display of Traffic	2.2.6.1.2.1.9	Display of Traffic
2.4.2.8.1.1.9	Altitude Band for the Display	2.2.6.1.2.1.10	Altitude Band for the Display
		2.2.6.5.2	Traffic Display Controls
2.4.2.8.1.1.10	Display of ACAS X Operating Mode	2.2.6.1.2.1.11	ACAS X Operating Mode and Selected Display Range Annunciation
		2.2.6.1.2.1.14	Status and Failure Annunciations
		2.2.6.6.2	Traffic Display Annunciations
2.4.2.8.1.3.1	Display Range	2.2.6.1.2.3.1	Display Range
2.4.2.8.1.3.2	Range Selection	2.2.6.1.2.3.2	Range Selection
2.4.2.8.1.4.1	Activation and Information Displayed	2.2.6.1.2.4.1	Activation
		2.2.6.1.2.4.2	Information Displayed

Tests		Requirements	
2.4.2.8.1.7.1 .1	Available Display Modes	2.2.6.1.2.7.1.1	Display Modes
2.4.2.8.1.7.1 .2	Display Mode Characteristics	2.2.6.1.2.7.1.1	Display Modes
2.4.2.8.1.7.2 .1	Failure Annunciations	2.2.6.1.2.7.2.2	Failure Annunciations
2.4.2.8.1.7.2 .2	Display of Traffic	2.2.6.1.2.7.2.3	Traffic Display
2.4.2.8.1.7.2 .3	ACAS X and Weather Information	2.2.6.1.2.7.2.4	ACAS X and Weather Information
2.4.2.8.1.7.2 .4	ACAS X and Navigation Information	2.2.6.1.2.7.2.5	ACAS X and Navigation Information
2.4.2.8.1.7.2 .6	EXPANDED Mode	2.2.6.1.2.7.2.7.1	Display Characteristics
2.4.2.8.1.7.2 .7.1	Display Characteristics	2.2.6.1.2.7.2.8.1	Display Characteristics
2.4.2.8.1.7.2 .7.2	Display Range and Orientation	2.2.6.1.2.7.2.8.2	Display Range and Range Rings
		2.2.6.1.2.7.2.8.3	Display Orientation
2.4.2.8.1.7.2 .8.1	Display Characteristics	2.2.6.1.2.7.2.9.1	Display Characteristics
		2.2.6.1.2.7.2.9.3	Ownship Symbol
		2.2.6.1.2.7.2.9.4	Display Orientation
		2.2.6.1.2.7.2.9.5	TA or RA Annunciation
2.4.2.8.1.7.2 .8.2	Display Orientation	2.2.6.1.2.7.2.9.6	Selection of Heading Up Display
2.4.2.8.1.7.3	EICAS or Systems Displays	2.2.6.1.2.7.3.3	Traffic Display Inhibit
2.4.2.8.1.7.4 .1	VSI Requirements	2.2.6.1.2.7.4.1.2	VSI Requirements
2.4.2.8.1.7.4 .2	Interference with Vertical Speed Scale	2.2.6.1.2.7.4.2.3	Interference with Vertical Speed Scale
2.4.2.8.2.1	Round Dial VSI	2.2.6.2.2	RA/VSI (Round dial VSI)
		2.2.6.2.2.1.1	Red Arcs
		2.2.6.2.2.1.2	Green Arcs
		2.2.6.2.2.1.3	Black Arcs
		2.2.6.2.2.1.4	Multi-aircraft Encounters
2.4.2.8.2.1.1	VSI Range	2.2.6.2.2	RA/VSI (Round dial VSI)
		2.2.6.2.3.3	VSI Scale
2.4.2.8.2.2	Vertical Speed Tape	2.2.6.2.3	RA/VSI (Integrated Tape VSI on a Primary Flight Display [PFD])
		2.2.6.2.3.1.1	Red Zone
		2.2.6.2.3.1.2	Green Zone
		2.2.6.2.3.1.3	Black Zones
		2.2.6.2.3.1.4	Multi-aircraft Encounters
2.4.2.8.2.2.1	Vertical Speed Tape Range	2.2.6.2.3	RA/VSI (Integrated Tape VSI on a Primary Flight Display [PFD])
2.4.2.8.2.3	Pitch Cues on the PFD	2.2.6.2.4.1	No-Fly Pitch Angles
		2.2.6.2.4.2	Mode Annunciation

Tests		Requirements	
2.4.2.8.2.5	HUD	2.2.6.2.4.3	Multi-aircraft Encounters
		2.2.6.4.3	Message on PFD
		2.2.6.2.6	Heads-up Display
		2.2.6.2.6.1	No-Fly Zone
		2.2.6.2.6.2	Mode Annunciation
		2.2.6.2.6.3	Flight Path Target
		2.2.6.2.6.4	Multi-aircraft Encounters
		2.2.6.2.6.5	Display Decluttering
		2.2.6.2.6.6	RA Guidance Availability
2.4.2.8.3.1	ACAS X/Mode S Controls	2.2.6.5.1	ACAS X/Mode S Controls
2.4.2.8.3.2	Traffic Display Controls	2.2.6.5.2	Traffic Display Controls
2.4.2.8.3.3	Shared Weather Radar/Traffic Display Controls	2.2.6.5.3.1	WX/ACAS X Mode
		2.2.6.5.3.2	WX-and-Traffic Mode
		2.2.6.5.3.3	Optional Mode Selection
2.4.2.8.4.1	Traffic Display Annunciations	2.2.6.6.1	Status and Failure Annunciations - General
2.4.2.8.4.2	RA Display Annunciations	2.2.6.6.2	Traffic Display Annunciations
2.4.2.9	Automatic Performance Monitoring and Self Test	2.2.7	Monitor Requirements
2.4.2.9.1	Automatic Performance Monitoring	2.2.7.1.1	Failure Response
2.4.2.9.2	Self-Test	2.2.7.1.3	Self-Test
2.4.2.9.3	Own Transponder ICAO 24-Bit Aircraft Address	2.2.7.2.6	ICAO 24-Bit Aircraft Address
2.4.2.10.1	ACAS X to Mode S Transponder Interference	2.1.10.1	Performance Compatibility with Ownship ACAS X
2.4.2.10.2	Mode S Transponder to ACAS X Interference	2.2.3.10	Compatibility With Own Mode S Transponder
2.4.2.11.2	Verification of Shared Use of 1090 MHz Receiver with an ADS-B Receiving Subsystem	2.1.13	Shared Use of 1090 MHz Receiver with an ADS-B Receiving Subsystem
2.4.2.11.3	Verification of Acquisition and Maintenance of Established Tracks Using Active Surveillance	2.2.4.6.4.2.2.1 Tests 2a, 2b	Acquisition Using Passive Position Reports
		2.2.4.6.4.2.2.1.1 Tests 2a, 2b	Extended Hybrid Surveillance Quality Requirements
		2.2.4.6.4.2.3.1.3 Tests 2a, 2b	Transitions to Extended Hybrid Surveillance
		2.2.4.6.4.2.3.1.7 Test 1 - Intruders 4,5,6,7,8,9	Transitions to Hybrid Surveillance
		2.2.4.6.4.2.3.2	Maintenance of Tracks Using Active Surveillance
		2.2.4.6.4.2.3.2.2 Test 1 - Intruders 1,2,3,7,8,9	Persistence of Active Surveillance

Tests		Requirements	
		2.2.4.6.4.2.3.2.3 Test 1 – Intruders 1-7	Active Surveillance Region
		2.2.4.6.4.2.3.2.7 Test 1 - Intruders 4-9	Validation of Airborne Position Message Data
		2.2.3.8.3.2.8.2	Altitude from Airborne Position Message
2.4.2.11.4	Verification of Maintenance of Established Tracks Using Passive Surveillance	2.2.3.8.3.2.8.1	TYPE Subfield of the ME Field
		2.2.3.8.3.2.8.2	Altitude from Airborne Position Message
		2.2.3.8.3.2.8.3	CPR Format from Airborne Position Message
		2.2.3.8.3.2.8.4	Encoded Latitude from Airborne Position Message
		2.2.3.8.3.2.8.5	Encoded Longitude from Airborne Position Message
		2.2.3.8.3.2.8.6	ADS-B Version Number from Aircraft Operational Status Message
		2.2.3.8.3.2.8.7	Navigational Integrity Category (NIC) from Airborne Position Message
		2.2.3.8.3.2.8.8	Navigational Accuracy Category for Position (NACp)
		2.2.3.8.3.2.8.8.1	NACp from Aircraft Operational Status Message
		2.2.3.8.3.2.8.8.2	NACp from Target State and Status Message
		2.2.3.8.3.2.8.9	Source Integrity Level (SIL)
		2.2.3.8.3.2.8.9.1	SIL from Aircraft Operational Status Message
		2.2.3.8.3.2.8.9.2	SIL from Target State and Status Message
		2.2.3.8.3.2.8.10	System Design Assurance (SDA)
		2.2.4.6.3 Test 10	Determination of Ownship Air-Ground Status
		2.2.4.6.4.2.2.1 Tests 5, 6, 7, 8, 9 10	Acquisition Using Passive Position Reports
		2.2.4.6.4.2.2.1.1 Tests 5, 6, 7, 8, 9	Extended Hybrid Surveillance Quality Requirements
		2.2.4.6.4.2.2.1.2 Test 4	Establishing an Extended Hybrid Surveillance Track
		2.2.4.6.4.2.2.1.3 Test 8, 9	Extended Hybrid Surveillance MTL

Tests		Requirements	
		2.2.4.6.4.2.2.1.4 Test 12	Determination of Estimated Signal Strength
		2.2.4.6.4.2.3.1.2 Test 5 Test 10	Persistence of Extended Hybrid Surveillance
		2.2.4.6.4.2.3.1.5 Test 10	Persistence of Hybrid Surveillance
		2.2.4.6.4.2.3.1.8 Test 2 – All intruders Test 6 Test 7 Test 15	Track Updates Using Airborne Position Messages
		2.2.4.6.4.2.3.1.9 Test 2 – Intruder 3, 4 Test 15	Tracking in the Absence of Airborne Position Messages
		2.2.4.6.4.2.3.1.10 Test 1 – All intruders Test 11 Test 14	Revalidation
		2.2.4.6.4.2.3.1.11 Test 10	Error Budget Allocated to ACAS X for Slant Range Validation
		2.2.4.6.4.2.3.2.3 Test 1 - Intruders 1,2 Test 3 Test 5	Active Surveillance Region
		2.2.4.6.4.2.3.2.7 Test 1 – Intruder 3 Test 13	Validation of Airborne Position Message Data
		2.2.7	Monitor Requirements
2.4.2.11.5	Verification of Requirements Related to Transitions between Passive and Active Surveillance	2.2.3.8.3.2.8.1	TYPE Subfield of the ME Field
		2.2.3.8.3.2.8.2 Test 3, Intruder 1 Test 3A, Intruder 1	Altitude from Airborne Position Message
		2.2.3.8.3.2.8.3	CPR Format from Airborne Position Message
		2.2.3.8.3.2.8.4	Encoded Latitude from Airborne Position Message
		2.2.3.8.3.2.8.5	Encoded Longitude from Airborne Position Message

Tests	Requirements
	2.2.3.8.3.2.8.6 ADS-B Version Number from Aircraft Operational Status Message
	2.2.3.8.3.2.8.7 Navigational Integrity Category (NIC) from Airborne Position Message
	2.2.3.8.3.2.8.8 Navigational Accuracy Category for Position (NACp)
	2.2.3.8.3.2.8.8.1 NACp from Aircraft Operational Status Message
	2.2.3.8.3.2.8.8.2 NACp from Target State and Status Message
	2.2.3.8.3.2.8.9 Source Integrity Level (SIL)
	2.2.3.8.3.2.8.9.1 SIL from Aircraft Operational Status Message
	2.2.3.8.3.2.8.9.2 SIL from Target State and Status Message
	2.2.3.8.3.2.8.10 System Design Assurance (SDA)
2.2.4.6.3 Test 5	Determination of Ownship Air-Ground Status
2.2.4.6.4.2.2.1.1 Test 4, 3A	Extended Hybrid Surveillance Quality Requirements
2.2.4.6.4.2.3.1.3 Test 4, Intruder 8	Transitions to Extended Hybrid Surveillance
2.2.4.6.4.2.3.1.5 Test 1 – All intruders Test 2 – Intruders 1,2	Persistence of Hybrid Surveillance
2.2.4.6.4.2.3.1.6 Test 2 – Intruders 1,2 Test 4 Test 5	Hybrid Surveillance Region
2.2.4.6.4.2.3.1.7 Test 2 – Intruder 3 Test 4, Intruders 5, 6, 7	Transitions to Hybrid Surveillance
2.2.4.6.4.2.3.1.9 Test 3 – Intruders 1, 5, 6	Tracking in the Absence of Airborne Position Messages

Tests		Requirements	
		2.2.4.6.4.2.3.1.10 Test 2 – Intruder 3 Test 3 – Intruders 2-6 Test 3A – Intruders 2,3 Test 4 – Intruders 6,7	Revalidation
		2.2.4.6.4.2.3.2.3 Test 1 – Intruders 1-4 Test 2 – All intruders Test 3 – Intruder 7 Test 4 Test 5	Active Surveillance Region
		2.2.4.6.4.2.3.2.4 Test 1 – Intruder 1 Test 2 – Intruder 3	Passive to Active Surveillance Transition
		2.2.4.6.4.2.3.2.5 Test 1 – Intruder 1 Test 2 – Intruder 3	Track Updates Using Active Surveillance
		2.2.4.6.4.2.3.2.7 Test 1, Intruder 1 Test 2, Intruder 2	Validation of Airborne Position Message Data
2.4.2.11.6	Verification of Error Budget in Computing Slant Range from Passive Data	2.2.4.6.4.2.3.1.11	Error Budget Allocated to ACAS X for Slant Range Validation
2.4.2.11.8	Verification of DF17 Decoding	2.2.3.8.3.2.8.1	TYPE Subfield of the ME Field
		2.2.3.8.3.2.8.3 Test 1	CPR Format from Airborne Position Message
		2.2.3.8.3.2.8.4 Test 1	Encoded Latitude from Airborne Position Message
		2.2.3.8.3.2.8.5 Test 1	Encoded Longitude from Airborne Position Message
2.4.2.11.9	Verification of Monitoring Requirements	2.2.7.2 Test 5	Monitoring of ACAS X Components
2.4.2.11.10	Verification of On-Ground ACAS X Range Determination Using ADS-B	2.2.4.6.4.2.4	NTA3/NTA6 Range Determination for Active CAS On-Ground Intruders
2.4.2.11.11	Verification of DF=18 ADS-B and ADS-R Only Passive Surveillance	2.2.5.5.4	ReceiveWgs84Observation
		2.2.5.5.9	ReceiveStateVectorPosition
		2.2.5.5.10	ReceiveStateVectorVelocity
2.4.3	Computer Performance Verification	2.2.7.2.1	Computer Monitoring
3.2.9	Number of Intruders Displayed	2.2.6.1.2.1.9	Display of Traffic

Tests		Requirements	
3.2.10	Display of Traffic on EICAS/SYSTEMS Displays	2.2.6.1.2.7.3.3	Traffic Display Inhibit
3.2.12	Red Visual Alert	2.2.6.4.2	RAs
3.4.2.4.1	Pitch Cues	3.2.13.1	Pitch Cues on PFD and HUD
3.4.2.4.2	Flight Director	3.2.13.2	Flight Director
3.4.2.4.3	Auto Pilot	3.2.13.3	Autopilot TCAS Mode to Perform RA Maneuvers
3.4.4.1	Mode C Surveillance Flight Tests	2.2.4.6.4.1.1	Reply Merging
		2.2.5.5.6	ReceiveDF0
		2.2.5.5.8	ReceiveModeCReplies

Table 2-93: Cross-Reference of Associated Tests to Requirements

Requirements	Tests		
2.1.7.2	Minimum Flight Crew Control Instructions	2.4.2.3.2	TA-Only Mode Selection
2.1.10.1	Performance Compatibility with Ownship ACAS X	2.4.2.10.1	ACAS X to Mode S Transponder Interference
2.1.13	Shared Use of 1090 MHz Receiver with an ADS-B Receiving Subsystem	2.4.2.11.2	Verification of Shared Use of 1090 MHz Receiver with an ADS-B Receiving Subsystem
2.2.3.1	Radiated Output Power	2.4.2.1.1.2	Radiated Output Power
2.2.3.2	Unwanted Output Power	2.4.2.1.1.6	Unwanted Output Power
2.2.3.3	Interrogation Spectrum	2.4.2.1.1.5	Interrogation Spectrum
2.2.3.4	Interrogation Jitter	2.4.2.1.1.8	Interrogation Repetition Interval and Jitter
2.2.3.5	Transmit Frequency and Tolerance	2.4.2.1.1.1	Transmit Frequency
2.2.3.6	Interference Limiting	2.4.2.1.7.4	Interference Limiting
2.2.3.6.1	Interference Limiting Formulas	2.4.2.1.7.4.1	Interrogation Control of Airborne ACAS X
		2.4.2.1.7.4.4	Mode S Surveillance Special Functionality Test
2.2.3.6.2	Interference Limiting Procedures	2.4.2.1.7.4.1	Interrogation Control of Airborne ACAS X
		2.4.2.1.7.4.4	Mode S Surveillance Special Functionality Test
		2.4.2.1.7.4.5	Interference Limiting and Required Interrogations
2.2.3.6.3	Interrogations from ACAS X On The Ground	2.4.2.1.7.4.2	Interrogation Control of ACAS X On The Ground
		2.4.2.1.7.4.4	Mode S Surveillance Special Functionality Test
2.2.3.6.4	Interrogations From ACAS X Above 18,000 Ft Barometric Altitude	2.4.2.1.7.4.1	Interrogation Control of Airborne ACAS X
2.2.3.7	Transmit Pulse Characteristics	2.4.2.1.1.4	ACAS X Transmit Pulse Characteristics
2.2.3.7.1	Mode C Transmissions	2.4.2.1.1.4	ACAS X Transmit Pulse Characteristics
2.2.3.7.2	Mode S Transmissions	2.4.2.1.1.4	ACAS X Transmit Pulse Characteristics
2.2.3.8.3.2.3.1	Subfields in MB for RA Report	2.4.2.4.3.3	ACAS X in Multi-Aircraft Conflict
		2.4.2.6	Transmission of Low-level Descend Inhibit (LDI) Information in RF Messages
2.2.3.8.3.2.4.1	Subfields in MU for a TCAS Resolution Message	2.4.2.4.3.4	Coordination Monitor

Requirements	Tests		
2.2.3.8.3.2.4.2	Subfields in MU for a TCAS Broadcast Interrogation Message	2.4.2.1.7.4.3	Correct Content of Transmitted TCAS Broadcast Interrogation Messages
2.2.3.8.3.2.6	SL TCAS II Sensitivity Level Report	2.4.2.3.1	Ground Control of Sensitivity Level
2.2.3.8.3.2.8.1	TYPE Subfield of the ME Field	2.4.2.11.4	Verification of Maintenance of Established Tracks Using Passive Surveillance
		2.4.2.11.5	Verification of Requirements Related to Transitions between Passive and Active Surveillance
		2.4.2.11.8	Verification of DF17 Decoding
2.2.3.8.3.2.8.2	Altitude from Airborne Position Message	2.4.2.11.3	Verification of Acquisition and Maintenance of Established Tracks Using Active Surveillance
		2.4.2.11.4	Verification of Maintenance of Established Tracks Using Passive Surveillance
		2.4.2.11.5	Verification of Requirements Related to Transitions between Passive and Active Surveillance
2.2.3.8.3.2.8.3	CPR Format from Airborne Position Message	2.4.2.11.4	Verification of Maintenance of Established Tracks Using Passive Surveillance
2.2.3.8.3.2.8.4	Encoded Latitude from Airborne Position Message	2.4.2.11.5	Verification of Requirements Related to Transitions between Passive and Active Surveillance Verification of DF17 Decoding
2.2.3.8.3.2.8.5	Encoded Longitude from Airborne Position Message		
2.2.3.8.3.2.8.6	ADS-B Version Number from Aircraft Operational Status Message	2.4.2.11.4	Verification of Maintenance of Established Tracks Using Passive Surveillance
2.2.3.8.3.2.8.7	Navigation Integrity Code (NIC) from Airborne Position Message	2.4.2.11.5	Verification of Requirements Related to Transitions between Passive and Active Surveillance
2.2.3.8.3.2.8.8	Navigation Accuracy Category for Position (NACp)		
2.2.3.8.3.2.8.8.1	NACp from Aircraft Operational Status Message		
2.2.3.8.3.2.8.8.2	NACp from Target State and Status Message		
2.2.3.8.3.2.8.9	Source Integrity Level (SIL)		
2.2.3.8.3.2.8.9.1	SIL from Aircraft Operational Status Message		
2.2.3.8.3.2.8.9.2	SIL from Target State and Status Message		

Requirements	Tests		
2.2.3.8.3.2.8.10	System Design Assurance (SDA)		
2.2.3.9.2.1	Detection	2.4.2.1.5.3	Mode S Extended Squitter Reception (opt.)
2.2.3.9.2.4	TCAS Broadcast Interrogations	2.4.2.1.10.2	Use of Directional Antenna for TCAS Broadcast Interrogations
		2.4.2.1.7.4.3	Correct Content of Transmitted TCAS Broadcast Interrogation Messages
2.2.3.9.3.1	Determining Own Coordination Protocol	2.4.2.4.3.1	Sense Selection and Communication
		2.4.2.4.3.7	Use of Received Collision Avoidance Coordination Capability Bits (CCCB)
2.2.3.9.3.2	Transmission of Coordination Interrogations to Other Active CAS Aircraft	2.4.2.4.3.1	Sense Selection and Communication
		2.4.2.4.3.3	ACAS X in Multi-Aircraft Conflict
2.2.3.9.3.3	Capacity and Handling Requirements for Incoming Coordination Interrogations	2.4.2.4.3.1	Sense Selection and Communication
		2.4.2.4.3.5	Transponder to ACAS X Interface
2.2.3.9.3.4	Coordination Deliver Delay Requirements	2.4.2.4.3.6	Coordination Timing
2.2.3.9.3.6.1.2	Aircraft Operational Status Message	2.4.2.4.3.7	Use of Received Collision Avoidance Coordination Capability Bits (CCCB)
2.2.3.9.3.6.2.1	Operational Coordination Message	2.4.2.4.3.7	Use of Received Collision Avoidance Coordination Capability Bits (CCCB)
2.2.3.9.5.1	RA Report	2.4.2.4.1	Transmission of RA Report to Mode S Sensor
2.2.3.9.5.3	ACAS X Unit Part Number and ACAS X Software Part Number	2.4.2.5	ACAS X Capability and Part Number Reporting
2.2.3.9.6.1	RA Broadcast Interrogations	2.4.2.4.2	Transmission of RA Broadcast Interrogation
2.2.3.9.8	Extended Squitter with Aircraft ID Message	2.4.2.1.5.3	Mode S Extended Squitter Reception (opt.)
2.2.3.10	Compatibility with Own Mode S Transponder	2.4.2.1.7.5	Surveillance Target Capacity and Overload, Mode S
		2.4.2.10.2	Mode S Transponder to ACAS X Interference
2.2.3.11	Aircraft Suppression Bus	2.4.2.1.1.7.1	Suppression Pulse Supplied by ACAS X
		2.4.2.1.1.7.2	Suppression Pulse Supplied by Other Avionics

Requirements	Tests		
2.2.3.12.2.1.3	Determination of Transponder OCM Transmit Capability and Setting of CCCB Value	2.4.2.4.3.8	Transmission of Own Collision Avoidance Coordination Capability Bits (CCCB)
2.2.3.12.2.2.1	Data Provided to Own Transponder for Use in Special Surveillance Replies (DF=0, 16)	2.4.2.3	External Parameter Selection
		2.4.2.5	ACAS X Capability and Part Number Reporting
2.2.3.12.2.2.4	Data Provided to Own Transponder for Use in Altitude and Identity Comm-B Replies (DF=20, 21)	2.4.2.3	External Parameter Selection
		2.4.2.4.1	Transmission of RA Report to Mode S Sensor
		2.4.2.5	ACAS X Capability and Part Number Reporting
2.2.3.12.2.3.2	Data Received in Altitude and Identity Comm-A Interrogations (UF=20, 21) from Mode S Ground Stations via Own Mode S Transponder	2.4.2.3.1	Ground Control of Sensitivity Level
2.2.3.12.3	ACAS X Operating Mode Control	2.4.2.3.2	TA-Only Mode Selection
2.2.4.1	Surveillance Update Rate	2.4.2.1.1.8	Interrogation Repetition Interval and Jitter
2.2.4.2	System Delay	2.4.2.1.7.2	Mode S Range Acquisition
2.2.4.4.1	Receiver Sensitivity and Bandwidth	2.4.2.1.2	Receiver Characteristics
2.2.4.4.1.1	In-Band Acceptance	2.4.2.1.2.1.1	Ability to Operate Over the Frequency Band 1087 to 1093 MHz for ATCRBS and 1089 to 1091 MHz for Mode S signals
		2.4.2.1.2.1.2	Sensitivity and Dynamic Range at 1090 MHz
2.2.4.4.1.2	Out-of-Band Rejection	2.4.2.1.2.2	Out-of-Band Rejection
2.2.4.4.2	Reply Detection and Decoding	2.4.2.1.3.1	Mode C Reply Reception, Narrow Pulse Discrimination, and TACAN and DME Discrimination
		2.4.2.1.4.1	Bracket Detection and Reply Decoding
		2.4.2.1.4.2	Wide Pulse Detection and Pulse Position Discrimination
2.2.4.4.2.1	Mode C Reply Reception	2.4.2.1.3.1	Mode C Reply Reception, Narrow Pulse Discrimination, and TACAN and DME Discrimination
		2.4.2.1.4.1	Bracket Detection and Reply Decoding

Requirements	Tests		
		2.4.2.1.4.2	Wide Pulse Detection and Pulse Position Discrimination
2.2.4.4.2.1b	Criteria for Mode C Pulse Detection	2.4.2.1.4.3	Narrow Pulse Rejection
2.2.4.4.2.1c	Criteria for Acceptance of Garbled Mode C Replies	2.4.2.1.4.2	Wide Pulse Detection and Pulse Position Discrimination
		2.4.2.1.4.4	Detection of Garbled Replies
		2.4.2.1.4.5	Detection of Interleaved Replies
2.2.4.4.2.1d	Phantom Rejection	2.4.2.1.4.6	Phantom Rejection
2.2.4.4.2.2	Mode S Squitter and Reply Reception	2.4.2.1.3.2	Mode S Squitter and Reply Reception, Narrow Pulse Discrimination, and TACAN and DME Discrimination
		2.4.2.1.5	Mode S Squitter and Reply Reception
2.2.4.4.2.2b	Criteria for Preamble Acceptance	2.4.2.1.5.1	Mode S Preamble Acceptance
2.2.4.4.2.2c	Criteria for Data Block Acceptance in Squitter and Asynchronous Transmissions	2.4.2.1.5.2	Mode S Squitter and Fruit Reply Reception
		2.4.2.1.5.3	Mode S Extended Squitter Reception (opt.)
2.2.4.4.2.2d	Additional Criterion for Data Block Acceptance in Discrete Transmissions	2.4.2.1.5.4	Mode S Error Correction
2.2.4.5	Interference Rejection and Control	2.4.2.1.6.1	Mode C Surveillance Initiation
2.2.4.5.1.2	Reply Link Interference	2.4.2.1.3	Reply Link Interference
2.2.4.5.1.2.1	Narrow Pulse Discrimination	2.4.2.1.3.1	Mode C Reply Reception
		2.4.2.1.3.2	Mode S Squitter and Reply Reception
2.2.4.5.1.2.2	TACAN and DME Discrimination	2.4.2.1.3.1	Mode C Reply Reception
		2.4.2.1.3.2	Mode S Squitter and Reply Reception
		2.4.2.1.4.7	TACAN and DME Rejection
2.2.4.5.2	Narrow Pulse Rejection	2.4.2.1.4.3	Narrow Pulse Rejection
2.2.4.5.3	TACAN and DME Signal Rejection	2.4.2.1.4.7	TACAN and DME Rejection
2.2.4.5.4	Control of ATCRBS Synchronous Garble	2.4.2.1.1.3.1	Control of Synchronous Garble by Whisper-Shout
		2.4.2.1.1.3.2	Determination of Whisper-Shout Sequence
2.2.4.5.4.1	Control of Synchronous Garble by Transmitter Power	2.4.2.1.1.3	Control of Synchronous Garble by Transmitter Power
2.2.4.5.4.1.1	Minimum Basic Whisper-Shout Sequence	2.4.2.1.1.3.1	Control of Synchronous Garble by Whisper-Shout

Requirements	Tests		
2.2.4.5.4.1.2	Higher Capability Whisper-Shout Sequences	2.4.2.1.1.3.1	Control of Synchronous Garble by Whisper-Shout
2.2.4.5.4.1.3	Determination of Whisper-Shout Based on Synchronous Garble	2.4.2.1.1.3.2	Determination of Whisper-Shout Sequence
2.2.4.5.4.1.4	Criteria for Selection of a Specific Whisper-Shout Sequence	2.4.2.1.1.3.2	Determination of Whisper-Shout Sequence
2.2.4.5.4.1.5	Surveillance In Areas of No ATCRBS Intruder Aircraft	2.4.2.1.1.3.2	Determination of Whisper-Shout Sequence
2.2.4.5.4.2.1	Directional Interrogation Beamwidth Control	2.4.2.1.1.4	ACAS X Transmit Pulse Characteristics
		2.4.2.1.10.1	Use of Directional Interrogations for Mode C Surveillance and Bearing Receive Radiation Pattern
2.2.4.5.4.2.2	Directional Interrogation Radiated Power	2.4.2.1.10.1	Use of Directional Interrogations for Mode C Surveillance and Bearing Receive Radiation Pattern
2.2.4.6	Surveillance Tracking Requirements	2.4.2.1.6.1	Mode C Surveillance Initiation
		2.4.2.1.8.2	Altitude and Range Tracking of Mode C and Mode S
2.2.4.6.1	Surveillance Target Track Capacity	2.4.2.1.6.4	Surveillance Target Track Capacity, Mode C
		2.4.2.1.7.5	Surveillance Target Capacity and Overload, Mode S
		2.4.2.1.8.1	Surveillance Target Capacity
2.2.4.6.1.1	Surveillance Overload	2.4.2.1.6.5	Surveillance Overload
		2.4.2.1.7.5	Surveillance Target Capacity and Overload, Mode S
		2.4.2.11.7	Surveillance Overload and Capacity Tests (test 3)
2.2.4.6.2.2.2	Determination for Intruders Equipped with Mode S Transponders	2.4.2.1.7.1	Mode S Surveillance Initiation
		2.4.2.1.7.2	Mode S Range Acquisition
2.2.4.6.3	Determination of Ownship Air-Ground Status	2.4.2.11.4 Test 10	Verification of Maintenance of Established Tracks Using Passive Surveillance
		2.4.2.11.5 Test 5	Verification of Requirements Related to Transitions between Passive and Active Surveillance
2.2.4.6.4	Range and Altitude Estimation	2.4.2.1.6	Mode C Target Surveillance Performance
2.2.4.6.4.1.1	Reply Merging	3.4.4.1	Mode C Surveillance Flight Tests
2.2.4.6.4.1.2	Track Initiation	2.4.2.1.6.1	Mode C Surveillance Initiation
		2.4.2.1.9.5	Mode C Azimuth Filtering
2.2.4.6.4.1.3		2.4.2.1.6.2	Mode C Surveillance Extension

Requirements	Tests		
	Maintenance of Established Tracks	2.4.2.1.6.3	Missing Mode C Replies
		2.4.2.1.9.5	Mode C Azimuth Filtering
2.2.4.6.4.1.4	Multipath False Targets	2.4.2.1.6.2	Mode C Surveillance Extension
2.2.4.6.4.2	Mode S Targets	2.4.2.1.7	Mode S Target Surveillance Performance
2.2.4.6.4.2.1	Squitter Processing	2.4.2.1.5.3	Mode S Extended Squitter Reception
		2.4.2.1.7.1	Mode S Surveillance Initiation
		2.4.2.1.7.4.4	Mode S Surveillance Special Functionality Test
2.2.4.6.4.2.2	Acquisition	2.4.2.1.7.2	Mode S Range Acquisition
		2.4.2.1.7.4.4	Mode S Surveillance Special Functionality Test
2.2.4.6.4.2.2.1	Acquisition Using Passive Position Reports	2.4.2.11.3	Verification of Acquisition and Maintenance of Established Tracks Using Active Surveillance
		2.4.2.11.4	Verification of Maintenance of Established Tracks Using Passive Surveillance
2.2.4.6.4.2.2.1.1	Extended Hybrid Surveillance Traffic Quality Requirements	2.4.2.11.3	Verification of Acquisition and Maintenance of Established Tracks Using Active Surveillance
		2.4.2.11.4	Verification of Maintenance of Established Tracks Using Passive Surveillance
		2.4.2.11.5	Verification of Requirements Related to Transitions between Passive and Active Surveillance
2.2.4.6.4.2.2.1.2	Establishing an Extended Hybrid Surveillance Track	2.4.2.11.4	Verification of Maintenance of Established Tracks Using Passive Surveillance
2.2.4.6.4.2.2.1.3	Extended Hybrid Surveillance MTL	2.4.2.11.4	Verification of Maintenance of Established Tracks Using Passive Surveillance
2.2.4.6.4.2.2.1.4	Determination of Estimated Signal Strength	2.4.2.11.4	Verification of Maintenance of Established Tracks Using Passive Surveillance
2.2.4.6.4.2.2.2	Acquisition Using Interrogations	2.4.2.1.7.2	Mode S Range Acquisition
		2.4.2.1.7.4.4	Surveillance Special Functionality Test
2.2.4.6.4.2.3	Maintenance of Established Tracks	2.4.2.1.7.3	Maintenance of Established Mode S Tracks
2.2.4.6.4.2.3.1.2	Persistence of Extended Hybrid Surveillance	2.4.2.11.4	Verification of Maintenance of Established Tracks Using Passive Surveillance

Requirements	Tests		
2.2.4.6.4.2.3.1.3	Transitions to Extended Hybrid Surveillance - Active to Extended Hybrid Surveillance Transition	2.4.2.11.3	Verification of Acquisition and Maintenance of Established Tracks Using Active Surveillance
		2.4.2.11.5	Verification of Requirements Related to Transitions between Passive and Active Surveillance
2.2.4.6.4.2.3.1.5	Persistence of Hybrid Surveillance	2.4.2.11.4	Verification of Maintenance of Established Tracks Using Passive Surveillance
		2.4.2.11.5	Verification of Requirements Related to Transitions between Passive and Active Surveillance
2.2.4.6.4.2.3.1.6	Hybrid Surveillance Region	2.4.2.11.5	Verification of Requirements Related to Transitions between Passive and Active Surveillance
2.2.4.6.4.2.3.1.7	Transitions to Hybrid Surveillance	2.4.2.11.3	Verification of Acquisition and Maintenance of Established Tracks Using Active Surveillance
		2.4.2.11.5	Verification of Requirements Related to Transitions between Passive and Active Surveillance
2.2.4.6.4.2.3.1.8	Track Updates Using Airborne Position Messages	2.4.2.11.4	Verification of Maintenance of Established Tracks Using Passive Surveillance
2.2.4.6.4.2.3.1.9	Tracking in the Absence of Airborne Position Messages	2.4.2.11.4	Verification of Maintenance of Established Tracks Using Passive Surveillance
		2.4.2.11.5	Verification of Requirements Related to Transitions between Passive and Active Surveillance
2.2.4.6.4.2.3.1.10	Revalidation	2.4.2.11.4	Verification of Maintenance of Established Tracks Using Passive Surveillance
		2.4.2.11.5	Verification of Requirements Related to Transitions between Passive and Active Surveillance
2.2.4.6.4.2.3.1.11	Error Budget Allocated to ACAS X for Slant Range Validation	2.4.2.11.4	Verification of Maintenance of Established Tracks Using Passive Surveillance
		2.4.2.11.6	Verification of Error Budget in Computing Slant Range from Passive Data
2.2.4.6.4.2.3.2.2	Persistence of Active Surveillance	2.4.2.11.3	Verification of Acquisition and Maintenance of Established Tracks Using Active Surveillance
2.2.4.6.4.2.3.2.3	Active Surveillance Region	2.4.2.11.3	Verification of Acquisition and Maintenance of Established Tracks Using Active Surveillance

Requirements	Tests		
		2.4.2.11.4	Verification of Maintenance of Established Tracks Using Passive Surveillance
		2.4.2.11.5	Verification of Requirements Related to Transitions between Passive and Active Surveillance
2.2.4.6.4.2.3.2.4	Passive to Active Surveillance Transition	2.4.2.11.5	Verification of Requirements Related to Transitions between Passive and Active Surveillance
2.2.4.6.4.2.3.2.5	Track Updates Using Active Surveillance	2.4.2.1.7.3	Maintenance of Established Mode S Tracks
		2.4.2.11.5	Verification of Requirements Related to Transitions between Passive and Active Surveillance: -Test 1, Intruder 1 -Test 2, Intruder 3
2.2.4.6.4.2.3.2.6	Power Programming	2.4.2.1.7.6	Mode S Power Programming
2.2.4.6.4.2.3.2.7	Validation of Airborne Position Message Data	2.4.2.11.3	Verification of Acquisition and Maintenance of Established Tracks Using Active Surveillance
		2.4.2.11.4	Verification of Maintenance of Established Tracks Using Passive Surveillance
		2.4.2.11.5	Verification of Requirements Related to Transitions between Passive and Active Surveillance
2.2.4.6.4.2.4	NTA3/NTA6 Range Determination for On-Ground ACAS X Intruders	2.4.2.11.10	Verification of On-Ground ACAS X Range Determination Using ADS-B
		2.4.2.1.7.6	Mode S Power Programming
2.2.4.6.5	Non-Altitude-Reporting Aircraft	2.4.2.8.1.1.6	Intruder Vertical Speed Arrow
2.2.4.6.6.1	Bearing Estimation - General Requirements	2.4.2.1.9.1	Bearing Accuracy with Standard Ground Plane
		2.4.2.8.1.1.7	No-Bearing Advisories
2.2.4.6.6.2	Bearing Acquisition with Standard Ground Plane	2.4.2.1.9.1	Bearing Accuracy with Standard Ground Plane
2.2.4.6.6.2.1	Accuracy, -10 Deg. to +10 Deg. Elevation	2.4.2.1.9.1	Bearing Accuracy With Standard Ground Plane
2.2.4.6.6.2.2	Accuracy, Greater Than 10 Deg. To +20 Deg. Elevation	2.4.2.1.9.1	Bearing Accuracy With Standard Ground Plane
2.2.4.6.6.3	Bearing Accuracy in the Presence of Interference	2.4.2.1.9.2	Reply Processing
2.2.4.6.6.3.1	Mode C Interleaved Replies	2.4.2.1.9.2.1	Mode C Interleaved Replies
2.2.4.6.6.3.2	Mode C Overlapped Replies	2.4.2.1.9.2.2	Mode C Overlapped Replies
2.2.4.6.6.3.3	Mode S Overlapped Replies	2.4.2.1.9.2.3	Mode S Overlapped Replies
2.2.4.6.6.4	Bearing Filter Performance	2.4.2.1.9.3	Bearing Filter Performance

Requirements	Tests		
2.2.4.6.6.4.1	Bearing Filter	2.4.2.1.9.3.1	Bearing Track and Coast
		2.4.2.1.9.3.2	Filter Lag
2.2.4.7	Antenna System	2.4.2.1.7.2	Mode S Range Acquisition
		2.4.2.1.7.3	Maintenance of Established Mode S Tracks
2.2.4.7.2	Radiation Pattern	2.4.2.1.9.4	Radiation Pattern
2.2.4.7.2.1	Transmit Radiation Pattern	2.4.2.1.9.4	Radiation Pattern
		2.4.2.1.10.1	Use of Directional Interrogations for Mode C Surveillance and Bearing Receive Radiation Pattern
2.2.4.7.2.2	Receive Radiation Pattern	2.4.2.1.10.1	Use of Directional Interrogations for Mode C Surveillance and Bearing Receive Radiation Pattern
2.2.4.7.3	Use of a Directional Antenna for Mode S Interrogations	2.4.2.1.10.2	Use of a Directional Antenna for TCAS Broadcast Interrogations
2.2.4.7.4.1	Squitter Listening	2.4.2.1.7.2	Mode S Range Acquisition
2.2.5.5	Combined STM/TRM Input Interfaces	2.4.2.7	System Integration Tests
2.2.5.5.1	ReceiveBaroAltObservation	2.4.2.7	System Integration Tests
2.2.5.5.2	ReceiveRadAltObservation	2.4.2.7	System Integration Tests
2.2.5.5.3	ReceiveHeadingObservation	2.4.2.7	System Integration Tests
2.2.5.5.4	ReceiveWgs84Observation	2.4.2.7	System Integration Tests
		2.4.2.11.11	Verification of DF=18 ADS-B and ADS-R Only Passive Surveillance
2.2.5.5.5	ReceiveDiscretes	2.4.2.4.1	Transmission of RA Report to Mode S Sensor
		2.4.2.5	ACAS X Capability and Part Number Reporting
		2.4.2.7	System Integration Tests
2.2.5.5.6	ReceiveDF0	2.4.2.1.7.3	Maintenance of Established Mode S Tracks
		2.4.2.7	System Integration Tests
		3.4.4.1	Mode C Surveillance Flight Tests
2.2.5.5.7	ReceiveModeCReply	2.4.2.7	System Integration Tests
2.2.5.5.8	ReceiveModeCReplies	2.4.2.7	System Integration Tests
		3.4.4.1	Mode C Surveillance Flight Tests
2.2.5.5.9	ReceiveStateVectorPosition	2.4.2.7	System Integration Tests
		2.4.2.11.11	Verification of DF=18 ADS-B and ADS-R Only Passive Surveillance
2.2.5.5.10	ReceiveStateVectorVelocity	2.4.2.7	System Integration Tests
		2.4.2.11.11	Verification of DF=18 ADS-B and ADS-R Only Passive Surveillance
2.2.5.5.11	ReceiveModeStatus	2.4.2.4	Coordination
		2.4.2.7	System Integration Tests
2.2.5.5.12	ReceiveTargetDesignation	2.4.2.7	System Integration Tests

Requirements	Tests		
2.2.5.5.13	ReceiveUF16UDS30	2.4.2.7	System Integration Tests
2.2.6.1.2.1.1	Ownship Symbol	2.4.2.8.1.1.1 (a)	Ownship Symbol
2.2.6.1.2.1.2	Range Rings	2.4.2.8.1.1.2	Range Rings
2.2.6.1.2.1.3	Traffic Symbology	2.4.2.8.1.1.3	Threat Symbology
2.2.6.1.2.1.4	Off-Scale Symbology	2.4.2.8.1.1.4	Off-Scale Symbology
2.2.6.1.2.1.5	Altitude Data Tag	2.4.2.8.1.1.5	Altitude Data Tag
2.2.6.1.2.1.6	Intruder Vertical Speed Arrow	2.4.2.8.1.1.6	Intruder Vertical Speed Arrow
2.2.6.1.2.1.7	Non-Altitude Reporting Intruders	2.4.2.8.1.1.5	Altitude Data Tag
2.2.6.1.2.1.8	Intruder Bearing	2.4.2.8.1.1.7	No-Bearing Advisories
2.2.6.1.2.1.9	Display of Traffic	2.4.2.8.1.1.8	Display of Traffic
		3.2.9	Number of Intruders Displayed
2.2.6.1.2.1.10	Altitude Band for the Display	2.4.2.8.1.1.9	Altitude Band for the Display
2.2.6.1.2.1.11	ACAS X Operating Mode and Selected Display Range Annunciation	2.4.2.8.1.1.4	Off-Scale Symbology
		2.4.2.8.1.1.10	Display of ACAS X Operating Mode
2.2.6.1.2.1.12	RA Annunciations	2.4.2.8.1.1.3	Threat Symbology
		2.4.2.8.1.1.4	Off-Scale Symbology
		2.4.2.8.1.1.7	No-Bearing Advisories
2.2.6.1.2.1.14	Status and Failure Annunciations	2.4.2.8.1.1.10	Display of ACAS X Operating Mode
2.2.6.1.2.2.2	Range Ring	2.4.2.8.1.1.2	Range Rings
2.2.6.1.2.3.1	Display Range	2.4.2.8.1.3.1	Display Range
2.2.6.1.2.3.2	Range Selection	2.4.2.8.1.3.2	Range Selection
2.2.6.1.2.3.3	Range Rings	2.4.2.8.1.1.2	Range Rings
2.2.6.1.2.4.1	Activation	2.4.2.8.1.4.1	Activation and Information Displayed
2.2.6.1.2.4.2	Information Displayed	2.4.2.8.1.4.1	Activation and Information Displayed
2.2.6.1.2.6.2	Ownship Symbol and Location	2.4.2.8.1.1.1	Ownship Symbol
2.2.6.1.2.7.1.1	Display Modes	2.4.2.8.1.7.1. 1	Available Display Modes
		2.4.2.8.1.7.1. 2	Display Mode Characteristics
2.2.6.1.2.7.1.2	Ownship Symbol and Location	2.4.2.8.1.1.1	Ownship Symbol
2.2.6.1.2.7.2.2	Failure Annunciations	2.4.2.8.1.7.2. 1	Failure Annunciations
2.2.6.1.2.7.2.3	Traffic Display	2.4.2.8.1.7.2. 2	Display of Traffic
2.2.6.1.2.7.2.4	ACAS X and Weather Information	2.4.2.8.1.7.2. 3	ACAS X and Weather Information

Requirements	Tests		
2.2.6.1.2.7.2.5	ACAS X and Navigation Information	2.4.2.8.1.7.2.4	ACAS X and Navigation Information
2.2.6.1.2.7.2.6.2	Ownship Symbol and Location	2.4.2.8.1.1.1	Ownship Symbol
2.2.6.1.2.7.2.7.1	Display Characteristics	2.4.2.8.1.7.2.6	EXPANDED Mode
2.2.6.1.2.7.2.7.2	Display Range and Range Ring	2.4.2.8.1.1.2	Range Rings
2.2.6.1.2.7.2.7.3	Ownship Symbol and Location	2.4.2.8.1.1.1	Ownship Symbol
2.2.6.1.2.7.2.8.1	Display Characteristics	2.4.2.8.1.7.2.7.1	Display Characteristics
2.2.6.1.2.7.2.8.2	Display Range and Range Rings	2.4.2.8.1.1.2	Range Rings
		2.4.2.8.1.7.2.7.2	Display Range and Orientation
2.2.6.1.2.7.2.8.3	Display Orientation	2.4.2.8.1.7.2.7.2	Display Range and Orientation
2.2.6.1.2.7.2.8.4	Ownship Symbol and Location	2.4.2.8.1.1.1	Ownship Symbol
2.2.6.1.2.7.2.9.1	Display Characteristics	2.4.2.8.1.7.2.8.1	Display Characteristics
2.2.6.1.2.7.2.9.2	Display Range and Range Rings	2.4.2.8.1.1.2	Range Rings
2.2.6.1.2.7.2.9.3	Ownship Symbol	2.4.2.8.1.1.1	Ownship Symbol
		2.4.2.8.1.7.2.8.1	Display Characteristics
2.2.6.1.2.7.2.9.4	Display Orientation	2.4.2.8.1.7.2.8.1	Display Characteristics
2.2.6.1.2.7.2.9.5	TA or RA Annunciation	2.4.2.8.1.7.2.8.1	Display Characteristics
2.2.6.1.2.7.2.9.6	Selection of Heading Up Display	2.4.2.8.1.7.2.8.2	Display Orientation
2.2.6.1.2.7.3.3	Traffic Display Inhibit	2.4.2.8.1.7.3	EICAS or Systems Displays
		3.2.10	Display of Traffic on EICAS/SYSTEMS Displays
2.2.6.1.2.7.4.1.2	VSI Requirements	2.4.2.8.1.7.4.1	VSI Requirements
2.2.6.1.2.7.4.2.2	Ownship Symbol and Location	2.4.2.8.1.1.1	Ownship Symbol
2.2.6.1.2.7.4.2.3	Interference with Vertical Speed Scale	2.4.2.8.1.7.4.2	Interference with Vertical Speed Scale
2.2.6.2.2	RA/VSI (Round dial VSI)	2.4.2.8.2.1	Round Dial VSI
		2.4.2.8.2.1.1	VSI Range
2.2.6.2.2.1.1	Red Arcs	2.4.2.8.2.1	Round Dial VSI
2.2.6.2.2.1.2	Green Arcs		
2.2.6.2.2.1.3	Black Arcs		
2.2.6.2.2.1.4	Multi-aircraft Encounters		

Requirements	Tests		
2.2.6.2.3 RA/VSI (Integrated Tape VSI on a Primary Flight Display [PFD])	RA/VSI (Integrated Tape VSI on a Primary Flight Display [PFD])	2.4.2.8.2.2	Vertical Speed Tape
		2.4.2.8.2.2.1	Vertical Speed Tape Range
2.2.6.2.3.1.1	Red Zone	2.4.2.8.2.2	Vertical Speed Tape
2.2.6.2.3.1.2	Green Zone	2.4.2.8.2.2	Vertical Speed Tape
2.2.6.2.3.1.3	Black Zones	2.4.2.8.2.2	Vertical Speed Tape
2.2.6.2.3.1.4	Multi-aircraft Encounters	2.4.2.8.2.2	Vertical Speed Tape
2.2.6.2.3.3	VSI Scale	2.4.2.8.2.1.1	VSI Range
2.2.6.2.4.1	No-Fly Pitch Angles	2.4.2.8.2.3	Pitch Cues on the PFD
2.2.6.2.4.2	Mode Annunciation		
2.2.6.2.4.3	Multi-aircraft Encounters		
2.2.6.2.6	Heads-up Display	2.4.2.8.2.5	HUD
2.2.6.2.6.1	No-Fly Zone		
2.2.6.2.6.2	Mode Annunciation		
2.2.6.2.6.3	Flight Path Target		
2.2.6.2.6.4	Multi-aircraft Encounters		
2.2.6.2.6.5	Display Decluttering		
2.2.6.2.6.6	RA Guidance Availability		
2.2.6.4.2	RAs	3.2.12	Red Visual Alert
2.2.6.4.3	Message on PFD	2.4.2.8.2.3	Pitch Cues on the PFD
2.2.6.5.1	ACAS X/Mode S Controls	2.4.2.8.3.1	ACAS X/Mode S Controls
2.2.6.5.2	Traffic Display Controls	2.4.2.8.1.1.9	Altitude Band for the Display
		2.4.2.8.3.2	Traffic Display Controls
2.2.6.5.3.1	WX/ACAS X Mode	2.4.2.8.3.3	Shared Weather Radar/Traffic Display Controls
2.2.6.5.3.2	WX-and-Traffic Mode		
2.2.6.5.3.3	Optional Mode Selection		
2.2.6.6.1	Status and Failure Annunciations - General	2.4.2.8.4.1	Traffic Display Annunciations
2.2.6.6.2	Traffic Display Annunciations	2.4.2.8.1.1.10	Display of ACAS X Operating Mode
		2.4.2.8.4.2	RA Display Annunciations
2.2.7	Monitor Requirements	2.4.2.9	Automatic Performance Monitoring and Self Test
		2.4.2.11.4	Verification of Maintenance of Established Tracks Using Passive Surveillance
2.2.7.1.1	Failure Response	2.4.2.9.1	Automatic Performance Monitoring
2.2.7.1.3	Self-Test	2.4.2.9.2	Self-Test
2.2.7.2	Monitoring of ACAS X Components	2.4.2.11.9	Verification of Monitoring Requirements
2.2.7.2.1	Computer Monitoring	2.4.3	Computer Performance Verification
2.2.7.2.2	Coordination Monitoring	2.4.2.4.3.4	Coordination Monitor

Requirements	Tests		
2.2.7.2.6	ICAO 24-Bit Aircraft Address	2.4.2.9.3	Own Transponder ICAO 24-Bit Aircraft Address
3.2.13.1	Pitch Cues on PFD and HUD	3.4.2.4.1	Pitch Cues
3.2.13.2	Flight Director	3.4.2.4.2	Flight Director
3.2.13.3	Autopilot TCAS Mode to Perform RA Maneuvers	3.4.2.4.3	Auto Pilot

This Page Intentionally Left Blank

3 INSTALLED EQUIPMENT PERFORMANCE

3.1 Test Conditions

3.1.1 Power Input

Ground tests may be conducted using the aircraft's electrical power generating system. Alternatively, an appropriate external power supply may be used unless otherwise specified or unless the tested function is known to be dependent upon power source qualities.

3.1.2 Associated Equipment

All electrically operated aircraft systems and equipments **shall** (2203) be operational before conducting electronic interference tests.

3.1.3 Environment

During the following tests, the equipment **shall** (2204) not be subjected to environmental conditions that exceed those specified by the manufacturer.

3.1.4 Adjustment of Equipment

The circuits of the equipment under test **shall** (2205) be properly aligned and otherwise adjusted in accordance with the manufacturer's recommended practices.

3.2 Equipment Installation

3.2.1 Equipment Accessibility

The equipment controls and display(s) installed for in-flight operation **shall** (2206) be readily accessible to the pilot from the normal seated position. The appropriate operator/crew member(s) **shall** (2207) have an unobstructed view of the display(s) when in normal sitting position.

3.2.2 Display Visibility

The display brilliance **shall** (2208) be such that the display can be interpreted under all probable cockpit conditions of ambient light. Filters and brightness adjustments are acceptable means of obtaining visibility in daylight.

3.2.3 Interference

The equipment **shall** (2209) not be the source of objectionable conducted or radiated interference nor be adversely affected by conducted or radiated interference from other certificated equipment or systems installed in the aircraft.

3.2.4 Physical Installation

Operation of the equipment **shall** (2210) not be adversely affected by aircraft maneuvering or changes in attitude encountered in normal flight operations.

3.2.5 Aircraft Power Source

The voltage and voltage tolerance characteristics of the equipment **shall** (2211) be compatible with the aircraft power source.

3.2.6 Transmission Lines

Transmission lines to the antennas **shall** (2212) have impedance, power handling and loss characteristics in accordance with the specifications of the equipment manufacturer. The VSWR for both bottom and top antennas as seen through the transmission lines **shall** (2213) be within the limits specified by the manufacturer.

The delay difference between the transmission lines to the top and bottom mounted antennas **shall** (2214) not exceed the amount specified by the equipment manufacturer.

3.2.7 Antenna Location

The horizontal distance between the top and the bottom antennas **shall** (2215) be not greater than 7.6 m (25 ft). Both antennas **shall** (2216) be mounted as near as possible to the center line of the fuselage. Antennas **shall** (2217) be located to minimize obstruction to their fields in the horizontal plane.

Note: The antenna spacing is specified in order to control apparent range jitter from reply to reply due to antenna switching.

3.2.7.1 Minimum Distance from Other Antennas

The spacing between the ACAS X antenna and the Mode S transponder antenna **shall** (2218) be such as to provide a minimum of 20 dB of isolation between the two antennas when the peak of the azimuth beam of the ACAS X antenna is steered in the direction of the Mode S transponder antenna.

Note: If both antennas are conventional omnidirectional matched quarter-wave stubs, this level of isolation is obtained by providing a spacing of at least 51 cm (20 in) between the centers of the two antennas. If either antenna is other than a conventional stub, the minimum spacing must be determined by measurement.

3.2.8 Altimetry System

Note: The altimetry system performance requirement is specified for the total system. This includes all errors associated with the static system, the pressure transducer and with any quantizer used to convert the transducer output to a form suitable for ACAS X or the Mode S altitude report or both.

Figure 3-1 is a block diagram of an altimetry system and shows the outputs to the transponder and to ACAS X. All of the associated error components, listed below the system elements, must be included in the assessment of total altimetry system error. The assessment must include consideration of both the altitude data being supplied to the ACAS X unit and that provided in the transponder's altitude report.

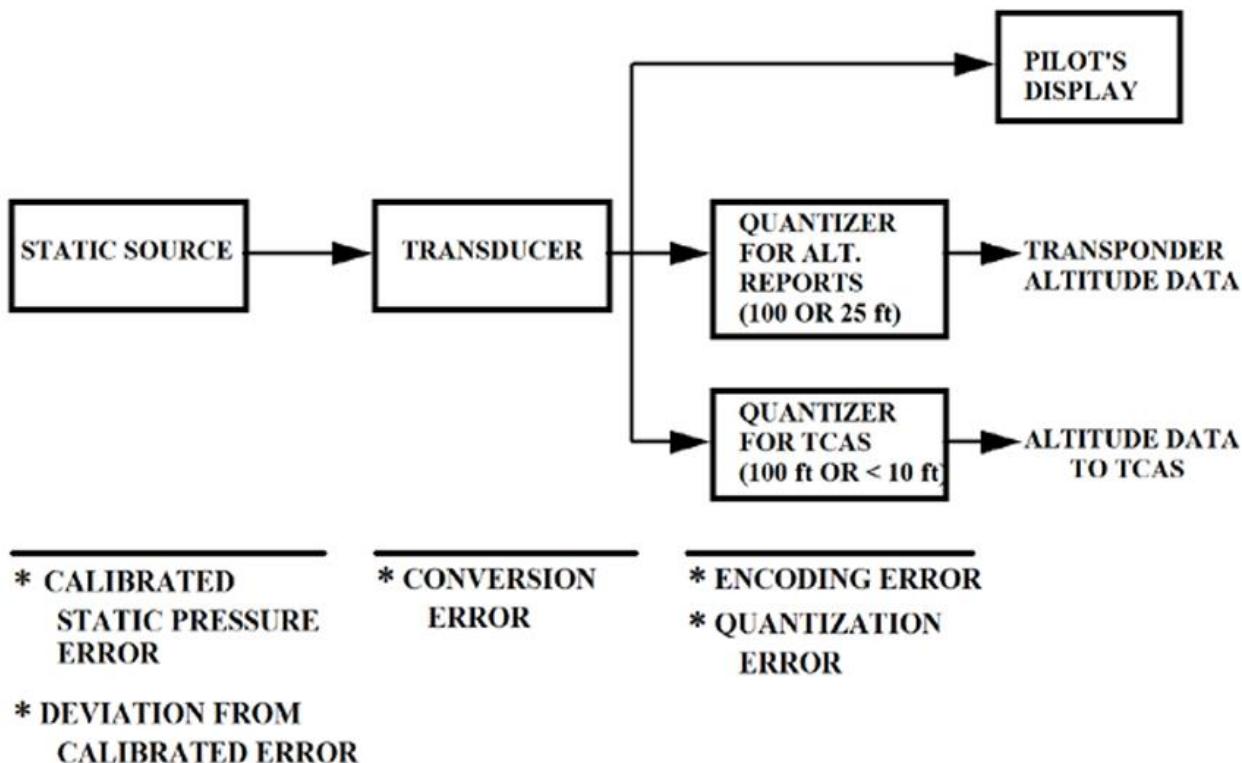


Figure 3-1: Altimetry System Elements and Error Components

3.2.8.1 Barometric Altimetry Performance Required for ACAS X Operation

Table 3-1 shows the altimetry performance requirements necessary for ACAS X operation. The table specifies the error bound of the altimetry system providing altimetry data to the ACAS X unit and also the error bound of the altitude report. The error magnitude **shall** (2219) not exceed the error bound more than 0.3 percent of the time on a probability basis. Intermediate values are found through linear interpolation of the nearest given values. Values at altitudes beyond those given are found through linear extrapolation of the two nearest end values.

Table 3-1: Specification Of Total Altimetry System Performance

(Error Bound Must Be Met On A 99.7% Probability Basis)

ALTITUDE	ERROR BOUND in feet
MSL	135
5 K	144
10 K	156
15 K	174
20 K	195
25 K	213
30 K	234
35 K	258
40 K	285

The altimetry data used for ACAS X **shall** (2220) be obtained from the source that provides that information for own Mode S altitude reports and **shall** (2221) be used at the finest quantization available within the aircraft installation. When the altimetry data is provided to ACAS X, it **shall** (2222) be indicated as being either finely quantized or coarsely quantized. Altitude information that is quantized to 10 ft or less **shall** (2223) be considered finely quantized. All other altitude information **shall** (2224) be considered coarsely quantized.

Notes:

1. *Where these requirements are not met, ACAS X operation may be unacceptable.*
2. *These requirements are for altimetry performance in all flight configurations in which RAs are generated including clean, maneuvering, holding and for approach and departure when above 500 feet AGL. They do not include any consideration of flight technical error, pilot blunders or barometric setting errors. All altimetry data used for ACAS X purposes is referenced to the standard atmospheric pressure reference datum of 1013.25 millibars (29.92 inches of mercury).*
3. *The altimetry system performance reflected in Table 3-1 is practically realizable with current altimetry equipment. In addition, when two Active CAS-equipped aircraft are in conflict and both aircraft a) meet the performance requirements of Table 3-1 and b) respond to their displayed RAs as modeled in the ADD (Volume II), the assurance that the conflict will be successfully resolved is very high.*

3.2.8.2 Altimetry System Error Assessment

Note: The accuracy of the installed altimetry system should be determined by tests, evaluation of previously collected data or both, to assure that the altimetry error budget specified in §3.2.8.1 is not exceeded.

As a minimum, altimetry performance **shall** (2225) be proven to meet the requirements at one altitude within each of the following altitude bands, provided an altitude in the band falls within the approved operational ceiling of the aircraft:

- a. 500 to 2500 feet above MSL.
- b. 2500 to 10,000 feet above MSL.
- c. 10,000 to 18,000 feet above MSL.
- d. 18,000 to 30,000 feet above MSL.
- e. 30,000 feet and greater above MSL.

Note: A recommended procedure for assessing altimetry system performance is described below. This procedure differentiates between fixed and variable errors. Variable errors are combined using the root-sum-square (RSS) methodology. Fixed errors are combined arithmetically. Next, assume that the total altimetry system error is a Gaussian random variable characterized by the accumulated variable and fixed errors.

The RSS methodology provides the standard deviation of the total variable error through calculation of the square root of the sum of squares of the variable error component standard deviations. This assumes that the variable error components are statistically uncorrelated. In the RSS methodology, variable error components are treated differently according to their distributions. For example, the standard deviation for each variable error component that is uniformly distributed (has a

rectangular distribution) is found by dividing the total error range by the square root of 12. The total fixed error is the arithmetic sum of the individual fixed error components.

To assess altimetry system performance in terms of Table 3-1, the altimetry system error is taken as Gaussian with a standard deviation equal to the standard deviation of the total variable error as determined by the RSS methodology and a mean equal to the total fixed error. Standard calculations and/or mathematical tables then can be used to calculate the probability that the magnitude of the altimetry system error will exceed the error bound of Table 3-1. This probability must be less than 0.3 percent.

3.2.8.3 Definitions

The following paragraphs describe the error components listed under each of the altimetry system elements identified in Figure 3-1.

a. **Static Source:** Static source errors arise from variations in the location and physical condition of flush ports or static probes and any errors associated with the transmission of air pressure to the transducer. The net result is a differential between the ambient pressure outside the aircraft and that presented to the transducer. Errors that should be considered for analysis are:

1. **Calibrated static pressure error:** This refers to the published static source error correction. (If this correction is automatically applied, then this error need not be included in the computation of total system accuracy).

This error is treated as a fixed error that shifts the mean of the altimetry system error distribution.

2. **Deviation from calibrated error:** This refers to the deviation of static source error from the published calibration data. This variable error component includes the errors associated with the initial calibration measurements of static source error, production tolerances, in-service degradation of the static source and expected variations in the flight regime and gross weight of the aircraft. If automatic correction of static pressure error is applied, then the residual errors must be included in this variable error component.

b. **Transducer:** Transducer errors result in a difference between the air pressure introduced to the transducer and the conditioned output provided to the quantizer. This is a conversion error and is further defined below:

1. **Conversion error:** This refers to the resultant of those variable error components identified with the conversion of the input static system pressure to an appropriate output. Conversion error is a variable error component that includes but may not be limited to: scale errors, temperature scale errors, balance errors, threshold errors, friction errors and hysteresis errors. In addition, the overall accuracy of the transducer is affected by the limitations in accuracy and readability of the reference barometric standard. Determination of total conversion error must include considerations of all such errors.

c. **Quantizer:** The quantizer, as distinguished from the transducer, contributes additional error:

1. **Encoding error:** This variable error component is associated with the correspondence between the ideal quantizer transition points and the actual

quantizer transition points for a specified transducer output. The largest errors are generally associated with mechanically driven or servo driven quantizers yet such errors do occur in an electrical analog-to-digital conversion.

2. **Quantization error:** This variable error component results from digitizing analog data. The peak quantization error is half the quantization range or interval. The data input to ACAS X may not be more coarsely quantized than that used in the Mode C report.

The peak Mode C quantization error is half the 100 foot quantization range or 50 feet. For purposes of the RSS treatment of errors, the standard deviation of quantization error, assuming a uniform distribution of aircraft altitude over the quantization range, is then 50 feet divided by the square root of three.

3.2.9 Number of Intruders Displayed (§2.2.6.1.2.1.9)

The number of intruders displayed by the traffic display is fixed and cannot be modified from the cockpit. Verify that the aircraft wiring either defines no limit to the number of displayed intruders or that the proper pins are grounded to limit the number of aircraft shown on the traffic display. Use a test set or fixed transponders to simulate more intruders than the traffic display is configured to display and verify that only the maximum number of intruders defined for the installation are shown on the traffic display. While the intruders are being simulated, exercise all ACAS X and transponder control functions and verify that the number of displayed intruders does not change.

3.2.10 Display of Traffic on EICAS/SYSTEMS Displays (§2.2.6.1.2.7.3.3)

If the traffic display is implemented on the same display used by the EICAS or SYSTEMS display, verify that: (a) the traffic information is not shown on the primary engine display under any condition or operating configuration and (b) the traffic display function is inhibited if one or more EICAS displays is inoperative.

3.2.11 Use of Flight Director Cues for RA Guidance (§3.2.13.2)

If an installation uses flight director cues for displaying RA guidance, verify that a pilot selectable RA mode and suitable mode annunciation is provided to both pilots.

3.2.12 Red Visual Alert (§2.2.6.4.2)

Verify that a red visual alert is provided in the primary field of view of each pilot whenever an RA is issued.

3.2.13 Integration with Aircraft Flight Controls

3.2.13.1 Pitch Cues on PFD and HUD

If implemented, interaction between No-Fly zones on PFD and HUD, **shall** (2033) be shown to be compatible with existing low speed/high speed warning and/or visual alert.

3.2.13.2 Flight Director

Flight Director RA guidance may be provided to support manually flown RAs. If a Flight Director TCAS mode is made available, it **shall** (2034) comply with requirements

elaborated in MOPS RTCA/DO-325 Appendix C §C2.1, C2.2, C2.3 (Installation Guidance for Autopilot/Flight Director/Autothrust coupled TCAS Resolution Advisory Mode)

Note: If Flight Director Guidance for RA is installed it is mandatory to consider its integration with other Flight Director Modes and associated alarms and must ensure adequate guidance performances, consistent with TCAS expected response. In addition, the behavior of an air vehicle in approved configurations, throughout the flight envelope, must be considered.

3.2.13.3 Autopilot TCAS Mode to Perform RA Maneuvers

An autopilot TCAS mode may be provided to perform RA maneuvers. If an Autopilot TCAS mode is made available, it **shall** (2035) comply with requirements elaborated in MOPS RTCA/DO-325 Appendix C §C2.1, C2.2, C2.3 (Installation Guidance for Autopilot/Flight Director/Autothrust coupled TCAS Resolution Advisory Mode).

Note: If an Autopilot RA mode is installed it is mandatory to consider its integration with other Autopilot Modes and associated alarms and must ensure adequate guidance performances, consistent with TCAS expected response. In addition behavior at the limit of flight envelope must be considered.

3.3 Minimum Installed Equipment Performance Requirements

When a transmission line is included as part of the installation, all minimum installed system performance requirements must be met. The installed equipment **shall** (2036) meet the requirements of §2.2.

In order to meet these requirements, test results supplied by the equipment manufacturer may be accepted in lieu of tests performed by the equipment installer.

However, performance characteristics such as interaction with other installed equipment and power sources, which cannot be tested by the equipment manufacturer, **shall** (2037) be tested by the installer.

3.4 Test Procedures for Installed Equipment Performance

The test procedures set forth below are considered satisfactory for use in determining required equipment performance when installed in aircraft. Tests are stated in a manner that will make maximum use of test data available before installation on the aircraft.

Although specific test procedures are cited, it is recognized that other methods may be preferred by the installer.

3.4.1 Conformity Inspection

Visually inspect the installed equipment to determine the use of acceptable workmanship and engineering practices. Verify that all mechanical and electrical connections have been made properly and that the equipment has been installed and located in accordance with the manufacturer's requirements.

3.4.2 General Test Procedures**3.4.2.1 Equipment Function**

Vary all controls used for in-flight operations through their full range to determine that the equipment is operating according to the manufacturer's instructions and that each control performs its intended function.

3.4.2.2 Interference Effects (Ground Test)

With the transmitter and receiver operating, individually operate each of the other electrically operated aircraft equipments to determine that no significant conducted or radiated interference exists. Evaluate all reasonable combinations of control settings and operating modes. Operate communication and navigation equipment on at least one low-band, one high-band and one mid-band frequency. Make note of systems or modes of operation that should also be evaluated during flight. If appropriate, repeat tests using emergency power.

Note: Electromagnetic interference tests are often conducted on all electronics systems in one test series, using procedures established by the aircraft manufacturer. If such tests include the ACAS X equipment, no further tests are required.

3.4.2.2.1 Suppression Bus

This test ensures that ACAS X and the Mode S transponder are properly installed on the mutual suppression bus such that when transmitting, each does not cause the other to receive and process erroneous signals.

Specifically:

- a. During the interval of an ACAS X transmission as defined in §2.1.10.1, demonstrate that the ACAS X transmission does not cause the Mode S transponder to generate spurious signals.
- b. During the interval of a Mode S transponder transmission as defined in §2.2.3.10 demonstrate that the Mode S transponder transmission does not cause ACAS X to generate spurious signals.

3.4.2.3 Accessibility

Determine that all equipment controls and displayed data are readily accessible and easily interpreted.

3.4.2.4 Integration with Aircraft Flight Controls**3.4.2.4.1 Pitch Cues (§3.2.13.1)**

When alerts and guidance information are displayed simultaneously from other systems, ensure the ACAS X TA or RA, if displayed (i.e. pitch cues on the PFD or HUD) do not contradict or prevent the crew from acting on higher priority alerts, such as overspeed or stall warning.

3.4.2.4.2 Flight Director (§3.2.13.2)

If Flight Director RA Guidance is installed, refer to tests defined in MOPS RTCA/DO-325 Appendix C §C2.1, C2.2, C2.3 (Installation Guidance for Autopilot/Flight Director/Autothrust coupled TCAS Resolution Advisory Mode)

3.4.2.4.3 Auto Pilot (§3.2.13.3)

If Auto Pilot coupled with ACAS is installed, the tests are defined in MOPS RTCA/DO-325 Appendix C §C2.1, C2.2, C2.3 (Installation Guidance for Autopilot/Flight Director/Autothrust coupled ACAS Resolution Advisory Mode)

3.4.3 Antenna Gain Performance

These paragraphs apply to all ACAS X antennas and specify antenna gain tests or antenna location criteria to ensure that the installed antenna gain is not degraded beyond an acceptable value. If a transponder suppression pattern is part of the antenna system, its gain performance must also be considered. Tests peculiar to antennas used for bearing estimation are described in §3.4.5.

The purpose of the antenna gain tests is to verify that the installed antenna patterns meet the success criteria for antenna gain specified in §3.4.3.1. This may be done using one or a combination of three distinct procedures: (a) full-scale antenna range measurements, (b) scaled model measurements or (c) theoretical calculations.

Another method of validating antenna gain is to employ distance-area conditions to ensure that the location of the antenna on the aircraft does not unduly degrade its required gain performance. This procedure, which is described in §3.4.3.5, can be used in place of or in combination with any of the other three antenna gain determination procedures.

3.4.3.1 Success Criteria

At an elevation angle of zero degrees relative to the fuselage reference plane, the gain of the forward ±45 degree azimuth sector of both the top and bottom antennas **shall** (2038) be no more than one dB below the gain of the antenna when installed on a standard ground plane as specified in §2.2.4.7.2. The radiation pattern gain, at zero degrees elevation, **shall** (2039) be within three dB of the gain of the ground-plane-installed antenna over 90 percent of the remainder of its azimuthal coverage.

Note: Where possible, it is recommended that the antenna be mounted on the forward fuselage, thereby minimizing blockage due to the vertical stabilizer and engine nacelles.

3.4.3.2 Full-Scale Anechoic Antenna Range Measurements of Gain

The gain characteristics of the antenna as mounted on the actual airframe may be measured directly in a calibrated anechoic antenna test range using standard controlled procedures for such measurements. Gain characteristics determined in this way require no further validation.

Note: Anechoic range measurements are generally impractical for determining full antenna gain patterns for large aircraft. However, such techniques may be practical for qualifying certain subregions of the coverage pattern or for validating model measurements or theoretical calculations.

3.4.3.3 Scaled Model Measurements of Gain

3.4.3.3.1 Aircraft Model

Aircraft models for antenna measurements are normally 1/10 to 1/40 scale. The scaling factor is dependent upon the availability of equipment, antenna scaling considerations, etc., with larger models resulting in greater accuracy.

It is necessary to construct only the major structural features of the aircraft. Detailed features such as windows, turbines, etc. are not required. The outside skin should be of conductive material.

Often, the fuselage and engine nacelles can be modeled from metal tubing and/or shaped metal screening; wings and stabilizers can be constructed of flat metal plates. Moving control surfaces are not required unless it is expected that they will have significant effects upon the antenna pattern.

Notes:

1. *As a general rule, obstructions that subtend angles at the antenna of less than a few degrees in elevation or azimuth do not have to be modeled. However, smaller obstructions, such as other antennas, that are within a few wavelengths of the antenna under test, may have to be modeled because they can act as resonant scatterers and could have a significant effect upon the radiation pattern.*
2. *If the swept area of propeller blades exceeds the limits given in (1) above, the blades can be worst-case modeled as a flat metal disk of radius equal to the blade length. If the radiation pattern using disks for propellers satisfies the success criteria given above, it can be assumed that the pattern modulation caused by the rotating blades will not significantly degrade the ACAS X system performance.*

3.4.3.3.2 Model Tests

- a. Mount the scaled model antenna in the center of a ground plane whose radius is equal in wavelengths to the ground plane used for testing the full-scale antenna (§2.2.4.7.2).
- b. Using a calibrated anechoic antenna test range, confirm that the gain of the scaled antenna (including possible multiple radiating elements, splitting or combining networks, impedance and mutual coupling effects) is within two dB of the full-scale antenna gain, for all azimuth and elevation angles for which the gain of the full-scale antenna is within six dB of the peak gain.
- c. Mount the scaled model antenna on the aircraft model at the intended installation location.
- d. Measure the antenna gain for all azimuth angles (for top or bottom antennas).
- e. Confirm that the scaled antenna meets the success criteria of §3.4.3.1.

Note: This measurement may be performed using either analog or digital instrumentation. Azimuth data should be acquired every two degrees or less.

3.4.3.4

Theoretical Calculations of Antenna Gain

The gain characteristics of the antenna as mounted on the actual airframe may be determined indirectly by a combination of radiation pattern calculations and measurements designed to validate those calculations. When using such techniques to determine the gain of a multi-element antenna it is necessary to show that the calculations include the inherent characteristics of the antenna elements and their drivers, splitting or combining networks, and any effects due to mutual coupling between those elements.

3.4.3.4.1

Validation of Theoretical Calculations

If radiation pattern calculations are used to prove the success criteria of §3.4.3.1, the manufacturer of the antenna must provide corroborating data demonstrating the success of the calculation technique in predicting the antenna gain on an airframe roughly similar in size and complexity to the airframe under qualification. Such data must be obtained by comparison with selected gain measurements made (a) on a full-size airframe using a calibrated ramp test antenna range or (b) on a scaled model airframe as indicated in §3.4.3.3.

3.4.3.5

Antenna Location Using Distance-Area Calculations

The extent to which the antenna installation successfully minimizes obstructions in the horizontal plane and minimizes effects of reflecting objects may be judged by the distances to such objects and their sizes. If these distances and sizes satisfy the condition given here, then the antenna installation may be considered to be validated with regard to antenna gain. The condition is: For a target aircraft at zero degree elevation angle and at an azimuth bearing between -90 degrees and +90 degrees,

$$\frac{A_1^2}{\lambda^2 D_1^2} + \sum \frac{A_2^2 G'_2}{\lambda^2 D_2^2 G_2} + \sum \frac{A_3 G'_3}{4\pi D_3^2 G_3} < 0.02$$

where $\lambda=0.9$ ft is the free-space wavelength at 1090 MHz. The first term is applicable only if there is a metallic obstruction between the target and the ACAS X antenna. The distance in feet to the obstruction is denoted D_1 , and the area in ft^2 of the obstruction projected in the direction of the ACAS X antenna is denoted A_1 . The second term is a summation over flat metallic reflectors, if any, that are oriented so as to cause a specular reflection between the ACAS X antenna and the target. The distance in feet to the reflector is denoted D_2 , the area in ft^2 of the reflector projected in the direction of the ACAS X antenna is denoted A_2 , the antenna gain in the direction of the reflector is denoted G'_2 and is dimensionless (i.e., gain in dB = $10 \log G'_2$), and the antenna gain in the direction of the target is denoted G_2 and is dimensionless. The third term is a summation over all other metallic objects that may cause reflections between the ACAS X antenna and the target. The parameters D_3 , A_3 , G'_3 , and G_3 have the same meanings as in the second term. In the case of other aircraft antennas in view of the ACAS X antenna, a minimum value $A=0.22 \text{ ft}^2$ is to be used if the actual area is smaller than 0.22 ft^2 .

3.4.4

Certification Flight Test Procedures

The bench tests in §2.4 of this document are designed to verify ACAS X performance under carefully controlled conditions. Flight tests (§3.4.4.1, §3.4.4.2, and §3.4.4.3) **shall** (2040) be performed to completely validate ACAS X surveillance and coordination designs. The tests **shall** (2041) be accomplished with all certifiable antenna configurations unless the

applicant can demonstrate that a worst-case configuration has been evaluated. Since the functions to be evaluated by these tests are sufficiently independent of installation factors such as aircraft type and specific antenna location, these flight tests can be carried out using a smaller non-carrier aircraft.

3.4.4.1

Mode C Surveillance Flight Tests

- a. A flight test under environmentally stressful conditions **shall** (2042) be performed to provide the data for a complete ACAS X Mode C surveillance design validation. This environment includes the combined effects of multipath interference, synchronous garbling, and fruit interference as a result of high aircraft density, and ground interrogator interference from multiple ground interrogators.
- b. Flight testing **shall** (2043) be accomplished in an area of high Mode C density such as the Los Angeles basin area, particularly in the vicinity of the Long Beach California and Santa Ana California airports, since previous tests of TCAS II have shown that this area provides the stressful environment necessary for proper evaluation of ACAS X Mode C surveillance. Flight paths **shall** (2044) include a representative mixture of the following conditions:
 1. Over-land flights at altitudes between 3000 and 6000 feet MSL;
 2. Over-water flights at altitudes between 3000 and 6000 feet MSL for a duration that is at least 20% of the total required flight duration defined in d. below; and
 3. Approach and departure flights to Long Beach and Santa Ana airports.
- c. The flight test **shall** (2045) be conducted on weekends between 10:00 a.m. and 3:00 p.m. when the ground visibility is greater than 5 NM with a ceiling of at least 10,000 feet MSL to ensure the highest peak traffic densities. Another location and/or time may be proposed by the applicant and will be considered as a suitable alternative to the Los Angeles area if the applicant can demonstrate that the surveillance test was conducted in an average density of at least 0.05 transponder-equipped aircraft per sq. NM and a peak density of at least 0.07 transponder-equipped aircraft per sq. NM. Density is defined as the number of other transponder-equipped aircraft within 5 NM of the ACAS X test aircraft divided by the area of a circle of 5 NM radius. Four actual aircraft target tracks occurring simultaneously within 5 NM of the ACAS X test aircraft is equivalent to an average density of 0.05 transponder-equipped aircraft per sq. NM. The alternative location must also contain a minimum of three civil or military secondary surveillance radars located within 30 NM of ownship in order to provide an interference environment similar to the Los Angeles area.
- d. For newly established systems and those that have made significant surveillance changes, flight testing **shall** (2046) be of sufficient duration to record at least 5,000 seconds of target track reports on targets-of-interest and a minimum duration of three hours. The data **shall** (2151) be accumulated during a continuous time period of at least three hours.

For systems that have previously demonstrated satisfactory performance and have not made significant changes that affect surveillance, the test flight testing **shall** (2593) be of sufficient duration to demonstrate satisfactory system performance in accordance with the changes made. If no changes affecting surveillance have been made, this flight test is optional.

A target-of-interest is defined as any target-of-opportunity that is:

- 1. Within a 5 NM range in the forward quadrant,
 - 2. Within a 3.5 NM range in the right and left quadrants,
 - 3. Within a 2 NM range in the aft quadrant,
 - 4. Within ± 5 deg. elevation, and
 - 5. Altitude reporting and airborne.
- e. The number of actual flight hours necessary to collect 5,000 track-seconds of data depends on the aircraft density. For example, a three hour recording in the Los Angeles basin which has a demonstrated minimum average density of 0.04 aircraft per sq. NM would result in approximately 5,000 track-seconds of data on targets-of-interest. At an average density of 0.06 the three hour recording will result in approximately 9,000 seconds of data on targets-of-interest. It is expected that the applicant will have prior knowledge of the density conditions and will select the flight time necessary to record the appropriate amount of data.
- Note: Mode C General Aviation aircraft cone of interest reductions have been influenced by equipage with Mode S transponders. Continued Mode S transponder equipage may be cause for future cone of interest re-evaluations.*
- f. The recorded data **shall** (2048) include as a minimum all raw reply reports prior to any duplicate reply elimination, reply merging, reply correlation, etc., processed surveillance track reports that have been declared established, TAs and RAs, internal processor clock, and time-of-day.
 - g. The applicant **shall** (2051) perform at least the following specific post-flight data analyses:
 - 1. Identify all recorded surveillance target track reports that are established (all tracks included in the STM report to the display) and that satisfy the criteria for a target-of-interest.
 - 2. Identify those established target-of-interest track reports that are associated with actual aircraft.
 - 3. Determine the total number of aircraft-seconds associated with actual aircraft and the total number of true track-seconds (i.e., those update intervals in which an established surveillance track report was either coasted or updated with a valid reply) associated with actual aircraft.
 - 4. Determine the surveillance track probability as the ratio of the total number of true track-seconds associated with actual aircraft to the total number of aircraft-seconds associated with actual aircraft.
 - 5. Identify those established target-of-interest track reports that are not associated with actual aircraft.
 - 6. Determine the total number of track-seconds that are not associated with actual aircraft. These are classified as false track-seconds.
 - 7. Determine the false track probability as the ratio of the total number of false track-seconds to the total number of aircraft-seconds associated with actual aircraft.
 - h. The data **shall** (2052) also be examined to determine whether any single event such as crossing targets, multipath-induced image targets, the inability to maintain track at closest-point-of-approach, or a synchronously garbled target dominates the false track rate or the rate at which actual targets are dropped. If the false tracks or dropped tracks

are due mostly to one type of interference mechanism or geometric condition, it could indicate a design deficiency in the area even though the overall surveillance track probability and false track ratio are acceptable. If this condition exists, the applicant **shall** (2053) present this information and provide an evaluation summary that indicates an acceptable design.

- i. The applicant **shall** (2054) present sufficient data in the form of reply plots, surveillance track plots, printouts, tabulations, etc., to substantiate that the Mode C surveillance performance on targets-of-interest, measured as a result of the surveillance flight test, meets or exceeds the following acceptance criteria:
 1. ACAS X surveillance probability for targets-of-interest > 95%, and
 2. ACAS X false track rate for targets-of-interest < 1.2%.

The applicant **shall** (2055) also present an evaluation summary on any single dominant failure event as discussed in h. above.

3.4.4.2

Mode S Surveillance Flight Tests

- a. A flight test under environmentally stressful conditions **shall** (2060) be performed to provide the data for a complete Mode S surveillance design validation. This environment includes the combined effects of fruit interference as a result of high aircraft density and ground interrogator interference from multiple ground interrogators.
- b. Flight testing **shall** (2061) be accomplished in an area of high Mode S density such as Atlanta, Chicago or New York. Previous tests have shown that these areas provide the most stressful environment necessary for evaluation of Mode S surveillance. Chicago and New York have provided average airborne densities of 0.029. Either Atlanta, Chicago or New York are considered adequate to test the Mode S surveillance; however, New York has provided the most dense Mode S traffic. The most effective flight paths have proven to be orbital flights of 5 NM radius near the major airport terminal: Hartsfield in Atlanta, JFK in New York and O'Hare in Chicago.
- c. The flight test **shall** (2062) be conducted at an altitude less than 10,000 feet MSL during peak traffic periods when the ground visibility is greater than 10 NM with a ceiling of at least 11,000 feet MSL to ensure the highest peak traffic densities. Other locations may be proposed by the applicant and will be considered as a suitable alternative to the above areas if the applicant can demonstrate that the ACAS X surveillance test was conducted in an environment whose combined or airborne average density of transponder-equipped aircraft equals or exceeds those of Atlanta, New York or Chicago and indicates a continuous minimum NTA count of 75. Density is defined as the number of other transponder-equipped aircraft within 10 NM of the ACAS X test aircraft divided by the area of a circle of 10 NM radius. Thirty-one actual aircraft targets occurring simultaneously within 10 NM of the ACAS X test aircraft is equivalent to a density of 0.1 transponder-equipped aircraft per sq NM. Alternate locations must also contain a minimum of three civil ATC or military secondary surveillance radars located within 30 NM of ownship in order to provide an interference environment similar to the above areas.

Note: Although Atlanta, New York and Chicago are presently the highest known densities, future development or test flights may define areas of greater density. If this occurs, for purposes of maximum system stress, the Mode S Surveillance Flight should be flown in these newly defined areas.

- d. Density estimates **shall** (2063) be obtained during a flight test of a minimum duration of 45 minutes. Track analysis data **shall** (2064) be obtained during the 45 minute flight and will contain an accumulation of 5,000 to 8,000 seconds of target track reports on targets-of-interest. More than 8,000 track-seconds need not be analyzed. However, an aircraft density estimate for at least 45 minutes **shall** (2065) be provided. A target-of-interest is defined as any target-of-opportunity that is:
 - 1. Within 10 NM,
 - 2. Within ± 10 degrees elevation, and
 - 3. Altitude reporting and airborne.

It is expected that the applicant will have prior knowledge of the density conditions in the flight area and will select the flight time that will provide the maximum density.
- e. The recorded data **shall** (2066) include as a minimum all raw reply reports prior to reply correlation, processed surveillance track reports that have been declared established, TAs and RAs, internal processor clock, and time-of-day. Recorded flight data **shall** (2067) be provided to the FAA or equivalent aviation authorities to permit an independent assessment of surveillance performance in accordance with paragraph g. below. Details concerning the format and data types of the data to be provided **shall** (2068) be coordinated with the FAA or equivalent aviation authorities in a timely manner. Sixty days or more in advance of the availability of the recorded flight data is recommended.
- f. The applicant **shall** (2069) perform at least the following specific post-flight data analyses:
 - 1. Identify all recorded surveillance target track reports that are established and that satisfy the criteria for a target-of-interest.
 - 2. Identify those established target-of-interest track reports that are associated with actual aircraft.
 - 3. Determine the total number of aircraft-seconds associated with actual aircraft and the total number of true track-seconds (i.e., those update intervals in which an established surveillance track report was either coasted or updated with a valid reply) associated with actual aircraft.
 - 4. Determine the surveillance track probability as the ratio of the total number of true track-seconds associated with actual aircraft to the total number of aircraft-seconds associated with actual aircraft.
 - 5. Identify those established target-of-interest track reports that are not associated with actual aircraft.
 - 6. Determine the total number of track-seconds that are not associated with actual aircraft. These are classified as false track-seconds.
 - 7. Determine the false track probability as the ratio of the total number of false track-seconds to the total number of aircraft-seconds associated with actual aircraft.
- g. The applicant **shall** (2070) present sufficient data in the form of reply plots, surveillance track plots, printouts, tabulations, etc., to substantiate that the Mode S surveillance performance on targets-of-interest, measured as a result of the surveillance flight test, meets or exceeds the following acceptance criteria:
 - 1. Surveillance probability for targets-of-interest $> 95\%$, and

2. False track rate for targets-of-interest = 0%.

3.4.4.3 Coordination Flight Tests

- a. Planned ACAS X coordination encounters **shall** (2071) be performed in order to evaluate the reliability of the overall air-to-air coordination data link for installed ACAS X and Mode S transponder equipment. This is meant to test the reliability of the RF link between aircraft systems during resolution advisory coordination. For systems for which the RF link has been previously verified and for which changes to the hardware/software components do not affect the RF link, this testing is not required.
- b. The encounters **shall** (2072) be flown at 3,000 feet AGL over calm water in order to increase the likelihood of multipath and in configurations that maximize the times during which an RA is present and coordination is taking place. Examples of such encounters are tail chases and parallel flight tracks flights. If the lateral separation necessary for multipath is larger than the separation that maintains the RA, the aircraft can periodically turn in towards each other during parallel flight tracks flights in order to continually initiate RAs.
- c. The encounters **shall** (2073) be flown so as to involve a reasonable mix of the four quadrants of the ACAS X and transponder antennas.
- d. The system under test **shall** (2074) have sufficient recording capability to record the coordination interrogations and replies received in response to the coordination interrogations. The recorded data **shall** (2075) include at least 2,000 ACAS X coordination scans.
- e. The criteria for acceptable performance is as follows:
 1. The probability of a single scan coordination link dropout **shall** (2076) be no greater than 0.1%.
 2. There **shall** (2077) be no coordination link dropouts that exceed one scan in duration.

3.4.4.4 Passive Surveillance Flight Tests

The following key objectives **shall** (2078) be demonstrated through flight test:

1. ACAS X Mode S surveillance performance is not degraded.
2. Ability to validate and track an aircraft passively. This includes:
 - Ability to track an aircraft passively based on validation criteria for the Hybrid Surveillance Region while in that region.
 - Ability to track an aircraft passively based on validation criteria for the Active Surveillance Region while in that region.
 - Ability to correctly exhibit revalidation intervals while in the Hybrid Surveillance Region.
 - Ability to correctly exhibit revalidation intervals while in the Active Surveillance Region.
3. Ability to correctly transition between surveillance regions. This includes:
 - For ADS-B Link Version 0:
 - Ability to correctly transition to active surveillance.
 - Ability to correctly transition from active surveillance back to passive surveillance.
 - For ADS-B Link Version 1 & 2:
 - Ability to correctly transition to the appropriate revalidation interval when going from the Hybrid Surveillance Region to the Active Surveillance Region.
 - Ability to correctly transition to the appropriate revalidation interval when going from the Active Surveillance Region back to the Hybrid Surveillance Region.

A ground test **shall** (2079) be performed to demonstrate correct performance of the DF=17 squitter capability of flight test aircraft and that correct ownship data is provided to the ACAS X prior to flight testing.

Objective 1 **shall** (2080) be met by performing the Mode S Flight Test specified in §3.4.4.2. Objectives 2 and 3 can be demonstrated with planned encounters or with targets-of-opportunity during the Passive Surveillance flight test. If Extended Hybrid Surveillance target-of-opportunity tests are conducted, the Extended Hybrid Surveillance Traffic Quality Requirements of §2.2.4.6.4.2.2.1.1 **shall** (2081) be met. If planned encounters are performed to demonstrate correct performance of DF=17 squitter capability of the planned encounter aircraft, a ground test **shall** (2082) be performed prior to flight testing. In addition, prior to flight testing, ownship position sources **shall** (2083) meet the requirements of §2.2.4.6.4.2.2.1.1 and these required quality standards **shall** (2084) be established with both targets-of-opportunity or planned encounters flights.

Note: It is required that the tracking function must have the capacity for active surveillance of at least 30 aircraft. As noted in §2.2.4.6.1.1, targets in excess of 30 will be deleted in order of decreasing range. Since New York consistently exhibits 30 or more tracks, minimum data may be accumulated on Extended Hybrid Surveillance as a result of their long ranges. These will frequently be deleted when the 30 tracks are exceeded. Los Angeles also frequently exhibits 30 tracks and the same condition will be noted.

These Passive Surveillance flight tests **shall** (2085) be performed in conjunction with the Mode S surveillance flight test, with the following additional requirements:

- a. This target-of-opportunity flight test **shall** (2088) be a minimum of one hour, and a minimum of 14 passive surveillance tracks **shall** (2089) be analyzed.
- b. If planned encounters are conducted, the flight program **shall** (2123) include those performance characteristics listed in section 2.

3.4.5

Bearing Estimation Tests

The purpose of these tests is to determine those azimuthal regions, if any, where the installed bearing estimation antenna and all associated bearing estimation subsystems fail to meet the success criteria for bearing estimation specified in §3.4.5.1. This may be done using one or a combination of four distinct procedures: (a) antenna range measurements, (b) airborne measurements, (c) scaled-modeled measurements or (d) theoretical calculations. Procedures (b), (c) and (d) require additional data to corroborate the validity and calibration accuracy of the procedure.

3.4.5.1

Success Criteria for Bearing Estimation Measurements

The test **shall** (2124) identify those azimuthal regions in which the bearing estimation system in the absence of external sources of interference or multipath, fails to provide target bearing estimation accuracy sufficient to support a pilot TA service. Specifically, the test **shall** (2125) identify all clock-position sectors within which the peak bearing estimation error for a Mode S reply persistently exceeds 30 degrees at zero degrees elevation relative to the fuselage reference plane. The clock position **shall** (2126) be identified if the error exceeds the stated bound at any azimuth angle within its sector. A transponder is used as the calibration source in the range or airborne measurements to enable the ACAS X equipment to operate in its normal manner when making reply bearing estimates. A Mode S transponder target is used rather than an ATCRBS transponder to avoid ATCRBS synchronous garble effects, to provide positive target identification and to minimize the effects of thermal noise on the measurement.

3.4.5.2

Antenna Range Measurements of Bearing Accuracy

Bearing estimation accuracy for the antenna and bearing estimation system as mounted on the actual airframe may be measured using a calibrated antenna range that allows precise echo-controlled, far-field, angle-of-arrival measurements over a small range of positive and negative elevation angles and over 360 degrees in azimuth.

- a. The location, the direction and the directivity of the test target antenna must be chosen to ensure that ground reflections or multipath reflections from structures or other aircraft near enough in range to provide overlapping signals are attenuated by at least 20 dB relative to the direct test signal. Figure 3-2 illustrates some test geometry considerations. The site should also be monitored carefully during the test to ensure that specular reflections from moving aircraft or ground vehicles do not contaminate the test results.
- b. The distance of the test target from the aircraft **shall** (2127) be such that the maximum dimensions of the aircraft subtend an angle of less than 10 degrees at the test target. The pitch attitude of the aircraft **shall** (2128) be such that the fuselage reference plane is within five degrees of horizontal.
- c. Point the aircraft toward the target transponder using an appropriate precise means of mechanical or visual bearing calibration. Interrogate the target transponder using the

normal ACAS X surveillance process. Record the target transponder relative bearing as indicated on the plan position display of TA information, or other special test display.

Note: Any bearing bias error measured using procedures of §2.4 should be subtracted from the indicated relative bearing.

- d. Rotate the aircraft or move the test set (being careful to account for multipath) one clock position (30 degrees) and repeat the measurement. Repeat the measurement for each clock position.

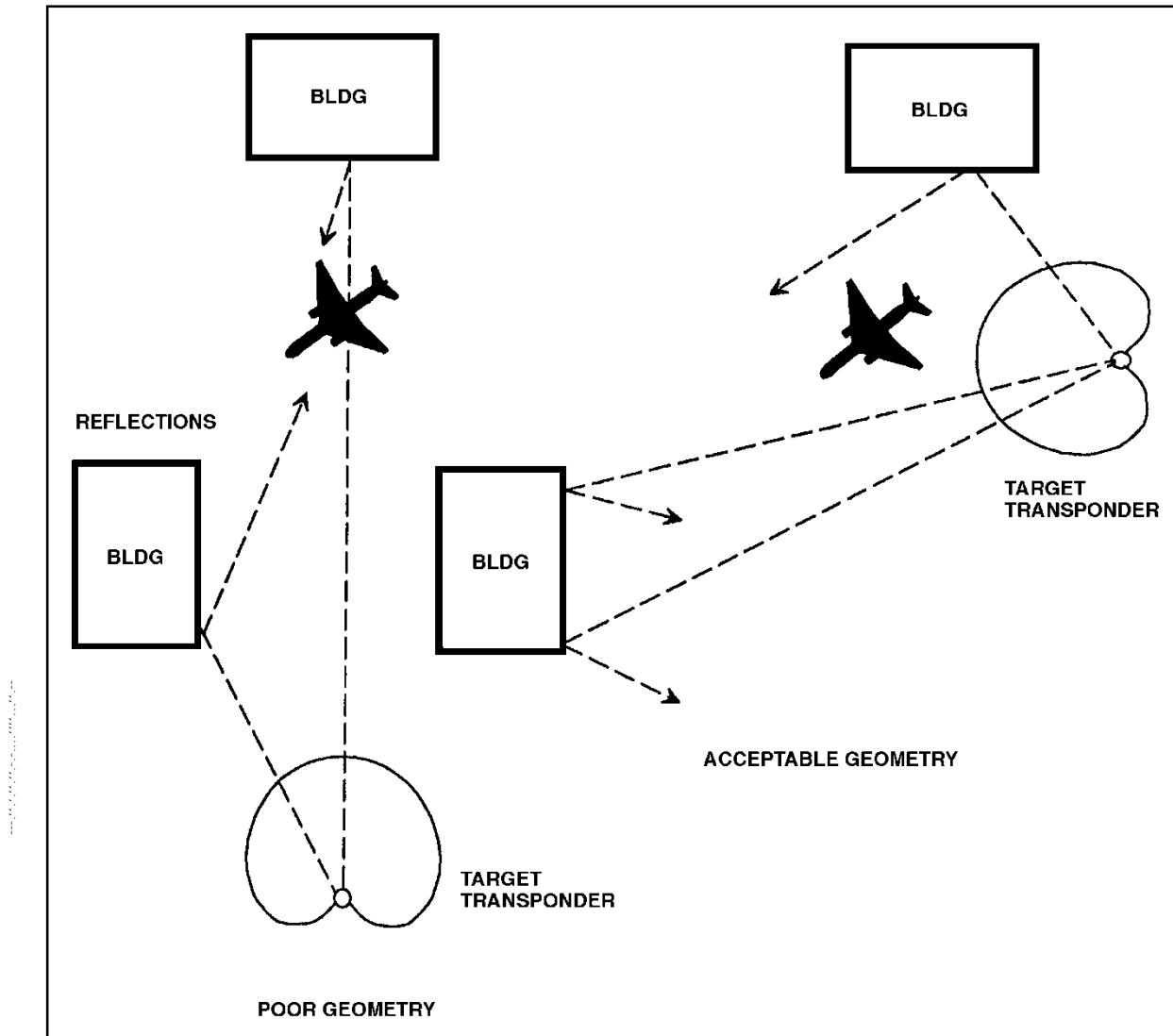


Figure 3-2: Ramp Test Geometry Considerations

3.4.5.3**Airborne Measurements of Bearing Accuracy**

Bearing estimation accuracy may optionally be measured by means of a flight test. The following suggested procedure could be followed in a region of low traffic density, but any other test that provides equivalent data would be acceptable.

A target aircraft equipped with a Mode S transponder may be used for calibration of bearing estimation accuracy. Such measurements are most conveniently made when the target aircraft and ownship under calibration have equal speed capability and fly at relatively high altitude for maximum immunity to multipath. The flight paths of the two aircraft are chosen to provide extended periods of constant target bearing. This is most conveniently accomplished at bearing angles of 0, 90, 180 and 270 degrees. However, calibration at each clock position requires a means of optical calibration (visual sighting) of the true bearing angle from the cockpits of both aircraft.

Calibration of angles other than the four cardinal bearings is best accomplished by means of slow overtaking encounters. (The bearing angle must change sufficiently slowly so that tracker lag does not introduce significant additional errors).

Manual readout of the bearing estimate may be accomplished directly from a plan position display of TA information. Alternatively, the bearing estimates may be automatically recorded or may be read from a special test display.

An alternative method for flight test calibration employs a Mode S transponder mounted on a building or tower. Bearing measurements may be calibrated by visual sighting or by means of conventional navigational techniques as ownship flies by the transponder.

3.4.5.4**Scaled Model Measurements of Bearing Accuracy****3.4.5.4.1****Aircraft Model**

The aircraft modeling criteria given in §3.4.3.3.1 are adequate to test for the effects of the aircraft structure upon the bearing estimation accuracy.

3.4.5.4.2**Antenna Model**

The scaled model antenna should include the inherent characteristics of the antenna elements and their drivers, splitting or combining networks and any effects due to mutual coupling between the elements. Model measurements should be made at sufficiently high signal-to-noise ratios to eliminate the effects of noise on the accuracy of the bearing estimates.

3.4.5.4.3**Model Tests**

These tests are similar to the tests of §3.4.3.3.2 but with additional bearing measurement requirements.

- a. Mount the scaled model antenna in the center of a ground plane whose radius is equal in wavelengths to the ground plane used for testing the full-scale antenna.
- b. Using a calibrated anechoic antenna test range, confirm that, on the ground plane, the bearing estimates are within 10 degrees of the full-scale antenna estimates for azimuth angles from zero to 360 degrees and elevation angles from -15 to +20 degrees.

-
- c. Mount the scaled model antenna on the aircraft model at the intended installation location.
 - d. Measure the antenna bearing estimates for elevation angles from -15 to +20 degrees and for all azimuth angles.

Note: This measurement can be performed using either analog or digital instrumentation. Azimuth data should be acquired every two degrees or less. Elevation angle data should be acquired in increments of five degrees or less.

3.4.5.5 Theoretical Calculations of Bearing Accuracy

The bearing estimation accuracy of the antenna as mounted on the actual airframe may be determined indirectly by a combination of radiation pattern calculations and measurements designed to validate those calculations. When using such techniques to determine the bearing accuracy of a multi-element antenna, it is necessary to show that the calculations include the inherent characteristics of the antenna elements and their drivers, splitting or combining networks and any effects due to mutual coupling between those elements.

3.4.5.5.1 Validation of Theoretical Calculations of Bearing Accuracy

If radiation pattern calculations are used to determine installed bearing estimation accuracy, the manufacturer of the antenna must provide corroborating data demonstrating the success of the calculation technique in predicting the bearing estimates on an airframe roughly similar in size and complexity to the airframe under qualification. Such data must be obtained by comparison with selected bearing measurements made (a) on a full-size airframe on an antenna range or (b) on a scaled model airframe as indicated in §3.4.5.4.

This Page Intentionally Left Blank

MEMBERSHIP

RTCA Special Committee 147
Aircraft Collision Avoidance Systems

RTCA Special Committee 147 Chairman

Ruy Brandao
J. Stuart Searight

Organization

Honeywell International, Inc.
FAA William J. Hughes Technical Center

Government Authorized Representative

Sheila Mariano

Organization

Federal Aviation Administration

RTCA Special Committee 147 Secretary

Donna Froehlich

Organization

Aurora Innovations

EUROCAE Working Group 75 Chairman

Bill Booth

Organization

EUROCONTROL

RTCA Program Director

Al Secen

Organization

RTCA, Inc.

RTCA Document Editor

Perla Domingo

Organization

RTCA, Inc.

EUROCAE Technical Manager

Alexander Engel

Organization

EUROCONTROL

Members

Deebu Abi
Eva Adamova
Michael Allouche
Luis Alvarez
Massimiliano Amirfeiz
Craig Anderson
Igor Andreev
Michael Angleraud

Organization

Garmin Ltd.
Honeywell International, Inc.
Israel Aerospace Industries
MIT Lincoln Laboratory
Leonardo SpA
The Boeing Company
VNIIRA-Navigator JSC
Safran

Cengiz Ari	SESAR JU
Thierry Arino	Egis Avia
Christian Aveneau	DSNA
Ben Bakker	EUROCONTROL
Chris Baum	Air Line Pilots Association (ALPA)
Raymond Bayh	BAE Systems, Inc.
Walter Bender	The Johns Hopkins University
Karim Benmeziane	BNAE
Peter Bess	Air Line Pilots Association (ALPA)
Piotr Binek	Becker Avionics
Rich Boll	National Business Aviation Association
Thierry Bourret	Airbus
Joseph Boyd	The MITRE Corporation
Guillaume Brat	NASA
Gabriel Brewer	The Boeing Company
Emily Bromberg	The MITRE Corporation
Jeffrey Brush	The Johns Hopkins University
Jose Cabanilla	Honeywell International, Inc.
Paul Campbell	Federal Aviation Administration (FAA)
Lawrence Capuder	MIT Lincoln Laboratory
Joslin Carino	Federal Aviation Administration (FAA)
Ken Carpenter	QinetiQ (EUROCAE Member)
Michael Castle	Aurora Innovations
David Cavone	Thales Group
Sherry Chappell	Federal Aviation Administration (FAA)
Zhong Chen	Honeywell International, Inc.
Kathryn Ciaramella	Federal Aviation Administration (FAA)
Michel Cochin	BNAE
Emilie Cowen	MIT Lincoln Laboratory
Giuseppe D'Angelo	Leonardo SpA
Jonathan Dahlgren	Honeywell International, Inc.
Garfield Dean	EUROCONTROL
Teddyson deGuzman	Federal Aviation Administration (FAA)
Juan Ignacio del Valle	EDA

Esther Delgado Pinedo	Indra Sistemas
Stanislaw Drozdowski	EUROCONTROL
Ann Drumm	MIT Lincoln Laboratory
Yu Duan	Honeywell International, Inc.
Robert Duffer	MIT Lincoln Laboratory
Tom Eich	L-3 Communications
Dave Elliott	The MITRE Corporation
Nourhan El Zarif	Honeywell International, Inc.
Alexander Engel	EUROCAE
Josh Estfan	L3Technologies
Julien Farjon	Sagem Avionics, Inc.
Jack Field	The MITRE Corporation
Craig Foster	UK National Air Traffic Services
Amy Fritz	The MITRE Corporation
Richard Fukutome	Honeywell International, Inc.
Wayne Gallo	Federal Aviation Administration (FAA)
Ryan Gardner	The Johns Hopkins University
Robin Garrity	SESAR JU
Jean Rene Gely	Thales Communications
William L. Geoghegan	National Air Traffic Controllers Association
Dimitra Giannakopoulou	NASA
Edward Glowacki	Federal Aviation Administration (FAA)
Jose Gonzalez	Federal Aviation Administration (FAA)
Josue Gonzalez	Federal Aviation Administration (FAA)
Carl Graf	Hensoldt Sensors GmbH
Yvonne Graner	DFS
Elena Gromova	GOSNIIAS
Alexandre Guignot	DGAC/DTA/STAC
Edward Hahn	Air Line Pilots Association (ALPA)
Kevin Hallworth	European Aviation Safety Agency (EASA)
Nathalie Hasevoets	EDA
Marc Henegar	Air Line Pilots Association (ALPA)
Jens Hennig	Hensoldt Sensors GmbH
Carlos Hernandez Medel	Everis Aeroespacial y Defensa, SLU
DO-385	© 2018 RTCA, Inc.

Eric Hoffman	EUROCONTROL
Marie Hogestad	Federal Aviation Administration (FAA)
Volker Huck	EUROCONTROL
Chris Jackman	Google
Randy Jacobson	Rockwell Collins, Inc.
Ravi Jain	Federal Aviation Administration (FAA)
Ian Jessen	MIT Lincoln Laboratory
Brian Jolly	EASA
Mischael Joseph	Federal Aviation Administration (FAA)
Robert Joslin	Federal Aviation Administration (FAA)
Damien Jourdan	Google
Gilles Jurquet	Thales Group
Bill Kaliardos	Federal Aviation Administration (FAA)
Randy Kenagy	Air Line Pilots Association (ALPA)
Alexey Khripunov	RPE CRTS LLC
Vladimir Khudoshin	VNIIRA-Navigator JSC
Pavel Klang	Honeywell International, Inc.
Robert Klaus	MIT Lincoln Laboratory
Barbara Kobzik-Juul	The Johns Hopkins University
Mykel Kochenderfer	MIT Lincoln Laboratory
Tatsuya Kotegawa	General Atomics Aeronautical Systems, Inc.
Andreas Krebber	DFS
Jan Kubalcik	Honeywell International, Inc.
Fabrice Kunzi	General Atomics Aeronautical Systems Inc.
Rene L. C. Eveleens	NLR
Maurice Labonde	Airbus Defence & Space
John Law	JL SURACAS Ltd.
Patrick Le Blaye	ONERA
Charles Leeper	The Johns Hopkins University
Ted Lester	The MITRE Corporation
Nicolas Leon	DSNA
Claude Le-Tallec	ONERA
Didier Lion	Safran
Huan Liu	Thales Group

Edward Londner	MIT Lincoln Laboratory
Jessica Lopez	The Johns Hopkins University
Felix Lopez Perez	Hensoldt Sensors GmbH
Edwin Lorenzo	The Johns Hopkins University
Erika Majchrzak	Lufthansa Technik AG
David Manda	Spire Global
Jean-Francois Marc	Thales Group
Charlene Mari	Egis Avia
Sheila Mariano	Federal Aviation Administration (FAA)
Steffen Marquard	DFS
Andre Marques	Airbus SAS
Johan Martensson	EUROCONTROL
Sean Martin	The Johns Hopkins University
Raymond McDonald	Regulus Group
Cynthia McLain	MIT Lincoln Laboratory
Jörg Meyer	Airbus Defence & Space
Christian Minor	MIT Lincoln Laboratory
Walter Monk	CSSI, Inc.
Robert Moss	MIT Lincoln Laboratory
Russell Nowy	QinetiQ
Jennifer O'Brien	The Johns Hopkins University
Wesley Olson	MIT Lincoln Laboratory
Michael Owen	MIT Lincoln Laboratory
Thomas Pagano	Regulus Group
Adam Panken	MIT Lincoln Laboratory
Bharath Parthasarathy	Garmin Ltd.
Darrell Pennington	Air Line Pilots Association (ALPA)
Christian Perez	MIT Lincoln Laboratory
Michael Petri	Regulus Group
Erik Petrini	Saab Aeronautics
Shirley Phillips	MIT Lincoln Laboratory
Eric Potier	EUROCONTROL
Sylvain Pouillard	Safran
Beatrice Raynaud	Egis Avia
DO-385	

Joseph Raynes	The Johns Hopkins University
Boris Resnick	IANS
Jean-Luc Robin	Airbus
Stacey Rowlan	L-3 Communications
Vsevolod Sadilov	IANS
Anshu Saksena	The Johns Hopkins University
Lucia Sanz	Egis Avia
Jonathan Saunders	Aurora Innovations
Aurora Schmidt	The Johns Hopkins University
Carlos Alberto Schroeder Reguse	Hensoldt Sensors GmbH
Al Secen	RTCA, Inc.
Taji Shafaat	The Boeing Company
Chris Shaw	EUROCONTROL
Eduard Shestak	Honeywell International, Inc.
Nishit Sheth	Federal Aviation Administration (FAA)
Vladislav Shifrin	RPE CRTS LLC
Ori Shloosh	IATAS
Alan Sigman	Federal Aviation Administration (FAA)
Josh Silbermann	The Johns Hopkins University
Dierk Steinbach	Hensoldt Sensors GmbH
Ben Strutin	Air Line Pilots Association (ALPA)
Brandon Suarez	General Atomics Aeronautical Systems, Inc.
Neal Suchy	Federal Aviation Administration (FAA)
Bengt-Goran Sundqvist	Saab
Rachel Szczesiul	The Johns Hopkins University
Tom Teller	MIT Lincoln Laboratory
Eric Thomas	Rockwell Collins International
Marc Thouzeau	Dassault Aviation
Dan Tillotson	Rockwell Collins, Inc.
Dwight Unruh	Rockwell Collins, Inc.
Eric Vallauri	Egis Avia
Alain Vallee	Safran
Jeremy Vanderhal	Federal Aviation Administration (FAA)
Samuel Wahyou	Honeywell International, Inc

Mingwei Wang	COMAC America
QuoQing Wang	Honeywell International, Inc.
Jerry Welch	MIT Lincoln Laboratory
Jared Wikle	MIT Lincoln Laboratory
Mark Williams	SAIC
Kevin Wilson	Honeywell International, Inc.
Samantha Witt	Regulus Group
James Won	MIT Lincoln Laboratory
Loren Wood	MIT Lincoln Laboratory
Andrey Yablokov	RPE CRTS LLC
Zhu Yanbo	ADCC
Tyler Young	The Johns Hopkins University
Andrew Zeitlin	The MITRE Corporation
Rong Zhang	Honeywell International, Inc.
Jinghua Zheng	Honeywell International, Inc.
Benjamin Zintak	The Johns Hopkins University
Farid Zizi	DGAC/DTA/STAC

This Page Intentionally Left Blank

APPENDIX A SURVEILLANCE PROCEDURES

This appendix was written for RTCA/DO-185B. The sections in this appendix describe example implementations for MOPS requirements as reference. For ACAS X, some of those algorithms or similar algorithms were included as suggested algorithms within the ADD. In this case, a note at the beginning of the section will provide information about similarities and differences of the described algorithm and the ADD.

A.1 Interference Limiting Procedure

The following describes one possible implementation of an interference limiting procedure. It varies the system parameters appearing in inequalities 1, 2, and 3 of §2.2.3.6.1 to maximize and maintain approximate equality between the estimated surveillance ranges for Mode S and ATCRBS targets.

In evaluating these inequalities 8-second averages of the Mode S parameters are used, and current or anticipated values of the ATCRBS parameters are used.

The procedure is illustrated in the flowchart of Figure A-1. Depending on the specific whisper-shout sequence in use, the first step in the control process is to reduce either the number of whisper-shout steps or the whisper-shout power level of the sequence tentatively scheduled for use during the present scan if either:

Inequality 3 is violated, or

Inequality 1 or 2 is violated and the Mode S surveillance range of the last scan does not exceed the ATCRBS surveillance range that would result from use of the scheduled whisper-shout sequence.

If the high resolution whisper-shout sequence is in use, whisper-shout steps for an airborne ACAS X are eliminated from that sequence in the order dictated by the limiting priority in §2.2.4.5.4.1.2 and the number of steps eliminated is just large enough to insure that neither of the above conditions is satisfied. If the minimum basic whisper-shout sequence or the single interrogation is in use, the power of each of the whisper-shout steps or the single interrogation in a directional beam for an airborne ACAS X is reduced in 1-dB steps with the order of beams dictated by the limiting priority in §2.2.4.5.4.1.1 and §2.2.4.5.4.1.4 and the amount of power reduction is just large enough to insure that neither of the above conditions is satisfied. The value of the number of whisper-shout steps for a high resolution sequence or the power level of the largest step for the minimum basic whisper-shout sequence or the single interrogation tentatively scheduled for use is initialized at the number used on the last scan.

For a ACAS X on the ground, the reduction or elimination of whisper-shout steps is modified from the process used for airborne ACAS X such that steps are first eliminated or reduced in the forward beam until it matches the sequence in the side beams and then the steps in the forward and side beams are eliminated or reduced until they match the sequence in the rear beam. Further reduction is accomplished by sequentially eliminating or reducing steps in all beams.

The relative magnitudes of the Mode S and ATCRBS surveillance ranges are determined from the estimated effective radiated power (ERP) seen by targets with Mode S and ATCRBS transponders located directly ahead of ownship. The ERP in a given direction is determined by the product of the power input to the antenna, and the antenna pattern gain in that direction. If the transponder sensitivities were identical, the Mode S range would be more or less than the ATCRBS range according to whether the Mode S transmitted power was more or less than the ATCRBS transmitted power. Since ATCRBS transponders may have somewhat lower sensitivities than Mode S transponders, the ATCRBS range is assumed to be greater than the Mode S range if, and only if, the ATCRBS power exceeds the Mode S power by 3 dB.

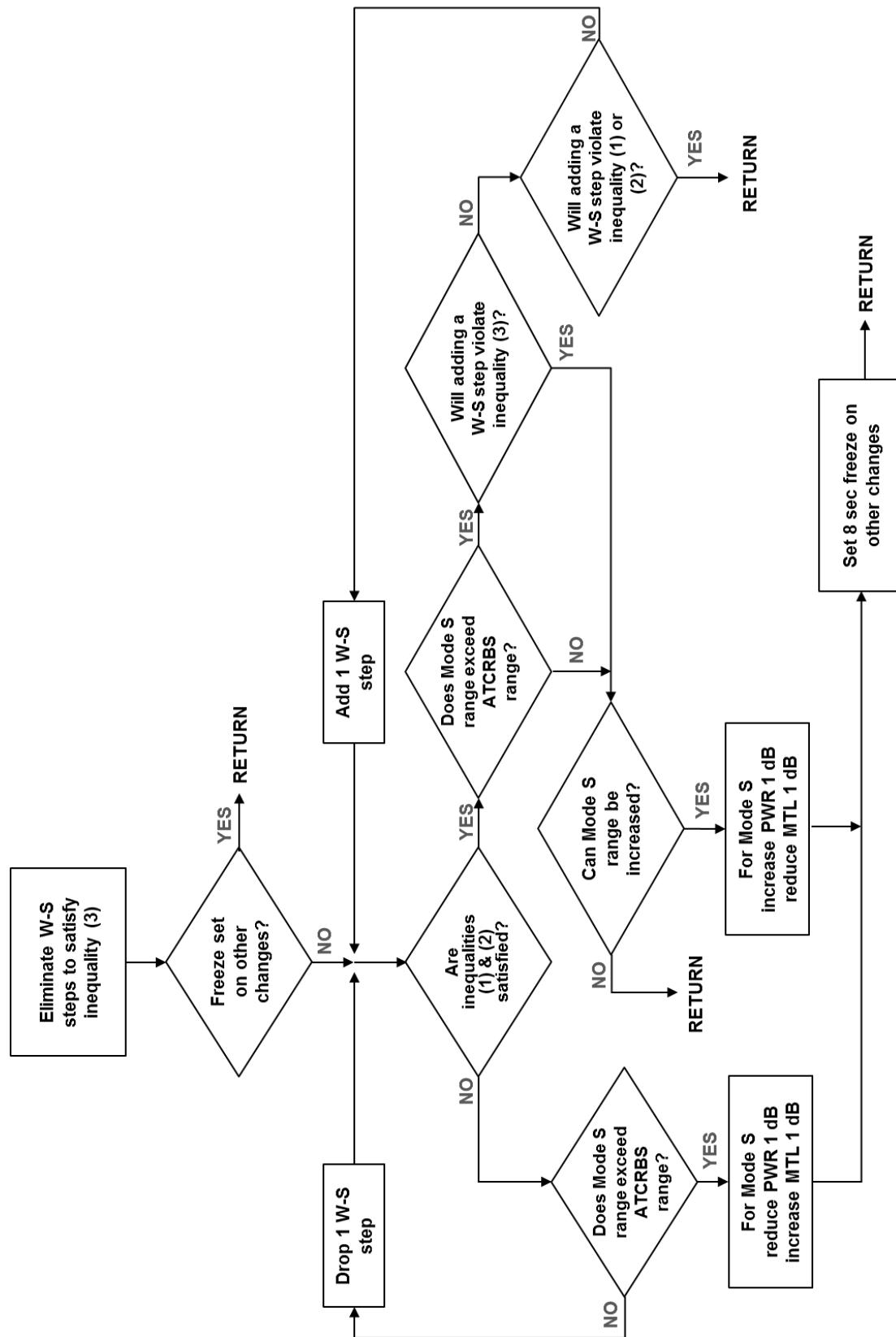


Figure A-1: Interference Limiting Flow Diagram.

The second step in the controlling process is to reduce the Mode S interrogation power for acquisition by 1 dB, and to increase the MTL for Mode S squitter listening by 1 dB from the values last used, if:

Inequality 1 or 2 is violated and the Mode S surveillance range of the last scan exceeds the ATCRBS surveillance range that would result from use of the scheduled whisper-shout sequence.

Once such a change has been made the only change allowed during the ensuing 8 seconds is a reduction in the number of whisper-shout steps if such is needed to satisfy Inequality 3. This 8-second freeze allows the effect of the Mode S changes to become apparent since the 8-second averages used in Inequalities 1 and 2 then will be determined by the behavior of the system since the change.

The third step is to add a whisper-shout step to those tentatively scheduled if using the **high resolution sequence or add 1 dB to each tentatively scheduled level in a beam** if using the minimum basic whisper-shout sequence or the single interrogation, when it is not prevented by an 8-second freeze, and the following conditions are satisfied:

Inequalities 1, 2, and 3 are satisfied and will continue to be satisfied after step e. is added, and

The Mode S surveillance range of the last scan exceeds the ATCRBS surveillance range that would result from use of the scheduled sequence.

As many steps are added as possible without violating d. or e. above.

Finally, if condition d. above is satisfied, but condition e. is not, an estimate is made of the effects of increasing the Mode S interrogation power for acquisition by 1 dB and reducing the MTL for Mode S squitters/fruit by 1 dB. If the estimate indicates that Inequalities 1 and 2 will not continue to be satisfied the 1 dB change is not made. If the estimate indicates that they will continue to be satisfied, the 1 dB change is made and no further changes in either the ATCRBS or Mode S parameters are made for the ensuing 8 seconds, except as described in connection with condition c.

A.2 Rejection of Reply Link Interference

One possible means of rejecting low level multipath signals is to use the dynamic minimum trigger level (DMTL) function described below. Variable receiver thresholds have usually been avoided in ATCRBS reply processors because they tend to discriminate against weak replies. However, when used in conjunction with the minimum required whisper-shout technique (§2.2.4.5.4.1.1), this disadvantage of dynamic thresholding is largely overcome.

ATCRBS Reply Reception

When listening for ATCRBS replies and upon receipt of the first pulse of an ATCRBS reply of amplitude A, when A exceeds MTL+13 dB, the receiver threshold is increased to $A-9 \text{ dB} \pm 1 \text{ dB}$ for a period of not less than 21 microseconds following the leading edge of the first reply pulse and is recovered in not more than 26 microseconds. The receiver threshold should at no time exceed $A-8 \text{ dB}$ except for possible overshoot during the first microsecond following the desensitizing pulse.

If A is less than MTL+11 dB, there is no need to raise the threshold.

Mode S Squitters and Reply Reception

When listening for Mode S squitters or replies and upon receipt of the first pulse of a Mode S transponder squitter or interrogation reply of amplitude A, where A exceeds MTL+10 dB, the receiver threshold is increased to $A-6 \text{ dB} \pm 1 \text{ dB}$ for a period of not less than 5 microseconds following the leading edge of the first reply pulse and is recovered in not more than 8 microseconds, unless a valid or qualifying preamble is received (§2.2.4.4.2.2), in which case the threshold is held at $A-6 \text{ dB} \pm 1 \text{ dB}$ for a period of not less than 115 microseconds and is recovered in not more than 120 microseconds. The receiver threshold should at no time exceed $A-5 \text{ dB}$ except for possible overshoot during the first microsecond following the desensitizing pulse. If A is less than MTL+8 dB, there is no need to raise the threshold.

The length of a Mode S reply cannot be determined with certainty by the DMTL system. Therefore, the Mode S DMTL extends for the duration of the long Mode S reply.

A.3 Mode S Error Correction

The following is an algorithm for error detection and correction of Mode S squitter replies (DF=11 and DF=17) and Mode S replies to addressed interrogations, (DF=0 and DF=16). This algorithm may be implemented in hardware or software. The configuration of the implementation may be inferred from the details of the corresponding figures.

Message Decoding - The function of message decoding is to first check the bit decision sequence for errors by means of a parity check, and if errors are detected, to use the confidence bit sequence to attempt to locate a burst error pattern spanning up to 24 bit decisions that would correct the message.

The inputs to the message decoder are:

Bit decision sequence (D_i)

Confidence bit sequence (C_i).

Reply length indicator bit (L).

Expected address (A_i , where $i=1\dots24$).

The derivation of the Bit Decision and the Confidence Bit Sequences is described in §2.2.4.4.2.2c.

The decoder outputs are:

Error correction enable bit, m (1 bit).

The decoded message, if $m=1$, (32 or 88 bits).

Message decoding is treated in three stages; error detection, error pattern location, and error correction.

Error Detector - The primary function of the error detection is to compute a 24-bit error syndrome. This computation is accomplished in two steps. First, the decoded address is calculated from the sequence of bit decisions. Then the syndrome is obtained by a comparison of the decoded address with the expected address. If the two addresses agree, then the error syndrome is zero (i.e., all component bits are zero), and the decoded message will ultimately be obtained directly from the bit decision sequence.

Initial Conditions for Error Detector - The error detector and initial conditions are shown in Figure A-2. The DB register need not be initialized and is used to store the bit decision sequence as it is produced by the bit processor. The CB register is used to store the confidence bit sequence as it is produced by the bit processor and is initialized by setting the 56 stages closest to the input to ONE. The A register, used to determine the decoded address A_i , $i=1\dots24$, is initialized by resetting all 24 stages to ZERO.

Confidence Test - As the confidence bits are being loaded into the CB register, a confidence test maintains a count of the number of high-confidence bits contained in the 24 elements closest to the input side of the register. (Note: The count is initialized to 24.) If the counter falls below 12 while the CB register is being loaded, the correction disable bit is set to ONE. Correction of an erroneous message is not allowed if the correction disable bit has been set.

Final Conditions of Error Detector - The bit processor produces sequentially the bit decision sequence in the order D_1, D_2, \dots, D_l and the confidence bit sequence in the order C_1, C_2, \dots, C_l , with C_i and D_i made available to the message decoder together after the i th bit interval has been processed. The lengths of these sequences (I) is determined by the message length indicator bit L ($I=56$ for $L=0$, $I=112$ for $L=1$.) The bit decision sequence is shifted into the DB register and the A register as each bit decision is produced. The confidence bit sequence is shifted into the CB register and checked by the confidence test as each bit decision is produced. The state of the error detector immediately after the last bit decision and confidence bit is entered into the message decoder is as shown in Figure A-3. At this instant, the decoded address

$A_1 \dots A_{24}$ has been compared bit-by-bit with the expected address $\bar{A}_1 \dots \bar{A}_{24}$ to produce, by parallel Modulo 2 addition, the 24-bit error syndrome denoted $S_1 \dots S_{24}$.

Error Detection Criterion - If the correction disable bit has been set after the error syndrome has been calculated, and the error syndrome is not all zeroes, error correction is not attempted. If the correction disable bit is set and the error syndrome is all zeroes, the message is accepted. If the correction disable bit is not set, error correction will be attempted. The case in which the error syndrome is all zeroes is taken as an entirely correct bit decision sequence, but is treated as a correctable error pattern in which no bit decisions are actually corrected.

Error Location - The second stage of message decoding attempts to locate a correctable error burst pattern in the bit decision sequence generated by the message bit processor. This function requires the error syndrome and the confidence bit sequence.

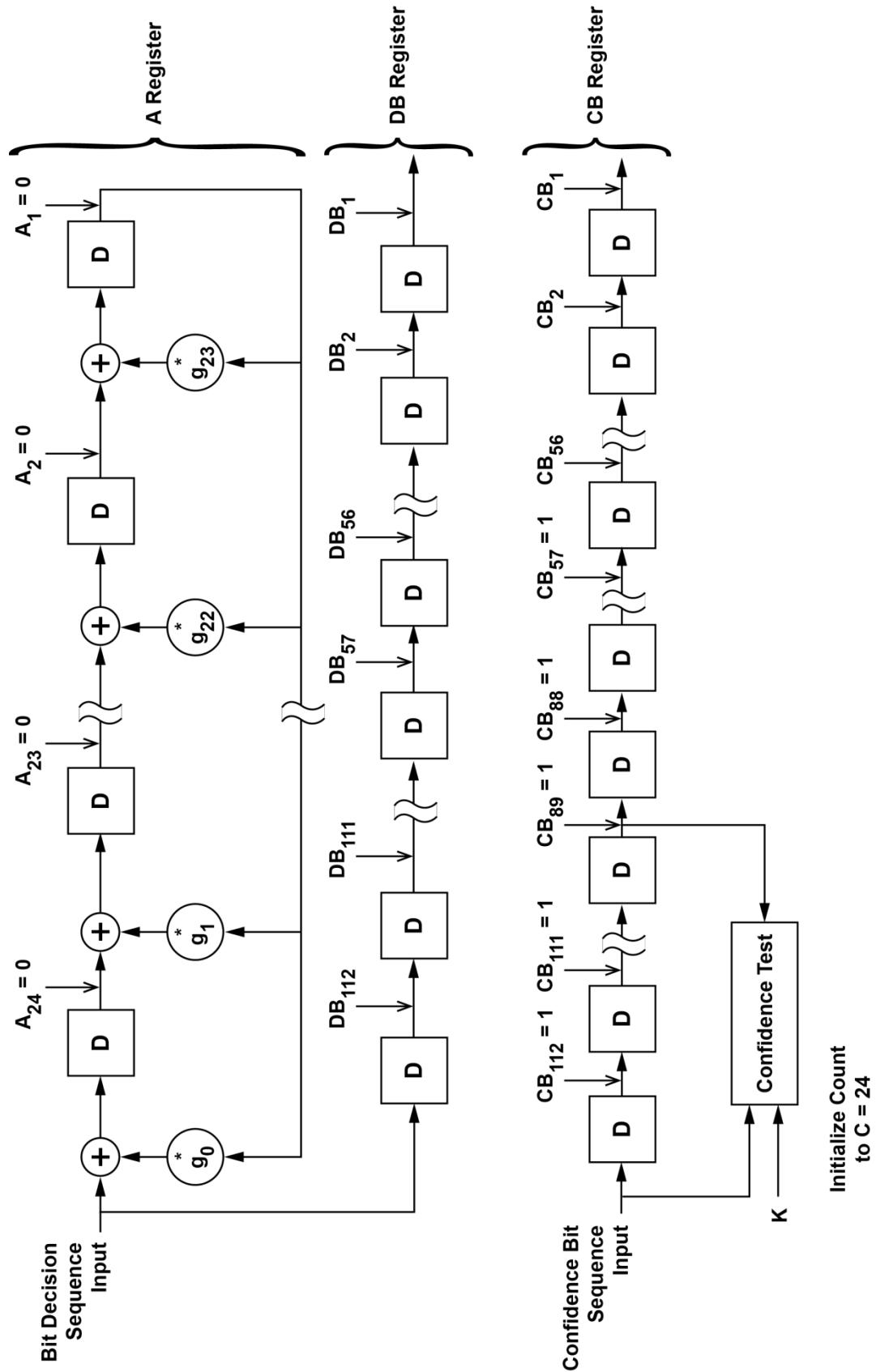


Figure A-2: Initial Conditions For Error Detection.

$$g_j = \begin{cases} 1, & j = 0, 3, 10, 12, 13, 14, \dots, 22, 23 \\ 0, & \text{otherwise} \end{cases}$$

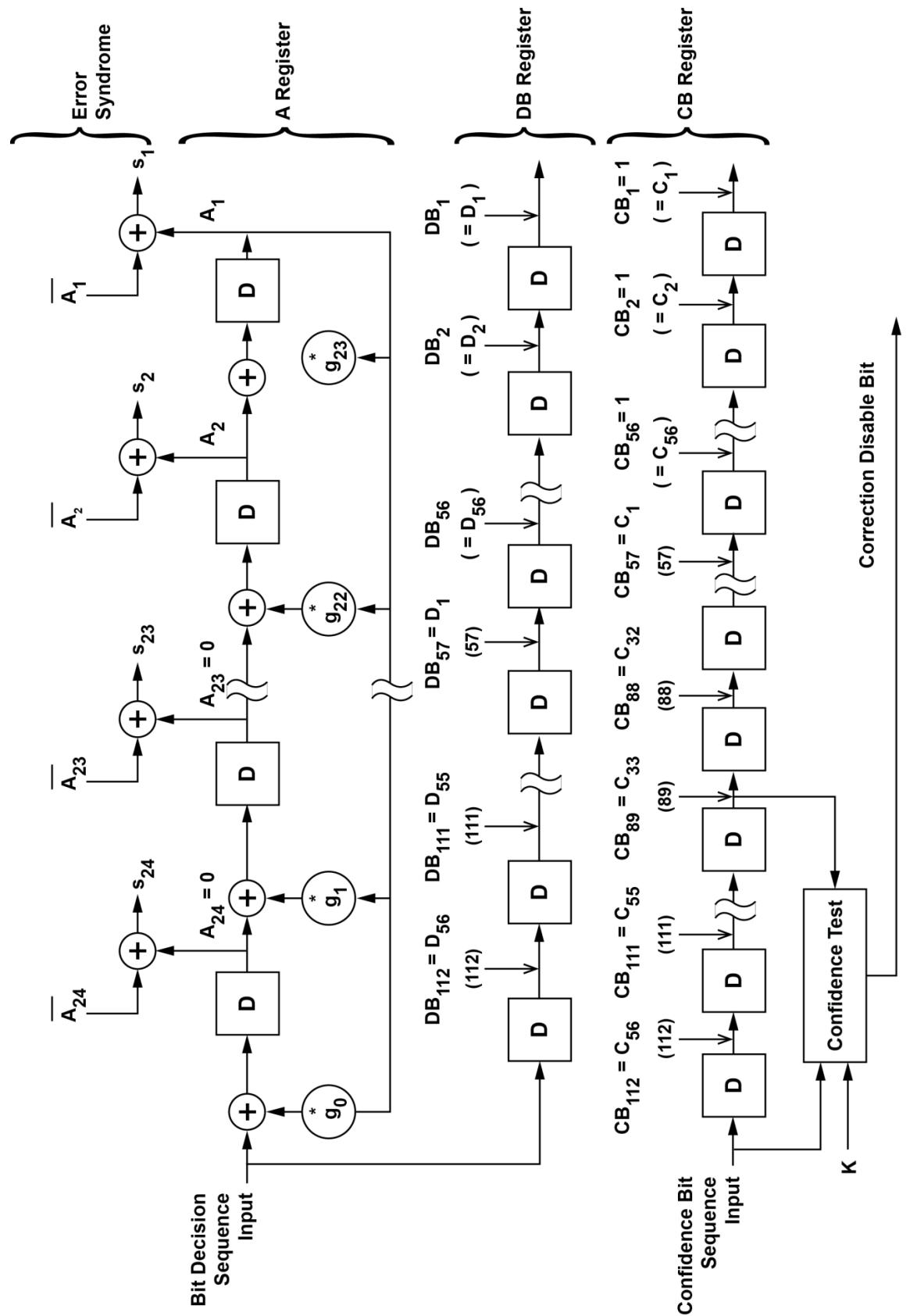


Figure A-3: Final Conditions Of Error Detection For 56 And 112 Bit Replies.

When the correction disable bit is false, and when all of the erroneous bit decisions are confined within a segment of the bit decision sequence spanning no more than 24 bits, and when none of the errors are associated with a high-confidence bit, the error pattern is a correctable error pattern. In every instance of a correctable error pattern, the error correction circuit will generate a decoded message.

The error pattern locating function attempts to locate erroneous bit decisions by performing logical operations on the error syndrome and confidence bit sequence. If a correctable error pattern is located, a bit signifying such an occurrence is set. When set, this bit allows the corrections indicated by the error pattern to be made in the decision sequence by the error pattern correction function. If this bit, denoted by m and called the correction enable bit, has not been set after a specified number of operations have been performed, the erroneous bit decisions cannot be located by the error location function.

Whenever the calculated address agrees with the expected address, the syndrome is zero and, with very high probability, every bit in the decision sequence is correct. As a result of the syndrome being zero, the error location function will immediately find a correctable error pattern consisting of 24 zeroes. The correction enable bit will be set even though the error pattern has no effect on the bit decision sequence.

The final value of the correction enable bit is used to indicate the presence or absence of a decoded message.

Initialization of Error Pattern Location Function - The error location function is initialized immediately after the error syndrome is generated by the error detector. The error location function requires three loading steps and is shown in Figure A-4.

The E register is loaded by a transfer of the error syndrome $S_1 \dots S_{24}$.

The M register is loaded by a transfer of the bit decision sequence $D_1 \dots D_L$.

The L register is loaded by a transfer of the confidence bit sequence $C_1 \dots C_L$, unless the correction disable bit is set, whereupon the loading of the L register is inhibited. The loading of the M and L registers is done in such a way as to permit digits to be shifted out of these registers in the reverse of the order in which they were received from the message bit processor.

Correctable Error Pattern Location - Once initialized, the error location function (E, L, and M registers) is shifted simultaneously, as rapidly as practical to free the function for the next reply. (The error location function should complete the operation while the error detector is being loaded with data from the next reply.) The number of shifts performed is determined by the reply length indicator bit L ($L=0$ implies 56 shifts, $L=1$ implies 112 shifts.) The input to the L register continues to enter ONE in this register while the registers are being shifted. If, after the specified number of shifts, no correctable error pattern has been located, the registers are deactivated.

The 24-bit E register generates a sequence of possible burst error patterns as it is being shifted, each of which would result in the observed error syndrome. After initialization, the E register contains a burst error pattern (which is the error syndrome itself) corresponding to the last 24 bits of the bit decision sequence.

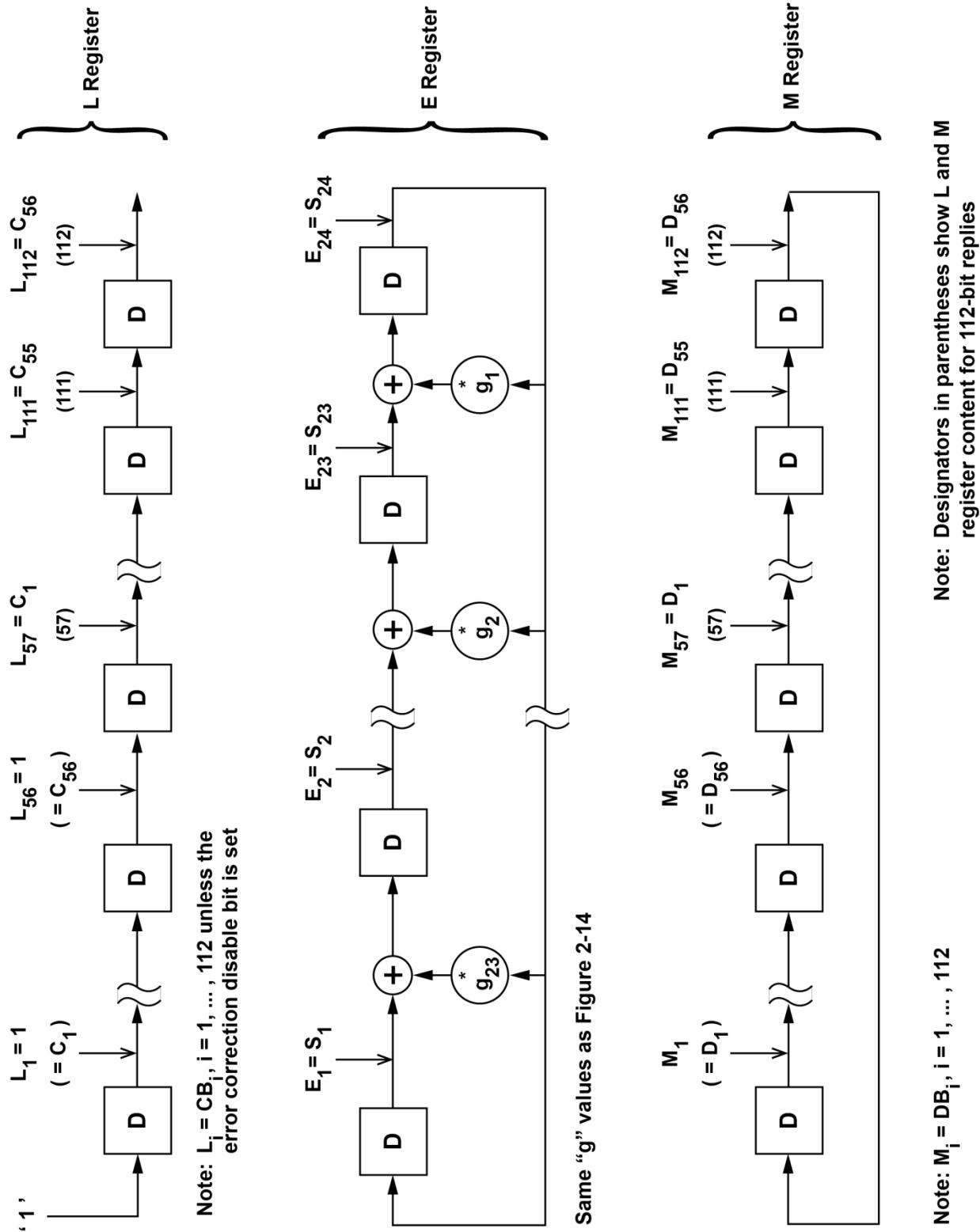


Figure A-4: Initialization Of Error Pattern Location Logic For 56 And 112 Bit Replies.

After j shifts, the E register contains a burst error pattern, which, if located in bits D_{l-j-23} through D_{l-j} , would result in the observed error syndrome. That is, every ONE in the E register corresponds to a bit decision that is presumed to be in error, while every ZERO in the E register corresponds to a bit decision that is presumed to be correct. The E and L registers are shifted together so that the possible burst error patterns may be compared with the proper 24-bit segment of the confidence bit sequence.

If a burst error pattern is found in which all error locations (ONES in the E register) align with low confidence bit positions (ZEROS in the L register), a correctable error pattern has been located and the correction enable bit m will be set to ONE (Figure A-5). The correction enable bit results in activation of the error pattern correction function. Note that the bit values in the L register that correspond to ZEROS in the E register have no effect on error pattern location.

Error Correction - Immediately after the correction enable bit has been set to ONE (before the next shift of the E, L and M registers), the error correction function is enabled to combine bit-by-bit the contents of the E register with the contents of the M register as shown in Figure A-6. The next 24 shifts of the E and M registers then result in the error pattern being shifted out of the E register (and replaced by ZEROS), causing those bit decisions that are combined (sum Modulo 2) with ONES in the error pattern to be changed. The error location function has no effect after the error pattern is shifted out of the E register, and so it is not necessary to disable this function. After the prescribed total number of shifts have been carried out, the final stage of the error correction function is as shown in Figure A-7.

A.4 Determination of Synchronous Garble Potential

The detection and determination of synchronous garble potential is monitored at the surveillance reply (Low Confidence Bit Measurement), track (Intruder Range Track Comparison), and beam (Mode C Aircraft Density Measurement) levels. The following algorithms describe an acceptable means for monitoring synchronous garble potential at the reply and track levels.

Low Confidence Bit Measurement

The Low Confidence Bit Measurement is added to the ATCRBS surveillance function prior to the formation of the reply buffer where redundant replies are merged. In the Low Confidence Bit Measurement, ATCRBS replies from the current and the previous scan are tested for the presence of low confidence bits.

A flowchart of the measurement algorithm is shown in Figure A-8, and assumes that replies are already arranged in increasing range order prior to the formation of the reply buffer. A list is created containing data for those replies that indicate low confidence bit settings in the altitude code. Each element in this list contains the following associated reply information:

reply range

antenna (top or bottom)

beam (front, back, right, left, or omni)

This list is retained for use during the current scan and the next scan.

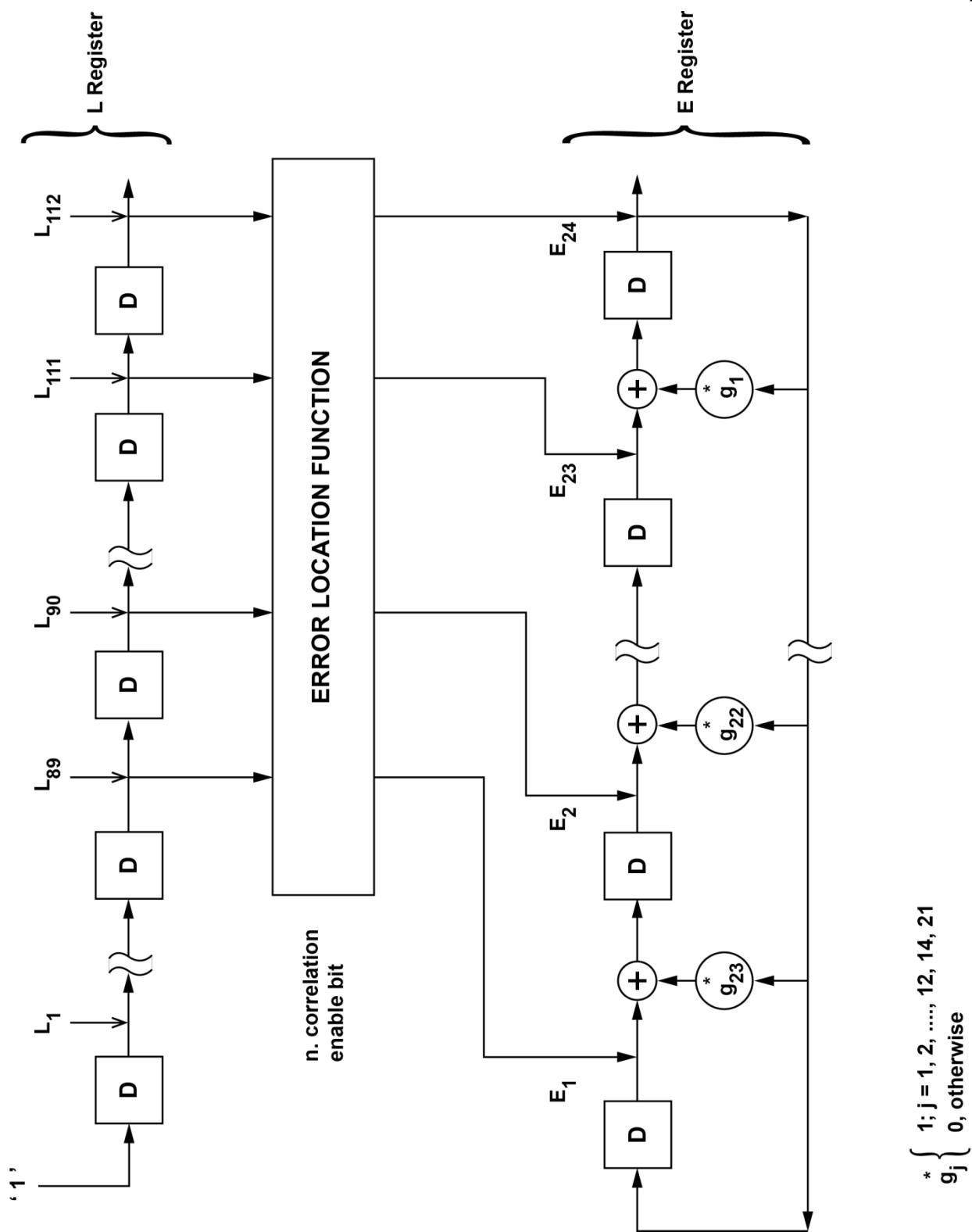


Figure A-5: Generation Of Correction Enable Bit.

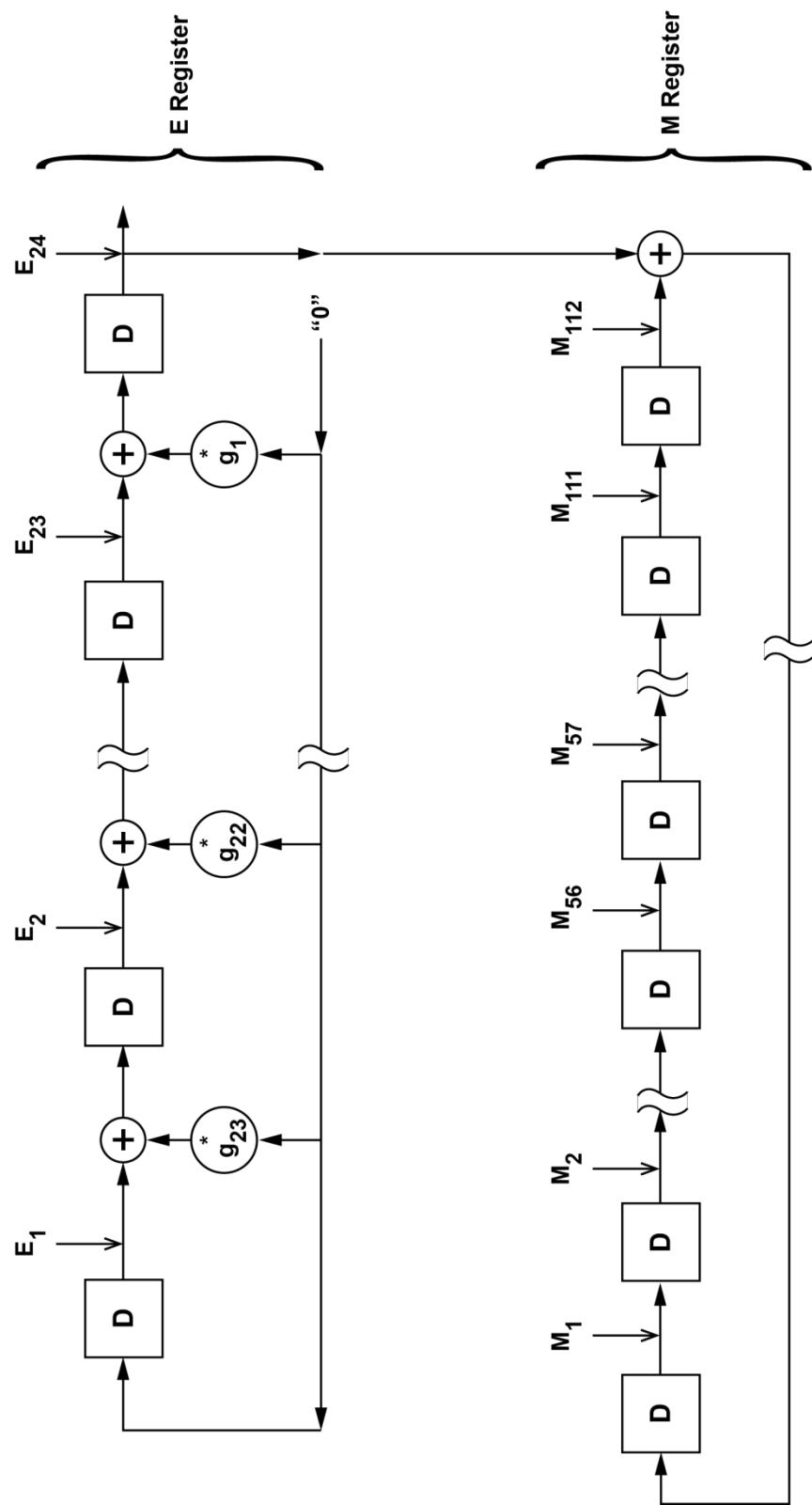


Figure A-6: Correction Of Decision Sequence.

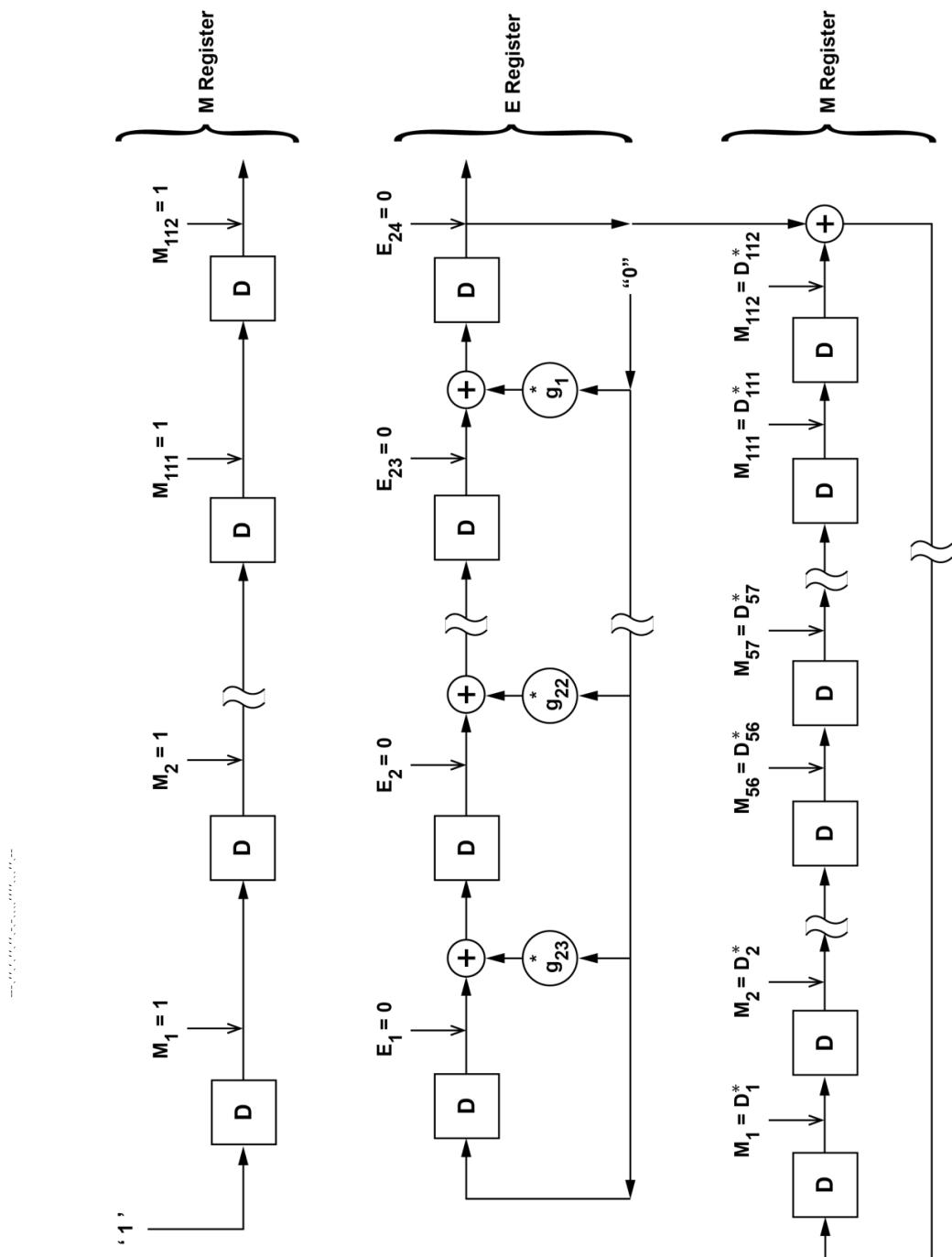


Figure A-7: Final State Of Error Correction Function.

For L=0 if M was set, For L=1 if M was set (in parentheses).

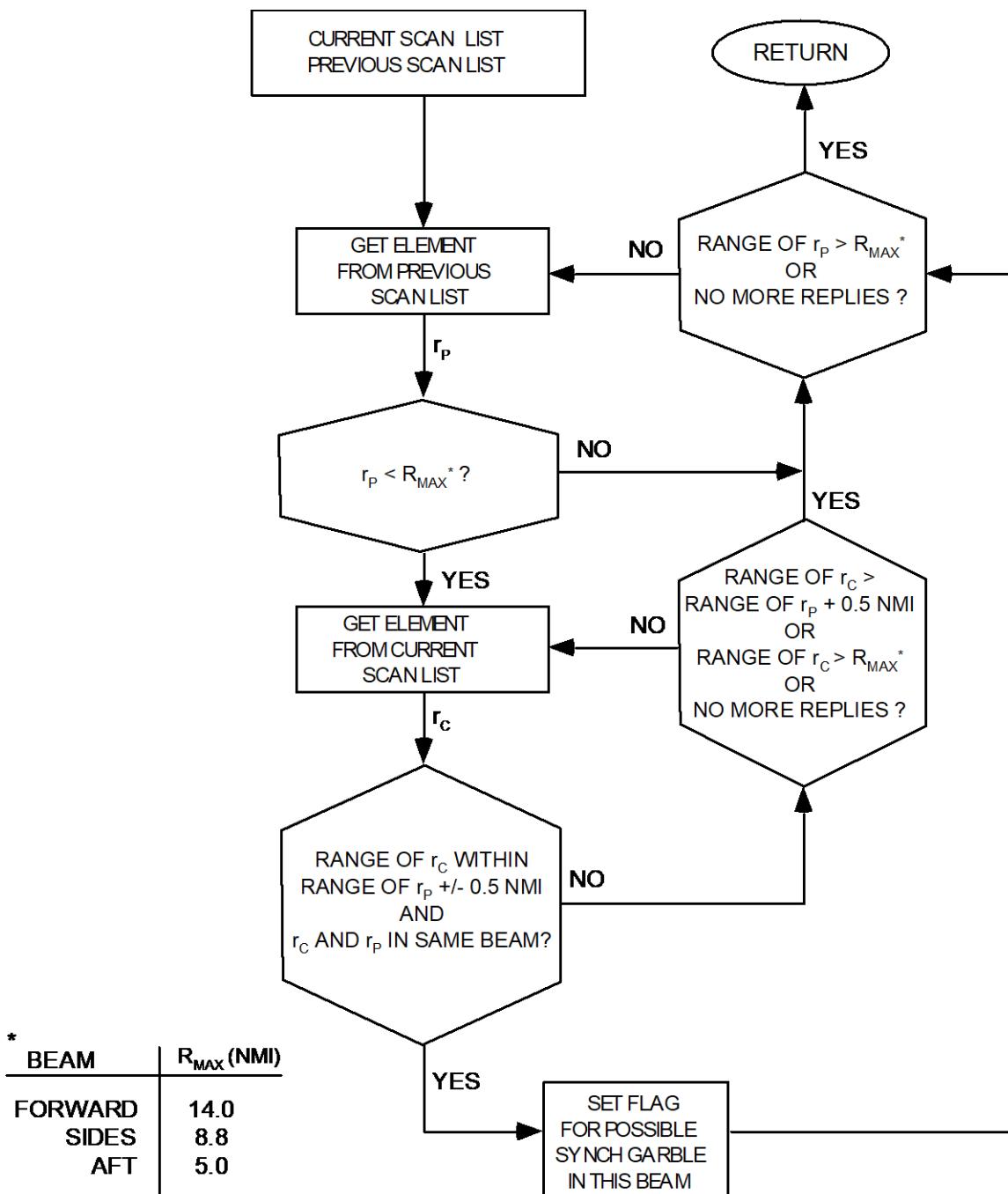


Figure A-8: Low Confidence Bit Measurement - Flow Diagram.

For each directional beam, a sequential comparison is made between each element in the previous scan list and the element from the current scan list. Starting with the first element in the previous scan list, r_p , the following item is checked:

r_p range is less than 14 NM in the forward beam, 8.8 NM in the side beams or 5.0 NM in the aft beam

If (a) is true, then the comparison starts for each element in the current scan list, r_c , with the following checks:

r_c is in same beam as r_p

r_c is within range of $r_p \pm 0.5$ NM

If both of the above are true then the flag, C_1 , is set indicating that the Low Confidence Bit Measurement has found a possible synchronous garble situation in this beam. If not true, the comparison continues with each remaining element in the current scan list until one of the following is true:

r_c range is greater than $r_p + 0.5$ NM.

r_c range is greater than 14 NM in the forward beam, 8.8 NM in the side beams or 5.0 NM in the aft beam

there are no more elements in the list

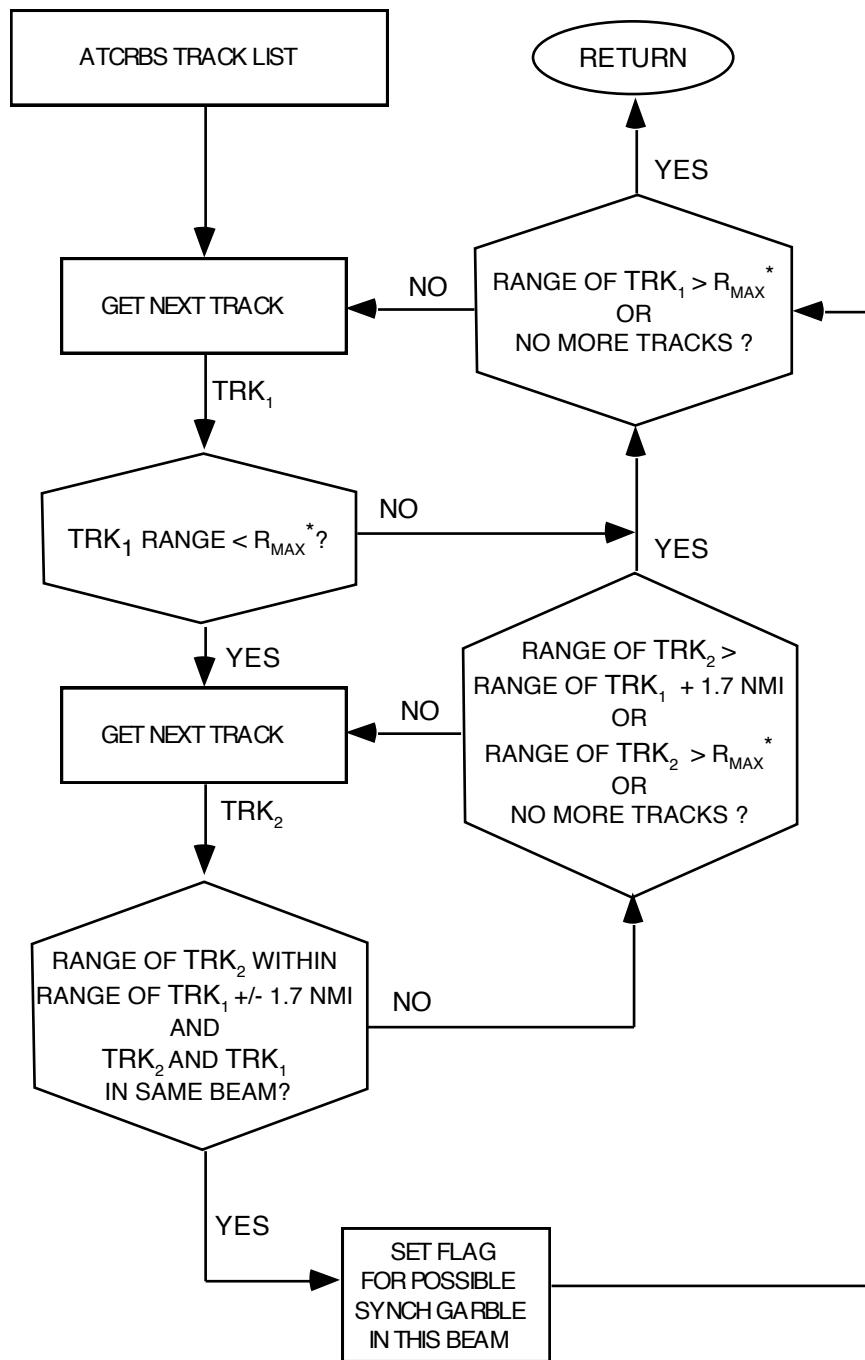
If any of the above are true, then the process moves to the next reply in the previous scan list, and the comparison is repeated until one of the following is true:

r_p range is greater than 14 NM in the forward beam, 8.8 NM in the side beams or 5.0 NM in the aft beam

there are no more elements in the list

Intruder Range Track Comparison

The Intruder Range Track Comparison is appended to the ATCRBS surveillance function. The Intruder Range Track Comparison is implemented in a similar fashion as the Low Confidence Bit measurement, except tracks within the same list are compared. Figure A-9 provides the process flow diagram and assumes that the tracks are arranged in increasing range order.



* R_{MAX} = CURRENT MAXIMUM RELIABLE SURVEILLANCE RANGE

Figure A-9: Intruder Range Track Comparison - Flow Diagram.

The range of the first track in the list (trk_1) is initially verified to be less than the current maximum reliable surveillance range in order to continue. If this is true, a comparison of the range of each of the following tracks (trk_2) is performed by checking:

trk_1 is in same beam as trk_2

trk_2 is within range of $\text{trk}_1 \pm 1.7$ NM.

If all of the above are true then the flag, T_r , is set indicating that the Intruder Range Track Comparison has found a possible synchronous garble situation in this beam. If not true, this comparison continues with each track in the list until one of the following is true:

trk_2 range is greater than $\text{trk}_1 + 1.7$ NM.

trk_2 range is greater than maximum reliable surveillance range

there are no more tracks in the list

If any of the above are true, then the process moves to the next track in the list, and the comparison is repeated until one of the following is true:

trk_1 range is greater than maximum reliable surveillance range

there are no more tracks in the list

A.5 Mode C Track Initiation

Note that a similar approach for the Mode C Track Initiation is documented in Appendix B to the ADD as a possible implementation.

Algorithm 209 AltCodeAgreement implements a modified approach for assessing correlation of reply code bits from the rules described below. The implementation differs in that a weighted sum of the number of bits in agreement is used as one component of a statistically derived correlation score which is compared to a threshold.

Algorithm 210 AltCodeEstimate implements the rules described below for determining the initial altitude track code estimate.

The following rules for assessing correlation of reply code bits and determining the initial altitude track code estimate for a target have been used successfully in experimental TCAS equipment (see Ref. F):

Three replies correlate only if:

All eight of their D, A and B code pulses agree, or

Seven of their D, A, and B code pulses agree and at least one of their C code pulses agree

A test for code agreement among the three replies is made individually for each of the reply code pulse positions. The initial test is based on the presence of code pulses alone. Agreement occurs for a given reply code pulse position if all three replies are detected with a ONE in that position or all three replies are detected with a ZERO in that position. The confidence associated with those pulse detections does not affect agreement. If agreement exists, the code estimate for that pulse position is set to the detected value.

When agreement among the three replies does not occur for a given reply code pulse position, the initial track pulse code estimate for that position is based on the values of the individual pulse codes and the confidence flags associated with those pulse codes in the three replies.

The confidence flag for a reply pulse position is set LOW whenever there exists another received reply (either real or phantom) that could have had a pulse within ± 0.121 microseconds of the same position. Otherwise, the confidence flag is set HIGH.

When agreement among the three replies fails for a given pulse position, the rules for estimating the initial track code for that position are based on the principle that LOW-confidence ONEs are suspect. The rules are as follows:

If in the most recent (third) reply the detected code for a given pulse position is HIGH confidence or a ZERO, the initial code estimate for that pulse position is set to the detected value. Otherwise,

if in the second reply the detected code for that pulse position is HIGH confidence or a ZERO, then the initial code estimate for that pulse position is set to the detected value in the second reply. Otherwise,

the initial code estimate for that pulse position is the same as the code detected in that pulse position in the first reply.

These rules can also be expressed as a truth table showing the agreement decision and the initial track code pulse estimate for a given pulse position. This is given in Table A-1 for all combinations of the three detected code bits and the three associated confidence bits.

Table A-1: Code Pulse Agreement

Determination of code pulse agreement and code pulse estimate for all possible combinations of code pulses and associated confidence bits. Reply 1 is the oldest and reply 3 is the most recent.

Reply 1		Reply 2		Reply 3		Code Agree	Code Est.
Code	Conf.	Code	Conf.	Code	Conf.		
1	-	1	-	1	-	Yes	1
0	-	0	-	0	-	Yes	0
-	-	-	-	1	High	No	1
-	-	-	-	0	High	No	0
-	-	-	-	0	Low	No	0
-	-	1	High	1	Low	No	1
-	-	0	High	1	Low	No	0
-	-	0	Low	1	Low	No	0
0	High	1	Low	1	Low	No	0
0	Low	1	Low	1	Low	No	0

- Defines a don't care value.

A.6 Mode C Track Maintenance

Note that an alternative approach for the Mode C Track Maintenance is documented in the Appendices to the ADD as a possible implementation.

Algorithm 201 AssociateModeCtoTarget in Appendix B to the ADD includes suggested algorithms to associate Mode C replies to existing tracks. The alternative approach uses the tracked state estimates produced by the STM trackers defined in the ADD. Algorithm 201 differs from the algorithm described below in that it uses a statistically derived score incorporating both the measurement error characteristics and the uncertainty in the tracked state estimate to determine which replies are used to update each existing track.

Appendix C of Volume II implements an alternative approach to determining if two tracks represent the same intruder. The algorithms found in Appendix C of Volume II differ from the approach below in that they are used to correlate and decorrelate tracks from multiple surveillance sources. In addition, the correlation processing utilizes a statistical approach utilizing the declared uncertainty of each tracked state estimate to determine if two tracks correlate with one another.

The following algorithms for surveillance of ATCRBS targets have been used successfully in experimental TCAS equipment (see Ref. F.).

The tracks are processed individually in increasing range order with input range precision of at least 1/128 NM and retained computational accuracy of at least 1/1024 NM. Range is estimated and predicted by a recursive (alpha-beta) tracker with *alpha* of 0.67 and *beta* of 0.25.

After each surveillance update a new range measurement is available for each target. Since the measurement includes errors, it must be smoothed based on previous measurements to obtain improved estimates of the current target position and velocity. The range and range rate estimation equations are as follows:

$$r_{est}(t) = r_{pred}(t) + \text{alpha} [r_{meas}(t) - r_{pred}(t)]$$

$$\dot{r}_{est}(t) = \dot{r}_{est}(t - T_p) + \left(\frac{\text{beta}}{T_p} \right) [r_{meas}(t) - r_{pred}(t)]$$

where T_p is the time difference between the current and previous measurements.

The gains, *alpha* and *beta*, determine the relative degree of reliance on current and previous measurements; gains of unity would place complete reliance on the current measurement and result in no smoothing.

The estimates obtained from the above equations are subsequently used to predict the range at the time of the next measurement as follows:

$$r_{pred}(t + T_n) = r_{est}(t) + [\dot{r}_{est}(t) \cdot T_n]$$

where T_n is the time difference between the next measurement and the current measurement.

The range correlation window is centered at the predicted range and has a half-window width as follows:

$$\left[\begin{array}{l} 760 \text{ ft if} \\ \text{coasted last} \\ \text{interval} \\ \\ 570 \text{ ft if} \\ \text{updated last} \\ \text{interval} \end{array} \right] + \left[\begin{array}{l} \text{if track is not established: 0} \\ \\ \text{if track is established:} \\ 2000 \text{ ft, if } r \geq 0.00 \text{ NM and } .LT. 0.17 \text{ NM} \\ 1000 \text{ ft, if } r \geq 0.17 \text{ NM and } .LT. 0.33 \text{ NM} \\ 600 \text{ ft, if } r \geq 0.33 \text{ NM and } .LT. 1.00 \text{ NM} \\ 240 \text{ ft, if } r \geq 1.00 \text{ NM and } .LT. 1.50 \text{ NM} \\ 0 \text{ ft, if } r \geq 1.50 \text{ NM} \end{array} \right]$$

If the track is above 10,000 ft, the term contained within the second pair of brackets is multiplied times four.

For the purposes of altitude correlation, altitude is estimated and predicted by an alpha-beta tracker with alpha of 0.28 and beta of 0.06. The tracker has retained computational accuracy of 100 ft divided by 16. The altitude prediction is rounded to the nearest 100 foot increment and converted to Gray code. The Gray codes of the predicted altitude ± 100 feet are also computed. (Note that the longer-term altitude predictions performed by the CAS logic require a more accurate altitude tracking procedure).

The reply(s) that lie in the range correlation window are tested for altitude correlation in increasing range order. The track is updated with the first reply that has exact agreement (in all bits) with any of the three

Gray codes computed above. If no reply matches, two additional Gray codes are computed and the process tried again. The two codes are the predicted altitude ± 200 feet.

The updating reply (if any) is eliminated from further consideration in updating other tracks, or in the track initiation process. If there is no updating reply, the range and altitude estimates are set equal to the corresponding predicted values. If this is the sixth consecutive surveillance update interval having no updating reply, the track is dropped. If there is an updating reply, and if the track is not identified as an image (§2.2.4.6.4.1.4), the track is flagged as established; that is, it is now available for use by the CAS logic. Once established, a track remains established until it is dropped, even if it subsequently satisfies the conditions for an image track (§2.2.4.6.4.1.3).

When all tracks have been processed, they are combined with the tracks that are newly initiated during the current scan, and then all the tracks are examined pairwise to determine if a given pair of tracks is likely to represent the same intruder. If:

the ranges differ by at most 0.082 NM

the range rates differ by at most 8.9 kt

either

the altitudes differ by at most 100 ft, or

the altitude rates differ by at most 10 ft/s and both tracks were initiated during the same scan,

only one of the tracks is retained, preference being given to the track showing the larger number of replies since initiation.

A.7 Multipath False Targets

Note that an alternative approach was used in the ADD as part of a possible implementation for identification of image tracks. Image tracks are identified as part of the track correlation processing in Appendix C to the ADD. Algorithm 233 CalculateImageRangeRate implements the image range rate calculation described below. This calculated range rate is used in the correlation processing where a statistical approach utilizing the declared uncertainty of each tracked state estimate is used to determine if a track correlates as an image to another existing track..

The following algorithms have been successfully used to identify image tracks.

If a track is identified as an image, it may be retained, but it cannot be flagged as established for use by the CAS logic. Those tracks that could have been formed by replies specularly reflected from the ground are referred to as image tracks. A track is identified as an image if there exists a track at shorter range (referred to as the real track) such that:

The difference between the real altitude and the image altitude is less than or equal to 200 ft for altitude-reporting targets, or both the image track and the real track are non-altitude-reporting, and

the difference between the measured image range rate and the calculated image range rate \dot{R}_I is less than or equal to 40 kt., where the calculated image range rate is either (for the single-reflection):

$$\dot{R}_I = \frac{1}{2} \left[\dot{R} + \frac{\sqrt{(2R_I - R)^2 - R^2 + (Z_O - Z)^2} (\dot{Z}_O + \dot{Z}) + R\dot{R} - (Z_O - Z)(\dot{Z}_O - \dot{Z})}{2R_I - R} \right]$$

or (for the double-reflection case):

$$\dot{R}_I = \frac{1}{R_I} \left[\sqrt{R_I^2 - R^2 + (Z_O - Z)^2} (\dot{Z}_O + \dot{Z}) + R\dot{R} - (Z_O - Z)(\dot{Z}_O - \dot{Z}) \right]$$

where:

R_I is the image range,
 R is the real range,
 Z is the real altitude for altitude-reporting targets or
 Z is set to own altitude for non-altitude-reporting targets, and
 Z_O is own altitude.

A.8 Mode S Squitter Processing

The following is one acceptable means of processing squitters and altitude replies to reduce unnecessary interrogations and minimize interference limiting and effects.

When a squitter with no parity error is first received, a running sum initialized at zero is associated with it. During each succeeding surveillance update interval the sum is decremented by 1 if no squitters or altitude replies with that address are received, and the sum is incremented by 16 for each reception within the surveillance update interval of either a squitter or an altitude reply. The process continues until the sum equals or exceeds 20. When the sum becomes less than or equal to -20, the address is removed from the system. When it equals or exceeds +20, the target is declared to be valid.

When a target has been declared to be valid, it is:

Interrogated for surveillance if its altitude is $\leq 10,000$ ft relative to own, or

Monitored for altitude using DF=0 or DF=4 replies or, in the absence of such replies, by interrogating once every 10 seconds if its altitude is $> 10,000$ ft relative to own.

When any of these conditions is satisfied, the running sum continues to be incremented and decremented even though its value may exceed 20.

Mode S squitter processing following an unsuccessful acquisition attempt is identical to the above but with the increment 16 replaced by 8. If a second failure to acquire occurs, the process is repeated with an increment of 4. After any subsequent failures, an increment of 2 is used.

A.9 Non-Altitude Reporting Aircraft

Note that an alternative approach for the Non-Altitude Reporting Aircraft (NAR) is documented in the ADD as a possible implementation. The alternative approach uses the STM trackers defined in the ADD to update tracks on both altitude reporting and NAR aircraft. In the case of NAR intruders, the vertical track is marked as invalid, and slant range is used in place of ground range. As described in Section A.6, Algorithm 201 AssociateModeCtoTarget is used to determine which Mode C replies are used to update each existing track (whether altitude reporting or Non-Altitude reporting). Unlike the approach described below, Non-Altitude reporting aircraft are accounted for in the statistical approach by applying zero weight to the altitude correlation score.

The following procedure for tracking altitude-unknown targets has been used successfully in experimental TCAS equipment (Ref. F.):

The altitudes of all replies having empty brackets are changed to a pseudo-altitude, such as 127,000 feet. These replies are used during track updating and initiation. When three of them from consecutive scans satisfy the criteria for a new track, a track is formed.

When a non-altitude-reporting target replies to more than one whisper-shout interrogation on a given scan, those replies are combined into a target report and discarded if they lie within 150 ft in range, 20 degrees in bearing, and their whisper-shout interrogations overlap in power.

Appendix A

A-22

Target reports and replies associated with non-altitude-reporting targets are used to update existing non-altitude-reporting target tracks. The range correlation window, centered on predicted track position, has a width (in units of 1/128 NM) of:

$$W = (7 - age) + 5 + \frac{up/down_coast_count}{2} + \frac{bias}{2}$$

but less than 15.

A newly initiated track has an age of 3. When the term $(7 - age)$ is negative, it is dropped from the formula. The variable *bias* is a measure of how well the track is following recent range measurements. The windows of adjacent tracks are not allowed to overlap. Regions of overlap are divided in half, and each half is allocated to the appropriate track.

The replies and/or reports that lie in a track's range window are examined to select the "best" one to use to update the track. If no reply or report is within 30 degrees of the predicted bearing then none is selected, and the track is coasted. If more than one reply or report, or both, lie in the window, and they have the same bearing within 15 degrees, the one having a range closest to the track range prediction is selected. If they do not have the same bearing within 15 degrees, the one having a range closest to the track range is selected.

The replies and/or reports that lie in the window are then discarded. That is, they are not eligible for further use by track update or by track initiation.

The variable *up/down_coast_count* is incremented by one for each coasted scan, and decremented by one for each updated scan.

The track updating equations are as follows. Let:

$r(t)$ = range measurement at time t

$$R_{meas}(t) = r(t) \cdot r(t) \text{ such that } r(t) = \sqrt{R_{meas}(t)}$$

Then, for straight line motion:

$$\dot{r}(t) = \dot{R}(t) / (2 \cdot r(t))$$

$R_{meas}(t)$ "measurements" are tracked with an alpha, beta, gamma filter.

$$R_{est}(t) = R_{pred}(t) + alpha [R_{meas}(t) - R_{pred}(t)]$$

$$\dot{R}_{est}(t) = \dot{R}_{pred}(t) + \left(\frac{beta}{T_p} \right) [R_{meas}(t) - R_{pred}(t)]$$

$$\ddot{R}_{est}(t) = \ddot{R}_{pred}(t) + \left(\frac{gamma}{T_p^2} \right) [R_{meas}(t) - R_{pred}(t)]$$

where T_p is the time difference between the current and previous measurements.

The gains, *alpha*, *beta* and *gamma* determine the relative degree of reliance on current and previous measurements: gains of unity would place complete reliance on the current measurements and result in no smoothing.

The gains are implicitly dependent on the age of the track, its history of updates and coasts, and how well the track followed the recent range measurements (as expressed in the *bias*):

$$\alpha = \alpha * 0.90 + 0.440 * 0.10$$

$$\begin{aligned} \text{beta} &= \text{beta} * 0.79 + 0.114 * 0.21 \\ \text{gamma} &= \text{gamma} * 0.70 + 0.012 * 0.30 \end{aligned}$$

After the above calculations, an additional one is made if the track was coasted on the last scan:

$$\text{alpha} = \text{alpha} * 0.750 + 0.250$$

$$\text{beta} = \text{beta} * 0.984 + 0.016$$

$$\text{gamma} = \text{gamma} * 0.998 + 0.002$$

Calculate the *bias* as:

$$\text{bias} = \frac{\text{bias} + [R_{\text{meas}}(t) - R_{\text{pred}}(t)]}{2}$$

If the *bias* is greater than 50 feet, and *alpha* is less than 0.58 then:

$$\text{alpha} = 0.580$$

$$\text{beta} = 0.220$$

$$\text{gamma} = 0.035$$

New tracks are initiated using replies and reports that did not lie in the range correlation windows of existing tracks. Three replies from consecutive scans are necessary to initiate a new track, and they must lie in an approximately straight line, and have bearing within 20 degrees of each other. The straight line criteria is that the third of three in a row must lie within 100 feet of a line through the first two.

A new track has the following initial values:

<i>age</i>	= 3
<i>up/down coast count</i>	= 0
<i>coasts</i>	= 0
<i>bias</i>	= 0
<i>alpha</i>	= 1.0067
<i>beta</i>	= 1.2355
<i>gamma</i>	= 0.7091

The estimates obtained from the above equations are subsequently used to predict the range at the time of the next measurement as follows:

$$R_{\text{pred}}(t + T_n) = R_{\text{est}}(t) + \dot{R}_{\text{est}}(t) \cdot T_n + \ddot{R}_{\text{est}}(t) \cdot \frac{T_n^2}{2}$$

$$\dot{R}_{\text{pred}}(t + T_n) = \dot{R}_{\text{est}}(t) + \ddot{R}_{\text{est}}(t) \cdot T_n$$

$$\ddot{R}_{\text{pred}}(t + T_n) = \ddot{R}_{\text{est}}(t)$$

where T_n is the time difference between the estimate and prediction.

$$r_{\text{pred}}(t + T_n) = \sqrt{R_{\text{pred}}(t + T_n)}$$

The initial track values are:

$$R_{est}(t_2) = R_{meas}(t_2)$$

$$\dot{R}_{est}(t_2) = \frac{3 \cdot R_{meas}(t_2) - 4 \cdot R_{meas}(t_1) + R_{meas}(t_0)}{2}$$

$$\ddot{R}_{est}(t_2) = R_{meas}(t_2) - 2 \cdot R_{meas}(t_1) + R_{meas}(t_0)$$

A.10 Bearing Estimation

Note that an alternative approach for the Bearing Estimation is documented in the ADD as a possible implementation. The STM Cartesian Tracker described in §2.2.2.1 of the ADD can be used for bearing estimation on active surveillance intruders in a similar manner as the Cartesian alpha-beta tracker described below is used for bearing estimation. The STM Cartesian Tracker performs state estimation using an Unscented Kalman Filter instead of the alpha-beta approach described below.

The following Cartesian alpha-beta estimation algorithm has been used successfully to develop smoothed target bearing estimates.

Initiation. The bearing measurement (if available) and range measurement of each reply are used to form x and y position estimates:

$$x = r \cos(\text{bearing})$$

$$y = r \sin(\text{bearing})$$

These measurements are used to form a least-squares estimate of the x and y positions and velocities. At least one reply from a bearing measurement antenna is required for initiation.

Prediction. A predicted x,y position for the next scan is formed by adding the product of the last-scan velocity estimate and the time since the last scan to the last-scan position estimate. The predicted x and y are converted to bearing using the tangent function in the appropriate quadrant.

Update. If a bearing measurement having at least one ungarbled code pulse is available, it is combined with the range measurement to form the x and y estimates. The update is made using a standard alpha-beta tracker in both x and y . The gains are the same in the x and y coordinates. The least-squares gains vary during the first 15 scans following initiation of the range-altitude track and then are held constant, as shown below. The gain values depend only on the age of the track, not the history of availability of reasonable estimates.

<u>Range-altitude</u> <u>Track Age (sec)</u>	<u>alpha</u>	<u>beta</u>
1	1.000	3.000
2	1.000	1.000
3	0.833	0.500
4	0.700	0.300
5	0.600	0.200
6	0.524	0.143
7	0.464	0.107
8	0.416	0.083
9	0.378	0.067
10	0.345	0.054
11	0.318	0.045
12	0.295	0.038
13	0.275	0.033
14	0.257	0.029
15 or more	0.242	0.025

If a valid bearing measurement is not available, the x and y estimates are set equal to the corresponding predicted values.

The position update equation in x is:

$$x_{est}(t) = x_{pred}(t) + alpha [x_{meas}(t) - x_{pred}(t)]$$

The velocity update equation in x is:

$$\dot{x}_{est}(t) = \dot{x}_{est}(t - T) + \left(\frac{beta}{T} \right) [x_{meas}(t) - x_{pred}(t)]$$

where T is the time difference between the current and previous measurements.

The y equations are analogous.

A.11 Mode S Tracker Using a Five-Second Update Rate

Note that an alternative approach for the Mode S tracking using a Five-second update rate is documented in the ADD as a possible implementation. The STM Active Surveillance described in §2.2 of the ADD has been used successfully for tracking Mode S intruders with a five second update rate. The Active Surveillance utilizes the same trackers for intruders under both normal (1-second rate) and reduced (5-second rate) surveillance.

The following describes one possible implementation for providing reliable tracked range and range rate estimates each scan during periods of reduced update rate tracking.

Track initiation and acquisition follows the suggested method based on the 1 second update interval. Upon receipt of a reply to an acquisition interrogation, the target is placed into the track state. The track is extended by interrogating at a 1 second update interval until an established range rate estimate can be made. To allow the range tracking algorithm time to provide valid rate estimates, the range rate is considered established 4 scans after acquisition as long as a minimum of 2 successful updates occurred in those 4 scans. Once established, it remains until the track is dropped. The target threat level, for use in determining the update interval, is not evaluated until the range rate is established.

Each active interrogation scan, all Mode S track parameters are updated by recursive alpha-beta filters for smoothing and predicting range and range rate based on the measured data and the time difference between the current and last interrogation. The form for providing smoothed range and range rate estimates and predicting future states is given by:

$$r_{est}(t) = r_{pred}(t) + 0.67 \cdot [r_{meas}(t) - r_{pred}(t)]$$

$$\dot{r}_{est}(t) = \dot{r}_{est}(t - T_p) + \frac{0.25}{T_p} [r_{meas}(t) - r_{pred}(t)]$$

$$r_{pred}(t + T_n) = r_{est}(t) + [\dot{r}_{est}(t) \cdot T_n]$$

where, t is the time of the estimate, T_p is the time since the previous measurement, T_n is the expected time until the next measurement, $r_{est}(t)$ is the smoothed range estimate, $r_{pred}(t)$ is the predicted range value, $r_{meas}(t)$ is the measured range, and $\dot{r}_{est}(t)$ is the estimated range rate.

For each of the nominal 5 second scan periods between active scans, the smoothed and estimated range and range rate are calculated using the above equations. The resultant rate estimates are then used to provide predicted position estimates during those scans where no active interrogation takes place.

A.12 Transition from On-The-Ground to Airborne Interference Limiting

The following describes one possible implementation of a procedure that enables ACAS X to quickly recover its normal airborne surveillance range capability following a transition from an on-the-ground status to an airborne status.

Whenever ACAS X is operating on the ground and is using three times the value of NTA in the interference limiting function to determine the Mode C and Mode S interrogation power levels that it transmits each surveillance update interval, ACAS X also simultaneously computes the Mode C and Mode S power levels that it could transmit using the direct value of NTA and an alpha value of one. Within the next surveillance update interval following indication of a transition from an on-the-ground status to an airborne status, ACAS X replaces the previous Mode C and Mode S transmitted power levels computed using three times NTA and an alpha value based on NTA and on the measured ACAS X distribution within 6 NM with the most recent power levels computed using NTA and an alpha of one. On successive surveillance update intervals, ACAS X performs interference limiting according to §2.2.3.6.1 and §2.2.3.6.2 as long as it remains airborne.

APPENDIX B ACAS X RF COMMUNICATION PATHS

B.1 Introduction

RF communication paths used by ACAS X are shown in Figure B-1. A brief overview of each path is given in sections 2 and 3 below. This information is presented as background material in order to provide clarification for the coordination and communication requirements contained within this document.

B.2 Air-to-Air Communication

Air-to-air communication takes place using Mode S uplink formats UF=0, 16 and Mode S downlink formats DF=0, 16. The short (56-bit) special surveillance interrogations and replies (UF, DF=0) are used for air-to-air surveillance. The long (112-bit) special surveillance interrogations and replies (UF, DF=16) are used for air-to-air coordination and communication. DF=11 replies are unique in that they are involved in both air-to-air and ground-to-air surveillance. (See §2.4 and §3.4 below.)

B.2.1 Coordination

ACAS X coordination is accomplished via air-air transmissions using UF, DF=16 interrogations and replies. An ACAS X-equipped aircraft will initiate a coordination interrogation/reply sequence with another Active CAS-equipped aircraft once per second as long as the initiating aircraft perceives the other aircraft to be a threat. Interrogations are sent by the ACAS X processor in one aircraft and received by the Mode S transponder in the other aircraft. The receiving transponder then passes the coordination information on to its associated Active CAS processor. UF=16 interrogations used for coordination contain a Resolution Message in the 56-bit message field; within this field is a vertical RA complement (VRC) which is used to restrict the choice of vertical sense by the receiving aircraft. Coordination information in replies (e.g., ARA, RAC) is not used; a coordination reply is simply a technical acknowledgment which informs the initiating ACAS X that its interrogation was received by the other aircraft's Mode S transponder.

B.2.2 TCAS Broadcast Interrogation Messages

UF=16 interrogations are also used to transmit TCAS Broadcast Interrogation Messages. In this case, the 56-bit message field contains the address of the initiating ACAS X. These Broadcast interrogations allow an ACAS X-equipped aircraft to announce its presence to other Active CAS-equipped aircraft in the vicinity. These Broadcast interrogations are sent every 8 to 10 seconds by an ACAS X processor and received by the Mode S transponder of every Active CAS-equipped aircraft within receiving range. The receiving transponder then passes the information on to its associated Active CAS processor, where the information is used for interference limiting purposes. The receiving aircraft's transponder does not reply to a TCAS Broadcast interrogation.

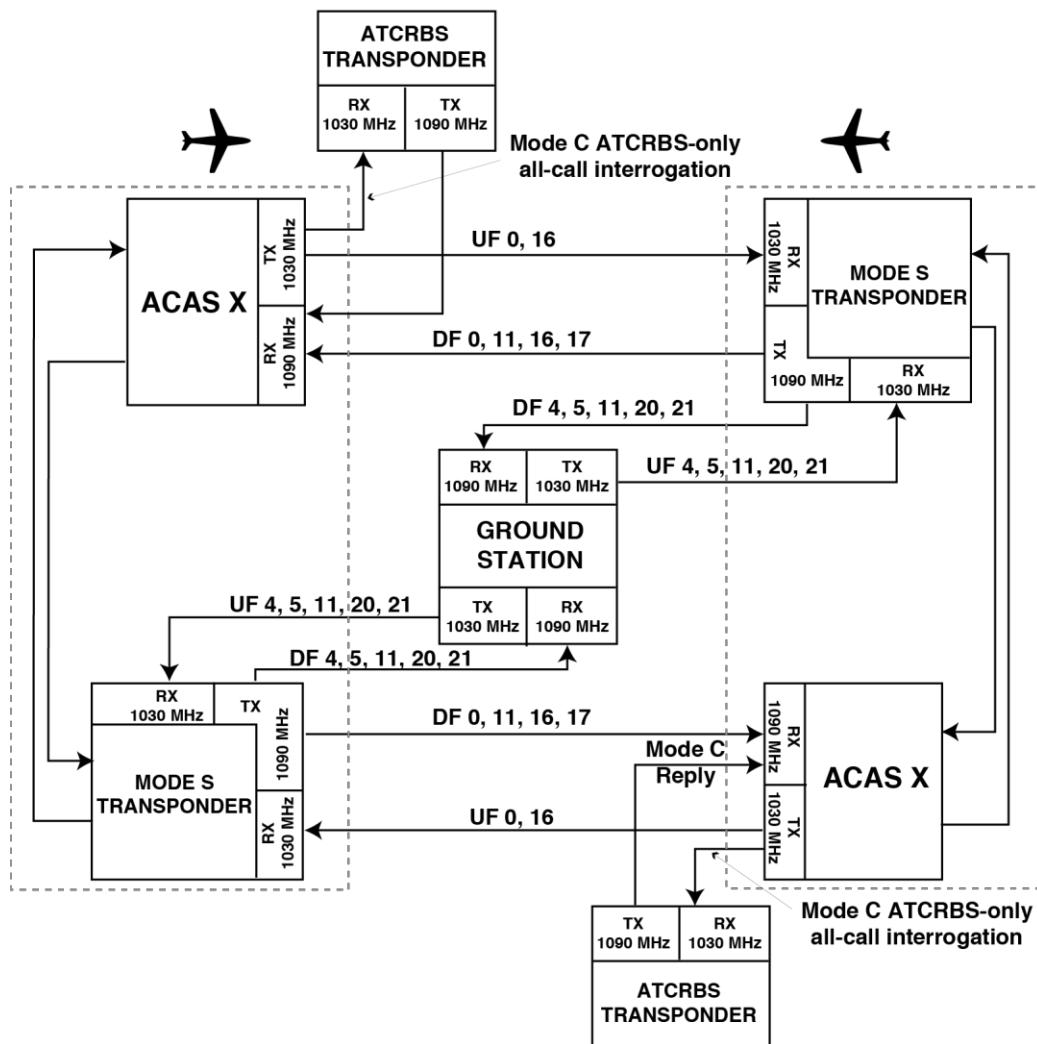


Figure B-1: ACAS X Communications

Note: This diagram shows a separated ACAS X processor and Mode S transponder. Current U.S. TCAS II air carrier implementations use a separated TCAS processor and Mode S transponder connected by two 100 kHz ARINC 429 buses. For ACAS X a manufacturer could choose some other option, e.g., housing the two functions in the same physical unit and communicating via shared memory or using a connection other than ARINC 429.

B.2.3 RA Broadcast Interrogation Messages

UF=16 interrogations are also used to transmit RA Broadcast Interrogation Messages. These interrogations are transmitted every 1 second by an ACAS X processor for the period that an RA is active. This allows RA activity to be monitored in areas where Mode S ground station surveillance coverage does not exist by using special RA broadcast signal receivers on the ground. RA Broadcast Interrogations are defined as uplink transmissions but are normally destined for ground equipment. In the RA Broadcast Interrogation, the 56-bit message field of the 112 bit long message describes the most recent RA that existed during the preceding 1 second period. The message field also includes the most recent Mode A and Mode C codes transmitted by the associated Mode S transponder.

B.2.4 Surveillance

Mode S "squitters" (DF=11 replies) are emitted automatically nominally once per second by a Mode S transponder. A squitter contains the aircraft's ICAO 24-bit Aircraft Address "in the clear" (i.e., parity does not overlay the address) and allows a receiving Active CAS-equipped aircraft to acquire the ICAO 24-bit Aircraft Address for surveillance initiation.

Following receipt of squitters, an Active CAS-equipped aircraft will use surveillance interrogations (UF=0) for acquisition and then tracking of a Mode S-equipped aircraft. A responding aircraft's Mode S transponder will reply with DF=0 replies, containing vertical status (VS), sensitivity level (SL), capability (RI), and altitude information. ACAS X periodically reports its sensitivity level and operational capability to the transponder for use in DF=0 replies.

DF=17 replies are identical to DF=11 replies but with an added 56-bit message field that can be used to convey various types of information (e.g., the aircraft's GNSS position). The DF=17 format is squittered automatically by some Mode S transponders.

For ACAS X surveillance of ATCRBS-equipped aircraft, ACAS X transmits Mode C ATCRBS-only all call interrogations [P1 and P3 spaced 21 microseconds apart to elicit a Mode C (altitude) reply, plus a short (0.8 microsecond) P4 pulse to prevent Mode S transponders from replying.] The "whisper-shout" technique (transmitting repeated Mode C interrogations at different power levels) is used to minimize synchronous garble.

B.3 Air-to-Ground, Ground-to-Air Communication

Communication between ACAS X and a Mode S ground sensor takes place using ground-transmitted UF=4, 5, 20, 21 interrogations and the aircraft's transponder-transmitted DF=4, 5, 20, 21 replies. The main use for all of these interrogations and replies is surveillance, and for this the short formats are sufficient (UF, DF=4, 5). When message bits are to be transmitted either in the uplink or downlink, the long formats are used (UF, DF=20, 21). UF=11 interrogations and DF=11 replies are also used in ground-based surveillance (§2.4).

B.3.1 Sensitivity Level Command Messages (Ground-to-Air)

A Mode S ground sensor can use the 56-bit message field of a UF=20 or 21 interrogation to transmit a Sensitivity Level Command Message to an ACAS X-equipped aircraft. The receiving aircraft's transponder replies with a corresponding DF=20 or 21 reply and passes the message on to its associated ACAS X processor.

ACAS X will not use the Sensitivity Level information that it receives through a Sensitivity Level Command Message. The reply will still be sent to be compatible with TCAS and to avoid a change for the ground station and transponder.

B.3.2 RA Report (Air-to-Ground)

Whenever ACAS X indicates to the transponder that it has an Active RA (ARA), the transponder sets a bit in its DF=4, 5, 20, 21 replies to indicate the availability of this information. A Mode S ground sensor can then request this information when sending a UF=4, 5, 20, 21 interrogation to the aircraft. When so

requested, the transponder will send the RA information in the 56-bit message field of the corresponding DF=20 or 21 reply. Possible uses for the downlinked information are display to air traffic controllers and data collection.

B.3.3 Data Link Capability Report (Air-to-Ground)

When requested by a Mode S ground sensor via a UF=4, 5, 20, 21 interrogation, the transponder will report its capability, including ACAS X capability, in the 56-bit message field of the corresponding DF=20 or 21 reply. Mode S ground sensors use the Data Link Capability Report to determine what ground-air-ground interactions an aircraft can support.

B.3.4 Surveillance

Mode S ground sensors typically obtain knowledge of a Mode S-equipped aircraft in their coverage area when the aircraft responds with a DF=11 reply to the sensor's Mode S-only all call (UF=11) interrogation. The aircraft thereafter is interrogated on a regular basis with discrete UF=4, 5 (or UF=20, 21) interrogations.

APPENDIX C DEGRADED SURVEILLANCE

C.1 Introduction

This appendix provides evidence of the robustness of the STM to off-nominal inputs – also referred to as “degraded surveillance”. The term “degraded” applies to any input where the update rate is less than nominal or when the observation input differs from the expected. For example, when track angle is provided rather than true heading, or when a lower than normal ADS-B quality is provided by an intruder. This appendix covers a large range of such degraded inputs.

C.2 Analysis Overview

For any allowed surveillance input, ACAS X has the objective of matching or improving upon TCAS II performance. The safety and operational suitability of ACAS X under nominal inputs has been extensively verified through simulation. The CSIM simulation framework developed for this end-to-end assessment contains models of the sensor systems (ATCRBS, Mode S, ownship, ADS-B, etc.) that generate observation values conforming to each type’s modalities. The safety and operational suitability testing supporting ACAS X was primarily conducted using these sensor models operating in nominal modes. To support this analysis, the CSIM framework has been modified to support the production of degraded surveillance inputs.

For each of the possible degraded inputs, a simulation of an ACAS X-equipped aircraft encountering an intruding aircraft over thousands of encounter geometries was run and the resulting Probability of a Near Mid-Air Collision (PNMAC) and alert rate were recorded. PNMAC is a metric used to assess the safety of the system. An NMAC is defined as the event of two aircraft coming within 100 ft vertically and 500 ft horizontally of each other. PNMAC, unlike the binary NMAC definition, assigns a probability based on the vertical separation of two aircraft; the horizontal separation criteria remains unchanged. The probability depends on the altitude of the aircraft and accounts for uncertainty in the altimetry systems. Alert rate, the percentage of encounters in which ACAS X issued an alert, is used as a measure of the operational suitability of the system.

The encounter geometries were provided by two encounters sets: the Lincoln Laboratory Correlated Encounter Model (LLCEM) was used to assess safety; the TCAS RA Monitoring System (TRAMS) model, operational suitability. LLCEM contains simulated trajectories of two aircraft that come within close proximity of one another. The model generates tracks using parameters, such as aircraft speeds, altitudes, and turn rates, that were distilled from recorded data in the NAS. TRAMS data is a collection of all aircraft tracks in the NAS that led to an aircraft equipped with TCAS II to issue an RA.

When testing each degraded input, the intruding aircraft was configured in such a way as to most stress the system. Of the intruder types and configurations that ACAS X protects against, an intruder equipped with a Mode C transponder provides the least amount of state information to ACAS X and so has been used in this analysis – unless the degraded mode involves ADS-B, in which case the intruder carried a Mode S transponder with hybrid surveillance.

To understand the impact of the degraded modes, the performance of ACAS X with each of the degraded inputs is compared against two baseline scenarios. The first baseline scenario assumes all inputs being fed to ACAS X conform to the nominal type and update rate while the second baseline assumes the aircraft is instead equipped with TCAS and the inputs are nominal.

C.3 Analysis of Degraded Surveillance Inputs

In this section, a summary of each of the degraded inputs is listed and the degraded modes are evaluated using the metrics described in Section 2.

C.3.1 ADS-B

ACAS X accepts measurements from the Automatic Dependent Surveillance – Broadcast (ADS-B) system and, using the information provided, will track potential threats and provide guidance to the pilot on how to avoid them. Both the availability and quality of ADS-B reports can vary in time or according to aircraft installation, so both are considered in this analysis.

C.3.1.1 Update Rate

C.3.1.1.1 Gating

Aircraft equipped with 1090ES ADS-B out broadcast position and velocity state vector reports each at a nominal rate of 2 Hz. ACAS X can and will process all ADS-B updates as they are made available to the system. The input requirement, however, stipulates that only the most recent report of each type need be provided each 1 Hz STM update cycle.

To test this degraded mode, only the most recent ADS-B report of each type was provided to the STM in each ACAS X update cycle; 1 contains the simulation results. There is no significant effect of gating on ACAS X performance.

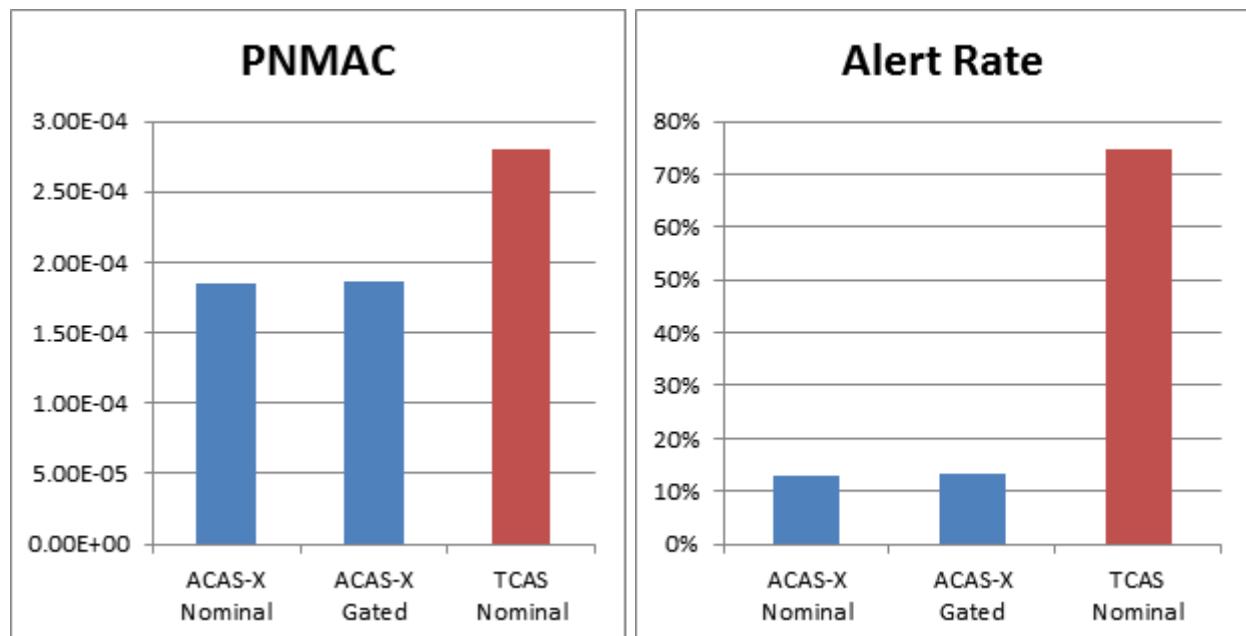


Figure C-1: Gating

C.3.1.1.2 Environment

In air traffic congested environments, ADS-B may, in accordance with the Minimum Aviation System Performance Standards (MASPS) for ADS-B, update the state vector reports less frequently than the nominal rate of 2 Hz – once every 3 and 5 seconds at 95% probability when two ADS-B equipped aircraft are within 3 and 10 nm, respectively. When ACAS X does not receive a state vector position or velocity report from an intruder in an update cycle, the corresponding passive track coasts and can eventually reset or drop if coasting for too long a duration. If the track should coast out, ACAS X would then revert to tracking with active surveillance.

To test the system performance when experiencing a reduced update rate, position and velocity reports were assigned a probability of reception such that at least one report of each type would be received with 95% probability in the prescribed update interval. The update interval was held constant for the duration of the simulated encounter; the constant interval differs from the ranged based minimum update interval defined in the ADS-B MASPS, but allows an assessment of the impact of the update interval on system performance. Figure C-2 contains the simulation results. ACAS X is robust against update delays in the passive track.

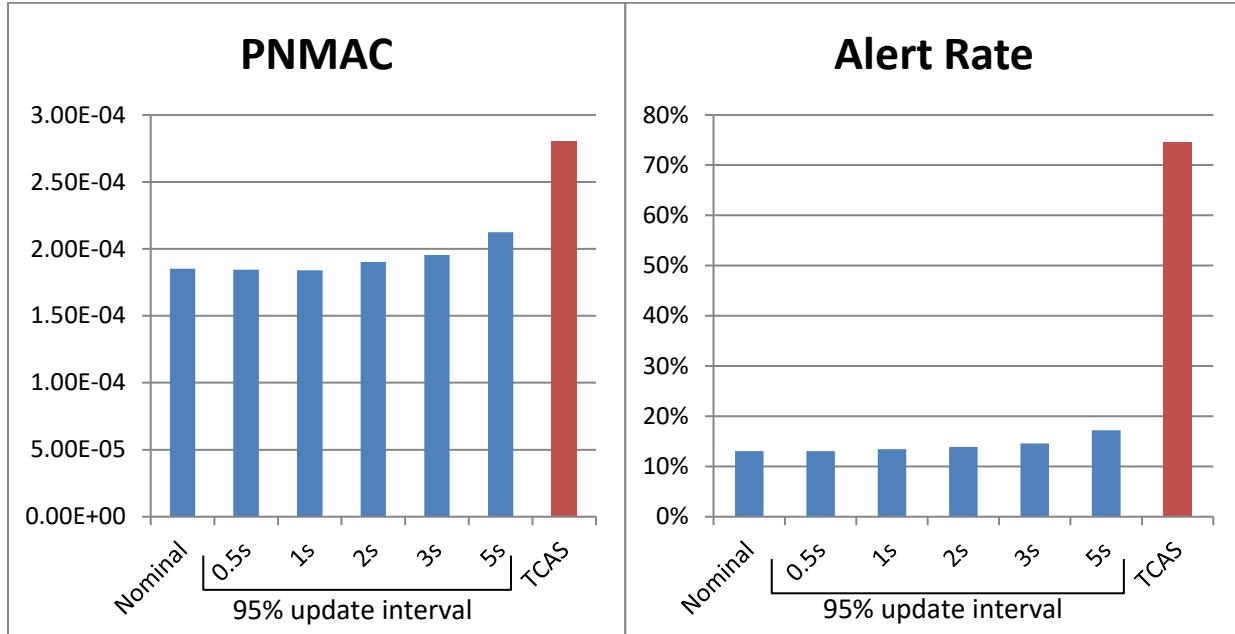


Figure C-2: Reduced Update Interval For ADS-B Report

C.3.1.2 Quality

Aircraft equipped with ADS-B out report the Navigation Accuracy Category for both position (NACp) and velocity (NACv). These parameters define the uncertainty in the reported position and velocity; their meaning and permissible values for ACAS X alerting are listed below.

C.3.1.2.1 NACp

The value of NACp represents the estimated position uncertainty (EPU). The EPU is a 95% accuracy bound on horizontal position and is defined as the radius of a circle, centered on the reported position, such that the probability of the actual position lying outside the circle is 0.05. NACp can range from 1, the largest EPU of 10 NM, to 11, the smallest EPU of 3 m or less. ACAS X will passively track intruders regardless of NACp, but will alert only on those intruders that are reporting their position with a NACp value of 7, EPU of 0.1 NM, or higher.

To test the range of NACp values, the intruding aircraft was set with a single NACp value for the duration of the simulated encounter and the reported position met the minimum requirements for that value; Figure C-3 contains the simulation results. ACAS X handles all NACp values with comparable performance.

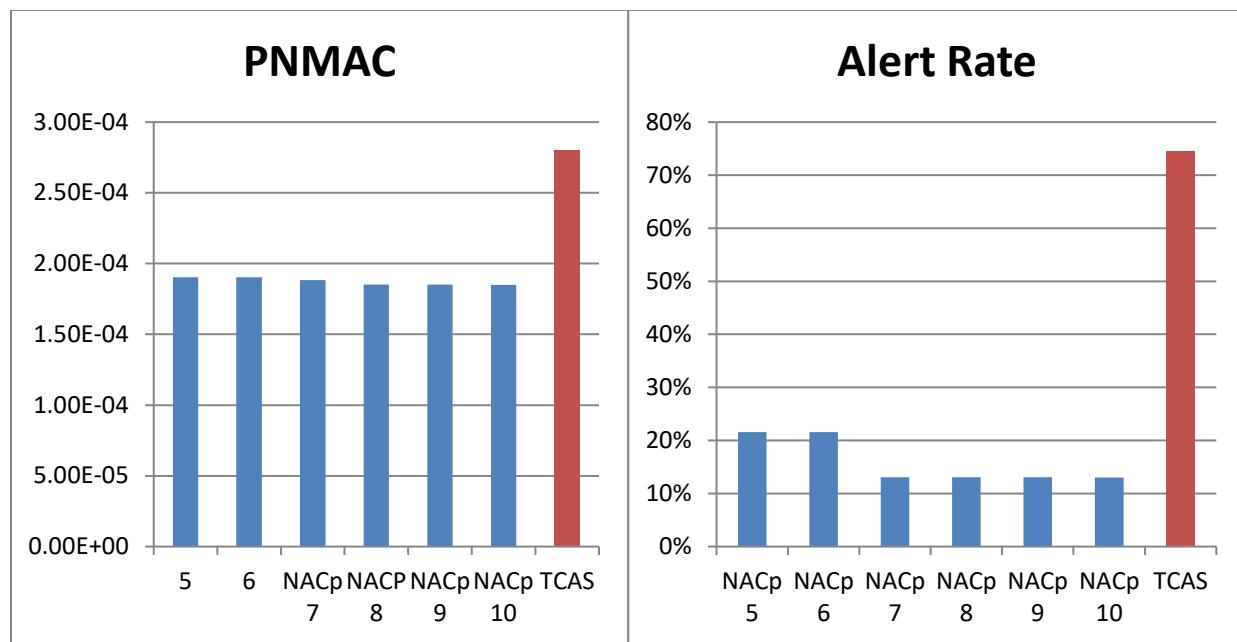


Figure C-3: NACp

C.3.1.2.2 NACv

The value of NACv represents the 95% accuracy figures of merit for horizontal and vertical velocity. NACv can range from 1, the largest horizontal velocity error of 10 m/s, to 4, the smallest horizontal velocity error of 0.3 m/s or less. ACAS X will passively track and alert on intruders regardless of NACv.

To test the range of NACv values, the intruding aircraft was set with a single NACv value for the duration of the simulated encounter and the reported velocity met the minimum requirements for that value; Figure C-5 contains the simulation results. The value of NACv has little impact on ACAS X performance.

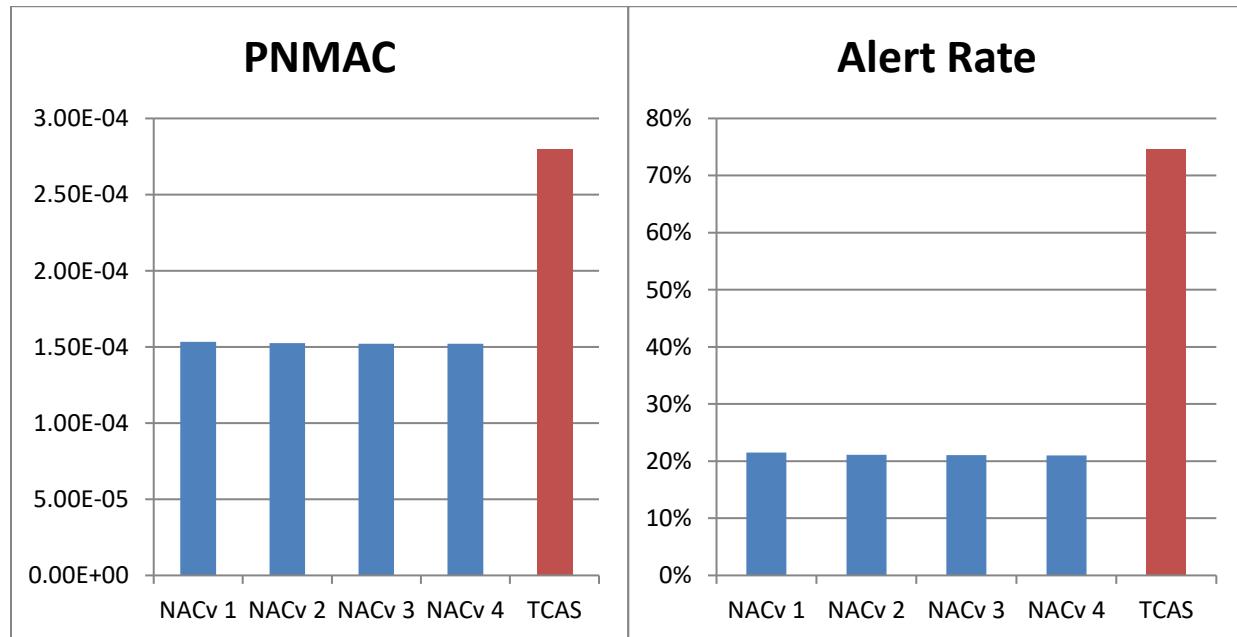


Figure C-5: NACv

C.3.2 Active

C.3.2.1 Bearingless

Active surveillance bearing observations are developed in the fashion as TCAS II. If surveillance fails to develop a bearing observation for an intruder, and if that failure persists over several update cycles, ACAS X will disregard the bearing and continue to track using only range and altitude – the intruder is then considered bearingless. ACAS X treats bearingless intruders as though they have zero cross-track velocity, therefore a decrease in ground-range separation implies the two aircraft are on a collision course. To test this degraded mode, the intruder's bearing was made unavailable to ACAS X for the duration of a simulated encounter. Figure C-6 contains the simulation results. ACAS X alerts more often, thereby increasing the separation and reducing the probability of NMAC. The system continues to alert less frequently than would TCAS.

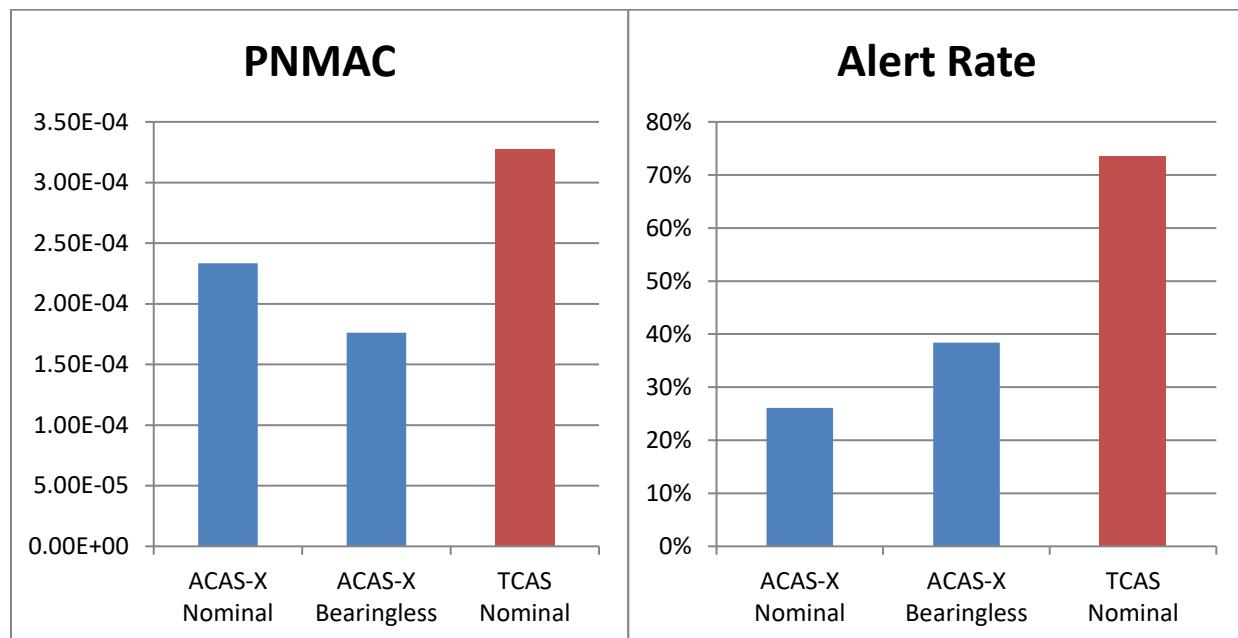


Figure C-6: Bearingless

C.3.2.2 Sensitivity to Stressing Bearing Error Models

ACAS X leverages active bearing measurements in developing a cross ground-range rate (cross-track velocity) estimate for intruders. Cross ground-range rate is one of several factors in ACAS X horizontal tau estimation, which influences the issuance of alerts by ACAS X.

Versions of TCAS II which incorporate a Horizontal Miss Distance (HMD) filter provide precedent for the use of active bearing measurements in determining whether to issue an alert. The HMD filter of TCAS II was a conservative application of active bearing measurements not sensitive to undetected bearing measurement failures. ACAS X builds on this experience by more tightly integrating the use of active bearing measurements into the alert logic, providing optimized timing of alerts based on state estimates informed by a series of active intruder bearing observations.

As a result ACAS X may be susceptible to specific modalities of bearing errors to an extent that TCAS II was not. A stressing analysis was conducted using an atypical bearing error model, which layered a 25% probability of transition into or out of a mode in which a parameterized bias was added to a representative, Hidden Markov Model based error for each bearing observation. The effect appears similar to step functions of variable widths superimposed on top of typical, flight-test validated bearing noise. This stressing error model is not representative of any observed or theorized errors, but intended to probe the

limits of ACAS X performance when subjected to repeated, sustained periods of bearing error. Analysis demonstrated that frequent, briefly-sustained bearing jumps of 30 – 60 degrees negatively impact ACAS X system safety, with maximum degradation observed at 45 degrees. Consideration of the potential for undetected bearing errors may be necessary during development or installation of ACAS X systems.

C.3.2.3 Active Update Delay

Should an intruder fail to reply to an ACAS X active interrogation, surveillance may reinterrogate that intruder several times within a single update cycle. This re-interrogation capability ensures a nominally successful reception. It would be useful, however, to understand the impact of missed detections on system performance.

To test the system performance when experiencing a reduced update rate, ATCRBS replies to interrogations were assigned a probability of reception such that at least one reply would be received with 95% probability in the prescribed update interval. The update interval was held constant for the duration of the simulated encounter. Figure C-77 contains the simulation results. ACAS X is robust against missed replies until those missed replies persist for longer than 3 seconds more than 5% of the time. Beyond this, ACAS X performance degrades as compared to a TCAS II unit operating under ideal conditions.

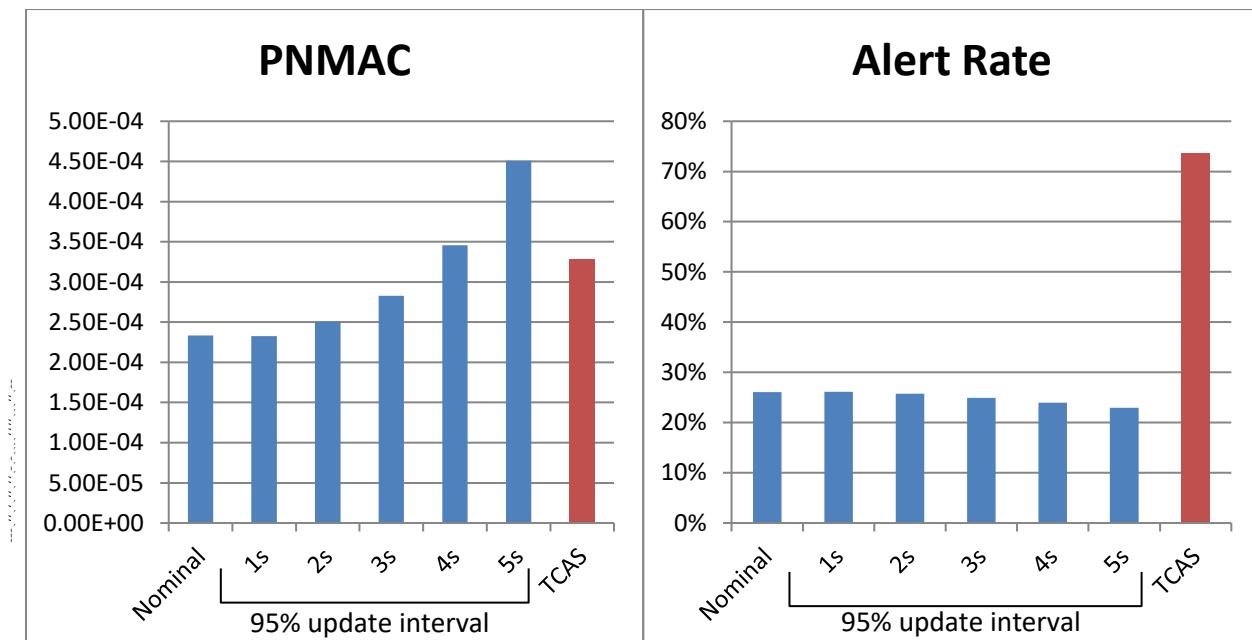


Figure C-7: Active update delay

C.3.2.4 Quantization Deception

This section describes an input mode that is expressly prohibited by the Mode S transponder specification, but was observed in recorded operational testing data.

In response to an interrogation from the ACAS X transponder, an aircraft will reply with a message that includes its altitude. Before being sent, the altitude is quantized and the amount of quantization is explicitly communicated. Current options for altitude quantization are either 100 ft or 25 ft for an aircraft equipped with a Mode S transponder; ATCRBS transponders are capable only of replying with 100 ft quantized altitude data. This analysis captures the expected difference in performance when a difference exists between the actual and reported altitude quantization levels received by ACAS X.

To test this faulty mode, the intruder altitude information was quantized to one level while reporting the other, persisting for the duration of the simulated encounters. Figure C-8 contains the simulation results. Should there be a mismatch between the reported and actual quantization of the intruder's altitude, ACAS X will continue to perform effectively.

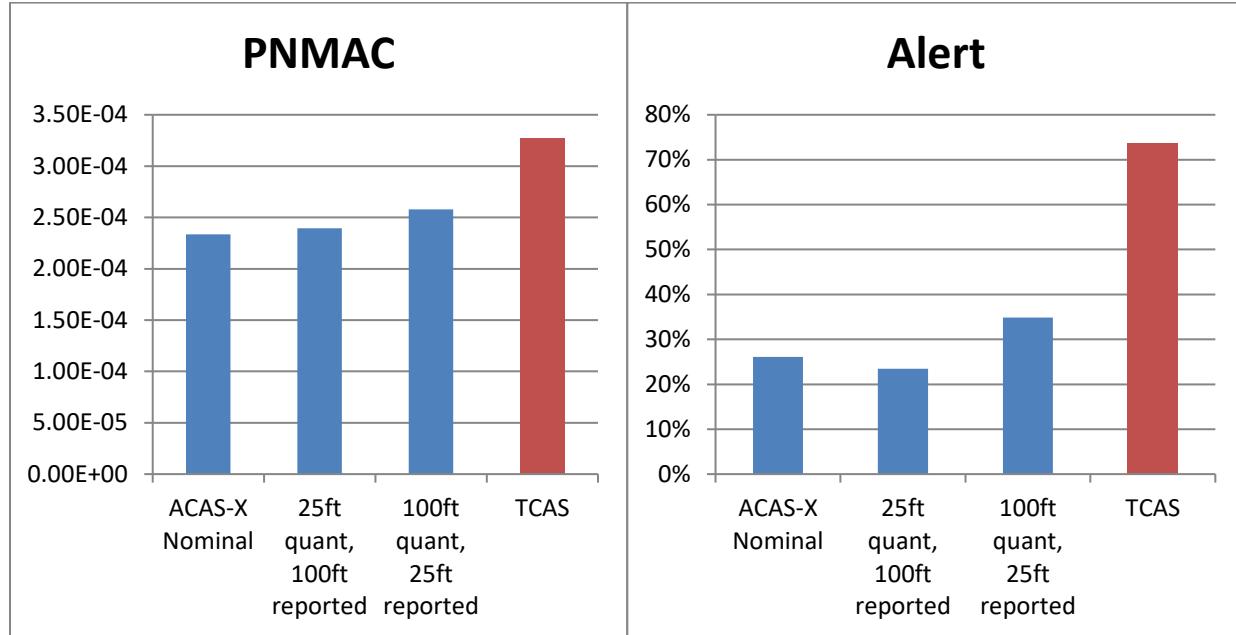


Figure C-8: Reported and Actual Quantization Mismatch

C.3.3 Ownership

C.3.3.1 Ownership Heading

ACAS X requires ownership heading be provided whereas TCAS II does not. The STM uses heading to translate the relative bearing and range of an actively tracked intruder to the relative-north and -east position of the intruder. Should heading information become unavailable, the surveillance front end system can either indicate that such a failure has occurred, or instead, provide an estimate of the track angle of the aircraft; both options are described below.

C.3.3.1.1 Headingless

When a heading source is unavailable, the STM still tracks in a body-fixed frame, but one that is relative to the nose of the aircraft instead of to true north. The rate estimates are then no longer of the intruder's relative-north and -east position, but of the intruder's position from the nose of the aircraft. Lag in the relative rate estimates, which is present both when heading is and is not available, can more easily affect the estimate of tau when ownership is maneuvering and no heading information is available. This vulnerability is due to the fact that the ownership heading directly impacts the relative bearing of an intruder with a one-to-one correspondence and that the heading can change much more quickly than can the bearing due solely to the intruder's movement.

To test this degraded mode, ownership heading was not provided to ACAS X for the duration of the encounter; Figure C-9 contains the simulation results. When not receiving heading, ACAS X is liable to alert later than it would under nominal conditions, shortening the time available to separate the aircraft and thereby reducing the achievable separation and increasing PNMAC. The vulnerability exists when the ACAS X aircraft turns. Despite the delayed alerts, ACAS X without heading inputs allows a comparable level of safety as does TCAS.

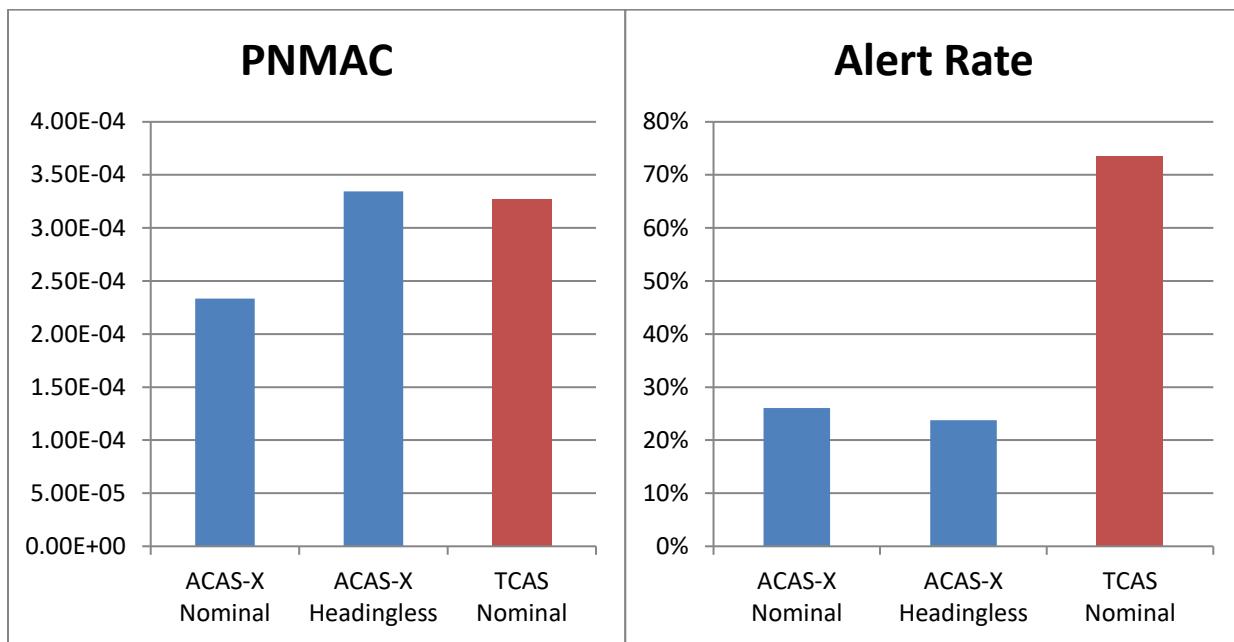
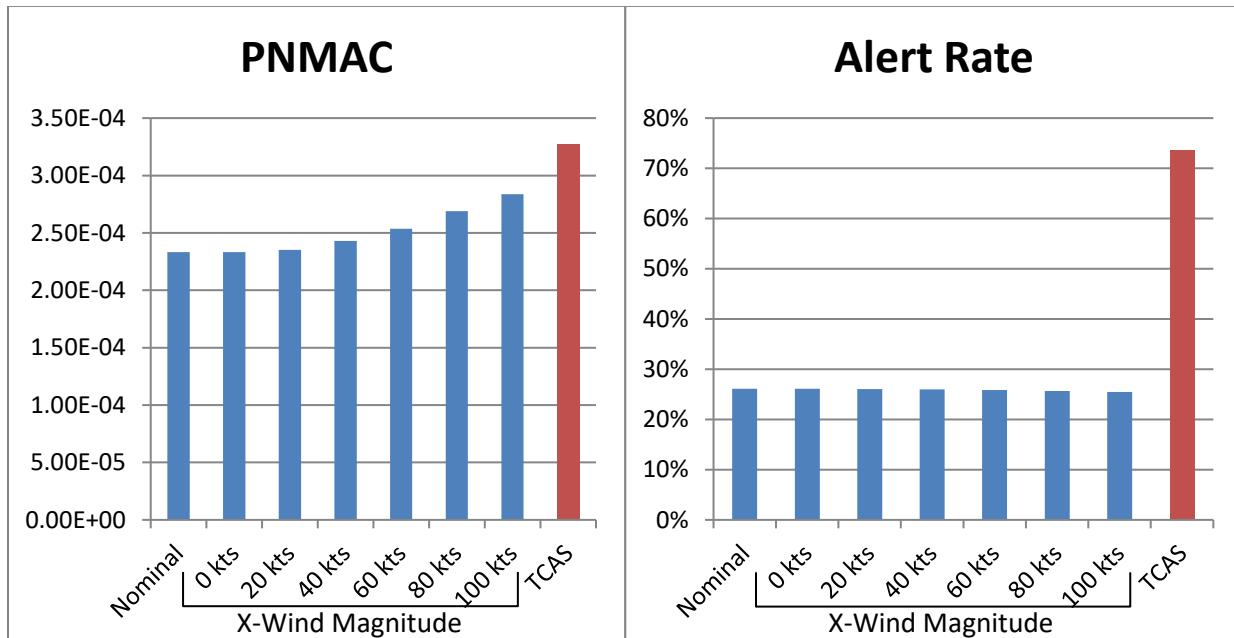


Figure C-9: No Heading Source Available

C.3.3.1.2 Track Angle

For an aircraft in motion, the track angle can differ from the heading in the presence of a crosswind: wind not aligned with the nose of the aircraft will exert a lateral force and can therefore affect the direction of flight. The difference between heading and track angle impacts intruder tracking only as the difference varies with time. A constant difference manifests itself only as a fixed rotation of the Cartesian body-fixed frame – the functional equivalent of rotating the intruder bearing by the ownship heading during nonmaneuvering flight. A varying difference, on the other hand, introduces an artificial bearing rate, which ultimately implies there will be some separation at the closest point of approach between the two aircraft. The crosswind, and therefore the difference between heading and track angle, changes as the magnitude of the crosswind changes or as the angle between the wind and the ownship changes, such as when ownship turns.

To test this degraded mode, the track angle was supplied to ACAS X and winds were simulated. The magnitude and the direction of the wind were held constant for each simulated encounter; any change in crosswind arises from ownship maneuvers, which are present in some but not all encounters. Figure C-30 contains the simulation results. Substituting track angle for the heading when unavailable provides the system with improved performance over TCAS; the system remains safe despite crosswind effects.

**Figure C-30: Track Angle Supplied as Ownship Heading****C.3.3.2 Ownship Altitude Quantization**

Ownship barometric altitude observations are nominally provided at 1 ft precision. There is an option for surveillance to provide altitude measurements that are quantized up to 100 ft precision. When surveillance exercises this option, the STM adjusts its tracking parameters to better track the ownship's altitude. Also affected are the conversions involving ownship altitude, such as the slant range to ground range conversion present in the active tracker.

To test this degraded mode, the ownship altitude measurements were quantized to 100 ft for the duration of the simulated encounter. Figure C-41 contains the simulation results. ACAS X is robust against coarsely quantized ownship altitude.

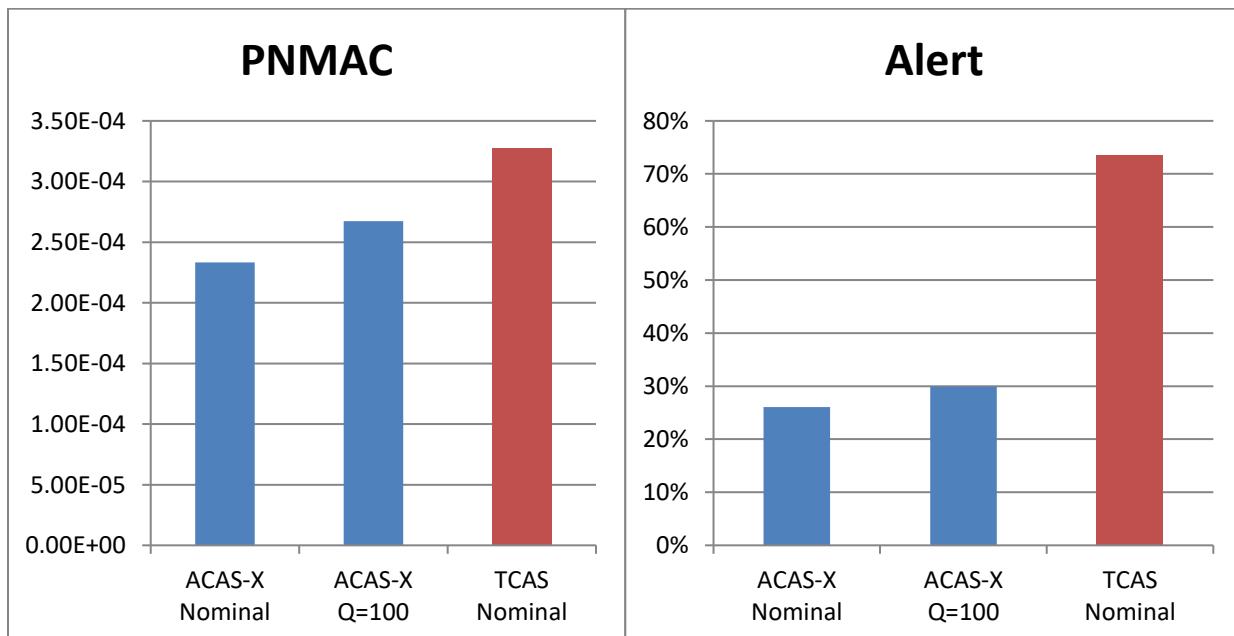


Figure C-41: – Ownship Coarse Altitude Quantization

C.3.4 TA Alerting Region

An analysis compares the dimensions of the TA alerting region with Normal and Reduced active surveillance regions.

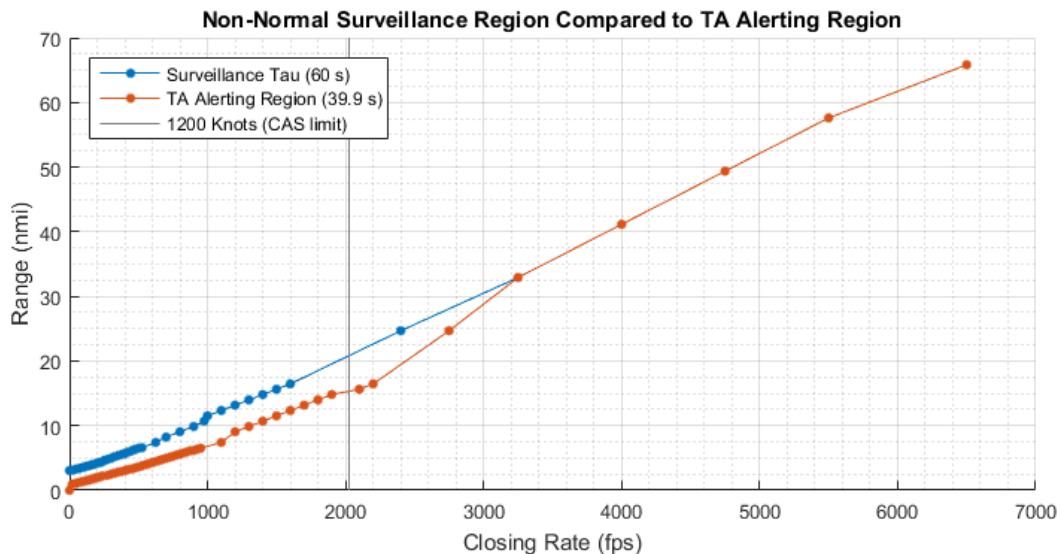


Figure C-12 – Non-Nominal Surveillance Region Compared to TA Alerting Region

One curve in Figure C-12 denotes the “Surveillance Tau” value of 60 seconds, which forms the border between Normal and Reduced surveillance regions. Details of the surveillance tau calculation and effect is found in §2.2.4.6.4.2.3.2.5. The Reduced active surveillance region lies above the 60 second surveillance tau curve, while the Normal active surveillance region is below the 60 second surveillance tau curve.

The second curve in Figure C-12 denotes the weighted tau value of 39.9 seconds as indicated within the ACAS-Xa entry table. Weighted tau is a critical factor in determining the issuance of TA or RA. No TA or RA will be issued at the maximum weighted tau value of 40 seconds. At weighted tau values below 40 seconds a TA could be issued. The weighted tau value of 39.9 seconds was chosen as the practical upper limit for the potential issuance of any TA. Note that a weighted tau value below 40 seconds does not indicate that a TA must be issued, only that TA issuance is not prohibited.

The geometry of the curves indicates that the potential TA issuance region (the area at and below the 39.9 second weighted tau curve) is fully encompassed by the Normal active surveillance region up to intruder ranges of 33 NM. At extreme active surveillance ranges (beyond 33 NM) and closing rates (above 3000 fps) the curves merge, indicating that a TA could be issued simultaneously with the transition to Normal active surveillance. There is no area where the curves cross, indicating that there is no possibility of a TA outside of the Normal active surveillance region.

C.4 Summary

In Table C-1 and Table C-2, the performance resulting from each degraded input is shown relative to the two baseline scenarios – nominal ACAS X and nominal TCAS. Table C-1 lists degraded modes affecting

intruders that were passively tracked; Table C-2, actively tracked. As an example, Table C-1 indicates that gated ADS-B reports increase the probability of NMAC by 0.7% and increase the alert rate by 1.3% when compared to ACAS X receiving ADS-B reports at the nominal rate. Gated ADS-B reports, however, still allow ACAS X a substantial increase in system performance, a decrease in PNMAC of 33.5% and a decrease in alert rate of 82.2%, when compared to TCAS.

Table C-1: ADS-B Degradations

	Relative difference to ACAS X with nominal ADS-B inputs		Relative difference to TCAS	
	PNMAC	Alerting	PNMAC	Alerting
ACAS X with Nominal ADS-B inputs			-33.9%	-82.5%
TCAS	51.3%	469.9%		
Gated	0.7%	1.3%	-33.5%	-82.2%
95% update interval - 3 s	5.6%	11.3%	-30.2%	-80.5%
95% update interval - 5 s	14.7%	31.6%	-24.2%	-76.9%
NACp 7	1.8%	-0.1%	-32.7%	-82.5%
NACp 8 (defined as nominal)	0.0%	0.0%	-33.9%	-82.5%
NACp 9	0.1%	-0.1%	-33.9%	-82.5%
NACp 10	-0.1%	-0.5%	-34.0%	-82.5%
NACv 1 (defined as nominal)	0.0%	0.0%	-33.9%	-82.5%
NACv 2	-0.4%	-2.4%	-34.2%	-82.9%
NACv 3	-0.5%	-3.3%	-34.2%	-83.0%
NACv 4	-0.8%	-3.3%	-34.5%	-83.0%

As seen in Table C-1, all ADS-B input modes provide ACAS X with improved safety (lower PNMAC) and operational suitability (fewer alerts) when compared to TCAS.

Table C-2: Degraded Inputs Affecting Active Tracks

	Relative Difference to ACAS X with nominal Mode C inputs		Relative Difference to TCAS	
	PNMAC	Alerting	PNMAC	Alerting
ACAS X with Nominal Mode C inputs			-28.7%	-64.6%
TCAS	40.3%	182.1%		
Bearingless	-24.5%	47.2%	-46.2%	-47.8%
95% update interval - 1 s	-0.4%	0.2%	-29.0%	-64.5%
95% update interval - 2 s	7.3%	-1.3%	-23.5%	-65.0%
95% update interval - 3 s	21.2%	-4.5%	-13.6%	-66.1%
95% update interval - 4 s	48.1%	-8.2%	5.6%	-67.5%
95% update interval - 5 s	93.3%	-12.1%	37.8%	-68.8%

25ft reporting 100	2.6%	-10.3%	-26.9%	-68.2%
100ft reporting 25	10.7%	33.7%	-21.1%	-52.6%
Headingless	43.4%	-8.9%	2.2%	-67.7%
Wind Magnitude 0 kts	0.0%	0.0%	-28.7%	-64.6%
Wind Magnitude 20 kts	0.8%	-0.1%	-28.1%	-64.6%
Wind Magnitude 40 kts	4.1%	-0.3%	-25.8%	-64.7%
Wind Magnitude 60 kts	8.7%	-0.8%	-22.5%	-64.8%
Wind Magnitude 80 kts	15.2%	-1.5%	-17.9%	-65.1%
Wind Magnitude 100 kts	21.6%	-2.4%	-13.3%	-65.4%
Ownship Coarse Altitude Quantization	14.6%	14.7%	-18.3%	-59.3%

All ACAS X degraded modes that affect active tracks, with the exception of headingless mode and unrealistically long update delays, permit improved performance over TCAS operating with nominal inputs. When suffering from complete heading loss, ACAS X provides comparable safety as does TCAS. It is expected that heading will be available a majority of the time and complete loss a rare event, so the overall performance of ACAS X compared to TCAS is better approximated by comparing the simulation results of the two systems with nominal inputs. The long active updates delays are unrealistic, and would result in safety impacts to all active surveillance systems, including TCAS II.

APPENDIX D CONVERSION OF REPORTED POSITIONS TO SLANT RANGE

This appendix provides useful guidance on computing range from own and reported position data. However, this section does not recommend a particular implementation and should be used for reference only. The equipment manufacturer must meet the computational accuracy requirements of §2.2.4.6.4.2.3.1.11.

D.1 Overview

This appendix provides useful guidance on computing range from own and reported position data. This appendix does not recommend a particular implementation and should be used for reference only.

First, the exact conversion equations from position to slant range are given. The computational requirements for the exact conversion equations are reasonable and could be used as is for modern processors and typical TCAS traffic loads.

Second, several approximate conversion equations from position to slant range are presented. For circumstances where Hybrid Surveillance is implemented as a software upgrade to existing processors, it may be desirable to use approximations to the conversion equations to reduce the computational requirements. The errors in the approximate equations are presented and compared to the computational accuracy requirements of §2.2.4.6.4.2.3.1.11, which requires a maximum 145 m processing error when calculating slant range.

D.2 Exact Conversion Equations

The following equations (Ref. N, Appendix 3.A.1, p. 115) give the exact conversion from latitude, longitude, and height above the WGS 84 ellipsoid to x, y, z earth-centered earth-fixed coordinates, based on the WGS 84 ellipsoidal earth model used by GNSS.

$$x = (N + h) \cos \varphi \cos \lambda \quad (D2-1)$$

$$y = (N + h) \cos \varphi \sin \lambda \quad (D2-2)$$

$$z = (N(1 - e^2) + h) \sin \varphi \quad (D2-3)$$

where:

$N = a / (1 - e^2 \sin^2 \varphi)^{1/2}$ = the length of a line normal to the ellipsoid between the point (φ, λ) on the surface of the ellipsoid and the point where it intersects the axis of the ellipsoid

h = height above the WGS 84 ellipsoid

φ = geodetic latitude

λ = geodetic longitude

$e^2 = (a^2 - b^2)/a^2$ = the square of the first eccentricity of the WGS 84 ellipsoid = $6.69437999014 \times 10^{-3}$

a = the semi-major axis of the WGS 84 ellipsoid = 6378137.0 m

b = the semi-minor axis of the WGS 84 ellipsoid = 6356752.3142 m

By converting both own and other aircraft's positions to x, y, z, the slant range can then be obtained by the standard equation:

$$r^2 = (x_{\text{other}} - x_{\text{own}})^2 + (y_{\text{other}} - y_{\text{own}})^2 + (z_{\text{other}} - z_{\text{own}})^2 \quad (D2-4)$$

Note that N is a function of ϕ and so the exact result requires calculating separate values of N for ownship and the other aircraft. However, N is a slowly varying function of latitude (see Table D-1) and could probably be calculated accurately enough by table lookup and interpolation with a small table of values.

The exact equations for slant range given above should not be difficult for modern computers to process in real time for the traffic loads typical for TCAS. They involve the calculation of four trigonometric functions (the sine and cosine of both latitude and longitude) and a square root to calculate x, y, and z. That must be done for ownship and for each of the other aircraft. An additional square root is then required for each of the other aircraft to calculate its slant range. Additional calculations may be required to extrapolate ownship's position to the time of validity of the other aircraft's reported position, and the conversion of ownship's position to x, y, z may therefore need to be performed for each report received from other aircraft.

Table D-1: Comparison of R and N for a Spherical Earth Radius as a Function of Latitude

Assumed spherical earth radius A = 6,366,707 m WGS 84 semi-major axis = 6,378,137.0 m WGS 84 semi-minor axis = 6,356,752.3142 m				
Latitude (deg)	R (m)	A – R (m)	N (m)	A – N (m)
89	6399574	-32867	6399587	-32880
80	6397643	-30936	6398943	-32236
70	6392033	-25326	6397072	-30365
60	6383454	-16747	6394209	-27502
50	6372956	-6249	6390702	-23995
40	6361816	4891	6386976	-20269
30	6351377	15330	6383481	-16774
20	6342888	23819	6380636	-13929
10	6337358	29349	6378781	-12074
0	6335439	31268	6378137	-11430
-10	6337358	29349	6378781	-12074
-20	6342888	23819	6380636	-13929
-30	6351377	15330	6383481	-16774
-40	6361816	4891	6386976	-20269
-50	6372956	-6249	6390702	-23995
-60	6383454	-16747	6394209	-27502
-70	6392033	-25326	6397072	-30365
-80	6397643	-30936	6398943	-32236
-89	6399574	-32867	6399587	-32880

D.3 Height Above Ellipsoid vs. Barometric Altitude

The h term in the above equations is the height above the WGS 84 ellipsoid, also known as HAE. This may be available for ownship, but is not usually provided in the Airborne Position Messages of other aircraft. For the purpose of range calculations for TCAS Hybrid Surveillance, it turns out to be sufficiently accurate to substitute uncorrected barometric altitude for HAE, both for own and other aircraft (large errors may result if it is not done for both).

The difference between uncorrected barometric altitude and true HAE is the sum of two effects. The geoid is the surface that represents Mean Sea Level in the WGS 84 system. The geoid is not the same as the WGS 84 ellipsoid. The geoid may be above or below the WGS 84 ellipsoid. The difference between the geoid and the WGS 84 ellipsoid is termed the geoidal height. Thus the first error effect is the geoidal height, which ranges from -104 m to +75 m (Ref. G, p. 83), where positive values indicate the geoid is above the ellipsoid, and negative values indicate it is below.

The second error effect is the difference between the uncorrected barometric altitude and the true (corrected) altitude above mean sea level (above the geoid). This is due to atmospheric pressure variations. The maximum variation likely to be encountered by an aircraft can be estimated by observing the range of adjustment provided in the Kollsman window of an aircraft altimeter. The pilot sets the value in this window to the reported sea level barometric pressure at or near ownship's location. This then corrects the altitude shown on the face of the altimeter to the true altitude with respect to mean sea level. The range of sea level pressure settings for one such altimeter was observed to be from 28.1 inches of mercury (in. Hg) to 31.0 in. Hg. In units of millibars, this is 952 mbar to 1050 mbar. The standard atmospheric pressure assumed in uncorrected barometric altitudes is 29.92 in. Hg (1013 mbar). Thus, the uncorrected barometric pressure difference measured by the altimeter could be in error (measured - true) from -61 to +37 mbar. At sea level, the pressure varies with altitude at a rate of 3.5 mbar per 30 m (Ref. O); these pressure differences would give altitude errors (reported - true) in the range of -523 m to +317 m (-1716 ft to +1040 ft).

The combination of both error effects could lead to errors (measured - true) in the range of -627 m to +392 m (-2057 ft to 1286 ft) when uncorrected barometric altitude is substituted for HAE. As a crosscheck, note that the Extended Squitter Airborne Velocity Message provides a range of ± 3125 ft (± 953 m) for the "Geometric Height Difference from Barometric Altitude" field (Ref. G, Appendix A, p. A-58).

Numerical tests performed by inserting a common altitude error into the h values used for both own and other aircraft, such as would occur if the uncorrected barometric altitude was substituted for HAE for both aircraft, and assuming the aircraft are approximately 15 NM apart, indicate that common errors in h of ± 1000 m result in an error in the calculated slant range r of no more than ± 5 m. This is negligible with respect to the 145 m processing error budget specified in §2.2.4.6.4.2.3.1.11.

D.4 Spherical Earth Approximation

One approximation to the slant range calculation is to assume a spherical earth. The equations are the same as those for the ellipsoidal earth, simplified by substituting the assumed constant earth radius for the varying value N and setting the eccentricity e to zero. In terms of calculation, this saves one square root per aircraft due to eliminating the calculation of N .

Table D-2 illustrates the error in the spherical earth approximation for one condition using an earth radius of 6,366,707 m. The delta latitude and the delta longitude at the equator assumed for these results was 0.175 deg. The delta longitude was adjusted with latitude by dividing the equatorial value by the cosine of the latitude in order to give approximately the same slant range at all latitudes. Ownship HAE was 40,000 feet, and other aircraft's HAE was 30,000 feet. Similar results are obtained at other relative bearings, except that the error at the equator can become roughly as large as the error at the pole when the aircraft are primarily north-south of each other.

Table D-2: Example of Errors in Spherical Earth Approximation

Aircraft 1 altitude = 40,000 ft Aircraft 2 altitude = 30,000 ft Aircraft 2 latitude = Aircraft 1 latitude + 0.175 deg Aircraft 2 longitude = Aircraft 1 longitude + Delta longitude			
Aircraft 1 latitude (deg)	Delta longitude (deg)	True slant range (NM)	Spherical earth slant range – true slant range (m)
89	10.027	14.368	-135
80	1.008	14.974	-135
70	0.512	14.998	-119
60	0.350	14.997	-95
50	0.272	14.987	-65
40	0.228	14.973	-33
30	0.202	14.960	-3
20	0.186	14.949	21
10	0.178	14.943	37
0	0.175	14.942	42
-10	0.178	14.947	37
-20	0.186	14.958	21
-30	0.202	14.973	-3
-40	0.228	14.992	-33
-50	0.272	15.013	-65
-60	0.350	15.036	-95
-70	0.512	15.060	-120
-80	1.008	15.102	-136
-89	10.027	15.667	-147

D.5 Approximation Assuming N is the Same for All Aircraft

A second approximation to the slant range calculation is to assume that N is the same for both own and other aircraft in the exact equations. This saves one square root per other aircraft. Some numerical results from such an approximation are given in Table D-3. Since N varies with latitude but not longitude, the example shows a case where the aircraft are north-south of each other, resulting in the greatest latitude difference. If they had the same latitude, the error would be zero. If the parameters used in Table D-3: are modified by setting the altitude of aircraft 2 to 40,000 ft (no altitude difference), the errors are reduced to under 0.5 m at all latitudes.

Table D-3: Example of Errors Using the Same N for Both Aircraft

<p>Aircraft 1 altitude = 40,000 ft Aircraft 2 altitude = 30,000 ft Aircraft 2 latitude = Aircraft 1 latitude + 0.25 deg Aircraft 2 longitude = Aircraft 1 longitude</p>			
Aircraft 1 latitude (deg)	Delta longitude (deg)	True slant range (NM)	Slant range assuming $N1=N2 - \text{true slant}$ range (m)
89	0.000	15.192	0.3
80	0.000	15.188	3.4
70	0.000	15.175	6.5
60	0.000	15.154	8.9
50	0.000	15.130	10.2
40	0.000	15.104	10.3
30	0.000	15.079	9.0
20	0.000	15.059	6.6
10	0.000	15.046	3.5
0	0.000	15.042	0.0
-10	0.000	15.046	-3.3
-20	0.000	15.059	-6.2
-30	0.000	15.079	-8.3
-40	0.000	15.103	-9.4
-50	0.000	15.129	-9.4
-60	0.000	15.154	-8.3
-70	0.000	15.174	-6.2
-80	0.000	15.187	-3.4
-89	0.000	15.192	-0.4

D.6 Ellipsoidal Differential Approximation

A third approximation to the slant range calculation is to treat the distances involved as if they were small enough to be treated as infinitesimals. The differential distance element on the surface of an ellipsoid is (Ref. P).¹

$$ds^2 = (R d\phi)^2 + (N \cos \phi d\lambda)^2 \quad (D6-1)$$

$$R = a(1 - e^2) / (1 - e^2 \sin^2 \phi)^{3/2} \quad (D6-2)$$

where R is the radius of curvature of the meridian ellipse.

To convert this to an equation for slant range, the assumption is made that the distances involved can be treated as if they are infinitesimal, and that the (ϕ, λ, h) coordinate system can be treated as linear, rather than curvilinear, over those distances. (The true coordinate axes are orthogonal at ownship's location, but two of them curve as they move away, while the approximation assumes they are straight lines.) Adjustments must also be made to account for the height of ownship above the reference ellipsoid, and to account for the height difference between the two aircraft. The resulting equation is:

$$r^2 = ((R + h) \Delta\phi)^2 + ((N + h) \cos \phi \Delta\lambda)^2 + \Delta h^2$$

R, N, h and $\cos \phi$ are evaluated at ownship's location, and the latitude and longitude differences are substituted for the infinitesimals. This involves only two trigonometric functions (sine and cosine of latitude) and one square root (the calculations of R and N can share the result), and only for ownship. Using table lookup and interpolation for R and N can reduce the computations further, as they are both slowly varying functions of latitude (see Table D-1). For each of the other aircraft, only a square root is required.

Various numerical results using this approximation are shown in Table D-4, Table D-5, and Table D-6. The HAE of one aircraft is 40,000 ft and the HAE of the other aircraft is 30,000 ft. In Table D-4, the delta longitude has been set to zero in order to show the accuracy of the approximation when only latitude differences are involved. The delta latitude is 0.25 deg. The approximation is excellent at all latitudes. The errors shown are due to having a combined height and latitude difference and the coordinate system not being truly linear over the distances involved. If the height difference is set to zero, the resulting errors are less than 1 m.

In Table D-5, the delta latitude has been set to zero in order to show the accuracy of the approximation when only longitude differences are involved. The delta longitude at the equator is 0.25 deg, and it is adjusted with latitude by dividing by the cosine of the latitude in order to keep the slant range approximately constant. The approximation is again excellent at all latitudes except close to the poles, where the assumptions underlying the approximation break down. Once again, if the height difference is set to zero the errors are less than a meter except for the values at ± 89 deg, which become 73 m.

In Table D-6, the delta latitude and the delta longitude at the equator have both been set to 0.175 deg. The same cosine correction is made to the delta longitude as a function of latitude. These are the same values used to generate Table D-2. The approximation is still quite good at most middle latitudes, but not as good as in Table D-4 and Table D-5, again showing the deviation from the assumed linearity over the distances

¹ This is a somewhat obscure publication, and derives the equation for the differential of arc length on a spheroid (ellipsoid of rotation) from the fundamental equations for the differential of arc length of an arbitrary smooth curve on a smooth surface, which is a complex approach. However, a heuristic derivation is as follows. R is the radius of curvature of the meridian ellipsoid. The equation for R can be derived using the general equation for the radius of curvature found in advanced calculus texts. The differential of arc length $R d\theta$ for a circle of radius R is a direct result of the definition of θ in radians. Similarly, $N \cos \phi$ is the radius of the circle of constant latitude and $N \cos \phi d\lambda$ gives the differential arc length along that circle. Assuming these two quantities are orthogonal in the infinitesimal limit gives the desired result.

involved. This approximation is better than the spherical earth approximation at latitudes within ± 80 deg. The error is less than the bias allowance for Mode S transponders within a range slightly larger than ± 50 deg latitude, which would cover most of the dense traffic regions of the world except northern Europe, including the United Kingdom. If the height difference is set to zero, about 6-7 m is subtracted from these errors, reducing the magnitude of the errors in the northern hemisphere but increasing it in the southern hemisphere, making the errors more symmetrical with respect to north-south latitude.

The net result of using an ellipsoidal differential approximation is a significant reduction in the calculations required to compute slant range, balanced by moderate errors at most latitudes. At higher latitudes the approximation becomes progressively less accurate, depending on the azimuth of the other aircraft.

Table D-4: Example of Errors in Ellipsoidal Differential Approximation with No Longitude Difference

Aircraft 1 altitude = 40,000 ft Aircraft 2 altitude = 30,000 ft Aircraft 2 latitude = Aircraft 1 latitude + 0.25 deg Aircraft 2 longitude = Aircraft 1 longitude			
Own latitude (deg)	Delta longitude (deg)	True slant range (NM)	Differential slant range – true slant range (m)
89	0.000	15.192	7
80	0.000	15.188	6
70	0.000	15.175	6
60	0.000	15.154	6
50	0.000	15.130	6
40	0.000	15.104	6
30	0.000	15.079	6
20	0.000	15.059	6
10	0.000	15.046	6
0	0.000	15.042	7
-10	0.000	15.046	7
-20	0.000	15.059	7
-30	0.000	15.079	7
-40	0.000	15.103	7
-50	0.000	15.129	7
-60	0.000	15.154	7
-70	0.000	15.174	7
-80	0.000	15.187	7
-89	0.000	15.192	7

Table D-5: Example of Errors in Ellipsoidal Differential Approximation with No Latitude Difference

Aircraft 1 altitude = 40,000 ft Aircraft 2 altitude = 30,000 ft Aircraft 2 latitude = Aircraft 1 latitude Aircraft 2 longitude = Aircraft 1 longitude + Delta longitude			
Own latitude (deg)	Delta longitude (deg)	True slant range (NM)	Differential slant range – true slant range (m)
89	14.325	15.153	79
80	1.440	15.190	7
70	0.731	15.186	7
60	0.500	15.179	7
50	0.389	15.171	7
40	0.326	15.162	7
30	0.289	15.154	7
20	0.266	15.148	7
10	0.254	15.143	7
0	0.250	15.142	7
-10	0.254	15.143	7
-20	0.266	15.148	7
-30	0.289	15.154	7
-40	0.326	15.162	7
-50	0.389	15.171	7
-60	0.500	15.179	7
-70	0.731	15.186	7
-80	1.440	15.190	7
-89	14.325	15.153	79

Table D-6: Example of Errors in Ellipsoidal Differential Approximation with Combined Latitude, Longitude and Height Differences

Aircraft 1 altitude = 40,000 ft Aircraft 2 altitude = 30,000 ft Aircraft 2 latitude = Aircraft 1 latitude + 0.175 deg Aircraft 2 longitude = Aircraft 1 longitude + Delta longitude			
Own latitude (deg)	Delta longitude (deg)	True slant range (NM)	Differential slant range – true slant range (m)
89	10.027	14.368	1253
80	1.008	14.974	126
70	0.512	14.998	64
60	0.350	14.997	43
50	0.272	14.987	31
40	0.228	14.973	24
30	0.202	14.960	18
20	0.186	14.949	14
10	0.178	14.943	10
0	0.175	14.942	7
-10	0.178	14.947	3
-20	0.186	14.958	-1
-30	0.202	14.973	-5
-40	0.228	14.992	-11
-50	0.272	15.013	-18
-60	0.350	15.036	-30
-70	0.512	15.060	-51
-80	1.008	15.102	-112
-89	10.027	15.667	-1153

D.7 Spherical Differential Approximation

A fourth approximation to the slant range equations is a spherical differential approximation. The spherical differential approximation is presented in Ref. G, §2.2.3.2.3.7, §2.2.3.2.3.8 and subparagraphs. The equations can be obtained from that given for the ellipsoid in §D.6 by substituting the assumed radius for the spherical earth (6,366,707 m) for both R and N. The spherical differential approximation in Ref. G does not include the correction for ownship's height. A height of 40,000 ft corresponds to 12,192 m, which would be added to the earth's radius. For extrapolations of no more than 2 s as specified in Ref. G this difference can be neglected. However, ignoring it for a slant range of 15 NM would result in an error on the order of 53 m, so it should be taken into account in the slant range calculation.

Table D-7, Table D-8, and Table D-9 give numerical results for the spherical differential approximation under the same conditions as Table D-4, Table D-5, and Table D-6 for the ellipsoidal differential approximation. The results include the height correction to the assumed earth radius. The errors are of the same order of magnitude as the spherical approximation, except at high latitudes where they can be much worse, but vary greatly for a given latitude depending on azimuth. These results can be better understood by comparing the spherical radius A used in these tables with the terms R and N used in the more exact ellipsoidal differential correction, as shown in Table D-1. The assumed radius A is less than R at the poles, and greater than R at the equator. Therefore the latitude term of the equation gives too small a result at the poles, and too large at the equator. On the other hand, A is always smaller than N, but A deviates more at the poles and less at the equator. Therefore, the longitude component is always too small, but less so at the equator. Close to the pole, the effect of A being too small partly corrects for the overestimation of the longitude component that results from the failure of the assumption of a linear coordinate system.

Table D-7: Example of Errors in Spherical Differential Approximation with No Longitude Difference

Aircraft 1 altitude = 40,000 ft Aircraft 2 altitude = 30,000 ft Aircraft 2 latitude = Aircraft 1 latitude + 0.25 deg Aircraft 2 longitude = Aircraft 1 longitude			
Own latitude (deg)	Delta longitude (deg)	True slant range (NM)	Differential slant range – true slant range (m)
89	0.000	15.192	-136
80	0.000	15.188	-128
70	0.000	15.175	-104
60	0.000	15.154	-67
50	0.000	15.130	-21
40	0.000	15.104	27
30	0.000	15.079	73
20	0.000	15.059	110
10	0.000	15.046	134
0	0.000	15.042	142
-10	0.000	15.046	134
-20	0.000	15.059	110
-30	0.000	15.079	74
-40	0.000	15.103	28
-50	0.000	15.129	-20
-60	0.000	15.154	-65
-70	0.000	15.174	-103
-80	0.000	15.187	-127
-89	0.000	15.192	-136

Table D-8: Example of Errors in Spherical Differential Approximation with No Latitude Difference

Aircraft 1 altitude = 40,000 ft Aircraft 2 altitude = 30,000 ft Aircraft 2 latitude = Aircraft 1 latitude Aircraft 2 longitude = Aircraft 1 longitude + Delta longitude			
Own latitude (deg)	Delta longitude (deg)	True slant range (NM)	Differential slant range – true slant range (m)
89	14.325	15.153	-64
80	1.440	15.190	-132
70	0.731	15.186	-125
60	0.500	15.179	-113
50	0.389	15.171	-97
40	0.326	15.162	-81
30	0.289	15.154	-66
20	0.266	15.148	-54
10	0.254	15.143	-46
0	0.250	15.142	-43
-10	0.254	15.143	-46
-20	0.266	15.148	-54
-30	0.289	15.154	-66
-40	0.326	15.162	-81
-50	0.389	15.171	-97
-60	0.500	15.179	-113
-70	0.731	15.186	-125
-80	1.440	15.190	-132
-89	14.325	15.153	-64

Table D-9: Example of Errors in Spherical Differential Approximation with Combined Latitude, Longitude and Height Differences

Aircraft 1 altitude = 40,000 ft Aircraft 2 altitude = 30,000 ft Aircraft 2 latitude = Aircraft 1 latitude + 0.175 deg Aircraft 2 longitude = Aircraft 1 longitude + Delta longitude			
Own latitude (deg)	Delta longitude (deg)	True slant range (NM)	Differential slant range – true slant range (m)
89	10.027	14.368	1112
80	1.008	14.974	-10
70	0.512	14.998	-55
60	0.350	14.997	-52
50	0.272	14.987	-34
40	0.228	14.973	-9
30	0.202	14.960	15
20	0.186	14.949	35
10	0.178	14.943	47
0	0.175	14.942	49
-10	0.178	14.947	40
-20	0.186	14.958	20
-30	0.202	14.973	-8
-40	0.228	14.992	-44
-50	0.272	15.013	-83
-60	0.350	15.036	-125
-70	0.512	15.060	-170
-80	1.008	15.102	-248
-89	10.027	15.667	-1294

APPENDIX E ANALYSIS OF VALIDATION/REVALIDATION RANGE TOLERANCE ERROR BUDGET

E.1 Introduction

To get the benefit of Hybrid Surveillance, the errors under the control of Hybrid Surveillance should be minimized as much as practical to ensure maximum usage of Hybrid Surveillance. Some sources of error are outside the control of Hybrid Surveillance. These include:

ACAS X active surveillance range error.

Intruder passive position sensor error.

Intruder position message LSB.

Uncompensated latency in the intruder position when received by ACAS X.

Ownship passive position sensor error.

Uncompensated latency in ownship passive position when received by ACAS X.

Errors that can be controlled to some degree by ACAS X Hybrid Surveillance are referred to as *hybrid data processing errors*. These errors include:

Differences in the time of applicability of the ACAS X range measurement and the intruder position data once it is received by ACAS X. Although this time difference cannot be eliminated, design options such as whether to use the UTC time synchronization will affect the size of the error.

Differences in the time of applicability of the ACAS X range measurement and ownship position data once it is received by ACAS X. It is possible to time align this data by extrapolating or interpolating ownship position data and ACAS X range data to a common time for the comparison. Ownship data can be extrapolated by ownship rates if available, and ACAS X range can be extrapolated by ACAS X range rate estimates if available.

Approximations in calculating the slant range from own and intruder latitude, longitude and altitude that might be used to simplify processing (See **Appendix D**).

A 290 m validation range threshold was chosen based on the analysis shown in Table E-1. The 340 m revalidation threshold was made 50 m larger than the validation threshold in order to provide some hysteresis and so reduce the chance of an intruder near the threshold going to active surveillance because of small changes in the ranging errors. The following sections explain how the values in Table E-1 were estimated.

Table E-1 Error Sources in the Hybrid Slant Range Calculation and Validation with ACAS X Range

Errors Outside the Control of ACAS X Hybrid Surveillance	
ACAS X active surveillance range error, 95%	69 m
Passive surveillance intruder position sensor error, 95%	93 m
Intruder position message LSB	5 m
Uncompensated latency in intruder position when received by ACAS X	185 m
Passive surveillance own position sensor error, 95%	93 m
Uncompensated latency in ownship position when received by ACAS X	77 m
Total errors outside control of ACAS X Hybrid Surveillance (assuming errors combine by root-sum-square)	250 m
Error budget for data processing by Hybrid Surveillance during validation	145 m
Resulting size of validation range tolerance (assuming errors combine by root-sum-square)	290 m (0.157 NM)
Revalidation range tolerance (validation range tolerance plus 50 m)	340 m (0.184 NM)

E.2 ACAS X Active Surveillance Range Error

ACAS X range accuracy is specified in §2.2.2.2.3, which requires that ACAS X measure the range of Mode S targets with signal power at least 6 dB above MTL with no more than 125 ft (38.1 m) bias and 50 ft (15.24 m) rms jitter.

The acceptable range error can be calculated from the bias and jitter using

$$\text{Acceptable range error} = \text{bias} + 2\sigma = 125 \text{ ft} + 2*50 \text{ ft} = 225 \text{ ft} \quad (69 \text{ m}) \quad (\text{E2-1})$$

E.3 Own and Intruder Position Sensor Error

14 CFR §91.227 requires that the NACp for all ADS-B Out units indicate a 95% accuracy of 0.05 NM or better. That corresponds to 93 m. That accuracy is assumed for both the intruder position sensor error and own position sensor error.

E.4 Intruder Message LSB

The intruder message LSB is specified in Ref. G, §A.1.4.2.3. The position squitter encoding using compact position reporting has a resolution of approximately 5.1 m.

E.5 Uncompensated Latency in Intruder Position When Received by ACAS X

14 CFR §91.227 requires the uncompensated latency for all ADS-B Out units to be less than 0.6 s. At the maximum aircraft ground speed of 600 kt (309 m/s) assumed in the ACAS X design, that corresponds to a distance of 185 m. That is the worst-case position error that is possible for an intruder whose uncompensated latency satisfies the specification. That much range error would only be observed in a head-on encounter with the intruder having a ground speed of 600 kt. In addition to the range error being smaller at slower speeds, smaller values would apply if the intruder's relative heading was different from 180 degrees, with the intruder's velocity vector, and the position error due to uncompensated latency, being

projected onto the line of sight between the two aircraft. Finally, 0.6 s is the maximum uncompensated latency that is allowed, and it can be expected that many actual values will be smaller than that.

The analysis in Table E-1 treats each of the errors as a normally distributed or Gaussian random variable. The magnitudes given in that table are treated as representing about two standard deviations of their respective distributions, so that 95% of all errors are smaller. They therefore combine by the square root of the sum of the squares procedure that applies to the standard deviations of all random variables, whether normally distributed or not. However, the 185 m worst-case range error due to uncompensated latency is an overestimate of the 95% value. If one knew both the distribution of relative headings and the distribution of uncompensated latencies over all intruders, one could calculate a value for two standard deviations of the overall distribution that was smaller than the worst-case of 185 m. However, the distribution of uncompensated latencies is unknown, and so a worst-case assumption would be needed there in any case, and one would have to assume something like a uniform distribution for the distribution of relative headings. It is much simpler to use the worst-case value as the two standard deviation value. It provides extra margin, and still allows the safety analysis in RTCA/DO-300A Appendix D to be carried through successfully.

E.6 Uncompensated Latency in Ownship Passive Position When Received by ACAS X

The uncompensated latency in the position report for ownship when reported to ACAS X is assumed to be 0.25 s or less. Since that quantity is determined by systems outside the scope of this MOPS, no requirement for it can be stated in this MOPS, and it does not appear to be constrained by any other specifications. If the actual value in a proposed system will exceed the assumed value by a substantial amount under realistic operational loads, this analysis should be revisited. At the maximum aircraft ground speed of 600 kt assumed in the ACAS X design an uncompensated latency of 0.25 s corresponds to a distance of 77 m. As with the intruder's uncompensated latency, the worst-case error is used as if it represents two standard deviations of the error distribution, even though it is an overestimate of that value.

E.7 Error Budget Allocated For Hybrid Surveillance Data Processing

The allocation of 145 m to Hybrid Surveillance data processing errors during validation is somewhat arbitrary. It is the same value that was used in the original RTCA/DO-300 and in RTCA/DO-300A. It was calculated there as the residual when errors outside the control of Hybrid Surveillance were subtracted from the 200 m validation range tolerance that had been a longstanding parameter of the Hybrid Surveillance design and that was mandated in the ICAO SARPS. As in the revised RTCA/DO-300A analysis, it is simply assumed and added to the errors outside the control of ACAS X Hybrid Surveillance to get a new value for the validation range tolerance.

E.8 Sample Allocation of Hybrid Data Processing Errors

The hybrid data processing error budget is 145 m (see Table E-1). Table E-2 shows some ways of how the error budget could be allocated.

Table E-2: Sample Allocation of Hybrid Data Processing Errors

Hybrid Data Processing Error	Sample Allocations		
Time Difference Between ACAS X Active Range Measurement and Intruder Position Data	450 msec = 140 m	400 msec = 124 m	300 msec = 93 m
Time Difference Between ACAS X Active Range Measurement and Ownship Position Data	0 m (ownship data extrapolated)	200 msec = 62 m	100 msec = 31 m
Approximation in Calculating Slant Range from Position	0 m (perfect calculation)	0 m (perfect calculation)	95 m (spherical Earth approx. at 60 deg latitude)
Total (Assuming Errors Root Sum Square)	140 m	140 m	136 m

FAR 25.1303

APPENDIX F TEST SUITE ENCOUNTER LIST

Table F-1: Test Group 10

Encounter	Test Group	Mode C	Mode S	ADS-B	ADS-R	Intruder RI	RA (Label 270)	Coordination	Designation/Xo	Multi-Threat	Prescriptive	Aural	Description
1000	10	X				3	5201 4010 4020 4310 5401			X	X	X	RA combinations from MOPS Table 2-49: Advisory 1 CrossingFlag=0 Label270=4010 MTE=1 ARASingleIntent=1 ARAUpDown=0 ARASTrength=4 ARACrossing=0; Advisory 5 CrossingFlag=0 Label270=4020 MTE=1 ARASingleIntent=1 ARAUpDown=0 ARASTrength=3 ARACrossing=0; Advisory 5 CrossingFlag=0 Label270=4020 MTE=0 ARASingleIntent=1 ARAUpDown=0 ARASTrength=3 ARACrossing=0; Advisory 10 CrossingFlag=0 Label270=5201 MTE=1 ARASingleIntent=0 ARAUpDown=1 ARASTrength=8 ARACrossing=0; Advisory 11 CrossingFlag=1 Label270=4310 MTE=1 ARASingleIntent=1 ARAUpDown=0 ARASTrength=6 ARACrossing=1; Advisory 11 CrossingFlag=0 Label270=4310 MTE=1 ARASingleIntent=1 ARAUpDown=0 ARASTrength=6 ARACrossing=0; Advisory 14 CrossingFlag=0 Label270=5401 MTE=1 ARASingleIntent=0 ARAUpDown=1 ARASTrength=7 ARACrossing=0; other advisories may also occur in this encounter.
1001	10	X				3	5001 4020 6020 6422 5201			X	X	X	RA combinations from MOPS Table 2-49: Advisory 2 CrossingFlag=0 Label270=5001 MTE=1 ARASingleIntent=1 ARAUpDown=1 ARASTrength=4 ARACrossing=0; Advisory 2 CrossingFlag=0 Label270=5001 MTE=0 ARASingleIntent=1 ARAUpDown=1 ARASTrength=4 ARACrossing=0; Advisory 5 CrossingFlag=0 Label270=4020 MTE=1 ARASingleIntent=0 ARAUpDown=0 ARASTrength=3 ARACrossing=0; Advisory 8 CrossingFlag=0 Label270=5201 MTE=1 ARASingleIntent=1 ARAUpDown=1 ARASTrength=5 ARACrossing=0; Advisory 22 CrossingFlag=0 Label270=6020 MTE=1 ARASingleIntent=0 ARAUpDown=0 ARASTrength=1 ARACrossing=0; Advisory 24 CrossingFlag=0

Appendix F
F-2

Encounter	Test Group	Mode C	Mode S	ADS-B	ADS-R	Intruder RI	RA (Label 270)	Coordination	Designation/Xo	Multi-Threat	Prescriptive	Aural	Description
													Label270=6422 MTE=1 ARASingleIntent=0 ARAUpDown=0 ARASTrength=14 ARACrossing=0; other advisories may also occur in this encounter.
1002	10	X				3	5401 4010 4020 6002 4022			X	X		RA combinations from MOPS Table 2-49: Advisory 1 CrossingFlag=0 Label270=4010 MTE=0 ARASingleIntent=1 ARAUpDown=0 ARASTrength=4 ARACrossing=0; Advisory 14 CrossingFlag=0 Label270=5401 MTE=1 ARASingleIntent=1 ARAUpDown=1 ARASTrength=7 ARACrossing=0; Advisory 20 CrossingFlag=0 Label270=6002 MTE=1 ARASingleIntent=1 ARAUpDown=1 ARASTrength=1 ARACrossing=0; Advisory 23 CrossingFlag=0 Label270=6002 MTE=1 ARASingleIntent=1 ARAUpDown=1 ARASTrength=1 ARACrossing=0; Advisory 23 CrossingFlag=0 Label270=6002 MTE=0 ARASingleIntent=1 ARAUpDown=1 ARASTrength=1 ARACrossing=0; Advisory 25 CrossingFlag=0 Label270=4022 MTE=1 ARASingleIntent=0 ARAUpDown=0 ARASTrength=15 ARACrossing=0; other advisories may also occur in this encounter.
1003	10	X				0 3	1000 5001 6002			X			RA combinations from MOPS Table 2-49: Advisory 21 CrossingFlag=0 Label270=6002 MTE=0 ARASingleIntent=1 ARAUpDown=1 ARASTrength=10 ARACrossing=0; other advisories may also occur in this encounter.
1004	10	X				3	5201 4010 4022 5001			X	X	X	RA combinations from MOPS Table 2-49: Advisory 1 CrossingFlag=0 Label270=4010 MTE=1 ARASingleIntent=0 ARAUpDown=0 ARASTrength=4 ARACrossing=0; other advisories may also occur in this encounter.
1005	10	X				3	5001 4010 4210 5022 4110			X	X	X	RA combinations from MOPS Table 2-49: Advisory 2 CrossingFlag=0 Label270=5001 MTE=1 ARASingleIntent=0 ARAUpDown=1 ARASTrength=4 ARACrossing=0; Advisory 3 CrossingFlag=1 Label270=4110 MTE=1 ARASingleIntent=1 ARAUpDown=0 ARASTrength=4 ARACrossing=1; Advisory 7 CrossingFlag=1 Label270=4210 MTE=1 ARASingleIntent=1 ARAUpDown=0 ARASTrength=5 ARACrossing=1; Advisory 26 CrossingFlag=0 Label270=5022

Encounter	Test Group	Mode C	Mode S	ADS-B	ADS-R	Intruder RI	RA (Label 270)	Coordination	Designation/Xo	Multi-Threat	Prescriptive	Aural	Description
													MTE=1 ARASingleIntent=0 ARAUpDown=1 ARASTrength=15 ARACrossing=0; other advisories may also occur in this encounter.
1006	10	X				3	5002 4210 4310			X	X		RA combinations from MOPS Table 2-49: Advisory 6 CrossingFlag=0 Label270=5002 MTE=1 ARASingleIntent=1 ARAUpDown=1 ARASTrength=3 ARACrossing=0; Advisory 6 CrossingFlag=0 Label270=5002 MTE=0 ARASingleIntent=1 ARAUpDown=1 ARASTrength=3 ARACrossing=0; Advisory 7 CrossingFlag=1 Label270=4210 MTE=1 ARASingleIntent=0 ARAUpDown=0 ARASTrength=5 ARACrossing=1; Advisory 11 CrossingFlag=1 Label270=4310 MTE=1 ARASingleIntent=0 ARAUpDown=0 ARASTrength=6 ARACrossing=1; other advisories may also occur in this encounter.
1007	10	X				3	5001 5002 4410 5101 4210			X	X		RA combinations from MOPS Table 2-49: Advisory 4 CrossingFlag=1 Label270=5101 MTE=1 ARASingleIntent=1 ARAUpDown=1 ARASTrength=4 ARACrossing=1; Advisory 4 CrossingFlag=1 Label270=5101 MTE=0 ARASingleIntent=1 ARAUpDown=1 ARASTrength=4 ARACrossing=1; Advisory 9 CrossingFlag=0 Label270=4210 MTE=0 ARASingleIntent=1 ARAUpDown=0 ARASTrength=8 ARACrossing=0; Advisory 13 CrossingFlag=0 Label270=4410 MTE=0 ARASingleIntent=1 ARAUpDown=0 ARASTrength=7 ARACrossing=0; other advisories may also occur in this encounter.
1008	10	X				3	5001 4020 5201			X	X		RA combinations from MOPS Table 2-49: Advisory 8 CrossingFlag=0 Label270=5201 MTE=0 ARASingleIntent=1 ARAUpDown=1 ARASTrength=5 ARACrossing=0; Advisory 27 CrossingFlag=0 Label270=4020 MTE=1 ARASingleIntent=0 ARAUpDown=0 ARASTrength=9 ARACrossing=0; other advisories may also occur in this encounter.
1009	10	X				3	1000 5201 4010			X	X		RA combinations from MOPS Table 2-49: Advisory 10 CrossingFlag=1 Label270=5201 MTE=1 ARASingleIntent=0 ARAUpDown=1 ARASTrength=8 ARACrossing=1; Advisory 12 CrossingFlag=0 Label270=5301 MTE=0 ARASingleIntent=1 ARAUpDown=1 ARASTrength=6 ARACrossing=0; Advisory

Appendix F
F-4

Encounter	Test Group	Mode C	Mode S	ADS-B	ADS-R	Intruder RI	RA (Label 270)	Coordination	Designation/Xo	Multi-Threat	Prescriptive	Aural	Description
							5301 4110						12 CrossingFlag=0 Label270=5301 MTE=1 ARASingleIntent=0 ARAUpDown=1 ARASTrength=6 ARACrossing=0; other advisories may also occur in this encounter.
1010	10		X			3	1000 5001 4010				X	X	RA combinations from MOPS Table 2-49: Advisory 27 CrossingFlag=0 Label270=4020 MTE=0 ARASingleIntent=1 ARAUpDown=1 ARASTrength=9 ARACrossing=0 (Duplicate); other advisories may also occur in this encounter.
1011	10		X			3	5001 4210 5301 5022 4010			X	X		RA combinations from MOPS Table 2-49: Advisory 7 CrossingFlag=0 Label270=4210 MTE=1 ARASingleIntent=1 ARAUpDown=0 ARASTrength=5 ARACrossing=0; other advisories may also occur in this encounter.
1012	10		X			3	5001 5002			X	X		RA combinations from MOPS Table 2-49: Advisory 19 CrossingFlag=0 Label270=5002 MTE=1 ARASingleIntent=1 ARAUpDown=1 ARASTrength=2 ARACrossing=0; Advisory 19 CrossingFlag=0 Label270=5002 MTE=0 ARASingleIntent=1 ARAUpDown=1 ARASTrength=2 ARACrossing=0; other advisories may also occur in this encounter.
1013	10		X			3	5201 4010 6002 5401				X	X	RA combinations from MOPS Table 2-49: Advisory 10 CrossingFlag=0 Label270=5201 MTE=0 ARASingleIntent=1 ARAUpDown=1 ARASTrength=8 ARACrossing=0; Advisory 14 CrossingFlag=0 Label270=5401 MTE=0 ARASingleIntent=1 ARAUpDown=1 ARASTrength=7 ARACrossing=0; Advisory 20 CrossingFlag=0 Label270=6002 MTE=0 ARASingleIntent=1 ARAUpDown=1 ARASTrength=1 ARACrossing=0; other advisories may also occur in this encounter.
1014	10		X			3	5201 4010			X	X		RA combinations from MOPS Table 2-49: Advisory 8 CrossingFlag=1 Label270=5201 MTE=1 ARASingleIntent=0 ARAUpDown=1 ARASTrength=5

Encounter	Test Group	Mode C	Mode S	ADS-B	ADS-R	Intruder RI	RA (Label 270)	Coordination	Designation/Xo	Multi-Threat	Prescriptive	Aural	Description
							4020 5301						ARACrossing=1; Advisory 12 CrossingFlag=1 Label270=5301 MTE=1 ARASingleIntent=1 ARAUpDown=1 ARASTrength=6 ARACrossing=1; Advisory 12 CrossingFlag=0 Label270=5301 MTE=1 ARASingleIntent=1 ARAUpDown=1 ARASTrength=6 ARACrossing=0; Advisory 12 CrossingFlag=1 Label270=5301 MTE=1 ARASingleIntent=0 ARAUpDown=1 ARASTrength=6 ARACrossing=1; other advisories may also occur in this encounter.
1015	10	X				3	5002 4410 4210				X		RA combinations from MOPS Table 2-49: Advisory 9 CrossingFlag=1 Label270=4210 MTE=0 ARASingleIntent=1 ARAUpDown=0 ARASTrength=8 ARACrossing=1; Advisory 15 CrossingFlag=1 Label270=4410 MTE=0 ARASingleIntent=1 ARAUpDown=0 ARASTrength=7 ARACrossing=1; other advisories may also occur in this encounter.
1016	10	X	X			2 3	1000 6002	X		X	X		RA combinations from MOPS Table 2-49: Advisory 23 CrossingFlag=1 Label270=6002 MTE=1 ARASingleIntent=1 ARAUpDown=1 ARASTrength=1 ARACrossing=1; Advisory 23 CrossingFlag=1 Label270=6002 MTE=0 ARASingleIntent=1 ARAUpDown=1 ARASTrength=1 ARACrossing=1; Advisory 23 CrossingFlag=0 Label270=6002 MTE=1 ARASingleIntent=0 ARAUpDown=1 ARASTrength=1 ARACrossing=0; other advisories may also occur in this encounter.
1017	10	X				3	1000 4010 5002	X			X	X	RA combinations from MOPS Table 2-49: Advisory 6 CrossingFlag=1 Label270=5002 MTE=0 ARASingleIntent=1 ARAUpDown=1 ARASTrength=3 ARACrossing=1; Advisory 28 CrossingFlag=1 Label270=5002 MTE=0 ARASingleIntent=1 ARAUpDown=1 ARASTrength=9 ARACrossing=1; other advisories may also occur in this encounter.
1018	10	X				3	5401 4210 4020 4110			X	X	X	RA combinations from MOPS Table 2-49: Advisory 3 CrossingFlag=1 Label270=4110 MTE=0 ARASingleIntent=1 ARAUpDown=0 ARASTrength=4 ARACrossing=1; Advisory 7 CrossingFlag=1 Label270=4210 MTE=0 ARASingleIntent=1 ARAUpDown=0 ARASTrength=5 ARACrossing=1; Advisory

Appendix F
F-6

Encounter	Test Group	Mode C	Mode S	ADS-B	ADS-R	Intruder RI	RA (Label 270)	Coordination	Designation/Xo	Multi-Threat	Prescriptive	Aural	Description
													17 CrossingFlag=1 Label270=4020 MTE=1 ARASingleIntent=1 ARAUpDown=0 ARASTrength=2 ARACrossing=1; other advisories may also occur in this encounter.
1019	10		X			3	5001 4210 5301 4010			X	X		RA combinations from MOPS Table 2-49: Advisory 7 CrossingFlag=0 Label270=4210 MTE=1 ARASingleIntent=0 ARAUpDown=0 ARASTrength=5 ARACrossing=0; other advisories may also occur in this encounter.
1020	10		X	X		3	1000 5201 4020 5401	X		X	X		RA combinations from MOPS Table 2-49: Advisory 10 CrossingFlag=1 Label270=5201 MTE=0 ARASingleIntent=1 ARAUpDown=1 ARASTrength=8 ARACrossing=1; Advisory 16 CrossingFlag=1 Label270=5401 MTE=0 ARASingleIntent=1 ARAUpDown=1 ARASTrength=7 ARACrossing=1; Advisory 16 CrossingFlag=1 Label270=5401 MTE=1 ARASingleIntent=0 ARAUpDown=1 ARASTrength=7 ARACrossing=1; other advisories may also occur in this encounter.
1021	10		X			3	1000 4020			X			RA combinations from MOPS Table 2-49: Advisory 5 CrossingFlag=1 Label270=4020 MTE=0 ARASingleIntent=1 ARAUpDown=0 ARASTrength=3 ARACrossing=1; other advisories may also occur in this encounter.
1022	10		X			3	6002 5101	X		X	X		RA combinations from MOPS Table 2-49: Advisory 21 CrossingFlag=1 Label270=6002 MTE=0 ARASingleIntent=1 ARAUpDown=1 ARASTrength=10 ARACrossing=1; other advisories may also occur in this encounter.
1023	10		X			3	1000 4010 6002	X		X			RA combinations from MOPS Table 2-49: Advisory 29 CrossingFlag=1 Label270=6002 MTE=0 ARASingleIntent=1 ARAUpDown=1 ARASTrength=11 ARACrossing=1; other advisories may also occur in this encounter.
1024	10		X			3	6002 4020 4022			X	X		RA combinations from MOPS Table 2-49: Advisory 23 CrossingFlag=1 Label270=6002 MTE=1 ARASingleIntent=0 ARAUpDown=1 ARASTrength=1 ARACrossing=1; Advisory 25 CrossingFlag=1 Label270=4022 MTE=1

Encounter	Test Group	Mode C	Mode S	ADS-B	ADS-R	Intruder RI	RA (Label 270)	Coordination	Designation/Xo	Multi-Threat	Prescriptive	Aural	Description
													ARASingleIntent=0 ARAUpDown=0 ARASTrength=15 ARACrossing=1; other advisories may also occur in this encounter.
1025	10	X				3	1000 4010 4020	X			X		RA combinations from MOPS Table 2-49: Advisory 17 CrossingFlag=0 Label270=4020 MTE=0 ARASingleIntent=1 ARAUpDown=0 ARASTrength=2 ARACrossing=0; other advisories may also occur in this encounter.
1026	10	X				3	1000 4410 6020				X	X	RA combinations from MOPS Table 2-49: Advisory 18 CrossingFlag=0 Label270=6020 MTE=0 ARASingleIntent=1 ARAUpDown=0 ARASTrength=1 ARACrossing=0; Advisory 22 CrossingFlag=0 Label270=6020 MTE=0 ARASingleIntent=1 ARAUpDown=0 ARASTrength=1 ARACrossing=0; other advisories may also occur in this encounter.
1027	10	X				3	5201 4010 4110 4022 5401			X	X		RA combinations from MOPS Table 2-49: Advisory 3 CrossingFlag=1 Label270=4110 MTE=1 ARASingleIntent=0 ARAUpDown=0 ARASTrength=4 ARACrossing=1; Advisory 10 CrossingFlag=1 Label270=5201 MTE=1 ARASingleIntent=1 ARAUpDown=1 ARASTrength=8 ARACrossing=1; other advisories may also occur in this encounter.
1028	10	X				3	5201 4010 4020 5001			X	X		RA combinations from MOPS Table 2-49: Advisory 8 CrossingFlag=0 Label270=5201 MTE=1 ARASingleIntent=0 ARAUpDown=1 ARASTrength=5 ARACrossing=0; other advisories may also occur in this encounter.
1029	10	X				3	5002 5101			X	X		RA combinations from MOPS Table 2-49: Advisory 6 CrossingFlag=1 Label270=5002 MTE=1 ARASingleIntent=1 ARAUpDown=1 ARASTrength=3 ARACrossing=1; Advisory 19 CrossingFlag=1 Label270=5002 MTE=0 ARASingleIntent=1 ARAUpDown=1 ARASTrength=2 ARACrossing=1; Advisory 19 CrossingFlag=1 Label270=5002 MTE=1 ARASingleIntent=0 ARAUpDown=1 ARASTrength=2 ARACrossing=1; other advisories may also occur in this encounter.

Appendix F
F-8

Encounter	Test Group	Mode C	Mode S	ADS-B	ADS-R	Intruder RI	RA (Label 270)	Coordination	Designation/Xo	Multi-Threat	Prescriptive	Aural	Description
1030	10		X			3	4410 4310				X		RA combinations from MOPS Table 2-49: Advisory 11 CrossingFlag=1 Label270=4310 MTE=0 ARASingleIntent=1 ARAUpDown=0 ARAStrength=6 ARACrossing=1; other advisories may also occur in this encounter.
1031	10		X			3	4010 6020 4410				X	X	RA combinations from MOPS Table 2-49: Advisory 13 CrossingFlag=0 Label270=4410 MTE=1 ARASingleIntent=1 ARAUpDown=0 ARAStrength=7 ARACrossing=0; Advisory 15 CrossingFlag=1 Label270=4410 MTE=1 ARASingleIntent=1 ARAUpDown=0 ARAStrength=7 ARACrossing=1; Advisory 18 CrossingFlag=0 Label270=6020 MTE=1 ARASingleIntent=1 ARAUpDown=0 ARAStrength=1 ARACrossing=0; Advisory 22 CrossingFlag=0 Label270=6020 MTE=1 ARASingleIntent=1 ARAUpDown=0 ARAStrength=1 ARACrossing=0; other advisories may also occur in this encounter.
1032	10		X			3	5201 4010 4020 4022 5401			X	X		RA combinations from MOPS Table 2-49: Advisory 5 CrossingFlag=1 Label270=4020 MTE=1 ARASingleIntent=1 ARAUpDown=0 ARAStrength=3 ARACrossing=1; Advisory 10 CrossingFlag=0 Label270=5201 MTE=1 ARASingleIntent=1 ARAUpDown=1 ARAStrength=8 ARACrossing=0; other advisories may also occur in this encounter.
1033	10		X			3	5001 5002 4410 4210			X	X		RA combinations from MOPS Table 2-49: Advisory 9 CrossingFlag=0 Label270=4210 MTE=1 ARASingleIntent=1 ARAUpDown=0 ARAStrength=8 ARACrossing=0; Advisory 19 CrossingFlag=0 Label270=5002 MTE=1 ARASingleIntent=0 ARAUpDown=1 ARAStrength=2 ARACrossing=0; other advisories may also occur in this encounter.
1034	10		X			3	5002 4410 4210			X	X		RA combinations from MOPS Table 2-49: Advisory 9 CrossingFlag=1 Label270=4210 MTE=1 ARASingleIntent=0 ARAUpDown=0 ARAStrength=8 ARACrossing=1; Advisory 15 CrossingFlag=1 Label270=4410 MTE=1 ARASingleIntent=0 ARAUpDown=0 ARAStrength=7 ARACrossing=1; other advisories may also occur in this encounter.

Encounter	Test Group	Mode C	Mode S	ADS-B	ADS-R	Intruder RI	RA (Label 270)	Coordination	Designation/Xo	Multi-Threat	Prescriptive	Aural	Description
1035	10	X				3	4310 5001 5002 5101 4210			X	X		RA combinations from MOPS Table 2-49: Advisory 9 CrossingFlag=0 Label270=4210 MTE=1 ARASingleIntent=0 ARAUpDown=0 ARAStrength=8 ARACrossing=0; Advisory 11 CrossingFlag=0 Label270=4310 MTE=0 ARASingleIntent=1 ARAUpDown=0 ARAStrength=6 ARACrossing=0; Advisory 11 CrossingFlag=0 Label270=4310 MTE=1 ARASingleIntent=0 ARAUpDown=0 ARAStrength=6 ARACrossing=0; other advisories may also occur in this encounter.
1036	10	X				3	5201 4010 5101 4310 5001				X		RA combinations from MOPS Table 2-49: Advisory 8 CrossingFlag=1 Label270=5201 MTE=0 ARASingleIntent=1 ARAUpDown=1 ARAStrength=5 ARACrossing=1; other advisories may also occur in this encounter.
1037	10	X				3	1000 5002 5101			X	X		RA combinations from MOPS Table 2-49: Advisory 19 CrossingFlag=1 Label270=5002 MTE=1 ARASingleIntent=1 ARAUpDown=1 ARAStrength=2 ARACrossing=1; other advisories may also occur in this encounter.
1038	10	X				3	4010 4020 4110			X	X		RA combinations from MOPS Table 2-49: Advisory 17 CrossingFlag=0 Label270=4020 MTE=1 ARASingleIntent=0 ARAUpDown=0 ARAStrength=2 ARACrossing=0; other advisories may also occur in this encounter.
1039	10	X				3	4110			X	X		RA combinations from MOPS Table 2-49: Advisory 17 CrossingFlag=0 Label270=4020 MTE=1 ARASingleIntent=1 ARAUpDown=0 ARAStrength=2 ARACrossing=0; Advisory 17 CrossingFlag=1 Label270=4020 MTE=0 ARASingleIntent=1 ARAUpDown=0 ARAStrength=2 ARACrossing=1; other advisories may also occur in this encounter.
1040	10	X				3	5201 4010 5101			X	X		RA combinations from MOPS Table 2-49: Advisory 4 CrossingFlag=1 Label270=5101 MTE=1 ARASingleIntent=0 ARAUpDown=1 ARAStrength=4 ARACrossing=1; Advisory 24 CrossingFlag=1 Label270=6422 MTE=1

Appendix F
F-10

Encounter	Test Group	Mode C	Mode S	ADS-B	ADS-R	Intruder RI	RA (Label 270)	Coordination	Designation/Xo	Multi-Threat	Prescriptive	Aural	Description
							6422 4110						ARASingleIntent=0 ARAUpDown=0 ARASTrength=14 ARACrossing=1; other advisories may also occur in this encounter.
1041	10		X			3	5002 4410 4210			X	X		RA combinations from MOPS Table 2-49: Advisory 6 CrossingFlag=0 Label270=5002 MTE=1 ARASingleIntent=0 ARAUpDown=1 ARASTrength=3 ARACrossing=0; other advisories may also occur in this encounter.
1042	10		X			3	5002 4410 6020 4210 5022			X	X		RA combinations from MOPS Table 2-49: Advisory 13 CrossingFlag=0 Label270=4410 MTE=1 ARASingleIntent=0 ARAUpDown=0 ARASTrength=7 ARACrossing=0; Advisory 26 CrossingFlag=1 Label270=5022 MTE=1 ARASingleIntent=0 ARAUpDown=1 ARASTrength=15 ARACrossing=1; other advisories may also occur in this encounter.
1043	10		X			3	5101 5301			X	X		RA combinations from MOPS Table 2-49: Advisory 12 CrossingFlag=1 Label270=5301 MTE=0 ARASingleIntent=1 ARAUpDown=1 ARASTrength=6 ARACrossing=1; other advisories may also occur in this encounter.
1044	10		X			3	5401 6020 4022			X	X		RA combinations from MOPS Table 2-49: Advisory 22 CrossingFlag=1 Label270=6020 MTE=0 ARASingleIntent=1 ARAUpDown=0 ARASTrength=1 ARACrossing=1; other advisories may also occur in this encounter.
1045	10		X			3	1000 5001 5401			X	X	X	RA combinations from MOPS Table 2-49: Advisory 16 CrossingFlag=1 Label270=5401 MTE=1 ARASingleIntent=1 ARAUpDown=1 ARASTrength=7 ARACrossing=1; other advisories may also occur in this encounter.
1046	10		X			3	5001 5002 5101 4210 4310 5022			X	X		RA combinations from MOPS Table 2-49: Advisory 9 CrossingFlag=1 Label270=4210 MTE=1 ARASingleIntent=1 ARAUpDown=0 ARASTrength=8 ARACrossing=1; other advisories may also occur in this encounter.

Encounter	Test Group	Mode C	Mode S	ADS-B	ADS-R	Intruder RI	RA (Label 270)	Coordination	Designation/Xo	Multi-Threat	Prescriptive	Aural	Description
1047	10	X				3	5201 5002 6020 5101 5022			X	X		RA combinations from MOPS Table 2-49: Advisory 8 CrossingFlag=1 Label270=5201 MTE=1 ARASingleIntent=1 ARAUpDown=1 ARAStrength=5 ARACrossing=1; other advisories may also occur in this encounter.
1048	10	X				3	1000 5001 4010 6002				X		RA combinations from MOPS Table 2-49: Advisory 29 CrossingFlag=0 Label270=6002 MTE=0 ARASingleIntent=1 ARAUpDown=1 ARAStrength=11 ARACrossing=0; other advisories may also occur in this encounter.
1049	10	X				3	5001 4010 5002				X		RA combinations from MOPS Table 2-49: Advisory 28 CrossingFlag=0 Label270=5002 MTE=0 ARASingleIntent=1 ARAUpDown=1 ARAStrength=9 ARACrossing=0; other advisories may also occur in this encounter.
1050	10	X				3	5001 4210 5301 5002 4010			X	X		RA combinations from MOPS Table 2-49: Advisory 7 CrossingFlag=0 Label270=4210 MTE=0 ARASingleIntent=1 ARAUpDown=0 ARAStrength=5 ARACrossing=0; Advisory 28 CrossingFlag=0 Label270=5002 MTE=1 ARASingleIntent=0 ARAUpDown=1 ARAStrength=9 ARACrossing=0; other advisories may also occur in this encounter.
1051	10	X				3	1000 5401 6002	X			X		RA combinations from MOPS Table 2-49: Advisory 20 CrossingFlag=1 Label270=6002 MTE=0 ARASingleIntent=1 ARAUpDown=1 ARAStrength=1 ARACrossing=1; other advisories may also occur in this encounter.
1052	10	X				3	5001 6020				X		RA combinations from MOPS Table 2-49: Advisory 30 CrossingFlag=0 Label270=6020 MTE=0 ARASingleIntent=1 ARAUpDown=0 ARAStrength=11 ARACrossing=0; other advisories may also occur in this encounter.
1053	10	X				3	5001 4010 6020			X	X		RA combinations from MOPS Table 2-49: Advisory 30 CrossingFlag=0 Label270=6020 MTE=1 ARASingleIntent=1 ARAUpDown=0 ARAStrength=11 ARACrossing=0; other advisories may also occur in this encounter.

Encounter	Test Group	Mode C	Mode S	ADS-B	ADS-R	Intruder RI	RA (Label 270)	Coordination	Designation/Xo	Multi-Threat	Prescriptive	Aural	Description
1054	10	X				3	5401 4210 4020 4022 4110			X	X	X	RA combinations from MOPS Table 2-49: Advisory 17 CrossingFlag=1 Label270=4020 MTE=1 ARASingleIntent=0 ARAUpDown=0 ARAStrength=2 ARACrossing=1; other advisories may also occur in this encounter.

Table F-2: Test Group 20

Encounter	Test Group	Mode C	Mode S	ADS-B	ADS-R	Intruder RI	RA (Label 270)	Coordination	Designation/Xo	Multi-Threat	Prescriptive	Aural	Description
2000	20	X				3	5001 6422			X	X		TID field for Multithreat: (#1 of 18) Verifies that most recent threat is displayed in the TID field of a multithreat encounter. Also verifies that the correct intruder is displayed in the TID field when the current threat is removed and the previously second threat becomes the main threat. All possible combinations of two intruders equipped with ACAS/TCAS, Mode S or Mode C, and with ownship being master or being slave. This encounter: CAS - CAS, ownship is master.
2001	20	X			0	3	5001 5101 6422			X	X		TID field for Multithreat: (#2 of 18) Verifies that most recent threat is displayed in the TID field of a multithreat encounter. Also verifies that the correct intruder is displayed in the TID field when the current threat is

Encounter	Test Group	Mode C	Mode S	ADS-B	ADS-R	Intruder RI	RA (Label 270)	Coordination	Designation/Xo	Multi-Threat	Prescriptive	Aural	Description
													removed and the previously second threat becomes the main threat. All possible combinations of two intruders equipped with ACAS/TCAS, Mode S or Mode C, and with ownship being master or being slave. This encounter: CAS - Mode S, ownship is master.
2002	20	X	X			3	5001 5101 6422			X	X		TID field for Multithreat: (#3 of 18) Verifies that most recent threat is displayed in the TID field of a multithreat encounter. Also verifies that the correct intruder is displayed in the TID field when the current threat is removed and the previously second threat becomes the main threat. All possible combinations of two intruders equipped with ACAS/TCAS, Mode S or Mode C, and with ownship being master or being slave. This encounter: CAS - Mode C, ownship is master.
2003	20		X			0 3	5001 6422			X	X		TID field for Multithreat: (#4 of 18) Verifies that most recent threat is displayed in the TID field of a multithreat encounter. Also verifies that the correct intruder is displayed in the TID field when the current threat is removed and the previously second threat becomes the main threat. All possible combinations of two intruders equipped with ACAS/TCAS, Mode S or Mode C, and with ownship being master or being slave. This encounter: Mode S - CAS, ownship is master.
2004	20		X			0	5001 5101			X	X		TID field for Multithreat: (#5 of 18) Verifies that most recent threat is displayed in the TID field of a multithreat encounter. Also verifies that the correct intruder is displayed in the TID field when the current threat is removed and the previously second threat becomes the main threat. All possible combinations of two intruders equipped with ACAS/TCAS, Mode S or Mode C, and with ownship being master or being slave. This encounter: Mode S - Mode S, ownship is master.
2005	20	X	X			0	5001 5101			X	X		TID field for Multithreat: (#6 of 18) Verifies that most recent threat is displayed in the TID field of a multithreat encounter. Also verifies that the correct intruder is displayed in the TID field when the current threat is

Appendix F
F-14

Encounter	Test Group	Mode C	Mode S	ADS-B	ADS-R	Intruder RI	RA (Label 270)	Coordination	Designation/Xo	Multi-Threat	Prescriptive	Aural	Description
													removed and the previously second threat becomes the main threat. All possible combinations of two intruders equipped with ACAS/TCAS, Mode S or Mode C, and with ownship being master or being slave. This encounter: Mode S - Mode C, ownship is master.
2006	20	X	X			3	5001 6422			X	X		TID field for Multithreat: (#7 of 18) Verifies that most recent threat is displayed in the TID field of a multithreat encounter. Also verifies that the correct intruder is displayed in the TID field when the current threat is removed and the previously second threat becomes the main threat. All possible combinations of two intruders equipped with ACAS/TCAS, Mode S or Mode C, and with ownship being master or being slave. This encounter: Mode C - CAS, ownship is master.
2007	20	X	X			0	5001 5101			X	X		TID field for Multithreat: (#8 of 18) Verifies that most recent threat is displayed in the TID field of a multithreat encounter. Also verifies that the correct intruder is displayed in the TID field when the current threat is removed and the previously second threat becomes the main threat. All possible combinations of two intruders equipped with ACAS/TCAS, Mode S or Mode C, and with ownship being master or being slave. This encounter: Mode C - Mode S, ownship is master.
2008	20	X					5001 5101			X	X		TID field for Multithreat: (#9 of 18) Verifies that most recent threat is displayed in the TID field of a multithreat encounter. Also verifies that the correct intruder is displayed in the TID field when the current threat is removed and the previously second threat becomes the main threat. All possible combinations of two intruders equipped with ACAS/TCAS, Mode S or Mode C, and with ownship being master or being slave. This encounter: Mode C - Mode C.
2009	20		X			3	5001 6422			X	X		TID field for Multithreat: (#10 of 18) Verifies that most recent threat is displayed in the TID field of a multithreat encounter. Also verifies that the

Encounter	Test Group	Mode C	Mode S	ADS-B	ADS-R	Intruder RI	RA (Label 270)	Coordination	Designation/Xo	Multi-Threat	Prescriptive	Aural	Description
													correct intruder is displayed in the TID field when the current threat is removed and the previously second threat becomes the main threat. All possible combinations of two intruders equipped with ACAS/TCAS, Mode S or Mode C, and with ownship being master or being slave. This encounter: CAS - CAS, ownship is slave.
2010	20	X				0 3	5001 5101 6422			X	X		TID field for Multithreat: (#11 of 18) Verifies that most recent threat is displayed in the TID field of a multithreat encounter. Also verifies that the correct intruder is displayed in the TID field when the current threat is removed and the previously second threat becomes the main threat. All possible combinations of two intruders equipped with ACAS/TCAS, Mode S or Mode C, and with ownship being master or being slave. This encounter: CAS - Mode S, ownship is slave.
2011	20	X	X			3	5001 5101 6422			X	X		TID field for Multithreat: (#12 of 18) Verifies that most recent threat is displayed in the TID field of a multithreat encounter. Also verifies that the correct intruder is displayed in the TID field when the current threat is removed and the previously second threat becomes the main threat. All possible combinations of two intruders equipped with ACAS/TCAS, Mode S or Mode C, and with ownship being master or being slave. This encounter: CAS - Mode C, ownship is slave.
2012	20		X			0 3	5001 6422			X	X		TID field for Multithreat: (#13 of 18) Verifies that most recent threat is displayed in the TID field of a multithreat encounter. Also verifies that the correct intruder is displayed in the TID field when the current threat is removed and the previously second threat becomes the main threat. All possible combinations of two intruders equipped with ACAS/TCAS, Mode S or Mode C, and with ownship being master or being slave. This encounter: Mode S - CAS, ownship is slave.
2013	20		X			0	5001 5101			X	X		TID field for Multithreat: (#14 of 18) Verifies that most recent threat is displayed in the TID field of a multithreat encounter. Also verifies that the

Appendix F
F-16

Encounter	Test Group	Mode C	Mode S	ADS-B	ADS-R	Intruder RI	RA (Label 270)	Coordination	Designation/Xo	Multi-Threat	Prescriptive	Aural	Description
													correct intruder is displayed in the TID field when the current threat is removed and the previously second threat becomes the main threat. All possible combinations of two intruders equipped with ACAS/TCAS, Mode S or Mode C, and with ownship being master or being slave. This encounter: Mode S - Mode S, ownship is slave.
2014	20	X	X			0	5001 5101			X	X		TID field for Multithreat: (#15 of 18) Verifies that most recent threat is displayed in the TID field of a multithreat encounter. Also verifies that the correct intruder is displayed in the TID field when the current threat is removed and the previously second threat becomes the main threat. All possible combinations of two intruders equipped with ACAS/TCAS, Mode S or Mode C, and with ownship being master or being slave. This encounter: Mode S - Mode C, ownship is slave.
2015	20	X	X			3	5001 6422			X	X		TID field for Multithreat: (#16 of 18) Verifies that most recent threat is displayed in the TID field of a multithreat encounter. Also verifies that the correct intruder is displayed in the TID field when the current threat is removed and the previously second threat becomes the main threat. All possible combinations of two intruders equipped with ACAS/TCAS, Mode S or Mode C, and with ownship being master or being slave. This encounter: Mode C - CAS, ownship is slave.
2016	20	X	X			0	5001 5101			X	X		TID field for Multithreat: (#17 of 18) Verifies that most recent threat is displayed in the TID field of a multithreat encounter. Also verifies that the correct intruder is displayed in the TID field when the current threat is removed and the previously second threat becomes the main threat. All possible combinations of two intruders equipped with ACAS/TCAS, Mode S or Mode C, and with ownship being master or being slave. This encounter: Mode C - Mode S, ownship is slave.

Encounter	Test Group	Mode C	Mode S	ADS-B	ADS-R	Intruder RI	RA (Label 270)	Coordination	Designation/Xo	Multi-Threat	Prescriptive	Aural	Description
2017	20	X					5001 5101			X	X		TID field for Multithreat: (#18 of 18) Verifies that most recent threat is displayed in the TID field of a multithreat encounter. Also verifies that the correct intruder is displayed in the TID field when the current threat is removed and the previously second threat becomes the main threat. All possible combinations of two intruders equipped with ACAS/TCAS, Mode S or Mode C, and with ownship being master or being slave. This encounter: Mode C - Mode C.

Table F-3: Test Group 30

Encounter	Test Group	Mode C	Mode S	ADS-B	ADS-R	Intruder RI	RA (Label 270)	Coordination	Designation/Xo	Multi-Threat	Prescriptive	Aural	Description
3100	31	X				3	5001 5301			X			Coordination Test Hazop01 (#1 of 4). Intruder is active CAS in TA/RA mode. Make sure RI field is set correctly to 3.
3101	31	X				3	5001 5301			X			Coordination Test Hazop01 (#2 of 4). Test shows that RI goes from 3 to 2 (transponder: {ri: 2} in STM report) when ownship descends below alt inhibit threshold. It also shows that RAs are not issued below threshold, but issued above. Test checks also that no uf16uds30 are generated below threshold.
3102	31	X				3	5001			X			Coordination Test Hazop01 (#3 of 4). Test shows that RI goes from 2 to 3 (transponder: {ri: 3} in STM report) when ownship climbs above altitude inhibit threshold. It also shows that RAs are issued above threshold, but

Appendix F
F-18

Encounter	Test Group	Mode C	Mode S	ADS-B	ADS-R	Intruder RI	RA (Label 270)	Coordination	Designation/Xo	Multi-Threat	Prescriptive	Aural	Description
													not issued below. Test checks also that no uf16uds30 are generated below threshold.
3103	31	X				3	5001				X		Coordination Test Hazop01 (#4 of 4). Test shows that RI goes from 2 to 3 (transponder: {ri: 2} in STM report) when ownship goes above altitude inhibit threshold or after 10 cycles of radio alt set to NaN (max radio alt exceeded). It also shows that RAs are issued after the 10 cycles of radio altitude set to NaN but not before, and that no uf16uds30 are generated before.
3110	31	X				3	5001 4210 5301				X		Coordination Test Hazop03 (#1 of 4). Tests switching from TA-only to TA/RA mode when altitude boundary was crossed and equipped intruder causes RA. Tests that ownship sends vrc 3 seconds after crossing altitude boundary of 1100 ft when there is no uf16uds30 received from intruder. Base encounter without radio altitude or uf16uds30 messages.
3111	31	X				3	5001 5002 4210 4010				X		Coordination Test Hazop03 (#2 of 4). Tests switching from TA to RA mode when altitude boundary was crossed and equipped intruder causes RA. Tests that ownship sends vrc 3 seconds after crossing altitude boundary of 1100 ft when there is no uf16uds30 received from intruder. (Ownship is master)
3112	31	X				3	5001 5002 4210 4010	X			X		Coordination Test Hazop03 (#3 of 4). Tests switching from TA to RA mode when altitude boundary was crossed and equipped intruder causes RA. Tests that ownship sends vrc immediately after crossing altitude boundary of 1100 ft when uf16uds30 received from intruder. (Ownship is master)
3113	31	X				3	5001 5002 4210 4010	X			X		Coordination Test Hazop03 (#4 of 4). Tests switching from TA to RA mode when altitude boundary was crossed and equipped intruder causes RA. Tests that ownship sends vrc immediately after crossing altitude

Encounter	Test Group	Mode C	Mode S	ADS-B	ADS-R	Intruder RI	RA (Label 270)	Coordination	Designation/Xo	Multi-Threat	Prescriptive	Aural	Description
													boundary of 1100 ft when uf16uds30 received from intruder. (similar to #3 but ownship is slave)
3120	31	X				3	5001 4210 5301				X		Coordination Test Hazop04 (#1 of 9). Intruder causing RA will be designated (DNA). Test verifies that ownship sends a vrc only after receiving a vrc from a TCAS II slave intruder. This encounter is the base encounter without DNA designation and without uf16uds30 coordination message.
3121	31	X				3			X		X		Coordination Test Hazop04 (#2 of 9). Intruder causing RA has been designated (DNA). Test verifies that ownship sends a vrc only after receiving a vrc from a TCAS II slave intruder. Intruder is TCAS II, slave. No uf16uds30 from intruder has been received. Test shows that ownship doesn't send a coordination message (vrc from ownship in TRM report is not populated).
3122	31	X				3		X	X		X		Coordination Test Hazop04 (#3 of 9). Intruder causing RA has been designated (DNA). Test verifies that ownship sends a vrc only after receiving a vrc from a TCAS II slave intruder. Intruder is TCAS II, slave. A uf16uds30 from intruder has been received. Test shows that ownship sends coordination message (vrc from ownship in TRM report is populated).
3123	31	X				3			X		X		Coordination Test Hazop04 (#4 of 9). Intruder causing RA has been designated (DNA). Test verifies that ownship sends a vrc only after receiving a vrc from a TCAS II slave intruder. Intruder is ACAS X, slave. No uf16uds30 from intruder has been received. Test shows that ownship doesn't send a coordination message (vrc from ownship in TRM report is not populated).
3124	31	X				3		X	X		X		Coordination Test Hazop04 (#5 of 9). Intruder causing RA has been designated (DNA). Test verifies that ownship sends a vrc only after receiving a vrc from a TCAS II slave intruder. Intruder is ACAS X, slave. A

Appendix F
F-20

Encounter	Test Group	Mode C	Mode S	ADS-B	ADS-R	Intruder RI	RA (Label 270)	Coordination	Designation/Xo	Multi-Threat	Prescriptive	Aural	Description
													uf16uds30 from intruder has been received. Test shows that ownship doesn't send a coordination message (vrc from ownship in TRM report is not populated).
3125	31	X			3			X		X			Coordination Test Hazop04 (#6 of 9). Intruder causing RA has been designated (DNA). Test verifies that ownship sends a vrc only after receiving a vrc from a TCAS II intruder. Intruder is TCAS II, master. No uf16uds30 from intruder has been received. Test shows that ownship doesn't send a coordination message (vrc from ownship in TRM report is not populated).
3126	31	X			3		X	X		X			Coordination Test Hazop04 (#7 of 9). Intruder causing RA has been designated (DNA). Test verifies that ownship sends a vrc only after receiving a vrc from a TCAS II slave intruder. Intruder is TCAS II, master. A uf16uds30 from intruder has been received. Test shows that ownship doesn't send a coordination message (vrc from ownship in TRM report is not populated).
3127	31	X			3			X		X			Coordination Test Hazop04 (#8 of 9). Intruder causing RA has been designated (DNA). Test verifies that ownship sends a vrc only after receiving a vrc from a TCAS II slave intruder. Intruder is ACAS X, master. No uf16uds30 from intruder has been received. Test shows that ownship doesn't send a coordination message (vrc from ownship in TRM report is not populated).
3128	31	X			3		X	X		X			Coordination Test Hazop04 (#9 of 9). Intruder causing RA has been designated (DNA). Test verifies that ownship sends a vrc only after receiving a vrc from a TCAS II slave intruder. Intruder is ACAS X, master. A uf16uds30 from intruder has been received. Test shows that ownship doesn't send a coordination message (vrc from ownship in TRM report is not populated).

Encounter	Test Group	Mode C	Mode S	ADS-B	ADS-R	Intruder RI	RA (Label 270)	Coordination	Designation/Xo	Multi-Threat	Prescriptive	Aural	Description
3130	31	X				0	1000 5001 5301			X			Coordination Test Hazop21 (#1 of 5). Ownship is slave, chooses an RA and wants to reverse RA before any uf16uds30 was received from intruder. Base encounter straight line with unequipped intruder.
3131	31	X				3	1000 5001 5301			X			Coordination Test Hazop21 (#2 of 5). Ownship is slave, chooses an RA and wants to reverse RA before any uf16uds30 was received from intruder. Base encounter straight line with equipped intruder.
3132	31	X				3	1000 5001 4210 4010			X			Coordination Test Hazop21 (#3 of 5). Ownship is slave, chooses an RA and wants to reverse RA before any uf16uds30 was received from intruder. Base encounter with equipped intruder straight line then descending then climbing. Test shows that slave (ownship) can reverse when not getting any coordination message from master.
3133	31	X				3	1000 5001 4210 5101 4010	X		X			Coordination Test Hazop21 (#4 of 5). Ownship is slave, chooses an RA and wants to reverse RA before any uf16uds30 was received from intruder. Base encounter as #3 plus uf16uds30. Test shows that slave (ownship) cannot reverse due to intruder sending uf16uds30, but reverses after intruder sends cancellation uf16uds30.
3134	31	X				2 3	1000 5001 4210 5101 4010	X		X			Coordination Test Hazop21 (#5 of 5). Ownship is slave, chooses an RA and wants to reverse RA before any uf16uds30 was received from intruder. Base encounter as #3 plus uf16uds30 and switching input RI. Test shows that slave (ownship) cannot reverse due to intruder sending uf16uds30, but reverses after intruder sends RI=2, even though intruder continues to send same uf16uds30.
3140	31	X				3	1000 5001 5301			X			Coordination Test Hazop27 (#1 of 2). Test verifies that slave reverses an RA right after receiving an incompatible uf16uds30 from the master. Ownship is slave. Base encounter without uf16uds30. Coordination Test Hazop31 (#1 of 2). Verifying that when an intruder is no longer a threat to ownship (goes from RA to COC) then ownship sends a cancellation message (cvc).

Appendix F
F-22

Encounter	Test Group	Mode C	Mode S	ADS-B	ADS-R	Intruder RI	RA (Label 270)	Coordination	Designation/Xo	Multi-Threat	Prescriptive	Aural	Description
3141	31	X				3	1000 5001 4210 4010 5201	X		X			Coordination Test Hazop27 (#2 of 2). Test verifies that slave reverses an RA right after receiving an incompatible uf16uds30 from the master. Ownership is slave. Encounter shows reversal right after conflicting uf16uds30 has been received. Coordination Test Hazop31 (#2 of 2). Ownership is receiving uf16uds30 with vrc 1 or 2 from the intruder, then intruder track continues but uf16uds30 suddenly stop. Verifying that the STM clears out the vrc for that intruder after 6 seconds.
3150	31	X				3	1000 5001 4210 4010			X			Coordination Test Hazop28 (#1 of 4). Reversal with ownership is master. Verifies that when reversing, the ownership output vrc changes and the cvc is set for the old value. The cvc has to remain that value until the RA is terminated or encounter ends. Descend to climb.
3151	31	X				3	1000 5201 4010 5101 5001			X			Coordination Test Hazop28 (#2 of 4). Reversal with ownership is master. Verifies that when reversing, the ownership output vrc changes and the cvc is set for the old value. The cvc has to remain that value until the RA is terminated or encounter ends. Climb to descend.
3152	31	X				3	1000 5001 5101	X		X			Coordination Test Hazop28 (#3 of 4). Reversal with ownership is slave. Verifies that ownership does not reverse (as it would do without receiving uf16uds30) when intruder as master sends uf16uds30 appropriately. Descend to climb.
3153	31	X				3	1000 4010 4110	X		X			Coordination Test Hazop28 (#4 of 4). Reversal with ownership is slave. Verifies that ownership does not reverse (as it would do without receiving uf16uds30) when intruder as master sends uf16uds30 appropriately. Climb to descend.
3160	31	X				3	1000 5001 5101	X		X			Coordination Test Hazop29 (#1 of 2). Verifying when multiple uf16uds30 messages arrive in the same cycle, the proper information gets used.

Encounter	Test Group	Mode C	Mode S	ADS-B	ADS-R	Intruder RI	RA (Label 270)	Coordination	Designation/Xo	Multi-Threat	Prescriptive	Aural	Description
													Encounter is similar to Hazop28 #3 but has additional conflicting uf16uds30.
3161	31	X				3	1000 4010 4110	X			X		Coordination Test Hazop29 (#2 of 2). Verifying when multiple uf16uds30 messages arrive in the same cycle, the proper information gets used. Encounter is similar to Hazop28 #4 but has additional conflicting uf16uds30.
3170	31	X				3	1000 5001 4210 4010				X		Coordination Test Hazop35a (#1 of 5). Verifying that a cancel vrc (cvc) with no associated new vrc clears all vrc received from that intruder: Base encounter. Without any uf16uds30 ownship reverses. (ownship is slave)
3171	31	X				3	1000 5001 4210 4010	X			X		Coordination Test Hazop35a (#2 of 5). Verifying that a cancel vrc (cvc) with no associated new vrc clears all vrc received from that intruder: Test verifies that ownship reverses right after uf16uds30 that had avoided reversing is cancelled. (ownship is slave)
3172	31	X				3	1000 5001 4210 5101 4010	X			X		Coordination Test Hazop35a (#3 of 5). Verifying that a cancel vrc (cvc) with no associated new vrc clears all vrc received from that intruder: Test verifies that ownship reverses right after uf16uds30 that had avoided reversing has timed out. (ownship is slave)
3173	31	X				3	1000 5001 4210 5101 4010	X			X		Coordination Test Hazop35a (#4 of 5). Verifying that a cancel vrc (cvc) with no associated new vrc clears all vrc received from that intruder: Using wrong cancellation. Test verifies that ownship reverses after the uf16uds30 vrc has timed out plus time for wrong uf16uds30. (ownship is slave)
3174	31	X				3	5001 4210 5101 4010	X			X		Coordination Test Hazop35a (#5 of 5). Verifying that a cancel vrc (cvc) with no associated new vrc clears all vrc received from that intruder: Using wrong cancellation for multiple seconds. Test verifies that ownship reverses after the uf16uds30 vrc has timed out plus the time of providing

Appendix F
F-24

Encounter	Test Group	Mode C	Mode S	ADS-B	ADS-R	Intruder RI	RA (Label 270)	Coordination	Designation/Xo	Multi-Threat	Prescriptive	Aural	Description
													the wrong cancellation per uf16uds30. Any uf16uds30 that doesn't change the vrc (like a wrong cancellation) keeps the last uf16uds30 coasting. This behavior is similar to TCAS. (ownship is slave)
3175	31	X				3	1000 5201 4010 4310 5001			X	X		Coordination Test Hazop35b (#1 of 21). Verifying that a cancel vrc (cvc) with no associated new vrc clears all vrc received from that intruder, testing for multithreat: Base encounter. Without any uf16uds30 ownship reverses. (ownship is slave)
3176	31	X				3	4310 1000 5001 4010 4110 5201 6422	X		X	X		Coordination Test Hazop35b (#2 of 21). Verifying that a cancel vrc (cvc) with no associated new vrc clears all vrc received from that intruder, testing for multithreat: RAC field indicates that intruder coordination message ends correctly after timing out 6 seconds after the last uf16uds30 was received (no cancellation was sent) during multithreat encounter (uf16uds30 from 1 intruder). Cancel for intruder correctly sent from ownship, as soon as threat went away. (ownship is slave)
3177	31	X				3	4310 1000 5001 4010 4110 5201 6422	X		X	X		Coordination Test Hazop35b (#3 of 21). Verifying that a cancel vrc (cvc) with no associated new vrc clears all vrc received from that intruder, testing for multithreat: RAC field indicates that intruder coordination message ends correctly after timing out 6 seconds after the last uf16uds30 was received (no cancellation was sent) during multithreat encounter (uf16uds30 from 1 intruder). Cancel for intruder correctly sent from ownship, as soon as threat went away. (ownship is slave)
3178	31	X				3	4310 1000 5001 4010 4110	X		X	X		Coordination Test Hazop35b (#4 of 21). Verifying that a cancel vrc (cvc) with no associated new vrc clears all vrc received from that intruder, testing for multithreat: RAC field indicates that intruder coordination messages end correctly after cancellations were received during multithreat encounter (uf16uds30 for multiple intruders, all cancelling at

Encounter	Test Group	Mode C	Mode S	ADS-B	ADS-R	Intruder RI	RA (Label 270)	Coordination	Designation/Xo	Multi-Threat	Prescriptive	Aural	Description
							5201 6422						the same time). Cancel for all intruders correctly sent from ownship, as soon as a threat went away. (ownship is slave)
3179	31	X				3	4310 1000 5001 4010 4110 5201 6422	X		X	X		Coordination Test Hazop35b (#5 of 21). Verifying that a cancel vrc (cvc) with no associated new vrc clears all vrc received from that intruder, testing for multithreat: RAC field indicates that intruder coordination messages end correctly after cancellations were received during multithreat encounter (uf16uds30 for multiple intruders, all cancelling at different times). Cancel for all intruders correctly sent from ownship, as soon as a threat went away. (ownship is slave)
3180	31	X				3	1000 5201 4010 4310 5001			X	X		Coordination Test Hazop35b (#6 of 21). Verifying that a cancel vrc (cvc) with no associated new vrc clears all vrc received from that intruder, testing for multithreat: Making sure that a vrc is properly cancelled and is no longer effective when the intruder later becomes a threat again. Base encounter without uf16uds30. (ownship is slave)
3181	31	X				3	4310 1000 5001 4010 5201 6422	X		X	X		Coordination Test Hazop35b (#7 of 21). Verifying that a cancel vrc (cvc) with no associated new vrc clears all vrc received from that intruder, testing for multithreat: Making sure that a vrc is properly cancelled and is no longer effective when the intruder later becomes a threat again. The earlier uf16uds30 vrc had been cancelled. Intruder geometry then causes RA that would be incompatible with former uf16uds30 message. Receiving RA for returning intruder shows that vrc was effectively cancelled. (ownship is slave)
3182	31	X				3	1000 5201 4010 4310 5001			X	X		Coordination Test Hazop35b (#8 of 21). Verifying that a cancel vrc (cvc) with no associated new vrc clears all vrc received from that intruder, testing for multithreat: Base encounter. Without any uf16uds30 ownship reverses. (ownship is master)

Appendix F
F-26

Encounter	Test Group	Mode C	Mode S	ADS-B	ADS-R	Intruder RI	RA (Label 270)	Coordination	Designation/Xo	Multi-Threat	Prescriptive	Aural	Description
3183	31	X				3	1000 5201 4010 4310 5001	X		X	X		Coordination Test Hazop35b (#9 of 21). Verifying that a cancel vrc (cvc) with no associated new vrc clears all vrc received from that intruder, testing for multithreat: RAC field indicates that intruder coordination message ends correctly after timing out 6 seconds after the last uf16uds30 was received (no cancellation was sent) during multithreat encounter (uf16uds30 from 1 intruder). Cancel for intruder correctly sent from ownship, as soon as threat went away. (ownship is master)
3184	31	X				3	1000 5201 4010 4310 5001	X		X	X		Coordination Test Hazop35b (#10 of 21). Verifying that a cancel vrc (cvc) with no associated new vrc clears all vrc received from that intruder, testing for multithreat: RAC field indicates that intruder coordination message ends correctly after timing out 6 seconds after the last uf16uds30 was received (no cancellation was sent) during multithreat encounter (uf16uds30 from 1 intruder). Cancel for intruder correctly sent from ownship, as soon as threat went away. (ownship is master)
3185	31	X				3	1000 5201 4010 4310 5001	X		X	X		Coordination Test Hazop35b (#11 of 21). Verifying that a cancel vrc (cvc) with no associated new vrc clears all vrc received from that intruder, testing for multithreat: RAC field indicates that intruder coordination messages end correctly after cancellations were received during multithreat encounter (uf16uds30 for multiple intruder, all cancelling at the same time). Cancel for all intruders correctly sent from ownship, as soon as a threat went away. (ownship is master)
3186	31	X				3	1000 5201 4010 4310 5001	X		X	X		Coordination Test Hazop35b (#12 of 21). Verifying that a cancel vrc (cvc) with no associated new vrc clears all vrc received from that intruder, testing for multithreat: RAC field indicates that intruder coordination messages end correctly after cancellations were received during multithreat encounter (uf16uds30 for multiple intruder, all cancelling at

Encounter	Test Group	Mode C	Mode S	ADS-B	ADS-R	Intruder RI	RA (Label 270)	Coordination	Designation/Xo	Multi-Threat	Prescriptive	Aural	Description
													different times). Cancel for all intruders correctly sent from ownship, as soon as a threat went away. (ownship is master)
3187	31	X				3	1000 5201 4010 4310 5001			X	X		Coordination Test Hazop35b (#13 of 21). Verifying that a cancel vrc (cvc) with no associated new vrc clears all vrc received from that intruder, testing for multithreat: Making sure that a vrc is properly cancelled and is no longer effective when the intruder later becomes a threat again. Base encounter without uf16uds30. (ownship is master)
3188	31	X				3	1000 5201 4010 4310 5001	X		X	X		Coordination Test Hazop35b (#14 of 21). Verifying that a cancel vrc (cvc) with no associated new vrc clears all vrc received from that intruder, testing for multithreat: Making sure that a vrc is properly cancelled and is no longer effective when the intruder later becomes a threat again. The earlier uf16uds30 vrc had been cancelled. Intruder geometry then causes RA that would be incompatible with former uf16uds30 message. Receiving RA for returning intruder shows that vrc was effectively cancelled. (ownship is master)
3189	31	X				3	1000 5201 4010 4310 5001			X	X		Coordination Test Hazop35b (#15 of 21). Verifying that a cancel vrc (cvc) with no associated new vrc clears all vrc received from that intruder, testing for multithreat: Base encounter. Without any uf16uds30 ownship reverses. (ownship is slave to intruder 1, master to 2 and 3)
3190	31	X				3	1000 5201 4010 4310 5001	X		X	X		Coordination Test Hazop35b (#16 of 21). Verifying that a cancel vrc (cvc) with no associated new vrc clears all vrc received from that intruder, testing for multithreat: RAC field indicates that intruder coordination message ends correctly after timing out 6 seconds after the last uf16uds30 was received (no cancellation was sent) during multithreat encounter (uf16uds30 from 1 intruder). Cancel for intruder correctly sent from ownship, as soon as threat went away. (ownship is slave to intruder 1, master to 2 and 3)

Appendix F
F-28

Encounter	Test Group	Mode C	Mode S	ADS-B	ADS-R	Intruder RI	RA (Label 270)	Coordination	Designation/Xo	Multi-Threat	Prescriptive	Aural	Description
		X							X	X	X		
3191	31		X			3	1000 5201 4010 4310 5001	X		X	X		Coordination Test Hazop35b (#17 of 21). Verifying that a cancel vrc (cvc) with no associated new vrc clears all vrc received from that intruder, testing for multithreat: RAC field indicates that intruder coordination message ends correctly after timing out 6 seconds after the last uf16uds30 was received (no cancellation was sent) during multithreat encounter (uf16uds30 from 1 intruder). Cancel for intruder correctly sent from ownship, as soon as threat went away. (ownship is slave to intruder 1, master to 2 and 3)
3192	31		X			3	1000 5201 4010 4310 5001	X		X	X		Coordination Test Hazop35b (#18 of 21). Verifying that a cancel vrc (cvc) with no associated new vrc clears all vrc received from that intruder, testing for multithreat: RAC field indicates that intruder coordination messages end correctly after cancellations were received during multithreat encounter (uf16uds30 for multiple intruder, all cancelling at the same time). Cancel for all intruders correctly sent from ownship, as soon as a threat went away. (ownship is slave to intruder 1, master to 2 and 3)
3193	31		X			3	1000 5201 4010 4310 5001	X		X	X		Coordination Test Hazop35b (#19 of 21). Verifying that a cancel vrc (cvc) with no associated new vrc clears all vrc received from that intruder, testing for multithreat: RAC field indicates that intruder coordination messages end correctly after cancellations were received during multithreat encounter (uf16uds30 for multiple intruder, all cancelling at different times). Cancel for all intruders correctly sent from ownship, as soon as a threat went away. (ownship is slave to intruder 1, master to 2 and 3)
3194	31		X			3	1000 5201 4010			X	X		Coordination Test Hazop35b (#20 of 21). Verifying that a cancel vrc (cvc) with no associated new vrc clears all vrc received from that intruder, testing for multithreat: Making sure that a vrc is properly cancelled and is

Encounter	Test Group	Mode C	Mode S	ADS-B	ADS-R	Intruder RI	RA (Label 270)	Coordination	Designation/Xo	Multi-Threat	Prescriptive	Aural	Description
							4310 5001						no longer effective when the intruder later becomes a threat again. Base encounter without uf16uds30. (ownship is slave to intruder 1, master to 2 and 3)
3195	31	X				3	6422	X		X	X		Coordination Test Hazop35b (#21 of 21). Verifying that a cancel vrc (cvc) with no associated new vrc clears all vrc received from that intruder, testing for multithreat: Making sure that a vrc is properly cancelled and is no longer effective when the intruder later becomes a threat again. The earlier uf16uds30 vrc had been cancelled. Intruder geometry then causes RA that would be incompatible with former uf16uds30 message. Receiving RA for returning intruder shows that vrc was effectively cancelled. (ownship is slave to intruder 1, master to 2 and 3)
3200	32	X				3		X			X		Coordination Parity Check: All combinations of vrc and cvc and vsb (parity) are tested. If a combination is invalid, it is indicated in the STMreport in degraded_surveillance.
3300	33	X				3	4010				X		Coordination Reach-back Test: (#1 of 4) Base encounter without coordination message, resulting in RA 4010 (climb) at 15s.
3301	33	X				3	5101	X			X		Coordination Reach-back Test: (#2 of 4) Coordination message at 14.45 s resulting in RA 5101 (crossing descend) at 15s
3302	33	X				3	5201 4010 5101	X			X		Coordination Reach-back Test: (#3 of 4) Coordination message at 15.45 s resulting in RA 4010 (climb) at 15s
3303	33	X				3	5101	X			X		Coordination Reach-back Test: (#4 of 4) Same encounter as #3 but this encounter is run with a reach-back (making fresh uf16uds30 available for TRM run) at 600 ms. The reach-back mechanism makes the coordination message at 15.45 s (like #3) still effective for the cycle at 15 s, hence similar to #2, RA 5101 (crossing descend) is issued at 15 s.

Table F-4: Test Group 40

Encounter	Test Group	Mode C	Mode S	ADS-B	ADS-R	Intruder RI	RA (Label 270)	Coordination	Designation/Xo	Multi-Threat	Prescriptive	Aural	Description
4100	41	X									X		Mode C Gillham Decode: (#1 of 7) Multiple Mode C intruders with 100 fps climb rate. After the track becomes stable with constant climb rate, an incorrect altitude decode would either cause the track to deviate from the expected value (if value deviates slightly and is accepted as track value) or would cause the track to coast (if value deviates significantly and is rejected as track value), indicated by the vertical coast flag of the degraded_surveillance field in the STMreport. All altitude values in the valid Mode C range are tested.
4101	41	X									X		Mode C Gillham Decode: (#2 of 7) Multiple Mode C intruders with 100 fps climb rate. After the track becomes stable with constant climb rate, an incorrect altitude decode would either cause the track to deviate from the expected value (if value deviates slightly and is accepted as track value) or would cause the track to coast (if value deviates significantly and is rejected as track value), indicated by the vertical coast flag of the degraded_surveillance field in the STMreport. All altitude values in the valid Mode C range are tested.
4102	41	X									X		Mode C Gillham Decode: (#3 of 7) Multiple Mode C intruders with 100 fps climb rate. After the track becomes stable with constant climb rate, an incorrect altitude decode would either cause the track to deviate from the expected value (if value deviates slightly and is accepted as track value) or would cause the track to coast (if value deviates significantly and is rejected as track value), indicated by the vertical coast flag of the degraded_surveillance field in the STMreport. All altitude values in the valid Mode C range are tested.

Encounter	Test Group	Mode C	Mode S	ADS-B	ADS-R	Intruder RI	RA (Label 270)	Coordination	Designation/Xo	Multi-Threat	Prescriptive	Aural	Description
4103	41	X								X			Mode C Gillham Decode: (#4 of 7) Multiple Mode C intruders with 100 fps climb rate. After the track becomes stable with constant climb rate, an incorrect altitude decode would either cause the track to deviate from the expected value (if value deviates slightly and is accepted as track value) or would cause the track to coast (if value deviates significantly and is rejected as track value), indicated by the vertical coast flag of the degraded_surveillance field in the STMreport. All altitude values in the valid Mode C range are tested.
4104	41	X									X		Mode C Gillham Decode: (#5 of 7) Multiple Mode C intruders with 100 fps climb rate. After the track becomes stable with constant climb rate, an incorrect altitude decode would either cause the track to deviate from the expected value (if value deviates slightly and is accepted as track value) or would cause the track to coast (if value deviates significantly and is rejected as track value), indicated by the vertical coast flag of the degraded_surveillance field in the STMreport. All altitude values in the valid Mode C range are tested.
4105	41	X									X		Mode C Gillham Decode: (#6 of 7) Multiple Mode C intruders with 100 fps climb rate. After the track becomes stable with constant climb rate, an incorrect altitude decode would either cause the track to deviate from the expected value (if value deviates slightly and is accepted as track value) or would cause the track to coast (if value deviates significantly and is rejected as track value), indicated by the vertical coast flag of the degraded_surveillance field in the STMreport. All altitude values in the valid Mode C range are tested.
4106	41	X									X		Mode C Gillham Decode: (#7 of 7) Multiple Mode C intruders with 100 fps climb rate. After the track becomes stable with constant climb rate, an incorrect altitude decode would either cause the track to deviate from the expected value (if value deviates slightly and is accepted as track

		Encounter	Test Group	Mode C	Mode S	ADS-B	ADS-R	Intruder RI	RA (Label 270)	Coordination	Designation/Xo	Multi-Threat	Prescriptive	Aural	Description
															value) or would cause the track to coast (if value deviates significantly and is rejected as track value), indicated by the vertical coast flag of the degraded_surveillance field in the STMreport. All altitude values in the valid Mode C range are tested.
4200	42	X							1000 4010 4310				X		Mode C TID field: (#1 of 1) RA to ground report TIDA field Mode C during RA where intruder altitude changes, hence altitude values are between the quantized values. This test verifies that the rounding before reporting the altitude to the ground is done correctly.

Table F-5: Test Group 50

		Encounter	Test Group	Mode C	Mode S	ADS-B	ADS-R	Intruder RI	RA (Label 270)	Coordination	Designation/Xo	Multi-Threat	Prescriptive	Aural	Description
5100	51	X						0	4010 5201 6002 4020 4310 5401				X		Turning Encounters: (#1 of 12) Ownship straight line and descending, intruder turning and climbing.

Encounter	Test Group	Mode C	Mode S	ADS-B	ADS-R	Intruder RI	RA (Label 270)	Coordination	Designation/Xo	Multi-Threat	Prescriptive	Aural	Description
5101	51		X			0	1000 5201 6002 4020			X			Turning Encounters: (#2 of 12) Ownship straight line and descending, intruder turning and climbing.
5102	51		X			0	1000 5201 6002 4020			X			Turning Encounters: (#3 of 12) Ownship straight line and descending, intruder turning and climbing.
5103	51		X			0	4110			X			Turning Encounters: (#4 of 12) Ownship turning, intruder straight line, both descending.
5104	51		X			0	5201 5301 4110			X			Turning Encounters: (#5 of 12) Ownship turning, intruder straight line, both descending.
5105	51		X			0	5002			X			Turning Encounters: (#6 of 12) Ownship turning, intruder straight line, both climbing.
5106	51		X	X		0	5201 6002 4020 5401			X			Turning Encounters: (#7 of 12) Ownship straight line and descending, intruder turning and climbing.
5107	51		X	X		0				X			Turning Encounters: (#8 of 12) Ownship straight line and descending, intruder turning and climbing.
5108	51		X	X		0				X			Turning Encounters: (#9 of 12) Ownship straight line and descending, intruder turning and climbing.
5109	51		X	X		0	4110			X			Turning Encounters: (#10 of 12) Ownship turning, intruder straight line, both descending.
5110	51		X	X		0	5201 5301 4110			X			Turning Encounters: (#11 of 12) Ownship turning, intruder straight line, both descending.

Appendix F
F-34

Encounter	Test Group	Mode C	Mode S	ADS-B	ADS-R	Intruder RI	RA (Label 270)	Coordination	Designation/Xo	Multi-Threat	Prescriptive	Aural	Description
5111	51	X	X			0	5002			X			Turning Encounters: (#12 of 12) Ownship turning, intruder straight line, both climbing.
5200	52	X				0	5002 4210			X			Long Forward Prediction: (#1 of 2) Messages arrive very early in execution cycle so that values have to be forward predicted for almost 1 second.
5201	52	X	X			0	5002 4210			X			Long Forward Prediction: (#2 of 2) Messages arrive very early in execution cycle so that values have to be forward predicted for almost 1 second.
5300	53	X				0	5001 5301			X			Altitude Inhibit: (#1 of 2) Verifying that radio altitude and not barometric altitude is used for inhibit. Base encounter with radio altitude above inhibit altitude, active RAs are issued.
5301	53	X				0				X			Altitude Inhibit: (#2 of 2) Verifying that radio altitude and not barometric altitude is used for inhibit. Radio altitude below inhibit altitude, no RAs, only TAs issued instead.
5310	53	X				0	5001 5301			X			Descend Inhibit (#1 of 4). Base encounter. With radio altitudes above inhibit altitudes, encounter causes RA 5001 descend at 4s and RA 5301 increase descend at 12s to end.
5311	51	X				0	5001 6002			X			Descend Inhibit (#2 of 4). Inhibits during ownship descending. Same as #1 but with modified radio altitudes. At 12s radio alt drops below 1450ft, causing RA to remain 5001 Descend instead of switching to 5301 Increase Descent. At 17s rad alt drops below 1000ft causing RA to change to RA 6002 Don't Climb. At 19s rad alt drops below 900ft causing RA to be suppressed.
5312	53	X				0	5001 6002 5301			X			Descend Inhibit (#3 of 4). Inhibits during ownship climbing. Same as #1 but with modified radio altitudes. Before 6s radio alt is below 1100 ft, no RA issued. At 6s radio alt climbs above 1100ft, causing RA to become 6002 Don't Climb. At 8s rad alt climbs above 1200ft allowing RA to change to RA 5001 Descend. At 11s radio alt climbs above 1650ft allowing strengthening, so that RA 5301 Increase Descent is issued at 18s to end.

Encounter	Test Group	Mode C	Mode S	ADS-B	ADS-R	Intruder RI	RA (Label 270)	Coordination	Designation/Xo	Multi-Threat	Prescriptive	Aural	Description
5313	53		X			0	5001 6002				X		Descend Inhibit (#4 of 4). Inhibits during ownship climbing. Same as #1 but with modified radio altitudes. Similar to #3 but with Increase Descend Inhibit at the end of encounter. Before 6s radio alt is below 1100 ft, no RA issued. At 6s radio alt climbs above 1100ft, causing RA to become 6002 Don't Climb. At 8s rad alt climbs above 1200ft allowing RA to change to RA 5001 Descend. Radio alt never climbs above 1650ft so that no strengthening is allowed, hence no RA 5301 Increase Descent is issued.
5400	54		X			0	5001 5301				X		Encounter for testing different discretes in OwnshipDiscretes: (#1 of 3) Base Encounter.
5401	54		X			0	5001				X		Encounter for testing different discretes in OwnshipDiscretes: (#2 of 3) Testing opflg, manual_SL, own_ground_display_mode_on, and on_surface.
5402	54		X			0	5001				X		Encounter for testing different discretes in OwnshipDiscretes: (#3 of 3) Testing opflg, manual_SL, own_ground_display_mode_on, and on_surface.
5410	54		X	X		0	5001 5301				X		Encounter for testing aoto_on (ADS-B only TA-only) flag: (#1 of 3) Base Encounter with intruder DF0 and ADS-B.
5411	54			X							X		Encounter for testing aoto_on (ADS-B only TA-only) flag: (#2 of 3) Intruder is ADS-B only and aoto_on set to true.
5412	54			X							X		Encounter for testing aoto_on (ADS-B only TA-only) flag: (#3 of 3) Intruder is ADS-B only and aoto_on set to false.
5500	55		X	X		0	1000 5001				X		Improper / Degraded Inputs: (#1 of 4).
5501	55		X	X		0	1000 5001				X		Improper / Degraded Inputs: (#2 of 4).
5502	55		X	X		3	4010	X			X		Improper / Degraded Inputs: (#3 of 4).

Encounter	Test Group	Mode C	Mode S	ADS-B	ADS-R	Intruder RI	RA (Label 270)	Coordination	Designation/Xo	Multi-Threat	Prescriptive	Aural	Description
5510	55	X	X			3	5001 5301		X				Improper / Degraded Inputs: (#4 of 4) Track changes from Mode S to Mode C and back to Mode S track. Test verifies that track id remains the same and RA doesn't get interrupted.
5600	56			X						X			Precision Encounters: (#1 of 4)
5601	56			X						X			Precision Encounters: (#2 of 4)
5602	56			X						X			Precision Encounters: (#3 of 4)
5603	56			X						X			Precision Encounters: (#4 of 4)

Table F-6: Test Group 60

Encounter	Test Group	Mode C	Mode S	ADS-B	ADS-R	Intruder RI	RA (Label 270)	Coordination	Designation/Xo	Multi-Threat	Prescriptive	Aural	Description
6000	60	X	X			0	1000 5001 5301	X			X		Designation encounters / Xo Testsubgroup1: (#1 of 14). Baseline encounter for testing Target Designation scenarios, it has no designations. The intruder has no CAS. The geometry of this encounter is head on ownship with a single intruder. Both aircraft are level at an altitude of 20,000 ft.
6001	60	X	X			0	5001	X	X		X		Designation encounters / Xo Testsubgroup1: (#2 of 14). This encounter tests that the system can successfully designate an intruder DNA. The intruder has no CAS. The geometry of this encounter is ownship with a head on intruder. Both aircraft are level at an altitude of 20,000 ft.

Encounter	Test Group	Mode C	Mode S	ADS-B	ADS-R	Intruder RI	RA (Label 270)	Coordination	Designation/Xo	Multi-Threat	Prescriptive	Aural	Description
6002	60		X	X		0	1000 5001 5301	X	X		X		Designation encounters / Xo Testsubgroup1: (#3 of 14). This encounter tests that the system can successfully designate an intruder CSPO-3000. The intruder has no CAS. The geometry of the encounter is ownship with a head on intruder. Both aircraft are level at an altitude of 13,000 ft.
6003	60		X	X		0	1000 5001 5301	X	X		X		Designation encounters / Xo Testsubgroup1: (#4 of 14). This encounter tests that the system can successfully undesignate an intruder that is designated DNA. The intruder has no CAS. The geometry of the encounter is ownship with a head on intruder. Both aircraft are level at an altitude of 20,000 ft.
6004	60		X	X		0	1000 5001 5301	X	X		X		Designation encounters / Xo Testsubgroup1: (#5 of 14). This encounter tests that the system can handle invalid designations. The intruder has no CAS. The geometry of the encounter is ownship with a head on intruder. Both aircraft are level at an altitude of 20,000 ft.
6005	60		X	X		0	1000 5001 5301	X	X		X		Designation encounters / Xo Testsubgroup1: (#6 of 14). This encounter tests that the system can handle an undesignation message when there are no designated intruders. The intruder has no CAS. The geometry of the encounter is ownship with a head on intruder. Both aircraft are level at an altitude of 20,000 ft.
6006	60		X	X		0	1000 5001 5301	X	X		X		Designation encounters / Xo Testsubgroup1: (#7 of 14). This encounter tests that the system handles a designation of a non-existent intruder. The intruder has no CAS. The geometry of the encounter is ownship with a head on intruder. Both aircraft are level at an altitude of 20,000 ft.
6007	60		X	X		3	1000 5001 5301	X			X		Designation encounters / Xo Testsubgroup1: (#8 of 14). Baseline encounter intended for testing Target Designation scenarios, it has no designations. The intruder has active CAS. The geometry of the encounter is ownship with a head on intruder. Both aircraft are level at an altitude of 20,000 ft.

Appendix F
F-38

Encounter	Test Group	Mode C	Mode S	ADS-B	ADS-R	Intruder RI	RA (Label 270)	Coordination	Designation/Xo	Multi-Threat	Prescriptive	Aural	Description
6008	60		X	X		3	5001	X	X		X		Designation encounters / Xo Testsubgroup1: (#9 of 14). This encounter tests that the system can successfully designate an intruder DNA. The intruder has active CAS. The geometry of the encounter is ownship with a head on intruder. Both aircraft are level at an altitude of 20,000 ft.
6009	60		X	X		3	1000 5001 5301	X	X		X		Designation encounters / Xo Testsubgroup1: (#10 of 14). This encounter tests that the system can successfully designate an intruder CSPO-3000. The intruder has active CAS. The geometry of the encounter is ownship with a head on intruder. Both aircraft are level at an altitude of 13,000 ft.
6010	60		X	X		3	1000 5001 5301	X	X		X		Designation encounters / Xo Testsubgroup1: (#11 of 14). This encounter tests that the system can successfully undesignate an intruder that is designated DNA. The intruder has active CAS. The geometry of the encounter is ownship with a head on intruder. Both aircraft are level at an altitude of 20,000 ft.
6011	60		X	X		3	1000 5001 5301	X	X		X		Designation encounters / Xo Testsubgroup1: (#12 of 14). This encounter tests that the system can handle invalid designations. The intruder has active CAS. The geometry of the encounter is ownship with a head on intruder. Both aircraft are level at an altitude of 20,000 ft.
6012	60		X	X		3	1000 5001 5301	X	X		X		Designation encounters / Xo Testsubgroup1: (#13 of 14). This encounter tests that the system handles an undesignation message when there are no designated intruders. The intruder has active CAS. The geometry of the encounter is ownship with a head on intruder. Both aircraft are level at an altitude of 20,000 ft.
6013	60		X	X		3	1000 5001 5301	X	X		X		Designation encounters / Xo Testsubgroup1: (#14 of 14). This encounter tests that the system handles a designation of a non-existent intruder. The intruder has active CAS. The geometry of the encounter is ownship with a head on intruder. Both aircraft are level at an altitude of 20,000 ft.

Encounter	Test Group	Mode C	Mode S	ADS-B	ADS-R	Intruder RI	RA (Label 270)	Coordination	Designation/Xo	Multi-Threat	Prescriptive	Aural	Description
6020	60		X	X		0	5002	X			X		Designation encounters / Xo Testsubgroup2: (#1 of 6). Baseline encounter for testing CSPO-3000 designations at low altitude, it has no designation. The intruder has no CAS. This encounter has ownship and an intruder flying in-parallel at the same velocity and level at 1050 ft. altitude.
6021	60		X	X		0	5002	X	X		X		Designation encounters / Xo Testsubgroup2: (#2 of 6). This encounter tests that the system can successfully designate an intruder CSPO-3000. The intruder has no CAS. This encounter has ownship and an intruder flying in-parallel at the same velocity and level at 1050 ft. altitude.
6022	60		X	X		0	5002	X	X		X		Designation encounters / Xo Testsubgroup2: (#3 of 6). This encounter tests that the system can successfully undesignate an intruder that is designated CSPO-3000. The intruder has no CAS. This encounter has ownship and an intruder flying in-parallel at the same velocity and level at 1050 ft. altitude.
6023	60		X	X		3	5002	X			X		Designation encounters / Xo Testsubgroup2: (#4 of 6). Baseline for testing CSPO-3000 designations at low altitude, it has no designation. The intruder has active CAS. This encounter has ownship and an intruder flying in-parallel at the same velocity and level at 1050 ft. altitude.
6024	60		X	X		3	5002	X	X		X		Designation encounters / Xo Testsubgroup2: (#5 of 6). This encounter tests that the system can successfully designate an intruder CSPO-3000. The intruder has active CAS. This encounter has ownship and an intruder flying in-parallel at the same velocity and level at 1050 ft. altitude.
6025	60		X	X		3	5002	X	X		X		Designation encounters / Xo Testsubgroup2: (#6 of 6). This encounter tests that the system can successfully undesignate an intruder that is designated CSPO-3000. The intruder has active CAS. This encounter has ownship and an intruder flying in-parallel at the same velocity and level at 1050 ft. altitude.
6030	60		X	X		0	5001 4210	X		X	X		Designation encounters / Xo Testsubgroup3: (#1 of 8). Baseline encounter for testing target designation in multi-threat scenarios. The intruder has

Appendix F
F-40

Encounter	Test Group	Mode C	Mode S	ADS-B	ADS-R	Intruder RI	RA (Label 270)	Coordination	Designation/Xo	Multi-Threat	Prescriptive	Aural	Description
							4010 4310						no CAS. The geometry of the encounter is ownship with a head on intruder. Second intruder a few thousand feet behind ownship traveling at the same speed and heading. All three aircraft are level at an altitude of 20,000 ft.
6031	60		X	X		0	5001 4210 4010 4310	X	X	X	X		Designation encounters / Xo Testsubgroup3: (#2 of 8). This encounter tests that the system correctly designates a target in a multi-threat scenario. The intruder has no CAS. The geometry of the encounter is ownship with a head on intruder. There is a second intruder a few thousand feet behind ownship traveling at the same speed and heading. All three aircraft are level at an altitude of 20,000 ft.
6032	60		X	X		0	5001 4210 4010 4310	X	X	X	X		Designation encounters / Xo Testsubgroup3: (#3 of 8). After a few seconds two intruders are designated DNA. This encounter tests designation limits. The intruder has no CAS. The geometry of the encounter is ownship with a head on intruder. There is a second intruder a few thousand feet behind ownship traveling at the same speed and heading. All three aircraft are level at an altitude of 20,000 ft.
6033	60		X	X		0	5001 4210 4010 4310	X	X	X	X		Designation encounters / Xo Testsubgroup3: (#4 of 8). This encounter tests that the system correctly carries out a sequence of designations: DNA, undesignate all, and DNA of a second intruder. The intruder has no CAS. The geometry of the encounter is ownship with a head on intruder. Second intruder a few thousand feet behind ownship traveling at the same speed and heading. All three aircraft are level at an altitude of 20,000 ft.
6034	60		X	X		3	5001 5301	X		X	X		Designation encounters / Xo Testsubgroup3: (#5 of 8). Baseline encounter for testing target designation in multi-threat scenarios. The intruder has active CAS. The geometry of the encounter is ownship with a head on intruder. There is a second intruder a few thousand feet behind ownship

Encounter	Test Group	Mode C	Mode S	ADS-B	ADS-R	Intruder RI	RA (Label 270)	Coordination	Designation/Xo	Multi-Threat	Prescriptive	Aural	Description
													traveling at the same speed and heading. All three aircraft are level at an altitude of 20,000 ft.
6035	60	X	X		3	5001 5301	X	X	X	X			Designation encounters / Xo Testsubgroup3: (#6 of 8). This encounter tests that the system correctly designates a target in a multi-threat scenario. The intruder has active CAS. The geometry of the encounter is ownship with a head on intruder. There is a second intruder a few thousand feet behind ownship traveling at the same speed and heading. All three aircraft are level at an altitude of 20,000 ft.
6036	60	X	X		3	5001 5301	X	X	X	X			Designation encounters / Xo Testsubgroup3: (#7 of 8). After a few seconds two intruders are designated DNA. This encounter tests designation limits. The intruder has active CAS. The geometry of the encounter is ownship with a head on intruder. There is a second intruder a few thousand feet behind ownship traveling at the same speed and heading. All three aircraft are level at an altitude of 20,000 ft.
6037	60	X	X		3	5001 5301	X	X	X	X			Designation encounters / Xo Testsubgroup3: (#8 of 8). This encounter tests that the system correctly carries out a sequence of designations: DNA, undesignate all, and DNA of a second intruder. The intruder has active CAS. The geometry of the encounter is ownship with a head on intruder. There is a second intruder a few thousand feet behind ownship traveling at the same speed and heading. All three aircraft are level at an altitude of 20,000 ft.
6040	60	X			0	5002	X			X			Designation encounters / Xo Testsubgroup4: (#1 of 20). This encounter is the baseline encounter for testing CSPO-3000 logic, it has no designations. The intruder has no CAS. Intruder flying in parallel with ownship. Both aircraft are traveling at the same heading and velocity. Both are level at 1050 ft. altitude.
6041	60	X			0	5002	X	X		X			Designation encounters / Xo Testsubgroup4: (#2 of 20). This encounter tests that the system can successfully designate an intruder CSPO-3000.

Appendix F
F-42

Encounter	Test Group	Mode C	Mode S	ADS-B	ADS-R	Intruder RI	RA (Label 270)	Coordination	Designation/Xo	Multi-Threat	Prescriptive	Aural	Description
													The intruder has no CAS. Intruder flying in parallel with ownship. Both aircraft are traveling at the same heading and velocity. Both are level at 1050 ft. altitude. The slant range of ownship in this encounter is approximately 1457 ft.
6042	60	X		0		X			X		X		Designation encounters / Xo Testsubgroup4: (#3 of 20). This encounter tests parallel operation without CSPO-3000 designation. The intruder has no CAS. Intruder flying in parallel with ownship. Both aircraft are traveling at the same heading and velocity. Both are level at 1050 ft. altitude. This encounter is part of a set designated to test CSPO-3000 logic. The slant range of ownship in this encounter is approximately 1822 ft.
6043	60	X		0		X	X	X	X		X		Designation encounters / Xo Testsubgroup4: (#4 of 20). This encounter tests parallel operation with CSPO-3000 designation. The intruder has no CAS. Intruder flying in parallel with ownship. Both aircraft are traveling at the same heading and velocity. Both are level at 1050 ft. altitude. The slant range of ownship in this encounter is approximately 1822 ft.
6044	60	X		0		X			X		X		Designation encounters / Xo Testsubgroup4: (#5 of 20). This encounter tests parallel operations without CSPO-3000 designations. The intruder has no CAS. Intruder flying in parallel with ownship. Both aircraft are traveling at the same heading and velocity. Both are level at 1050 ft. altitude. The slant range of ownship in this encounter is approximately 2186 ft.
6045	60	X		0		X	X	X	X		X		Designation encounters / Xo Testsubgroup4: (#6 of 20). This encounter tests parallel operation with CSPO-3000 designation. The intruder has no CAS. Intruder flying in parallel with ownship. Both aircraft are traveling at the same heading and velocity. Both are level at 1050 ft. altitude. The slant range of ownship in this encounter is approximately 2186 ft.

Encounter	Test Group	Mode C	Mode S	ADS-B	ADS-R	Intruder RI	RA (Label 270)	Coordination	Designation/Xo	Multi-Threat	Prescriptive	Aural	Description
6046	60	X				0		X			X		Designation encounters / Xo Testsubgroup4: (#7 of 20). This encounter tests parallel operations without CSPO-3000 designations. The intruder has no CAS. Intruder flying in parallel with ownship. Both aircraft are traveling at the same heading and velocity. Both are level at 1050 ft. altitude. The slant range of ownship in this encounter is approximately 2551 ft.
6047	60	X				0		X	X		X		Designation encounters / Xo Testsubgroup4: (#8 of 20). This encounter tests parallel operation with CSPO-3000 designation. The intruder has no CAS. Intruder flying in parallel with ownship. Both aircraft are traveling at the same heading and velocity. Both are level at 1050 ft. altitude. The slant range of ownship in this encounter is approximately 2551 ft.
6048	60	X				0		X			X		Designation encounters / Xo Testsubgroup4: (#9 of 20). This encounter tests parallel operations without CSPO-3000 designations. The intruder has no CAS. Intruder flying in parallel with ownship. Both aircraft are traveling at the same heading and velocity. Both are level at 1050 ft. altitude. The slant range of ownship in this encounter is approximately 2915 ft.
6049	60	X				0		X	X		X		Designation encounters / Xo Testsubgroup4: (#10 of 20). This encounter tests parallel operation with CSPO-3000 designation. The intruder has no CAS. Intruder flying in parallel with ownship. Both aircraft are traveling at the same heading and velocity. Both are level at 1050 ft. altitude. The slant range of ownship in this encounter is approximately 2915 ft.
6050	60	X				3	5002	X			X		Designation encounters / Xo Testsubgroup4: (#11 of 20). This encounter is the baseline encounter for testing CSPO-3000 logic, it has no designations. The intruder has active CAS. Intruder flying in parallel with ownship. Both aircraft are traveling at the same heading and velocity. Both are level at 1050 ft. altitude.

Appendix F
F-44

Encounter	Test Group	Mode C	Mode S	ADS-B	ADS-R	Intruder RI	RA (Label 270)	Coordination	Designation/Xo	Multi-Threat	Prescriptive	Aural	Description
6051	60	X			3	5002	X	X		X			Designation encounters / Xo Testsubgroup4: (#12 of 20). This encounter tests that the system can successfully designate an intruder CSPO-3000. The intruder has active CAS. Intruder flying in parallel with ownship. Both aircraft are traveling at the same heading and velocity. Both are level at 1050 ft. altitude. The slant range of ownship in this encounter is approximately 1457 ft.
6052	60	X			3		X				X		Designation encounters / Xo Testsubgroup4: (#13 of 20). This encounter tests parallel operations without CSPO-3000 designations. The intruder has active CAS. Intruder flying in parallel with ownship. Both aircraft are traveling at the same heading and velocity. Both are level at 1050 ft. altitude. The slant range of ownship in this encounter is approximately 1822 ft.
6053	60	X			3		X	X			X		Designation encounters / Xo Testsubgroup4: (#14 of 20). This encounter tests parallel operation with CSPO-3000 designation. The intruder has active CAS. Intruder flying in parallel with ownship. Both aircraft are traveling at the same heading and velocity. Both are level at 1050 ft. altitude. The slant range of ownship in this encounter is approximately 1822 ft.
6054	60	X			3		X				X		Designation encounters / Xo Testsubgroup4: (#15 of 20). This encounter tests parallel operations without CSPO-3000 designations. The intruder has active CAS. Intruder flying in parallel with ownship. Both aircraft are traveling at the same heading and velocity. Both are level at 1050 ft. altitude. The slant range of ownship in this encounter is approximately 2186 ft.
6055	60	X			3		X	X			X		Designation encounters / Xo Testsubgroup4: (#16 of 20). This encounter tests parallel operation with CSPO-3000 designation. The intruder has active CAS. Intruder flying in parallel with ownship. Both aircraft are

Encounter	Test Group	Mode C	Mode S	ADS-B	ADS-R	Intruder RI	RA (Label 270)	Coordination	Designation/Xo	Multi-Threat	Prescriptive	Aural	Description
													traveling at the same heading and velocity. Both are level at 1050 ft. altitude. The slant range of ownship in this encounter is approximately 2186 ft.
6056	60	X				3		X			X		Designation encounters / Xo Testsubgroup4: (#17 of 20). This encounter tests parallel operations without CSPO-3000 designations. The intruder has active CAS. Intruder flying in parallel with ownship. Both aircraft are traveling at the same heading and velocity. Both are level at 1050 ft. altitude. The slant range of ownship in this encounter is approximately 2551 ft.
6057	60		X					3		X	X		Designation encounters / Xo Testsubgroup4: (#18 of 20). This encounter tests parallel operation with CSPO-3000 designation. The intruder has active CAS. Intruder flying in parallel with ownship. Both aircraft are traveling at the same heading and velocity. Both are level at 1050 ft. altitude. The slant range of ownship in this encounter is approximately 2551 ft.
6058	60		X					3		X			Designation encounters / Xo Testsubgroup4: (#19 of 20). This encounter tests parallel operations without CSPO-3000 designations. The intruder has active CAS. Intruder flying in parallel with ownship. Both aircraft are traveling at the same heading and velocity. Both are level at 1050 ft. altitude. The slant range of ownship in this encounter is approximately 2915 ft.
6059	60		X					3		X	X	X	Designation encounters / Xo Testsubgroup4: (#20 of 20). This encounter tests parallel operation with CSPO-3000 designation. The intruder has active CAS. Intruder flying in parallel with ownship. Both aircraft are traveling at the same heading and velocity. Both are level at 1050 ft. altitude. The slant range of ownship in this encounter is approximately 2915 ft.

Appendix F
F-46

Encounter	Test Group	Mode C	Mode S	ADS-B	ADS-R	Intruder RI	RA (Label 270)	Coordination	Designation/Xo	Multi-Threat	Prescriptive	Aural	Description
6060	60		X			0	1000 4010 4020			X			Designation encounters / Xo Testsubgroup5: (#1 of 8). Baseline encounter, no Target Designation. The intruder has no CAS. Intruder is approaching ownship from far away causing an RA to be issued about 200 seconds into the encounter.
6061	60		X			0			X		X		Designation encounters / Xo Testsubgroup5: (#2 of 8). This encounter tests CSPO-3000 logic on a designated intruder that will cause an RA. The intruder has no CAS. Intruder is approaching ownship from far away causing an RA to be issued about 200 seconds into the encounter.
6062	60		X			0	1000 4010 4020		X		X		Designation encounters / Xo Testsubgroup5: (#3 of 8). This encounter tests CSPO-3000 logic on a designated intruder that will cause an RA. A second before the RA is to be issued the intruder is undesignated. The intruder has no CAS. Intruder is approaching from far away causing an RA to be issued about 200 seconds into the encounter.
6063	60		X			0	1000 4010 4020		X		X		Designation encounters / Xo Testsubgroup5: (#4 of 8). This encounter tests how the system handles designating a target CSPO-3000 while it has an active RA. The intruder has no CAS. Intruder is approaching ownship from far away causing an RA to be issued about 200 seconds into the encounter.
6064	60		X			3	1000 4010 4020				X		Designation encounters / Xo Testsubgroup5: (#5 of 8). Baseline encounter, no Target Designation. The intruder has active CAS. Intruder is approaching ownship from far away causing an RA to be issued about 200 seconds into the encounter.
6065	60		X			3			X		X		Designation encounters / Xo Testsubgroup5: (#6 of 8). This encounter tests CSPO-3000 logic on a designated intruder that will cause an RA. The intruder has active CAS. Intruder is approaching ownship from far away causing an RA to be issued about 200 seconds into the encounter.

Encounter	Test Group	Mode C	Mode S	ADS-B	ADS-R	Intruder RI	RA (Label 270)	Coordination	Designation/Xo	Multi-Threat	Prescriptive	Aural	Description
6066	60	X				3	1000 4010 4020		X		X		Designation encounters / Xo Testsubgroup5: (#7 of 8). This encounter tests CSPO-3000 logic on a designated intruder that will cause an RA. A second before the RA is to be issued the intruder is undesignated. The intruder has active CAS. Intruder is approaching ownship from far away causing an RA to be issued about 200 seconds into the encounter.
6067	60	X				3	1000 4010 4020		X		X		Designation encounters / Xo Testsubgroup5: (#8 of 8). This encounter tests how the system handles designating a target CSPO-3000 while it has an active RA. The intruder has active CAS. Intruder is approaching ownship from far away causing an RA to be issued about 200 seconds into the encounter.
6100	60	X	X			0	5001		X		X		Designation encounters / Xo: Tests designating a target with DNA. The intruder has active CAS. The geometry of this encounter is a level ownship with a head on intruder.
6101	60	X				3			X		X		Designation encounters / Xo: Tests that ACAS will drop the designation of a target that is designated DNA and then becomes invalid for DNA, after it has been invalid for 30 seconds. The intruder has active CAS. The geometry of this encounter is a level ownship with a head on intruder.
6102	60	X	X						X				Designation encounters / Xo: Tests that a DNA designated target becomes invalid for DNA when the ownship loses reporting of baro alt. The intruder has active CAS. The geometry of this encounter is a level ownship with a head on intruder 300 ft. below in alt.
6103	60	X	X	X		3	4010		X	X			Designation encounters / Xo: Tests that a DNA-designated target which becomes involved in a multi-threat RA while designated will display the RA even though it is currently DNA. The intruder has active CAS. The geometry of this encounter is a level ownship with two head on intruders one of them 300 ft. below ownship's altitude.
6104	60	X				3			X		X		Designation encounters / Xo: Intruder is designated as DNA. Tests that designation is dropped by ACAS when ownship's radio alt starting above

Appendix F
F-48

Encounter	Test Group	Mode C	Mode S	ADS-B	ADS-R	Intruder RI	RA (Label 270)	Coordination	Designation/Xo	Multi-Threat	Prescriptive	Aural	Description
													600 ft, drops below 400 ft. and then climbs above 600 ft. The intruder has active CAS. The geometry of this encounter is a level head on intruder. Ownship descends below 400 ft. then climbs above 650 ft.
6105	60		X			3			X		X		Designation encounters / Xo: Intruder is designated as DNA. Tests that designation is dropped by ACAS after ownship's radio altitude drops below 50ft. The intruder has active CAS. The geometry of this encounter is a level head on intruder with ownship descending below 50 ft.
6106	60		X			3			X		X		Designation encounters / Xo: Intruder is designated as DNA. Tests that ACAS undesignates intruder as intruder's slant range increases beyond 6 NM. The intruder has active CAS. The geometry of this encounter is a level ownship with intruder flying away behind ownship.
6107	60		X	X		0	1000 5001 5301	X	X		X		Designation encounters / Xo: This encounter tests that the system can undesignate a target that is currently designated as DNA. The intruder has active CAS. The geometry of this encounter is a level ownship with a head on intruder.
6108	60	X	X			3			X				Designation encounters / Xo: This encounter tests that the system can transition from a state with an intruder that is both designated DNA and in an invalid timeout window to a state where the intruder becomes valid, preventing the loss of the DNA designation. The intruder has active CAS. The geometry of this encounter is a level ownship with a head on intruder.
6109	60	X	X			3			X				Designation encounters / Xo: This encounter tests that the system will automatically undesignate an intruder that is both designated DNA and in an invalid timeout window when ownship's radio alt starts above 600 ft., drops below 400 ft., and climbs back above 600 ft. The intruder has active CAS. The geometry of this encounter is a level head on intruder. Ownship descends below 400 ft. then climbs above 650 ft.

Encounter	Test Group	Mode C	Mode S	ADS-B	ADS-R	Intruder RI	RA (Label 270)	Coordination	Designation/Xo	Multi-Threat	Prescriptive	Aural	Description
6110	60	X	X			3			X				Designation encounters / Xo: This encounter tests that the system will automatically undesignate an intruder that is both designated DNA and in an invalid timeout window when ownship's radio alt starts above 600 ft. and then drops below 50 ft. The intruder has active CAS. The geometry of this encounter is a level head on intruder with ownship descending below 50 ft.
6111	60	X	X			3			X				Designation encounters / Xo: This encounter tests that the system will automatically undesignate an intruder that is both designated DNA and in an invalid timeout window when the intruder moves beyond a slant range of 6 NM. The intruder has active CAS. The geometry of this encounter is a level ownship with intruder flying away behind ownship.
6112	60	X	X			3	1000 4010 4310		X				Designation encounters / Xo: This encounter tests that the system will undesignate an intruder that is both designated DNA and in an invalid timeout window when the pilot undesignates the intruder. The intruder has active CAS. The geometry of this encounter is a level ownship with a head on intruder.
6113	60		X	X		3	1000 4010 4310		X	X	X		Designation encounters / Xo: This encounter tests that the system will set the DNA flag for an intruder that is both marked DNA and in a multi-threat RA after the multi-threat RA clears. The intruder has active CAS. The geometry of this encounter is a level ownship with two intruders, one head on and the second directly behind overtaking ownship.
6114	60	X	X			3	6422		X	X			Designation encounters / Xo: This encounter tests that the system will clear the designation for an intruder that is both marked DNA and in a multi-threat RA after the intruder becomes invalid for designation from a loss of a valid Mode S address and the 30 second timeout window expires. The intruder has active CAS. The geometry of this encounter is a level ownship with three intruders, all head on with one at a high altitude and two at a lower altitude.

Appendix F
F-50

Encounter	Test Group	Mode C	Mode S	ADS-B	ADS-R	Intruder RI	RA (Label 270)	Coordination	Designation/Xo	Multi-Threat	Prescriptive	Aural	Description
6115	60		X			3	1000 6422		X	X	X		Designation encounters / Xo: This encounter tests that the system will clear the designation for an intruder that is both marked DNA and in a multi-threat RA after the intruder's track is dropped and the 30 second timeout window expires. The intruder has active CAS. The geometry of this encounter is a level ownship with three intruders, all head on with one at a high altitude and two at a lower altitude.
6116	60	X	X			3	4010 6422		X	X			Designation encounters / Xo: This encounter tests that the system will set the DNA flag for an intruder that is both marked DNA and in a multi-threat RA after undesigned intruder's RA clears. The intruder has active CAS. The geometry of this encounter is a level ownship with two intruders, one is 200 ft. directly below ownship, the other is head on 100 ft. above ownship.
6117	60		X			3	4022		X	X	X		Designation encounters / Xo: This encounter tests that the system will automatically clear DNA designation for an intruder that is both marked DNA and in a multi-threat RA after the intruder's slant range becomes larger than 6 NM. The intruder has active CAS. The geometry of this encounter is ownship on a slow descend with two head on intruders, one above and one below descending at the same rate. There is a third intruder flying away from ownship.
6118	60		X	X		3	1000 4010 4310		X	X	X		Designation encounters / Xo: This encounter tests that the system will automatically clear DNA designation for an intruder that is both marked DNA and in a multi-threat RA after the pilot undesignates the intruder. The intruder has active CAS. The geometry of this encounter is a level ownship with two intruders. One intruder is head on and the second intruder is closing in from behind ownship.
6119	60		X			3	1000 5001		X		X		Designation encounters / Xo: The encounter starts with a DNA-designated intruder whose track is dropped (starting the timeout timer).

Encounter	Test Group	Mode C	Mode S	ADS-B	ADS-R	Intruder RI	RA (Label 270)	Coordination	Designation/Xo	Multi-Threat	Prescriptive	Aural	Description
													The track is then reacquired but with an invalid Mode S address. The encounter tests that the designation is dropped after the timeout expires. The intruder has active CAS. The geometry of this encounter is a level ownship with a head on intruder 300 feet above ownship.
6120	60	X				3			X		X		Designation encounters / Xo: This encounter tests that the system will maintain a DNA designation for an intruder in which the intruder's track is dropped but then reacquired before the timeout expires. The intruder has active CAS. The geometry of this encounter is a level ownship with a head on intruder 300 feet above ownship.
6121	60	X				3	4010 4310		X		X		Designation encounters / Xo: This encounter tests that the system will not mark an intruder DNA that is designated by the pilot while it has an active CSPO-3000 RA. Before the CSPO-3000 RA clears, the DNA is removed by the pilot. The intruder has active CAS. The geometry of this encounter is a level ownship with a head on intruder 300 ft. below ownship. Ownship slowly overtakes the intruder.
6122	60	X				3	1000 4010		X		X		Designation encounters / Xo: This encounter tests that the system will enable the DNA flag on an intruder that has been designated DNA while having active CSPO-3000 RA after the CSPO-3000 RA clears. The intruder has active CAS. The geometry of this encounter is a level ownship with a head on intruder 300 ft. below ownship. Ownship slowly overtakes the intruder.
6123	60	X	X			3			X				Designation encounters / Xo: This encounter tests that the system will undesignate an intruder that is designated CSPO-3000 but then becomes an invalid track from a loss of a valid Mode S address after the 30 second invalid timeout window. The intruder has active CAS. The geometry of this encounter is a level ownship with a single intruder flying 700 ft. below ownship in the same direction and with the same speed.

Appendix F
F-52

Encounter	Test Group	Mode C	Mode S	ADS-B	ADS-R	Intruder RI	RA (Label 270)	Coordination	Designation/Xo	Multi-Threat	Prescriptive	Aural	Description
6124	60	X	X			3			X				Designation encounters / Xo: This encounter tests that the system will designate a valid intruder as CSPO-3000. The intruder has active CAS. The geometry of this encounter is a level ownship with a single intruder flying 700 ft. below ownship in the same direction and with the same speed.
6125	60	X	X			3			X				Designation encounters / Xo: This encounter tests that the system will remove the designation of an intruder that was both marked CSPO-3000 and had its track dropped after the 30 second invalid timeout window. The intruder has active CAS. The geometry of this encounter is a level ownship with a single intruder flying 700 ft. below ownship in the same direction and with the same speed.
6126	60	X	X			3			X				Designation encounters / Xo: This encounter tests that the system will undesignate a target that is designated as CSPO-3000 after the pilot undesignates the target. The intruder has active CAS. The geometry of this encounter is a level ownship with a single intruder flying 700 ft. below ownship in the same direction and with the same speed.
6127	60		X			3			X		X		Designation encounters / Xo: This encounter tests that the system will undesignate an intruder that is designated CSPO-3000 after the intruder climbs above an altitude of 14,500 ft. The intruder has active CAS. The geometry of this encounter is a level ownship with a single intruder directly above that is slowly climbing away from ownship.
6128	60	X	X			3			X				Designation encounters / Xo: This encounter tests that the system will undesignate an intruder that is both designated CSPO-3000 and in an invalid timeout window after the intruder climbs above an altitude of 14,500 ft. The intruder has active CAS. The geometry of this encounter is a level ownship with a single intruder directly above that is slowly climbing away from ownship.

Encounter	Test Group	Mode C	Mode S	ADS-B	ADS-R	Intruder RI	RA (Label 270)	Coordination	Designation/Xo	Multi-Threat	Prescriptive	Aural	Description
6129	60	X	X			3			X				Designation encounters / Xo: This encounter tests that the system will undesignate an intruder that is both designated CSPO-3000 and in an invalid timeout window after the pilot undesignates the intruder. The intruder has active CAS. The geometry of this encounter is a level ownship with a single intruder directly above that is slowly climbing away from ownship.
6130	60	X	X			3			X				Designation encounters / Xo: This encounter tests that the system will undesignate an intruder that is both designated CSPO-3000 and in an invalid timeout window after the 30 second timeout window expires. The intruder has active CAS. The geometry of this encounter is a level ownship with a single intruder flying 700 ft. above ownship in the same direction and with the same speed.
6131	60	X	X			3			X				Designation encounters / Xo: This encounter tests that the system will maintain the CSPO-3000 designation of an intruder that is both designated CSPO-3000 and in an invalid timeout window after it becomes valid for CSPO-3000. The intruder has active CAS. The geometry of this encounter is a level ownship with a single intruder flying 700 ft. above ownship in the same direction and with the same speed.
6132	60		X			3			X		X		Designation encounters / Xo: This encounter tests that the system will remove the CSPO-3000 designation of an intruder that is both designated CSPO-3000 and has a dropped track after the timeout window expires. The intruder has active CAS. The geometry of this encounter is a level ownship with a single intruder flying 700 ft. above ownship in the same direction and with the same speed.
6133	60		X			3			X		X		Designation encounters / Xo: This encounter tests that the system will maintain the CSPO-3000 designation of an intruder that is both designated CSPO-3000 and has a dropped track after the track of the intruder is reacquired before the invalid timeout window expires. The

Encounter	Test Group	Mode C	Mode S	ADS-B	ADS-R	Intruder RI	RA (Label 270)	Coordination	Designation/Xo	Multi-Threat	Prescriptive	Aural	Description
													intruder has active CAS. The geometry of this encounter is a level ownship with a single intruder flying 700 ft. above ownship in the same direction and with the same speed.
6134	60	X	X			3			X				Designation encounters / Xo: This encounter tests that the system will remove the CSPO-3000 designation of an intruder that is both designated CSPO-3000 and has a dropped track after it is reacquired with an invalid Mode S address and the invalid timer expires. The intruder has active CAS. The geometry of this encounter is a level ownship with a single intruder flying 700 ft. above ownship in the same direction and with the same speed.
6135	60		X			3			X		X		Designation encounters / Xo: This encounter tests that the system will remove the CSPO-3000 designation of an intruder that is both designated CSPO-3000 and has a dropped track after the pilot designates another intruder. The intruder has active CAS. The geometry of this encounter is a level ownship with a single intruder flying 700 ft. above ownship in the same direction and with the same speed.
6136	60		X			3	1000 5001 5301		X		X		Designation encounters / Xo: This encounter tests that the system will enable the CSPO-3000 protection mode on an intruder that is both designated CSPO-3000 and has an active Xa RA after the Xa RA clears. The intruder has active CAS. The geometry of this encounter is a level ownship with a head on intruder.
6137	60		X			3	1000 5001 5301		X		X		Designation encounters / Xo: This encounter tests that the system will not apply the CSPO-3000 protection mode to an intruder that is both designated CSPO-3000 and has an active Xa RA after the pilot removes the CSPO-3000 designation. The intruder has active CAS. The geometry of this encounter is a level ownship with a head on intruder.

Encounter	Test Group	Mode C	Mode S	ADS-B	ADS-R	Intruder RI	RA (Label 270)	Coordination	Designation/Xo	Multi-Threat	Prescriptive	Aural	Description
6138	60	X				3			X		X		Designation encounters / Xo: This encounter tests that the system will enable the CSPO-3000 protection mode on an intruder that is both designated CSPO-3000 and has an active RA being suppressed in DNA mode after the RA clears. The intruder has active CAS. The geometry of this encounter is a level ownship with a head on intruder.
6139	60	X				3	1000 5001		X		X		Designation encounters / Xo: This encounter tests that the system will not apply the CSPO-3000 protection mode to an intruder that is both designated CSPO-3000 and has an active Xa RA being suppressed in DNA mode after the pilot removes the CSPO-3000 designation. The intruder has active CAS. The geometry of this encounter is a level ownship with a head on intruder.
6140	60	X				3	6422		X	X	X		Designation encounters / Xo: This encounter tests that the system will remove a DNA designation for an intruder in which the intruder's track is dropped after the pilot designates another intruder. The intruder has active CAS. The geometry of this encounter is a level ownship with three intruders all head on with one at a high altitude and two at a lower altitude.
6150	60	X	X			0	5001	X	X		X		Designation encounters / Xo: Tests designating a target with DNA. The intruder has no active CAS. The geometry of this encounter is a level ownship with a head on intruder.
6151	60	X	X			0			X				Designation encounters / Xo: Tests that ACAS will drop the designation of a target that is designated DNA and then becomes invalid for DNA, after it has been invalid for 30 seconds. The intruder has no CAS. The geometry of this encounter is a level ownship with a head on intruder.
6152	60	X	X						X				Designation encounters / Xo: Tests that a DNA designated target becomes invalid for DNA when the ownship loses reporting of baro alt. The intruder has no CAS. The geometry of this encounter is a level ownship with a head on intruder 300 ft. below in alt.

Appendix F
F-56

Encounter	Test Group	Mode C	Mode S	ADS-B	ADS-R	Intruder RI	RA (Label 270)	Coordination	Designation/Xo	Multi-Threat	Prescriptive	Aural	Description
6153	60	X	X	X		0	4010		X	X			Designation encounters / Xo: Tests that a DNA-designated target that becomes involved in a multi-threat RA while designated will display the RA even though it is currently DNA. The intruder has no CAS. The geometry of this encounter is a level ownship with two head on intruders one of them 300 ft. below ownship's altitude.
6154	60		X			0			X		X		Designation encounters / Xo: Intruder is designated as DNA. Tests that designation is dropped by ACAS when ownship's radio alt starting above 600 ft, drops below 400 ft. and then climbs above 600 ft. The intruder has no CAS. The geometry of this encounter is a level head on intruder. Ownship descends below 400 ft. then climbs above 650 ft.
6155	60		X			0			X		X		Designation encounters / Xo: Intruder is designated as DNA. Tests that designation is dropped by ACAS after ownship's radio altitude drops below 50ft. The intruder has no CAS. The geometry of this encounter is a level head on intruder with ownship descending below 50 ft.
6156	60		X			0			X		X		Designation encounters / Xo: Intruder is designated as DNA. Tests that ACAS undesignates intruder as intruder's slant range increases beyond 6 NM. The intruder has no CAS. The geometry of this encounter is a level ownship with intruder flying away behind ownship.
6157	60		X	X		0	1000 5001 5301	X	X		X		Designation encounters / Xo: This encounter tests that the system can undesignate a target that is currently designated as DNA. The intruder has no CAS. The geometry of this encounter is a level ownship with a head on intruder.
6158	60	X	X			0			X				Designation encounters / Xo: This encounter tests that the system can transition from a state with an intruder that is both designated DNA and in an invalid timeout window to a state where the intruder becomes valid, preventing the loss of the DNA designation. The intruder has no

Encounter	Test Group	Mode C	Mode S	ADS-B	ADS-R	Intruder RI	RA (Label 270)	Coordination	Designation/Xo	Multi-Threat	Prescriptive	Aural	Description
													CAS. The geometry of this encounter is a level ownship with a head on intruder.
6159	60	X	X			0			X				Designation encounters / Xo: This encounter tests that the system will automatically undesignate an intruder that is both designated DNA and in an invalid timeout window when ownship's radio alt starts above 600 ft., drops below 400 ft., and climbs back above 600 ft. The intruder has no CAS. The geometry of this encounter is a level head on intruder. Ownship descends below 400 ft. then climbs above 650 ft.
6160	60	X	X			0			X				Designation encounters / Xo: This encounter tests that the system will automatically undesignate an intruder that is both designated DNA and in an invalid timeout window when ownship's radio alt starts above 600 ft. and then drops below 50 ft. The intruder has no CAS. The geometry of this encounter is a level head on intruder with ownship descending below 50 ft.
6161	60	X	X			0			X				Designation encounters / Xo: This encounter tests that the system will automatically undesignate an intruder that is both designated DNA and in an invalid timeout window when the intruder moves beyond a slant range of 6 NM. The intruder has no CAS. The geometry of this encounter is a level ownship with intruder flying away behind ownship.
6162	60	X	X			0	1000 4010 4310		X				Designation encounters / Xo: This encounter tests that the system will undesignate an intruder that is both designated DNA and in an invalid timeout window when the pilot undesignates the intruder. The intruder has no CAS. The geometry of this encounter is a level ownship with a head on intruder.
6163	60		X	X		0	1000 4010 6020		X	X	X		Designation encounters / Xo: This encounter tests that the system will set the DNA flag for an intruder that is both marked DNA and in a multi-threat RA after the multi-threat RA clears. The intruder has no CAS. The

Encounter	Test Group	Mode C	Mode S	ADS-B	ADS-R	Intruder RI	RA (Label 270)	Coordination	Designation/Xo	Multi-Threat	Prescriptive	Aural	Description
													geometry of this encounter is a level ownship with two intruders, one head on and the second directly behind overtaking ownship.
6164	60	X	X			0	5001 5101 6422		X	X			Designation encounters / Xo: This encounter tests that the system will clear the designation for an intruder that is both marked DNA and in a multi-threat RA after the intruder becomes invalid for designation from a loss of a valid Mode S address and the 30 second timeout window expires. The intruder has no CAS. The geometry of this encounter is a level ownship with three intruders all head on with one at a high altitude and two at a lower altitude.
6165	60	X	X			0	5001 5101 6422		X	X			Designation encounters / Xo: This encounter tests that the system will clear the designation for an intruder that is both marked DNA and in a multi-threat RA after the intruder's track is dropped and the 30 second timeout window expires. The intruder has no CAS. The geometry of this encounter is a level ownship with three intruders all head on with one at a high altitude and two at a lower altitude.
6166	60	X	X			0	4010		X	X			Designation encounters / Xo: This encounter tests that the system will set the DNA flag for an intruder that is both marked DNA and in a multi-threat RA after undesignated intruder's RA clears. The intruder has no CAS. The geometry of this encounter is a level ownship with two intruders, one is 200 ft. directly below ownship, the other is head 100 ft. above ownship.
6167	60		X			0	5201 4010 4022 5001		X	X	X		Designation encounters / Xo: This encounter tests that the system will automatically clear DNA designation for an intruder that is both marked DNA and in a multi-threat RA after the intruder's slant range becomes larger than 6 NM. The intruder has no CAS. The geometry of this encounter is ownship on a slow descend with two head on intruders, one

Encounter	Test Group	Mode C	Mode S	ADS-B	ADS-R	Intruder RI	RA (Label 270)	Coordination	Designation/Xo	Multi-Threat	Prescriptive	Aural	Description
													above and below descending at the same rate. There is a third intruder flying away from ownship.
6168	60	X	X		0	1000 4010 4310		X	X	X			Designation encounters / Xo: This encounter tests that the system will automatically clear DNA designation for an intruder that is both marked DNA and in a multi-threat RA after the pilot undesignates the intruder. The intruder has no CAS. The geometry of this encounter is a level ownship with two intruders. One intruder is head on and the second intruder is closing in from behind ownship.
6169	60	X			0	1000 5001		X		X			Designation encounters / Xo: The encounter starts with a DNA-designated intruder whose track is dropped (starting the timeout timer). The track is then reacquired but with an invalid Mode S address. The encounter tests that the designation is dropped after the timeout expires. The intruder has no CAS. The geometry of this encounter is a level ownship with a head on intruder 300 feet above ownship.
6170	60	X			0			X		X			Designation encounters / Xo: This encounter tests that the system will maintain a DNA designation for an intruder in which the intruder's track is dropped but then reacquired before the timeout expires. The intruder has no CAS. The geometry of this encounter is a level ownship with a head on intruder 300 feet above ownship.
6171	60	X			0	4010 4310		X		X			Designation encounters / Xo: This encounter tests that the system will not mark an intruder DNA that is designated by the pilot while it has an active CSPO-3000 RA. Before the CSPO-3000 RA clears, the DNA is removed by the pilot. The intruder has no CAS. The geometry of this encounter is a level ownship with a head on intruder 300 ft. below ownship. Ownship slowly overtakes the intruder.
6172	60	X			0	1000 4010		X		X			Designation encounters / Xo: This encounter tests that the system will enable the DNA flag on an intruder that has been designated DNA while having active CSPO-3000 RA after the CSPO-3000 RA clears. The intruder

Appendix F
F-60

Encounter	Test Group	Mode C	Mode S	ADS-B	ADS-R	Intruder RI	RA (Label 270)	Coordination	Designation/Xo	Multi-Threat	Prescriptive	Aural	Description
													has no CAS. The geometry of this encounter is a level ownship with a head on intruder 300 ft. below ownship. Ownship slowly overtakes the intruder.
6173	60	X	X			0			X				Designation encounters / Xo: This encounter tests that the system will undesignate an intruder that is designated CSPO-3000 but then becomes an invalid track from a loss of a valid Mode S address after the 30 second invalid timeout window. The intruder has no CAS. The geometry of this encounter is a level ownship with a single intruder flying 700 ft. below ownship in the same direction and with the same speed.
6174	60	X	X			0			X				Designation encounters / Xo: This encounter tests that the system will designate a valid intruder as CSPO-3000. The intruder has no CAS. The geometry of this encounter is a level ownship with a single intruder flying 700 ft. below ownship in the same direction and with the same speed.
6175	60	X	X			0			X				Designation encounters / Xo: This encounter tests that the system will remove the designation of an intruder that was both marked CSPO-3000 and had its track dropped after the 30 second invalid timeout window. The intruder has no CAS. The geometry of this encounter is a level ownship with a single intruder flying 700 ft. below ownship in the same direction and with the same speed.
6176	60	X	X			0			X				Designation encounters / Xo: This encounter tests that the system will undesignate a target that is designated as CSPO-3000 after the pilot undesignates the target. The intruder has no CAS. The geometry of this encounter is a level ownship with a single intruder flying 700 ft. below ownship in the same direction and with the same speed.
6177	60		X			0			X	X			Designation encounters / Xo: This encounter tests that the system will undesignate an intruder that is designated CSPO-3000 after the intruder climbs above an altitude of 14,500 ft. The intruder has no CAS. The

Encounter	Test Group	Mode C	Mode S	ADS-B	ADS-R	Intruder RI	RA (Label 270)	Coordination	Designation/Xo	Multi-Threat	Prescriptive	Aural	Description
													geometry of this encounter is a level ownship with a single intruder directly above that is slowly climbing away from ownship.
6178	60	X	X			0			X				Designation encounters / Xo: This encounter tests that the system will undesignate an intruder that is both designated CSPO-3000 and in an invalid timeout window after the intruder climbs above an altitude of 14,500 ft. The intruder has no CAS. The geometry of this encounter is a level ownship with a single intruder directly above that is slowly climbing away from ownship.
6179	60	X	X			0			X				Designation encounters / Xo: This encounter tests that the system will undesignate an intruder that is both designated CSPO-3000 and in an invalid timeout window after the pilot undesignates the intruder. The intruder has no CAS. The geometry of this encounter is a level ownship with a single intruder directly above that is slowly climbing away from ownship.
6180	60	X	X			0			X				Designation encounters / Xo: This encounter tests that the system will undesignate an intruder that is both designated CSPO-3000 and in an invalid timeout window after the 30 second timeout window expires. The intruder has no CAS. The geometry of this encounter is a level ownship with a single intruder flying 700 ft. above ownship in the same direction and with the same speed.
6181	60	X	X			0			X				Designation encounters / Xo: This encounter tests that the system will maintain the CSPO-3000 designation of an intruder that is both designated CSPO-3000 and in an invalid timeout window after it becomes valid for CSPO-3000. The intruder has no CAS. The geometry of this encounter is a level ownship with a single intruder flying 700 ft. above ownship in the same direction and with the same speed.
6182	60		X			0			X	X			Designation encounters / Xo: This encounter tests that the system will remove the CSPO-3000 designation of an intruder that is both designated

Appendix F
F-62

Encounter	Test Group	Mode C	Mode S	ADS-B	ADS-R	Intruder RI	RA (Label 270)	Coordination	Designation/Xo	Multi-Threat	Prescriptive	Aural	Description
													CSPO-3000 and has a dropped track after the timeout window expires. The intruder has no CAS. The geometry of this encounter is a level ownship with a single intruder flying 700 ft. above ownship in the same direction and with the same speed.
6183	60	X		0				X		X			Designation encounters / Xo: This encounter tests that the system will maintain the CSPO-3000 designation of an intruder that is both designated CSPO-3000 and has a dropped track after the track of the intruder is reacquired before the invalid timeout window expires. The intruder has no CAS. The geometry of this encounter is a level ownship with a single intruder flying 700 ft. above ownship in the same direction and with the same speed.
6184	60	X	X		0			X					Designation encounters / Xo: This encounter tests that the system will remove the CSPO-3000 designation of an intruder that is both designated CSPO-3000 and has a dropped track after it is reacquired with an invalid Mode S address and the invalid timer expires. The intruder has no CAS. The geometry of this encounter is a level ownship with a single intruder flying 700 ft. above ownship in the same direction and with the same speed.
6185	60		X		0			X		X			Designation encounters / Xo: This encounter tests that the system will remove the CSPO-3000 designation of an intruder that is both designated CSPO-3000 and has a dropped track after the pilot designates another intruder. The intruder has no CAS. The geometry of this encounter is a level ownship with a single intruder flying 700 ft. above ownship in the same direction and with the same speed.
6186	60		X		0	1000 5001 5301		X		X			Designation encounters / Xo: This encounter tests that the system will enable the CSPO-3000 protection mode on an intruder that is both designated CSPO-3000 and has an active Xa RA after the Xa RA clears. The

Encounter	Test Group	Mode C	Mode S	ADS-B	ADS-R	Intruder RI	RA (Label 270)	Coordination	Designation/Xo	Multi-Threat	Prescriptive	Aural	Description
													intruder has no CAS. The geometry of this encounter is a level ownship with a head on intruder.
6187	60	X			0	1000 5001 5301		X		X			Designation encounters / Xo: This encounter tests that the system will not apply the CSPO-3000 protection mode to an intruder that is both designated CSPO-3000 and has an active Xa RA after the pilot removes the CSPO-3000 designation. The intruder has no CAS. The geometry of this encounter is a level ownship with a head on intruder.
6188	60	X			0			X		X			Designation encounters / Xo: This encounter tests that the system will enable the CSPO-3000 protection mode on an intruder that is both designated CSPO-3000 and has an active RA being suppressed in DNA mode after the RA clears. The intruder has no CAS. The geometry of this encounter is a level ownship with a head on intruder.
6189	60	X			0	1000 5001		X		X			Designation encounters / Xo: This encounter tests that the system will not apply the CSPO-3000 protection mode to an intruder that is both designated CSPO-3000 and has an active Xa RA being suppressed in DNA mode after the pilot removes the CSPO-3000 designation. The intruder has no CAS. The geometry of this encounter is a level ownship with a head on intruder.
6190	60	X			0	5001 5101 6422		X	X	X			Designation encounters / Xo: This encounter tests that the system will remove a DNA designation for an intruder in which the intruder's track is dropped after the pilot designates another intruder. The intruder has no CAS. The geometry of this encounter is a level ownship with three intruders all head on with one at a high altitude and two at a lower altitude.

Table F-7: Test Group 70

Encounter	Test Group	Mode C	Mode S	ADS-B	ADS-R	Intruder RI	RA (Label 270)	Coordination	Designation/Xo	Multi-Threat	Prescriptive	Aural	Description
7000	70	X				3	5001 5002 5301 5022		X	X			Branch Coverage.
7001	70	X				3	1000 5001 5002 4210 4020 5301 4310			X			Branch Coverage.
7002	70	X				0	1000 5002			X			Branch Coverage.
7003	70	X				0	1000 4020			X			Branch Coverage.
7004	70	X				3	1000 5001 4210 4410			X			Branch Coverage.
7005	70	X				3	1000 4410 4310			X			Branch Coverage.
7006	70	X				3				X			Branch Coverage.
7007	70	X					1000 4010 4020						Branch Coverage: All intruders are Mode C.

Encounter	Test Group	Mode C	Mode S	ADS-B	ADS-R	Intruder RI	RA (Label 270)	Coordination	Designation/Xo	Multi-Threat	Prescriptive	Aural	Description
7008	70	X					1000 4010						Branch Coverage: All intruders are Mode C.
7009	70	X					5001 5002 5101 5022			X			Branch Coverage: All intruders are Mode C.
7010	70	X											Branch Coverage: All intruders are Mode C.
7011	70	X					1000 5001 5101			X			Branch Coverage: Exercising more than maximum TRM-supported number of intruders. Ownship flying level at 10000 ft and 31 intruders flying head-on at 10000 ft. All intruders are Mode C.
7012	70	X					4010 4310						Branch Coverage: All intruders are Mode C.
7013	70		X			0	1000 4010 4020		X		X		Branch Coverage: Exercises SetTargetDesignationInvalid. dropped target undesignation. DNA undesignation message in the 4 s cycle.
7014	70		X			3					X		Branch Coverage: Exercises ReinitializeRangeTracker. A sequence of DFO messages with NaN baro_alt_ft followed by one with slant_range_ft outlier.
7015	70	X		X									Branch Coverage: Exercises SetDisplayDataPassive. Mode C replies with all baro_alt_ft NaN.
7016	70		X	X		2 3	1000 6002 4010 6422	X		X	X		Branch Coverage: Exercises OffsetTrackByHeading and ReceiveHeadingObservation. All heading_true_rad NaN except for the first.
7017	70		X	X		2 3	1000 6002 4010 6422	X		X	X		Branch Coverage: Exercises ReceiveHeadingObservation. The first instance of heading_degraded set to true.

Appendix F
F-66

Encounter	Test Group	Mode C	Mode S	ADS-B	ADS-R	Intruder RI	RA (Label 270)	Coordination	Designation/Xo	Multi-Threat	Prescriptive	Aural	Description
7018	70		X	X		2 3	1000 6002 4010 6422	X		X	X		Branch Coverage: Exercises ReceiveHeadingObservation. The 3 s cycle heading_degraded set to true.
7019	70		X	X		2 3	1000 6002 4010 6422	X		X	X		Branch Coverage: Exercises ReceiveHeadingObservation. All heading_true_rad and all Chi_rel_rad set to NaN.
7020	70	X	X	X	X	3			X				Branch Coverage: Exercises MergeTargetDesignations. All tracks geometrically identical, differing only in duration. Mode S track 0-10 s, ADS-R non-ICAO 0-50 s, Mode C 20-50 s. Mode S target designated DNA at 4 s.
7021	70		X	X		2 3	1000 6002 4010 6422	X		X	X		Branch Coverage: Exercises SetCoordination. All type_capability set to 4.
7022	70		X	X		2 3	1000 6002 4010 6422	X		X	X		Branch Coverage: Exercises SetCoordination. All daa set to 1 and all ca_operational to 0.
7023	70		X	X		2 3	1000 6002 4010 6422	X		X	X		Branch Coverage: Exercises SetCoordination. All daa set to 2 and all ca_operational to 0.
7024	70		X	X	X	0	1000 5001				X		Branch Coverage: Exercises RemoveStaleTracks. baro_alt_ft for ownship and intruder in the 3 s and 4 s cycles set to NaN and ADS-B messages dropped in cycles 5 s to 8 s. Level-level head-on encounter at 10000 ft.

Encounter	Test Group	Mode C	Mode S	ADS-B	ADS-R	Intruder RI	RA (Label 270)	Coordination	Designation/Xo	Multi-Threat	Prescriptive	Aural	Description
7025	70	X	X	X	X	3			X				Branch Coverage: Exercises MergeTargetDesignations. Intruder 1 begins transmitting DF0 and ADS-R non-ICAO messages at 10 s and stops transmitting DF0 messages at 15 s, at 21 s only the non-ICAO ADS-R track remains. Intruder 2 transmits Mode C messages. Intruder 1 is designated DNA at 15 s. Ownship at 5000 ft flying level and north at 300 ft/s, intruder 1 at 3000 ft flying level and north at 300 ft/s, intruder 2 starts at 2500 ft flying level and north at 300 ft/s, at 10 s starts climbing at 25 ft/s, levels off at 3000 ft at 30 s.
7026	70	X	X	X	X	3			X				Branch Coverage: Exercises MergeTargetDesignations. All tracks are geometrically identical, differing only in duration. Mode S track 0-10 s, ADS-R non-ICAO 0-50 s, Mode C 20-50 s. Mode S target designated DNA at 4 s.
7027	70		X	X		3	1000 4010	X		X			Branch Coverage: Exercises Improper/Degraded Inputs.
7028	70	X		X	X				X				Branch Coverage.
7029	70	X					1000 4010						Branch Coverage: Exercises ReceiveHeadingObservation. All heading_true_rad set to NaN except for one at the 5 s cycle.
7030	70	X					1000 4010						Branch Coverage: Exercises ReceiveHeadingObservationSet. all heading_true_rad set to NaN except for one instance at the 5 s cycle and all Chi_rel_rad set to NaN.
7031	70	X					1000 4010						Branch Coverage: Exercises AltCodeEstimate. Chi_rel_rad set to 0 at the 2 s cycle and coded_alt set to 1 for the 1.0-3 s cycles.
7032	70		X			0	1000 5002			X			Branch Coverage.
7033	70	X				0	1000 4010 4020		X	X			Branch Coverage: Exercises dropped target undesignation.
7034	70	X											Branch Coverage: All intruders are Mode C.

Encounter	Test Group	Mode C	Mode S	ADS-B	ADS-R	Intruder RI	RA (Label 270)	Coordination	Designation/Xo	Multi-Threat	Prescriptive	Aural	Description
7035	70			X	X					X			Branch Coverage: Uses ADS-R non-ICAO messages. Level-level head-on encounter at 10000 ft.
7036	70		X			0			X		X		Branch Coverage: Exercises UpdateDNAValidity. DNA designation outside proximate traffic range.
7037	70			X	X				X		X		Branch Coverage: Exercises ReceiveTargetDesignation. ADS-R non-ICAO messages and intruder designated DNA at 5 s and 6 s cycles. Level-level head-on encounter at 10000 ft.
7038	70		X	X	X	0	5001 5301						Branch Coverage: Exercises EncodeTIDRange. Range must be >12.55 miles while having an RA and reporting TID as TIDA, TIDR and TIDB (tid_type=1). Mode C doesn't build a track at this high closing speed but ADS-B with DFO does. ADS-B doesn't report TID as type 1, but when setting rebroadcast=false with nonICAO=true, it does. Encounter has just enough distance and closing speed to get an RA right outside the 12.55 miles range.
7039	70		X	X		0	5001	X	X		X		Branch Coverage: Exercises UpdateDNAValidity. Target within DNA proximity hysteresis interval. Requires a DNA designated target in ground range 36456.69-36456.84 ft. Occurs at 52 s.
7040	70		X	X	X	0	1000 5001						Branch Coverage: Exercises IsImageTrack, AddDecorrelatedTrackToTarget, CorrelatePosition, GetTracksToDecorrelate, and MergeTargets. Intruder with ADS-R non-ICAO messages. Level-level head-on encounter at 10000 ft.
7041	70		X			0			X		X		Branch Coverage: Exercises UpdateDNAValidity. DNA designated target out of proximate range while landing with NaN radalt.
7042	70		X			0	1000 4010		X		X		Branch Coverage: Exercises SetTargetDesignation. Xo mode invalidation due to ownship altitude while RA is active for the designated intruder.
7043	70		X	X		0	5001 5301				X		Branch Coverage: Exercises EncodeTIDRange. Range must be >12.55 miles while having an RA and reporting TID as TIDA, TIDR and TIDB (tid_type=1). Mode C doesn't build a track at this high closing speed but ADS-B with DFO does. ADS-B doesn't report TID as type 1, but when setting rebroadcast=false

Encounter	Test Group	Mode C	Mode S	ADS-B	ADS-R	Intruder RI	RA (Label 270)	Coordination	Designation/Xo	Multi-Threat	Prescriptive	Aural	Description
													with nonICAO=true, it does. Encounter has just enough distance and closing speed to get an RA right outside the 12.55 miles range.
7044	70			X						X			Branch Coverage: Exercises SetDisplayDataPassive. An encounter with Mode C replies and baro_alt_ft set to NaN.
7045	70	X											Branch Coverage: Exercises AddModeCTrackToReport. Mode C messages arrive before ownship heading is initialized and manual_SL set to 1. Level-level encounter at 10000 ft.
7046	70	X	X			3	5001 5101 5301			X			Branch Coverage: Exercises TID field for multithreat.
7047	70		X	X		3	1000 5002 4410 4210	X		X			Branch Coverage: Exercises ShouldReverse. Head on encounter. Ownship altitude starts at 15025 ft. Intruder altitude starts at 14825 ft. Both planes are climbing at 2000 fpm. Horizontal closing happens at 100 seconds.
7048	70	X					5001 5101 5301			X			Branch Coverage: Exercises TID field for multithreat.
7049	70	X					1000 4010 4020						Branch Coverage: All intruders are Mode C.
7050	70	X					1000 4010			X			Branch Coverage: All intruders are Mode C.
7051	70		X	X	X	0					X		Branch Coverage: Exercises AddModeCTrackToReport. ADS-R intruder with several messages' surv_mode set to 1. Level-level head-on encounter at 10000 ft.
7052	70		X	X		0					X		Branch Coverage: Exercises RemoveStaleTracks. All surv_mode values set to 0. Ownship and intruder begin 80 NM apart. Ownship flies level at 6320 ft,

Appendix F
F-70

Encounter	Test Group	Mode C	Mode S	ADS-B	ADS-R	Intruder RI	RA (Label 270)	Coordination	Designation/Xo	Multi-Threat	Prescriptive	Aural	Description
													toward northeast at 146 kts. Intruder descends from 4700 ft at 19 fps, flying toward northeast at 260 kts.
7053	70	X											Branch Coverage: Exercises UpdateModeCTrack. MODE_C_REPLY messages at time 8 s are 1 ms apart but fall into different cycles. Both aircraft maneuvering on the ground (alt=0) with opflg=false.
7054	70		X	X	X	0	1000 5001				X		Branch Coverage: Exercises AddADSBTrackToReport and CorrelateID. All DF0 surv_mode fields set to 0. Ownship and two intruders start about 0.6 NM apart. Ownship and one intruder fly level at 10000 ft, the other intruder flies level at 11000 ft. Ownship flies due north. Intruders fly due south. Intruders' ADS-B messages are received as rebroadcasts. Intruder at 10000 ft transmits only Mode S messages and intruder at 11000 ft only ADS-B messages.
7055	70		X			0 3	1000 4010 4110	X		X	X		Branch Coverage: Exercises FilterTracksForTRM. Encounter has more than maximum TRM supported number of intruders. Intruder with address 9, ri set to 3 (TCASRA). At second 40, UF16UDS30 coordination message sets vrc to 2. Ownship flying level at 10000 ft and 31 intruders flying head-on at 10000 ft.
7056	70		X	X		0 3	1000 5001 5101 6422 5301	X		X	X		Branch Coverage: Exercises ArbitrateConflictingSenses. Head on encounter with three intruders. Ownship is level at 6000 ft. Intruder 1 is level at 6100 ft. Intruder 2 is level at 5600 ft. Intruder 3 appears between 85 s and 88 s transmitting only ADS-B messages.
7057	70	X									X		Branch Coverage: All intruders are Mode C.
7058	70	X					1000 5001 5101			X			Branch Coverage: Exercises more than maximum TRM supported number of intruders. Ownship flying level at 10000 ft and 31 intruders flying head-on at 10000 ft.
7059	70		X			3	4010	X	X		X		Branch Coverage: Exercises AssociateICAOtoTarget and RemoveStaleTracks. Intruder designated CSPO-3000 at 15 s, DF0 messages between 19 s and 38 s

Encounter	Test Group	Mode C	Mode S	ADS-B	ADS-R	Intruder RI	RA (Label 270)	Coordination	Designation/Xo	Multi-Threat	Prescriptive	Aural	Description
													missing, baro_alt_ft set to NaN at 19 s and 39 s to 49 s and. Overtaking encounter with intruder overtaking ownship. Ownship is flying level at 6000 ft. Intruder is flying level at 5800 ft. Overtake happens at 63 seconds.
7060	70	X				0	1000 4010 4110		X	X	X		Branch Coverage: Exercises FilterTracksForTRM. ri set to 3 (TCASRA) for intruder with address 9. At second 40 intruder is designated CSPO-3000. Ownship flying level at 10000 ft and 31 intruders flying head-on at 10000 ft. Exercising more than max TRM supported number of intruders.
7061	70	X					1000 4010						Branch Coverage: Exercises EncodeTIDAltitude. Ownship and Mode C intruder altitude negative through 12 s. Level-level head-on encounter at 10000 ft.
7062	70	X								X			Branch Coverage: Exercises multiple Mode C intruders with outlier/invalid altitude codes.
7063	70	X								X			Branch Coverage: Exercises multiple Mode C intruders with outlier/invalid altitude codes.
7064	70		X	X		0	4010 4310	X	X	X	X		Branch Coverage: Exercises SetTargetDesignation. DFO baro_alt_ft values set to NaN for intruder with address 2 for 4 s - 9 s. Intruder designated DNA at 5 s. Intruder to invalid at 7 s and multithreat_ra is still true.
7065	70		X	X		0	1000 5001 5301			X			Branch Coverage: Exercises ReceiveDFO. DFO message at 3 s with slant_range_ft set to NaN, Chi_rel_rad to NaN, and baro_alt_ft NaN.
7066	70		X	X		0	1000 5001 5301			X			Branch Coverage: Exercises ReceiveDFO. DFO message at 3 s with slant_range_ft set to NaN, Chi_rel_rad to NaN, and baro_alt_ft NaN and message Mode S address set to 11.
7067	70		X	X		0	1000 5001 5301			X			Branch Coverage: Exercises ReceiveDFO and ReceiveBaroAltObservation. Intruder with an ADS-B track and DFO message at 3 s with slant_range_ft set to NaN, Chi_rel_rad to NaN, and baro_alt_ft NaN.
7068	70	X					5001 5002		X	X			Branch Coverage: All intruders are Mode C.

Appendix F
F-72

Encounter	Test Group	Mode C	Mode S	ADS-B	ADS-R	Intruder RI	RA (Label 270)	Coordination	Designation/Xo	Multi-Threat	Prescriptive	Aural	Description
							5101 5022						
7069	70		X	X		0	5001 5301	X	X	X			Branch Coverage: Exercises SetPendingTargetDesignation. Three designation messages in one cycle at 5 s.
7070	70		X	X		3	5001	X	X	X	X		Branch Coverage: Exercises AdjustTargetDesignationValidity. More than maximum TRM supported number of intruders (31) and designated target is not sent to the TRM for processing.
7071	70		X	X		3	1000 5201 4410 5101 5301	X		X			Branch Coverage.
7072	70		X	X		0	1000 5001 5002 5101			X			Branch Coverage.
7073	70	X					1000 4010 4310			X			Branch Coverage: Exercises ReceiveHeadingObservation. All heading_true_rad set to NaN except for one at the 5 s cycle.
7074	70		X	X	X	0	1000 5001			X			Branch Coverage: Exercising multiple ADS-R tracks. Ownship flying level at 1000 ft and two intruders flying head-on at 10000 and 11000 ft.
7075	70		X	X	X	0					X		Branch Coverage: Exercises ADS-R without matching Mode S track. Ownship flying level at 1000 ft and two intruders flying head-on at 10000 and 11000 ft.
7076	70	X					1000 4010 4310			X			Branch Coverage: Exercises ReceiveHeadingObservation. All heading_true_rad set to NaN except for one instance at the 5 s cycle and all Chi_rel_rad to NaN.

Encounter	Test Group	Mode C	Mode S	ADS-B	ADS-R	Intruder RI	RA (Label 270)	Coordination	Designation/Xo	Multi-Threat	Prescriptive	Aural	Description
7077	70	X					1000 4010			X			Branch Coverage: Exercises AltCodeEstimate. Set Chi_rel_rad to 0 at the 2 s cycle and set coded_alt to 1 for the 1-3 s cycles.
7078	70	X								X			Branch Coverage: Exercises AddModeCTrackToReport. Mode C message arrives before ownship heading is initialized and all manual_SL values are 1. Level-level encounter at 10000 ft.
7079	70		X					X		X			Branch Coverage: Level-level head-on encounter at 10000 ft.
7080	70	X								X			Branch Coverage: Exercises UpdateMode CTrack. Two aircraft maneuvering on the ground (alt=0) with opflg=false. MODE_C_REPLY messages at time 8 s are 1 ms apart but fall into different cycles.
7081	70	X					1000 4010			X			Branch Coverage: Exercises EncodeTIDAltitude. Ownship and Mode C intruder have negative altitude through 12 s. Level-level head-on encounter at 10000 ft.
7082	70		X	X		3			X		X		Branch Coverage: Overtaking encounter with intruder overtaking Ownship. Ownship starts 2325 ft descending at a constant rate till 970 ft then climbs till the end of the encounter. Intruder starts level at 1850 ft then starts descending after 50 seconds for the remainder of the encounter.
7083	70		X	X		0 3					X		Branch Coverage: Ownship and Intruder begin 80 NM apart. Ownship flies level at 6320 ft, toward northeast at 146 kts. Intruder descends from 4700 ft at 19 fps, flying toward northeast at 260 kts.
7084	70	X									X		Branch Coverage: Exercises DroppedIntrudersAdjustment. Ownship starts just below nars_threshold and climbs just above it. Intruder is NAR and triggers a TA after ownship climbs above nars_threshold. Overtaking encounter with intruder overtaking ownship.
7085	70	X									X		Branch Coverage: Exercises GillhamDecode. Last coded_alt set to 4096. Multiple Mode C intruders with 100 fps climb rate.
7086	70		X			3					X		Branch Coverage: Exercises SlantRange. More than maximum TRM supported number of intruders. One intruder has invalid altitude.

Encounter	Test Group	Mode C	Mode S	ADS-B	ADS-R	Intruder RI	RA (Label 270)	Coordination	Designation/Xo	Multi-Threat	Prescriptive	Aural	Description
7087	70	X					1000 4010			X			Branch Coverage: Exercises EncodeTIDAltitude. Ownship and Mode C intruder have negative altitudes through 12 s. Level-level head-on encounter at 10000 ft.
7088	70		X			0	1000 5001 5301			X			Branch Coverage: Exercises slant_range_ft=NaN at track initialization. Level-level head-on encounter at 10000 ft.
7089	70		X			0	5001 5301	X		X			Branch Coverage: Exercises UF16UDS30 message without matching Mode S address track. Level-level head-on encounter at 10000 ft.
7090	70			X						X			Branch Coverage: Exercises multiple intruders without RA.
7091	70		X			0	1000 5001 5301			X			Branch Coverage: Exercises target designation DESIGNATION_GLOBAL_TARA. Level-level head-on encounter at 10000 ft.
7092	70		X			0	5001			X			Branch Coverage: Exercises simultaneous outlier detection count limit reached for altitude and bearing measurements. Level-level head-on encounter at 10000 ft.
7093	70	X											Branch Coverage: Multiple Mode C intruders.
7094	70	X											Branch Coverage: Multiple Mode C intruders.
7095	70	X											Branch Coverage: Exercises Mode C track initialization and processing for intruder with undefined bearing. Multiple Mode C intruders.
7096	70		X	X		3			X	X			Branch Coverage: Exercises AdvanceMode STrackFile. DFO slant_range_ft set to NaN for three seconds, starting at 60282.999. Ownship flies level at 5800 ft. Intruder flies level at 6000 ft. Intruder approaches ownship from a bearing of 90 degrees.
7097	70		X	X		2 3	1000 6002	X	X	X			Branch Coverage: Exercises UpdateAdsbQualityHistory. All sda values set to 0
7098	70	X											Branch Coverage: Exercises GillhamDecode. Last coded_alt changed to 4096. Multiple Mode C intruders with 100 fps climb rate.

Encounter	Test Group	Mode C	Mode S	ADS-B	ADS-R	Intruder RI	RA (Label 270)	Coordination	Designation/Xo	Multi-Threat	Prescriptive	Aural	Description
7099	70	X				0			X		X		Branch Coverage: Exercises CheckTargetDesignationTimers. Dropped target undesignation combined with surv_mode set to 1 and DF0 messages missing in cycles 29 s - 34 s.
7100	70	X											Branch Coverage: Exercises DroppedIntrudersAdjustment. Ownship starts just below nars_threshold and climbs just above it. Intruder is NAR and triggers a TA after ownship climbs above nars_threshold. Overtaking encounter with intruder overtaking ownship.
7101	70	X	X			3		X	X		X		Branch Coverage: Exercises UpdateDroppedTargetDesignationState. Intruder designated DNA immediately followed by BARO_ALT_OBS message with toa and baro_alt_ft set to NaN.
7102	70	X	X			3	1000 4010	X	X		X		Branch Coverage: Exercises UpdateDroppedTargetDesignationState. Intruder designated DNA, undesignated 2 s later and immediately followed by BARO_ALT_OBS message with toa and baro_alt_ft set to NaN.
7103	70	X	X	X		0	1000 5001						Branch Coverage: Exercises MergeTargets. Uses ADS-B non-icao messages, which should not be valid per MOPS. Level-level head-on encounter at 10000 ft.
7104	70	X	X	X		0	5001		X				Branch Coverage: Exercises SetIntruderDesignationData. Uses ADS-B non-icao messages, which should not be valid per MOPS. Level-level head-on encounter at 10000 ft. Intruder designated DNA in the 6 s cycle.
7105	70	X	X			0	5001 5301				X		Branch Coverage: Exercises EncodeTIDRange. Uses ADS-B non-icao messages, which should not be valid per MOPS. Range must be >12.55 miles while having an RA and reporting TID as TIDA, TIDR and TIDB (tid_type=1). Mode C doesn't build a track at this high closing speed but ADS-B with DF0 does. ADS-B doesn't report TID as type 1, but when setting rebroadcast=false with nonICAO=true, it does. Encounter has just enough distance and closing speed to get an RA right outside the 12.55 miles range.
7106	70	X	X	X		0	1000 5001						Branch Coverage: Exercises CorrelatePosition. Uses ADS-B non-icao messages, which should not be valid per MOPS. The first message for the

Appendix F
F-76

Encounter	Test Group	Mode C	Mode S	ADS-B	ADS-R	Intruder RI	RA (Label 270)	Coordination	Designation/Xo	Multi-Threat	Prescriptive	Aural	Description
													ADS-R track has a latitude that is 1 degree grater the ADS-B reported latitude. Level-level head-on encounter at 10000 ft.
7107	70	X	X	X	0	1000 5001	X						Branch Coverage: Exercises MergeTargets. Uses ADS-B non-icao messages, which should not be valid per MOPS. Coordination message at second 3. Level-level head-on encounter at 10000 ft.

APPENDIX G VARIABLE VALIDATION INTERVALS

G.1

Derivation of the Revalidation Time Interval

This appendix was originally developed for the minimum performance specification of hybrid surveillance, RTCA/DO-300A. The original safety analysis supporting hybrid surveillance is documented in RTCA/DO-300A Appendix D. Updates to the safety analysis are provided in Appendix C of this MOPS.

This appendix provides the derivation for the equation and table in §2.2.4.6.4.2.3.1.10 (Revalidation) that are used to determine the time interval until the next revalidation attempt.

If the passive position reports remain reasonably close to the true position of the intruder, the normal case, then the transition to active surveillance will occur prior to the earliest possible time for a TA against the intruder. Any safety concerns for that condition are addressed by the safety analysis in RTCA/DO-300A, Appendix D. This analysis considers the case where, subsequent to a successful revalidation, the intruder accelerates toward the TCAS aircraft while at the same time its passive reports diverge from its true position in such a manner that they indicate that the intruder is less of a potential threat than it really is. The analysis assumes that nothing is known about the possible failure modes of the passive range reports, and so makes no assumption about how far they might deviate from the true position of the intruder. All that is known is that the last revalidation was successful, and so at that time the passive range was close to the active range. It is also assumed that, given a successful revalidation, the range rate estimated from previous passive position reports is roughly accurate. Finally, it is assumed that the TCAS aircraft and the intruder can accelerate toward each other at an acceleration of at most $1/3\text{ g}$ (11 ft/s^2). That limit on the acceleration sets a lower bound on how soon the intruder can physically satisfy the range tau requirements for the transition to active surveillance. If the next revalidation occurs at or prior to that lower bound, any divergence between the true and reported range will be caught before active surveillance is required. The $1/3\text{ g}$ value was chosen by the SC-147 Surveillance Working Group because that is the lateral acceleration assumed in the calculation of the TCAS threat detection parameter DMOD, which makes a similar worst-case acceleration argument.

Two models were examined for calculating the worst-case (shortest) time interval from the time of a successful validation until the intruder could pass the range and range rate condition for transition from passive to active surveillance. The simpler model assumes that the TCAS aircraft and the intruder can simply accelerate in range directly toward each other at a constant 11 ft/s^2 , up to a maximum closing speed of 1200 knots. Such constant acceleration is not realistic in many aircraft geometries, as aircraft typically accelerate by turning rather than by substantial speed changes. However, when a more realistic model was evaluated that modeled both aircraft turning toward each other, each with a lateral acceleration of $11/2\text{ ft/s}^2$, identical results were obtained given the quantization used in the table. For assumed total accelerations on the order of 1 g (32 ft/s^2) (lateral accelerations of $\frac{1}{2}\text{ g}$) the more complicated model allowed a few more cells of the range and range rate space to go 10 seconds longer between validations than the simpler model. However, for the $1/3\text{ g}$ (11 ft/s^2) acceleration assumption, the two models give identical results. For that reason, only the simpler model will be presented.

The model assumes that the estimated range and range rate at the time of successful revalidation, which are based upon passive reports, are accurate because of the successful

validation. The range must be accurate within the validation tolerance or revalidation would not succeed and the track would transition to active surveillance. The argument for the range rate being accurate is more indirect. It is based on the likelihood that if the current passive range measurement is accurate, then the measurements in the recent past that primarily determine the range rate estimate are also likely to have been accurate. That is enforced to some degree by the requirement that a passive range report must fall within a correlation window centered about the predicted range of the track, where that predicted range is based on the previous range and range rate estimates. If a passive range report does not fall within the correlation window, the track will coast. If it coasts repeatedly, it will transition to active surveillance. Thus, there is a limit on how much the range reports are allowed to jump around from update to update.

It is assumed that the TCAS aircraft and the intruder can simply accelerate in range directly toward each other at a constant 11 ft/s^2 , up to a maximum closing speed of 1200 knots. The range equation is then:

$$r(t) = r_0 + v_0 t_a + 0.5 a t_a^2 + v_{\max} (t - t_a)$$

where:

t is the time since the last validation (s),
 r_0 is the range estimate at validation (ft),
 v_0 is the range rate estimate at validation (ft/s),
 a is the assumed acceleration, -11 ft/s^2 ,
 v_{\max} is the maximum assumed closure rate, -2025 ft/s (-1200 knots), and
 $t_a = \min(t, (v_{\max} - v_0)/a)$ is the length of the acceleration interval.

The corresponding velocity equation is:

$$v(t) = v_0 + a t_a$$

The range tau condition for transitioning to active surveillance is then:

$$\tau(t) = -\frac{(r(t) - s_{\text{mod}})}{\min(v(t), v_{\min})} \leq t_{\text{thr}}$$

where:

s_{mod} is a range offset of 18228 ft (3 NM) that is used in the range condition for transition to active surveillance,
 v_{\min} is a minimum closure rate assumed in the tau calculation, -10.13 ft/s (-6 kt), and
 t_{thr} is the range tau threshold value of 60 seconds for transitioning to active surveillance.

To create Table 2-19 in §2.2.4.6.4.2.3.1.10, $\tau(t)$ was calculated at projection times t of 10, 20, 30, 40 and 60 seconds for values of r_0 of 3 through 30 NM in increments of 1 NM and for values of v_0 from -1200 to +1200 knots in increments of 100 knots. For each combination of r_0 and v_0 the longest of those projection times that resulted in a $\tau(t)$ of

greater than or equal to the 60 second threshold t_{thr} was entered into the corresponding table cell.

Each cell in the table represents a range of values for r_0 and v_0 that is 1 NM by 100 kts in size. Some cells in the table are such that all values for r_0 and v_0 in that cell satisfy the range tau condition for the transition to active surveillance. Those cells are marked with an ‘A’. Other cells contain both values that are within the active surveillance region and those that are not. However, there is no reason for the table to be accessed for range and range rate values that require a transition to active surveillance. The times in those cells are for the r_0 and v_0 values that fall outside the active surveillance region.

For some combinations of r_0 and v_0 near the boundary of the active surveillance region, the simple acceleration model shows that there is not even 10 seconds before the intruder might enter the active surveillance region. Rather than specify a variety of shorter revalidation intervals, which would lead to excessive interrogation rates, such table cells have been set to the minimum value of 10 seconds. An intruder that is in one of those cells at the last successful revalidation and whose passive position reports continue to be accurate, the normal case, will transition to active surveillance at the appropriate time. However, if that intruder has the kind of failure analyzed here, that failure will not be detected for 10 seconds, at the next revalidation attempt. The intruder will then be inside the nominal range and range rate boundary for the transition to active surveillance, but by at most 10 seconds.

Instead of using a pre-calculated table of validation intervals as presented above, one might instead wish to calculate the maximum safe validation interval in real time. The necessary equations are derived below from the τ equation and the variable definitions used above, specifically:

$$\tau(t) = -\frac{r_0 + v_0 t_a + 0.5 a t_a^2 + v_{max}(t - t_a) - s_{mod}}{\min(v_0 + a t_a, v_{min})} \leq t_{thr}$$

where:

t is the time since the last successful validation (s),

r_0 is the range estimate at validation (ft),

v_0 is the range rate estimate at validation (ft/s),

a is the assumed acceleration, -11 ft/s²,

v_{max} is the maximum assumed closure rate, -2025 ft/s (-1200 knots),

v_{min} is the minimum closure rate assumed in the τ calculation, -10.13 ft/s (-6 kt),

$t_a = \min(t, (v_{max} - v_0)/a)$ is the length of the acceleration interval in seconds,

s_{mod} is a range offset of 18228 ft (3 NM), and

t_{thr} is the range tau threshold value of 60 seconds for transitioning to active surveillance.

In principle, it is necessary to use multiple equations in the real time calculations because of the $\min()$ function used in the definitions of $\tau(t)$ and t_a . The timeline must be divided into two or three segments, depending on the initial range rate v_0 . Within each of these time segments the results of the $\min()$ functions can be reduced to single values or expressions. The first time segment applies only if the initial range rate is more positive than v_{min} (-6 kt),

so that the intruder is either moving away from the TCAS aircraft or converging more slowly than v_{min} , and covers the time interval during which the intruder accelerates toward the TCAS aircraft until it is converging at a rate of v_{min} . The second time segment applies from the time the intruder is converging at or faster than v_{min} until the time that the intruder has accelerated to the maximum closure rate of v_{max} . The third segment applies after the intruder has reached a closure rate of v_{max} and so has stopped accelerating. It is not known a priori which of these segments will contain the time when the condition $\tau(t) = 60$ seconds is satisfied, although it will be shown below that the first time segment can be ignored, and that the third time segment applies only at large initial ranges r_0 and high initial closure rates v_0 , resulting in most solutions falling during the second time segment. It will be argued that the equation for the second time segment can also be used for the third time segment without any loss of safety and very little increase in interrogations. Therefore, the final result is that only a single equation, for the second time segment, needs to be solved.

The first time segment is only applicable if $v_0 > v_{min}$. If the first time segment is applicable, it will end at time $t_1 = (v_{min} - v_0)/a$, the time it takes to accelerate from v_0 to v_{min} . During that time interval, the τ equation can be simplified by using the fixed constant closure rate of v_{min} (-6 kt) and by ignoring the term that calculates the range closure after the acceleration has stopped upon reaching the range rate v_{max} . The following simplified τ equation is the result:

$$\tau(t) = -\frac{r_0 + v_0 t + 0.5at^2 - s_{mod}}{v_{min}}.$$

If $\tau(t_1) \leq t_{thr}$, then the maximum safe interval until the next validation falls somewhere during the first time segment. The appropriate time t can be found by solving the quadratic equation $\tau(t) = t_{thr}$ for t , giving:

$$t = \frac{-v_0 \pm \sqrt{v_0^2 - 2a(r_0 + v_{min}t_{thr} - s_{mod})}}{a}.$$

That equation has two solutions, but only the more positive one (obtained when using the minus sign) is of interest. The other solution will give a negative value for t . [To see this, note that the expression $(r_0 + v_{min}t_{thr} - s_{mod})$ will always be positive in value, as otherwise the intruder would already satisfy the τ condition for active surveillance. Therefore the expression inside the square root will always give a positive value that is larger than v_0^2 and a positive real root that is larger than v_0 .]

However, because of the small constant range rate v_{min} used in the τ equation during that first time segment, $\tau(t) = 60$ can only occur during that first time segment for a very small range of initial conditions r_0 and v_0 , such a small range that this version of the τ equation can be ignored for practical purposes.

In order to see why that is so, observe that the range versus time trajectory is a parabola opening downward with a fixed shape due to the fixed acceleration. Its location in range versus time space is determined by r_0 (the range at time 0) and v_0 (the slope at time 0). For $\tau(t) = t_{thr}$ to hold, the range at time t must be $s_{mod} - v_{min} t_{thr} = 3.1\text{NM}$, and therefore $r_0 + v_0 t + 0.5at^2 = 3.1\text{NM}$. The initial range r_0 must be greater than 3.1 NM in order that the active interrogation conditions not apply at the time of validation. v_0 must be more positive than v_{min} in order for the first time segment to be applicable. For the same reason, the range rate at time t must not be more negative than v_{min} . Given those constraints, the largest possible r_0 occurs when $v_0 = 0$ (the top of the parabola is reached just at time 0) and the range rate just reaches v_{min} at the same time that $\tau(t) = 60$ is satisfied. Since it takes only 0.92 seconds to accelerate to v_{min} at $a = -11 \text{ ft/s}^2$, that gives a maximum r_0 that is only a few feet larger than the minimum value of 3.1 NM.

The maximum possible v_0 will occur when $r_0 = 3.1 \text{ NM}$ on the outbound leg of the parabola. Since the range trajectory is a parabola, when it reaches 3.1 NM and $\tau(t) = t_{thr}$ is satisfied on the inbound leg, the range rate will be the negative of the initial range rate v_0 . That cannot be more negative than v_{min} , or the conditions for the first time segment would no longer apply, and so v_0 cannot be greater than $-v_{min}$ or $+6 \text{ kt}$. At larger initial range rates, the range rate would become more negative than v_{min} before the tau condition is satisfied, and so the tau equation for the first time segment would not be applicable.

Since the first time segment will give a valid solution over such a narrow range of initial conditions, and since any such solution will give a safe validation interval much smaller than 10 seconds, the first time segment can be ignored in practice, as the tau equation for the second time segment, presented below, gives almost the same validation interval under those initial conditions. The two equations give identical results if the actual range rate just reaches v_{min} at the same time that $\tau(t) = t_{thr}$. For example, for $v_0 = +6 \text{ kt}$ and $r_0 = 3.1 \text{ NM}$, both equations give a solution of $t = t_1 = 1.84 \text{ s}$.

The second time segment starts at t_1 and ends when the intruder reaches the maximum closure rate v_{max} , and so it ends at $t_2 = (v_{max} - v_0)/a$. During the second time segment the following simplified version of the equation for $\tau(t)$ applies:

$$\tau(t) = -\frac{r_0 + v_0 t + 0.5at^2 - s_{mod}}{v_0 + at}.$$

If $\tau(t_2) \leq t_{thr}$, then the maximum safe interval until the next validation falls somewhere during the second time segment. The appropriate time t can be found by solving the quadratic equation $\tau(t) = t_{thr}$ for t , giving:

$$t = \frac{-(v_0 + at_{thr}) \pm \sqrt{(v_0 + at_{thr})^2 - 2a(r_0 + v_0 t_{thr} - s_{mod})}}{a}$$

Once again that gives two solutions but only the more positive one (obtained when using the minus sign) is of interest for the same reasons described above. One probably does not want to pre-test the condition $\tau_{\text{au}}(t_2) \leq t_{\text{thr}}$ but rather simply calculate t and determine if it falls between zero and t_2 in order to verify that it is a valid solution. If the solution does not fall within the second time segment and $t_2 \geq 60$ then the validation interval can be set to 60 seconds without further calculation.

During the third time segment, the intruder has reached the maximum closure rate and has stopped accelerating. The simplified equation for $\tau_{\text{au}}(t)$ during the third time segment is:

$$\tau_{\text{au}}(t) = -\frac{r_0 - (1/2a)(v_0 - v_{\max})^2 + v_{\max}t - s_{\text{mod}}}{v_{\max}}.$$

Solving that version of the τ_{au} equation for $\tau_{\text{au}}(t) = t_{\text{thr}}$ gives:

$$t = -\frac{r_0 - (1/2a)(v_0 - v_{\max})^2 - s_{\text{mod}}}{v_{\max}} - t_{\text{thr}}.$$

If a valid solution is not found in the first two time segments, then this equation provides the desired answer.

Note that since the intruder has a range greater than 3.1 NM (otherwise it would satisfy the conditions for active interrogation), if it has a range rate of greater than or equal to +300 kt, a validation interval of 60 seconds will always be chosen. That can be seen by solving for t using the equation for the second time segment with $v_0 = +300$ kt and $r_0 = 3.1$ NM, which gives a time of 62 seconds for the maximum safe interval between validations. Larger values of r_0 and v_0 will give larger safe intervals, all of which will be truncated to 60 seconds.

Given all the above, an algorithm for computing the validation interval given that the intruder does not satisfy the conditions for active interrogation might be:

1. If $v_0 \geq +300$ kt then set the validation interval to 60 seconds.
2. Otherwise, calculate $t_2 = (v_{\max} - v_0)/a$ and

$$t = \frac{-(v_0 + at_{\text{thr}}) - \sqrt{(v_0 + at_{\text{thr}})^2 - 2a(r_0 + v_0 t_{\text{thr}} - s_{\text{mod}})}}{a}.$$

If $0 < t \leq t_2$ then t is a valid solution. It must be truncated to the next smaller integer number of seconds. If less than 10 seconds it must be set to the minimum validation interval of 10 seconds. If greater than 60 seconds it must be set to the maximum validation interval of 60 seconds. It could optionally be further quantized to the set of values 10, 20, 30, 40 and 60 seconds used in the table of validation intervals.

3. If, in step 2, $t > t_2$, but $t_2 \geq 60$, then the validation interval can be set to 60 seconds, as any valid solution for the third time segment must have a value greater than t_2 .

4. If, in step 2, $t > t_2$, and $t_2 < 60$, then it is necessary to compute

$$t = -\frac{r_0 - (1/2)a(v_0 - v_{max})^2 - s_{mod}}{v_{max}} - t_{thr}$$

and that gives the correct solution for t . The resulting value must then be quantized to the appropriate integer value as described in step 2.

A calculation similar to that described above was carried out in Excel for each entry in a range versus range rate table like Table 2-19 above. The results were quantized to the values 10, 20, 30, 40 and 60 seconds used in that table. The resulting table was then compared to Table 2-19. The two tables were identical.

Another observation is that the equation in step 2 of the above procedure could also be used for the third time segment, ignoring the t_2 boundary. That would correspond to the assumption that the intruder's acceleration does not stop when the closure rate reaches 1200 kt, allowing the closure rate to grow larger than that. The resulting times will be smaller than the times calculated by the above procedure, and so will lead to shorter revalidation times. Hence, safety will not be compromised, but slightly higher interrogation rates will result. However, the equation in step 4 of the above procedure is only used for ten cells of Table 2-19 that represent intruders at ranges at or beyond 24 NM and with closure rates of 1000 kt or greater. In three of those cells the calculated safe revalidation intervals are 10 seconds or smaller, and so the minimum value of 10 seconds would be used, and the remaining cells have values less than 21 seconds. If the 10-20-30-40-60 second quantization scheme is used as in Table 2-19, eight of those cells are quantized to 10 seconds and the remaining two cells are set to 20 seconds. If the equation in step 2 is used instead, and the results are quantized as in Table 2-19, only the one cell at 30 NM and -1200 kt would change, from 20 seconds to 10 seconds. Therefore it is attractive to trade a very small increase in interrogation rates for a simplification that eliminates steps 3 and 4 of the above procedure. That additional simplification has been adopted in the equation presented in §2.2.4.6.4.2.3.1.10.

This Page Intentionally Left Blank

APPENDIX H 1090 MHz SPECTRUM REDUCTION ANALYSIS

H.1

Introduction

This specification includes hybrid surveillance RTCA/DO-300A requirements which significantly reduce the utilization of 1030 MHz and 1090 MHz by ACAS X. This appendix is included unchanged from RTCA/DO-300A. It documents the simulation, improvements and associated rationale achieved by RTCA/DO-300A hybrid surveillance. These improvements are also achieved by ACAS X.

The principle motivation for updating RTCA/DO-300 to Version A, the version that is also used for ACAS X, was to modify the Hybrid Surveillance algorithms to reduce the TCAS use of the 1090 MHz channel. A number of algorithm changes were proposed by members of the SC-147 Surveillance Working Group (SWG). These proposed changes were evaluated using a Lincoln Laboratory TCAS surveillance simulation to understand the associated impact on the 1090 MHz channel. This appendix provides a high level description of the changes to the surveillance algorithms, the simulation used to evaluate these changes, and the results produced by the simulation.

H.2

Simulation and Input Data Set

A high fidelity TCAS surveillance simulation has been developed and updated over the last two decades to analyze the impact of changes to the TCAS surveillance algorithms. The simulation takes as input aircraft position tracks with associated transponder equipages (i.e., ATCRBS, Mode S only, and Mode S with TCAS). The surveillance activity for each TCAS aircraft present in the dataset is simulated in one second intervals. The TCAS surveillance metrics of interest are output for each aircraft, such as number of interrogations and surveillance range for TCAS aircraft and reply rates and transponder utilization for victim transponders.

The most important areas for reduction in RF channel usage are dense terminal regions. Hybrid Surveillance can make the greatest impact here as these areas experience particularly high levels of RF interference. The analysis was therefore focused on the New York City area as this airspace is one of the most dense aircraft environments in the country. An input data set was generated from radar tracks covering a circular area centered around the JFK radar with a radius of 120 NM. These radar tracks were taken from Sunday, November 29, 2009 between 17:00 and 18:00 EST as this hour was observed to be of particularly high Mode S density.

The following assumptions were made in this TCAS-based simulation for the analysis:

1. All TCAS in the simulated scenario are equipped with the surveillance modification under test.
2. For the Hybrid Surveillance modifications, all Mode S aircraft are ADS-B equipped.
3. TCAS does not perform altitude monitoring using DF=0 and DF=4 FRUIT messages (§2.2.4.6.4.2.1).
4. There is no upper limit on the number of aircraft TCAS instantaneously tracks.

5. Information in airborne position messages (lat/lon/alt) is always valid and is always accurate.
6. Own position information (lat/lon/alt) is always available and is always accurate.
7. All Mode S aircraft with 1090ES are crosslink capable.
8. Any aircraft marked as being non-crosslink capable is tracked actively.
9. Aircraft on the ground at an airport are able to communicate with all other aircraft on the ground at the same airport.

H.3 Evaluate Algorithm Changes

Many changes to the surveillance algorithms were proposed by members of the SWG. A number of these proposed changes were removed from consideration due to minimal impact on the 1090 MHz channel and/or difficulty to implement. The following subsections summarize those algorithm changes that were picked to be incorporated into Change 2 of RTCA/DO-185B and Version A of RTCA/DO-300.

H.3.1 Algorithm Changes to RTCA/DO-185B

H.3.1.1 Reduced Surveillance Volume when Operating on Airport Surface

There were two parts that contributed to this algorithm change. The first part was a change that reduced the full TCAS interrogation power by 10 dB when TCAS is powered on. Prior to RTCA/DO-185B Change 2, TCAS units were initialized to full interrogation power (maximum surveillance range) when TCAS was first powered-on. This initialization value lengthened the amount of time required for a newly powered-on TCAS in a busy environment to decrease its power, through interference limiting, to the proper level. For example, it would take a TCAS unit 104 seconds (13 steps \times 8 seconds freeze time per step) to reach the maximum 13 dB on-ground attenuation in a busy terminal environment. While the TCAS unit was “frozen” between eight second steps, interrogations would have been sent to all aircraft within communication range, resulting in increased utilization of those aircraft’s transponders. The issue was exacerbated by the fact that aircraft powering on TCAS on the ground are usually located at the peak of local aircraft density (the busy airport surface).

The second part of this algorithm change was to reduce the relative altitude threshold for targets of interest while TCAS is operating on the ground. While it was intended for TCAS to be activated only immediately prior to taking the active runway, analysis of surveillance data had indicated that many aircraft were activating their TCAS well before reaching the active runway (Ref. V, Appendix B). Although RTCA/DO-185B limits TCAS interrogations more significantly while ownship is on the ground than when airborne, it still requires a TCAS to interrogate aircraft that are within $\pm 10,000$ ft of ownship, even when using interference limiting algorithms. Reduction of the TCAS altitude surveillance volume to ± 3000 ft while on the ground was found to reduce unnecessary interrogations without significant loss of traffic situation awareness.

H.3.1.2 Eliminating Range Monitoring Interrogations to On-Ground TCAS Aircraft when Own TCAS is on the Ground

The interference limiting algorithms described in RTCA/DO-185B require that a TCAS unit adjust its receiver sensitivity and interrogation power level based upon, among other

parameters, the number of other TCAS-equipped aircraft within 3 and 6 NM (NTA3 and NTA6). A TCAS unit above 2000 ft AGL includes only airborne TCAS in the NTA3 and NTA6 values; a TCAS unit \leq 2000 ft AGL includes both airborne and on-ground TCAS in the NTA3 and NTA6 values. TCAS typically determines these values by simply counting the number of TCAS-equipped aircraft that are currently under track and within the specified range. However, RTCA/DO-185B does not allow TCAS to actively track other aircraft on the ground. To include TCAS aircraft operating on the airport surface in the NTA3 and NTA6 calculations, a TCAS unit \leq 2000 ft AGL is required to monitor other TCAS aircraft on the ground through an interrogation every 5 seconds. It was found that when ownship is on the ground, these monitoring interrogations for determining NTA3 and NTA6 were an unnecessary use of the RF spectrum as other interference limiting procedures exist to restrict interrogations by TCAS operating on the airport surface. This modification precludes TCAS operating on the airport surface from interrogating other TCAS operating on the airport surface.

H.3.2 Algorithm Changes to RTCA/DO-300

H.3.2.1 Passively Monitor TCAS Aircraft when Below 2000 ft AGL

This algorithm change was based on the same principle described in §H.3.1.2 where RTCA/DO-185B required TCAS below 2000 ft to interrogate other TCAS aircraft on the ground for the sole purpose of determining NTA3 and NTA6. With the change in §H.3.1.2, when own TCAS is on the ground, these interrogations were eliminated to reduce TCAS use of the 1090 MHz channel. However, a TCAS with Hybrid Surveillance could continue to monitor such targets through the passive reception of ADS-B messages. Therefore a change was made to the TCAS Hybrid Surveillance algorithms such that monitoring interrogations for the calculation of NTA3 and NTA6 would be replaced with the passive monitoring of these targets.

H.3.2.2 Two Validation Attempts

To ensure that a target would always be tracked actively prior to an alert, the original version of RTCA/DO-300 required that any failure to validate a passive track would require an immediate transition to active tracking. The cause for such a failure could include TCAS not receiving a reply to the validation interrogation. Since failing to receive a reply to an interrogation attempt is not uncommon, it was observed that this safeguard could prematurely and undesirably require TCAS to transition a valid passive target to active tracking. To avoid passive tracks unnecessarily transitioning to active tracking, the validation algorithms were changed such that the transition to active tracking would only occur if validation replies were not received for multiple consecutive surveillance cycles. The simulation performed for this analysis assumed that this transition would occur after two consecutive surveillance cycles; however the requirement specified in §2.2.4.6.4.2.3.1.10 of this document allows validation to be attempted in up to five consecutive surveillance cycles. The simulation results remain valid for the validation attempt within 5 instead of within 2 consecutive surveillance cycles. The difference between the requirement and the simulation was considered very small and beneficial in the sense that it produces less spectrum congestion.

H.3.2.3 Variable Validation Rate

The original version of RTCA/DO-300 required that all passively tracked targets have their ADS-B position information periodically validated with a TCAS interrogation. The requirements were that an ADS-B equipped target close in either range or altitude (but not both) be validated once every 10 seconds while a target neither close in range nor altitude be validated once every 60 seconds. It was determined that a reduction in TCAS use of the

1090 MHz channel could be achieved by simply modifying these validation rates. New logic to determine the validation rate for a passive track was developed to still remain within the constraints of the existing RTCA/DO-300 safety study.

Notionally, this new logic is the following: if an ADS-B equipped target is close in both altitude and range, the target should be tracked with active surveillance (i.e. only use TCAS active interrogations to calculate the relative position of the target). If the target is not close in altitude, then the target should be validated once every 60 seconds. If the target is close in altitude but not range, the target should be validated at a rate between 10 and 60 seconds with this variable rate based on a ‘worst case’ calculation. This calculation is considered ‘worst case’ as it assumes the other aircraft will accelerate toward ownship with the TCAS MOPS defined maximum for combined acceleration of $1/3\text{ g}$. The variable validation rate is given in this document as both an equation and a table in §2.2.4.6.4.2.3.1.10.

H.3.2.4 Use of Short Replies for Validation

With the original version of RTCA/DO-300, validation was performed using crosslink interrogations which elicited replies containing the ADS-B position of the target aircraft. This allowed for a very straightforward validation technique as the ADS-B position information included in the reply could be directly compared with slant range and bearing derived from the active interrogation; i.e., no interpolation or time alignment of passive and active position was necessary. While this validation technique was simple to implement, it had a negative side effect from the point of view of 1090 MHz channel use.

The reply elicited by a crosslink interrogation is twice the length of a reply to a standard TCAS interrogation. Therefore the use of crosslink interrogations for validation purposes negated some of the benefit yielded by Hybrid Surveillance. It was determined that a reduction in 1090 MHz channel use could be achieved by requiring TCAS to perform validation using standard interrogations which elicit replies that are half the length of those from crosslink interrogations. The additional processing required to perform this alternative validation technique (i.e., extrapolation for time alignment) was determined to be negligible.

H.3.2.5 Extended Hybrid Surveillance

The original version of RTCA/DO-300 contained two requirements that were eliminated in RTCA/DO-300A. The first was that a TCAS with Hybrid Surveillance had to acquire a target with the procedure described in RTCA/DO-185B (i.e., acquisition using active interrogations) before that target could be tracked passively. The second was that all passive tracks were required to be validated with a periodic interrogation. It was determined that a reduction in the TCAS use of the 1090 MHz channel could be accomplished by relaxing these requirements for target aircraft that are considered non-threatening and reporting high quality ADS-B information.

This surveillance algorithm change was achieved by allowing a TCAS with Hybrid Surveillance to passively acquire and track qualified ADS-B equipped target aircraft without validation. Surveillance of such aircraft was governed by rules that became known as Extended Hybrid Surveillance. In the most basic sense, a target aircraft would qualify for Extended Hybrid Surveillance if it was reporting high quality ADS-B information and its squitters were received at a power level below -68 dBm. Since received power level can be considered a rough surrogate for range, this change allowed for distant ADS-B equipped targets to be acquired passively without any interrogations. Furthermore, this change

eliminated the need to periodically validate ADS-B equipped aircraft that were far away and non-threatening.

H.3.2.6

Passive Only Surveillance on the Ground

As noted in §H.3.1.2, data collections showed that many more TCAS were operating on the airport surface than previously expected. While the algorithm changes described in §H.3.1.1, §H.3.1.2 and §H.3.2.1 reduced the number of interrogations used by TCAS when on the ground, it was recognized that a TCAS with Hybrid Surveillance could eliminate most interrogations by performing only passive surveillance on targets that are reporting high quality ADS-B information. Prior to RTCA/DO-300A, the TCAS Hybrid Surveillance requirements did not distinguish between airborne operation and operation on the ground. While this modification to the Hybrid Surveillance MOPS was evaluated in simulation as an independent change, it was considered part of Extended Hybrid Surveillance during the MOPS modification process due to a large similarity in the requirements.

H.4

Results

The surveillance algorithm changes described in §H.3 were evaluated using the simulation described in §H.2. These algorithm changes were compared against two baseline scenarios to understand the relative impact on the 1090 MHz channel. The first baseline scenario assumed all TCAS aircraft were equipped with an RTCA/DO-185B compliant unit while the second baseline assumed all TCAS aircraft were equipped with an RTCA/DO-300 compliant unit.

The metric chosen to understand the impact of each algorithm change was the 1090 MHz Receiver Occupancy. This metric describes the percentage of time a receiver with a minimum triggering level (MTL) of -74 dBm is occupied due to the decoding of TCAS generated replies. The calculation is performed through the summation of relevant replies received multiplied by the length of the reply. For example, if an antenna received 100 short reply messages (each 64 μ s in length) and 50 long reply message (each 120 μ s in length) within one second, the 1090 MHz Receiver Occupancy would be $(100 * 0.000064) + (50 * 0.000120) = 1.24\%$.

Within the simulation, the 1090 MHz TCAS Receiver Occupancy was calculated for those TCAS aircraft that were less than 30 NM from the JFK sensor. This metric was then averaged over all such TCAS aircraft and over the entire hour of the input data set. This averaging resulted in one value of 1090 MHz Receiver Occupancy for the scenario under consideration. Results were then compared against the baseline cases to highlight the relative reduction in 1090 MHz Receiver Occupancy. The relative reduction provided by each surveillance algorithm change is provided in the second and third columns of Table H-1. As an example, Table H-1 indicates that Extended Hybrid Surveillance provided a 26% reduction in 1090 MHz Receiver Occupancy when compared against a baseline of RTCA/DO-185B (column 2, row 9) and an 11% reduction when compared against the original version of RTCA/DO-300 (column 3, row 9).

As it was understood that multiple surveillance algorithm changes would be reflected in the final version of RTCA/DO-300A, several combinations of algorithm changes were also examined. The bottom two rows of Table H-1 show the relative reductions in 1090 MHz Receiver Occupancy provided by combinations of multiple surveillance algorithm

changes. The algorithm changes included in each group (each column) are indicated by shaded cells. As an example, Group A included the two modifications that were specific to RTCA/DO-185B. This group provided a 24% reduction in 1090 MHz Receiver Occupancy when compared against a baseline of RTCA/DO-185B.

Column H of Table H-1 most closely represents the modifications that were included in Version A of RTCA/DO-300. This column and the requirements in this document differ in only two manners. First, the modification of “Passively Monitor TCAS Aircraft when Below 2000 ft AGL (§H.3.2.1)” was not included in Version A of RTCA/DO-300. However, the majority of this modification is captured in “Eliminate Range Monitoring Interrogations to TCAS Aircraft when on the Ground (§H.3.1.2)” which is included in the updated version of RTCA/DO-185. Second, as explained in §H.3.2.2, the modification of “Two Validation Attempts (§H.3.2.2)” was simulated differently than the text drafted for Version A of RTCA/DO-300. These two differences between column H and requirements specified in Version A of RTCA/DO-300 are considered to be insignificant with regard to the TCAS use of the 1090 MHz channel.

Table H-1: Simulated Reduction in 1090 MHz Receiver Occupancy

Algorithm Change	% 1090 MHz Interference Reduction		Group A	Group B	Group C	Group D	Group E	Group F	Group G	Group H
	Vs DO-185B	Vs DO-300								
Reduced Surveillance Volume when Operating on Airport Surface (§H.3.1.1)	10%	NA								
Eliminate Range Monitoring Interrogations to TCAS Aircraft when on the Ground (§H.3.1.2)	14%	NA								
Original DO-300	17%	NA								
Passively Monitor TCAS Aircraft when Below 2000 ft AGL (§H.3.2.1)	36%	23%								
Two Validation Attempts (§H.3.2.2)	19%	2%								
Variable Validation Rate (§H.3.2.3)	30%	16%								
Use of Short Replies for Validation (§H.3.2.4)	34%	20%								
Extended Hybrid Surveillance (§H.3.2.5)	26%	11%								
Passive Only Surveillance on the Ground (§H.3.2.6)	49%	38%								
% Reduction in 1090 MHz Interference Relative to DO-185B			24%	68%	66%	71%	88%	83%	83%	89%
% Reduction in 1090 MHz Interference Relative to DO-300			NA	62%	59%	66%	86%	79%	80%	87%

This Page Intentionally Left Blank

APPENDIX I ACAS X EQUIPMENT CLASS REQUIREMENTS COMPLIANCE MATRIX**I.1 ACAS X Equipment Compliance Matrix**

The following Table identifies the performance requirements for the various classes of ACAS X equipment. Requirements in a given section apply unless the subparagraph or requirement numbers are listed separately or an exception is specified. This matrix is for reference only. Manufacturers should specify each requirement for which they are demonstrating compliance.

Equipment Class	Article Label	Article Name	ACAS MOPS Paragraph Compliance
-	A	Directional Antenna	§§2.1.1, 2.1.2, 2.1.3, 2.1.4, 2.1.5, 2.2.4.3, 2.2.4.5.4 (1248) (1283 ref only), 2.2.4.5.4.2.1, 2.2.4.5.4.2.2, 2.2.4.5.4.2.3 (ref), 2.2.4.6.3, 2.2.4.7, 2.3 and 2.4 (as applicable)
-	B	Omni-Directional Antenna	§§2.1.1, 2.1.2, 2.1.3, 2.1.4, 2.1.5, 2.2.4.3, 2.2.4.5.4 (1283 ref only), 2.2.4.5.4.2.2 (1299), 2.2.4.7, 2.3 and 2.4 (as applicable)
-	C	Display of Resolution Advisories	§§2.1.1, 2.1.2, 2.1.3, 2.1.4, 2.1.5, 2.2.6.2 (RA Only), 2.2.6.3 (Aural Annunciations), 2.3 and 2.4 (as applicable)
1 ACAS Xa Only	D	ACAS Xa Unit	§§2.1.1, 2.1.2, 2.1.3, 2.1.4, 2.1.5, 2.1.6, 2.1.7, 2.1.10, 2.1.11, 2.1.13, 2.2.3.6, 2.2.3.8, 2.2.3.9.2.1, 2.2.3.9, 2.2.3.10, 2.2.3.11, 2.2.3.12 (except 2.2.3.12.9.5), 2.2.4, 2.2.4.1, 2.2.4.5, 2.2.4.5.5, 2.4.6, 2.2.4.7.4, 2.2.5, 2.2.7, 2.3 and 2.4 (as applicable)
	E	Control Panel	§§2.1.1, 2.1.2, 2.1.3, 2.1.4, 2.1.5, 2.1.7, 2.2.3.12.3, 2.2.6.1.2.1.11 (ref), 2.2.6.1.2.4.4, 2.2.6.5 (except 2.2.6.5.4), 2.2.6.7, 2.3 and 2.4 (as applicable)
	F	Traffic Displays	§§2.1.1, 2.1.2, 2.1.3, 2.1.4, 2.1.5, 2.1.8, 2.1.12, 2.2.3.12.9 (display/receive ACAS X data), 2.2.4.2, 2.2.5.6.1.1 (display ACAS X data), 2.2.5.6.1.2 (ref), 2.2.6.1, 2.2.6.3 (Aural Annunciations), 2.2.6.4, 2.2.6.6, 2.2.6.7, 2.3 and 2.4 (as applicable)
2 ACAS X	D1	ACAS X Unit with DNA	§§2.1.1, 2.1.2, 2.1.3, 2.1.4, 2.1.5, 2.1.6, 2.1.7, 2.1.11, 2.1.13, 2.2.3.6, 2.2.3.8, 2.2.3.9, 2.2.3.10, 2.2.3.11, 2.2.3.12, 2.2.4, 2.2.4.1, 2.2.4.5, 2.2.4.5.5, 2.2.4.6, 2.2.2.4.7.4, 2.2.5, 2.2.7, 2.2.8 (2576, 2578, 1877, 1888, 1889, 2148, 1892, 1893, 1894, 1901, 1896, 2157, 2153, 1906, 1907, 1908, 1909, 2032, 1912, 1913, 1914, 1915, 1916, 2154, 2155, 2156, 2141, 2143, 2145, 2146, 2144, 2138 and 2139), 2.3 and 2.4 (as applicable)
	D2	ACAS X Unit with CSPO-3000	§§2.1.1, 2.1.2, 2.1.3, 2.1.4, 2.1.5, 2.1.6, 2.1.7, 2.1.11, 2.1.13, 2.2.3.6, 2.2.3.8, 2.2.3.9, 2.2.3.10, 2.2.3.11, 2.2.3.12, 2.2.4, 2.2.4.1, 2.2.4.5, 2.2.4.5.6, 2.2.4.6, 2.2.4.7.4, 2.2.5, 2.2.7, 2.2.8 (2577, 2578, 1877, 1888, 1889, 2148, 2136, 2586, 1897, 1898, 1901, 1896, 2157, 2152, 1910, 2032, 1917, 2141, 2143, 2145, 2146, 2144, 2138 and 2139), 2.3 and 2.4 (as applicable)
	E1	Control Panel with DNA	§§2.1.1, 2.1.2, 2.1.3, 2.1.4, 2.1.5, 2.1.7, 2.2.3.12.3, 2.2.6.1.2.1.11 (ref), 2.2.6.1.2.4.4, 2.2.6.5, 2.2.6.5.4 (Xo), 2.2.6.7, 2.2.8.2.1, 2.2.8.2.3 (2149), 2.3 and 2.4 (as applicable)
	E2	Control Panel with CSPO-3000	§§2.1.1, 2.1.2, 2.1.3, 2.1.4, 2.1.5, 2.1.7, 2.2.3.12.3, 2.2.6.1.2.1.11 (ref), 2.2.6.1.2.4.4, 2.2.6.5, 2.2.6.5.4 (Xo), 2.2.6.7, 2.2.8.2.1, 2.2.8.2.3 (2149), 2.3 and 2.4 (as applicable)
	F1	Traffic Displays with DNA	§§2.1.1, 2.1.2, 2.1.3, 2.1.4, 2.1.5, 2.1.8, 2.1.12, 2.2.3.12.9 (display/receive ACAS X data), 2.2.4.2, 2.2.5.6.1.1 (display ACAS X data), 2.2.5.6.1.2 (ref), 2.2.5.5.6.1.3 (ref), 2.2.6.1, 2.2.6.3 (Aural Annunciations), 2.2.6.4, 2.2.6.6, 2.2.6.7, 2.2.7, 2.2.8.2, 2.2.8.2.3 (2149), 2.2.8.7.1 (ref), 2.3 and 2.4 (as applicable)
	F2	Traffic Display with CSPO-3000	§§2.1.1, 2.1.2, 2.1.3, 2.1.4, 2.1.5, 2.1.8, 2.1.12, 2.2.3.12.9 (display/receive ACAS X data), 2.2.4.2, 2.2.5.6.1.1 (display ACAS X data), 2.2.5.6.1.2 (ref), 2.2.6.6.1.3 (ref), 2.2.6.1, 2.2.6.3 (Aural Annunciations), 2.2.6.4, 2.2.6.6, 2.2.6.7, 2.2.7, 2.2.8.2, 2.2.8.2.3 (2149), 2.2.8.2.3.2, 2.2.8.7.2 (refer to Note), 2.3 and 2.4 (as applicable)

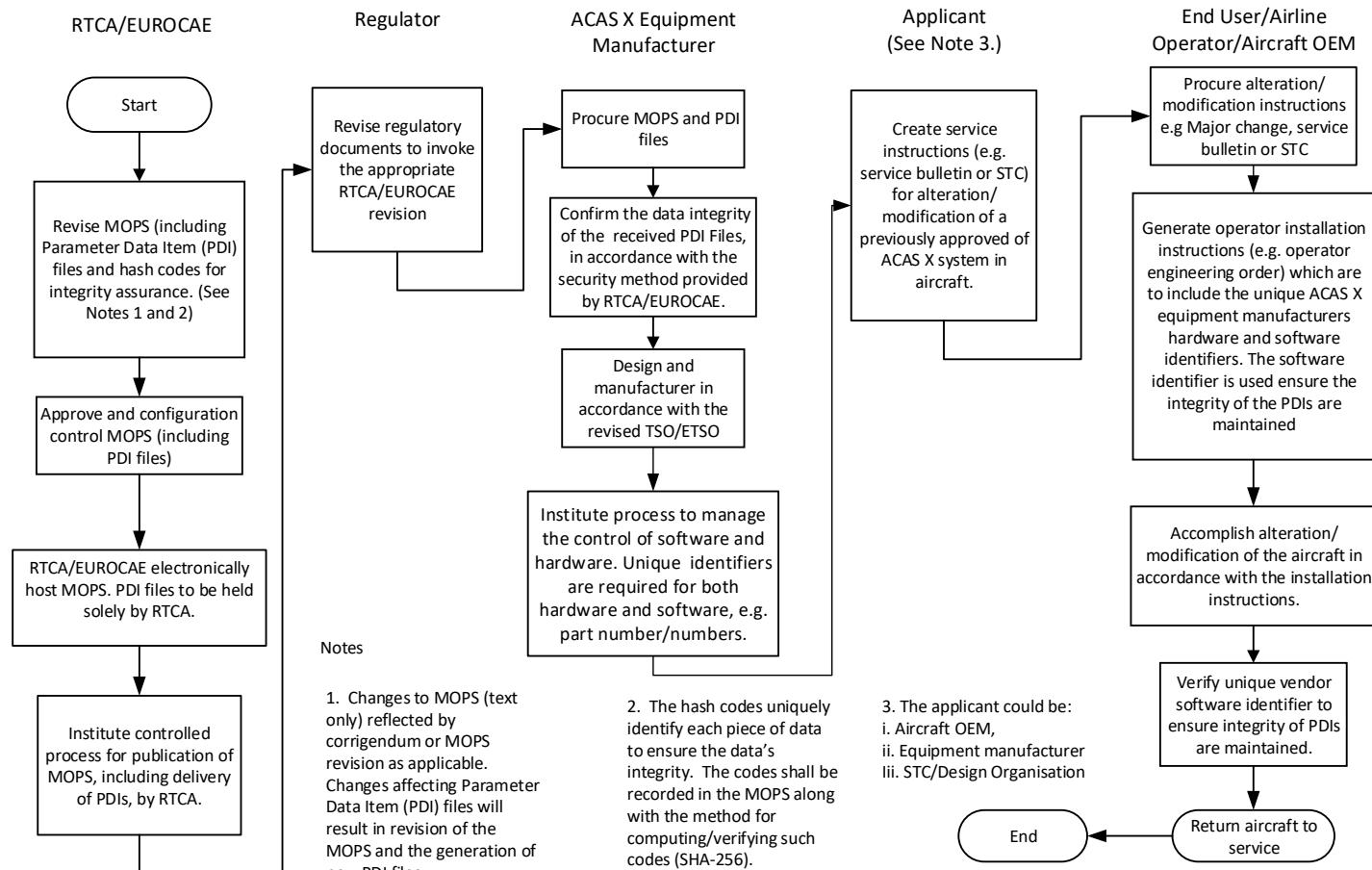
This Page Intentionally Left Blank

APPENDIX J PDIF LIFE CYCLE PROCESS FLOW AND THE ROLES AND RESPONSIBILITIES OF THE STAKEHOLDERS

J.1 PDIF Life Cycle Process Flow and the Roles and Responsibilities of the Stakeholders

The following diagrams specify the ACAS X Parameter Data Item File (PDIF) life cycle process flows and the roles and responsibilities of pertinent stakeholders.

Creation, Configuration Control And Distribution of Revised ACAS X Equipment



This Page Intentionally Left Blank

RTCA, Inc.
1150 18th Street NW, Suite 910
Washington, DC 20036
USA

**Minimum Operational Performance Standards for
Airborne Collision Avoidance System X (ACAS X)
(ACAS Xa and ACAS Xo)**

Volume II: Algorithm Design Description

Copies of this document may be obtained from

RTCA, Inc.
Telephone: 202-833-9339
Facsimile: 202-833-9434
Internet: www.rtca.org

Please visit the RTCA Online Store for document pricing and ordering information.

FOREWORD

This document was prepared jointly by Special Committee 147 (SC-147) and EUROCAE Working Group 75(WG-75). It was approved by the RTCA Program Management Committee (PMC) on September 20, 2018 and by the EUROCAE Council on October 2, 2018.

RTCA, Incorporated is a not-for-profit corporation formed to advance the art and science of aviation and aviation electronic systems for the benefit of the public. The organization develops consensus-based recommendations on contemporary aviation issues. RTCA's objectives include, but are not limited to:

- Coalescing aviation system user and provider technical requirements in a manner that helps government and industry meet their mutual objectives and responsibilities;
- Analyzing and recommending solutions to the system technical issues that aviation faces as it continues to pursue increased safety, system capacity and efficiency;
- Developing consensus on the application of pertinent technology to fulfill user and provider requirements, including development of Minimum Operational Performance Standards (MOPS) for equipment and electronic systems that support aviation; and
- Assisting in developing the appropriate technical material upon which positions for the International Civil Aviation Organization (ICAO) and the International Telecommunication Union (ITU) and other appropriate international organizations can be based.

The recommendations of RTCA are often used as the basis for government and private sector decisions as well as the foundation for many Federal Aviation Administration Technical Standard Orders (TSO).

Since RTCA is not an official agency of the United States Government, its recommendations may not be regarded as statements of official government policy unless so enunciated by the United States Government organization or agency having statutory jurisdiction over any matters to which the recommendations relate.

DISCLAIMER

This publication is based on material submitted by various participants during the SC approval process. Neither the SC nor RTCA has made any determination whether these materials could be subject to valid claims of patent, copyright or other proprietary rights by third parties, and no representation or warranty, expressed or implied is made in this regard. Any use of or reliance on this document shall constitute an acceptance thereof "as is" and be subject to this disclaimer.

This Page Intentionally Left Blank

EXECUTIVE SUMMARY

ACAS X is an airborne collision avoidance system designed to reduce the risk of mid-air collision while minimizing unnecessary alerts. The system described in this document is proposed as the replacement for the existing TCAS II system. The ACAS X design consists of two main modules: (1) the Surveillance and Tracking Module (STM), which detects aircraft in the vicinity and tracks their position, and (2) the Threat Resolution Module (TRM), which identifies threats and provides resolution guidance. This document provides the Algorithm Design Description (ADD) for both ACAS X modules.

This document contains modifications informed by operational suitability and performance evaluations conducted via modeling and simulation as well as analysis of flight tests in 2013 and 2015 as well as the Operational Evaluation conducted in 2017. This system is proposed as the next generation airborne collision avoidance system with equipage beginning in the 2020-2023 timeframe.

This page intentionally left blank.

TABLE OF CONTENTS

	Page
1 Introduction.....	1
1.1 Purpose.....	1
1.2 Scope	1
1.3 Background.....	1
1.4 Context.....	2
1.5 Prescription	2
1.6 Specification Language: Julia.....	3
1.6.1 Data Types.....	3
1.6.2 Built-In Values and Functions.....	3
1.7 Notation	4
1.7.1 Variables.....	5
1.7.2 Physical Quantities	6
1.7.3 Statistical Quantities	7
1.7.4 Functions	8
1.7.5 Diagrams	8
1.8 References	8
2 Surveillance and Tracking Module Description.....	11
2.1 System Overview	11
2.2 Active Surveillance	12
2.2.1 Input Processing	13
2.2.1.1 Mode S Surveillance	13
2.2.1.2 Mode C Surveillance	21
2.2.2 Intruder Horizontal Estimation	26
2.2.2.1 Cartesian Tracker	27
2.2.2.2 Range Tracker	34
2.2.3 Intruder Vertical Estimation	41
2.3 Passive Surveillance	46
2.3.1 Input Processing	47
2.3.1.1 State Vector Position Report	47
2.3.1.2 State Vector Velocity Report	60
2.3.1.3 Mode Status Report	63
2.3.2 Intruder Horizontal Estimation	65
2.3.3 Intruder Vertical Estimation	70
2.4 Ownship Tracking	70
2.4.1 Input Processing	70
2.4.2 Ownship Discretes	70
2.4.3 Target Designation	71
2.4.4 Barometric Altitude Estimation	75
2.4.5 Radio Altitude Estimation	79
2.4.6 Heading Estimation	79
2.4.7 WGS84 State Estimation	85
2.5 Coordination Processing.....	86
2.5.1 Input Processing	86
2.5.1.1 Active Coordination	86
2.5.1.2 Coordination Capability Report	89
2.6 Track Management	91
2.6.1 Sensitivity Level Calculation	91
2.6.2 Transponder Mode Management	93
2.6.3 Surveillance Source Correlation	94

2.6.4	STM Report Generation	95
2.6.4.1	Mode S Track File	101
2.6.4.2	ADS-B Track File	108
2.6.4.3	Mode C Track File	112
2.6.5	Track Deletion	114
2.6.6	Active Validation	119
2.6.7	Target Designation	123
2.6.8	Proximity Estimation	146
3	Threat Resolution Module Description.....	149
3.1	System Overview	149
3.1.1	Inputs and Outputs to the TRM	155
3.1.2	System Architecture	156
3.1.3	Intruder Designation and Protection Modes	157
3.2	State and Cost Estimation	158
3.2.1	Intruder Prep	162
3.2.2	State Estimation	165
3.2.2.1	Tau Estimation	172
3.2.2.2	Combine Vertical Beliefs	177
3.2.2.3	Set Belief-Based Online Cost	178
3.2.3	Offline Cost Estimation	183
3.2.4	Online Cost Estimation Independent of Coordination	189
3.2.4.1	Advisory Restart Cost	194
3.2.4.2	Altitude Dependent Clear of Conflict Cost	195
3.2.4.3	Altitude Inhibit Cost	196
3.2.4.4	Bad Transition Cost	197
3.2.4.5	Critical Interval Protection Cost	200
3.2.4.6	Initialization Cost	205
3.2.4.7	Prevent Early COC Cost	205
3.2.4.8	Safe Crossing Resolution Advisory Deferral Cost	207
3.2.5	Update Intruder Inputs Mid-Cycle	209
3.2.6	Online Cost Estimation Dependent on Coordination	211
3.2.6.1	Response Estimation	216
3.2.6.2	Compatibility Cost	220
3.2.6.3	Coordinated Resolution Advisory Deferral Cost	224
3.2.6.4	Coordination Delay Cost	226
3.2.6.5	Crossing No Alert Cost	227
3.2.6.6	Low Altitude Parallel Resolution Advisory Deferral Cost	229
3.2.6.7	Max Reversal Cost	234
3.2.6.8	Prevent Early Weakening Cost	236
3.2.6.9	Restrict Clear of Conflict Due To Reversal	237
3.2.6.10	SA01 Heuristic	238
3.2.6.11	Time-Based Non-Compliance Cost	241
3.2.7	Individual Cost Estimation	246
3.3	Action Selection	248
3.3.1	Single Intruder Rate Selection	251
3.3.2	Multiple Intruder Rate Selection	252
3.3.2.1	Sandwich Prevention Cost for Unequipped Intruders	255
3.3.2.2	Cost Fusion for Unequipped Intruders	257
3.3.2.3	Multi Threat Level Off (MTLO) Determination	258
3.3.2.4	Balancing Intruder Costs	261
3.3.2.5	Action Arbitration	264
3.4	Coordination Selection	267
3.5	Track Threat Assessment	272
3.5.1	Determine Intruder Alert	273

3.5.2	Determine Code	275
3.5.3	Determine Score.....	276
3.6	Display Logic	278
3.6.1	Determine Crossing.....	280
3.6.2	Determine Display Data.....	281
3.6.3	Determine Aural Inhibit.....	286
3.6.4	Determine Traffic Alert.....	287
3.7	Generate TRM Output	288
3.7.1	TA-Only Mode Processing.....	291
3.7.2	TA/RA Mode Processing	296
3.7.3	Standby Mode Processing	305
3.7.4	Resolution Advisory Output Messages.....	306
3.7.4.1	Common RA Output Message Data.....	308
3.7.4.2	RA Report Message.....	311
3.7.4.3	Broadcast Interrogation Message	316
3.7.5	Intruder Management	318
3.8	Standard States and Conversions	324
4	Acknowledgments	345
Appendix A STM Housekeeping		A-1
Appendix B Mode C Processing Implementation		B-1
Appendix C Correlation Processing Implementation		C-1
C.1	Track Extrapolation	C-1
C.2	Track Decorrelation	C-5
C.3	Track Correlation	C-10
Appendix D Constant Variables		D-1
D.1	Geometric Constants	D-1
D.2	Global Constants	D-2
Appendix E Data Structure Definitions		E-1
E.1	STM Output Data Structures	E-1
E.2	STM Internal Data Structures	E-3
E.3	STM Output-TRM Input Data Structures	E-18
E.4	TRM Output-STM Input Data Structures	E-21
E.5	TRM Internal Data Structures	E-28
E.6	TRM Context-Specific State Data Structures.....	E-38
Appendix F Parameter File Specification		F-1
F.1	ACAS X Parameter Data Item File (PDIF) Checksums	F-1
F.2	STM Parameters	F-2
F.3	TRM Parameters	F-12
Appendix G Data Table Format Specification		G-1

Appendix H Math Utilities	H-1
Appendix I Data Management Utilities	I-1
Appendix J Julia Standard Library Functions	J-1
Appendix K Design Notes	K-1
K.1 Overview	K-1
K.2 Interfacing with ACAS X	K-1
K.2.1 General	K-1
K.2.2 Input Message Entry Points	K-1
K.2.3 Report Generation	K-3
K.2.3.1 Module Memory Management	K-3
K.3 Specific Implementation Details	K-5
K.3.1 Julia Pseudocode	K-5
K.3.1.1 Julia Math and Array Notation	K-5
K.3.1.2 Automatic Data Type Conversions	K-6
K.3.1.3 Handling Symbols	K-7
K.3.2 Single versus Multi-Threaded Execution	K-7
K.3.3 32-Bit versus 64-Bit Applications	K-7
K.3.4 Floating Point Precision	K-7
K.3.5 Subsystem Modularity	K-8
K.3.6 Running Julia as a Development Tool	K-8
K.3.7 Static Allocation	K-9
K.3.8 Target IDs	K-12
K.4 Error Checking	K-13
K.4.1 Divide-by-Zero Errors	K-13
K.4.2 Invalid Input	K-15

TABLE OF TABLES

	Page
Table 1-1 Algorithm Prescription Categorization	3
Table 1-2 Data Types	4
Table 1-3 Julia Keywords	4
Table 1-4 Fundamental Units	5
Table 1-5 Key to Common Variable Prefixes	6
Table 1-6 Fundamental Physical Quantities	7
Table 1-7 Derived Physical Quantities	7
Table 2-1 Summary of Entry Points to the STM	11
Table 2-2 STM Input Variables - DF0 Message	14
Table 2-3 STM Input Variables - Mode C Reply Message	22
Table 2-4 Kalman F, G, H, Q, and R Matrix Definitions	40
Table 2-5 STM Input Variables - State Vector Position Report	48
Table 2-6 STM Input Variables - State Vector Velocity Report	61
Table 2-7 STM Input Variables - Mode Status Report	64
Table 2-8 STM Input Variables - Ownership Discretes	71
Table 2-9 STM Input Variables - Target Designation	72
Table 2-10 STM Input Variables - Barometric Altitude Observation	76
Table 2-11 STM Input Variables - Radio Altitude Observation	79
Table 2-12 STM Input Variables - Heading Observation	80
Table 2-13 STM Input Variables - WGS84 Observation	86
Table 2-14 STM Input Variables - UF16UDS30 Message	87
Table 2-15 STM Input Variables - Coordination Capability Report	90
Table 3-1 TRM Internal Variables - VerticalTRMUpdate (Figure 3-1)	151
Table 3-2 TRM Algorithm Output Variables - StateAndCostEstimation	159
Table 3-3 TRM Algorithm Output Variables - StateEstimation	167
Table 3-4 TRM Algorithm Output Variables - OnlineUncoordinatedCostEstimation	190
Table 3-5 TRM Algorithm Output Variables - OnlineCostEstimation	213
Table 3-6 TRM Algorithm Output Variables - IndividualCostEstimation	247
Table 3-7 TRM Algorithm Output Variables - ActionSelection	250
Table 3-8 TRM Algorithm Output Variables - CoordinationSelection	268
Table 3-9 TRM Algorithm Output Variables - TrackThreatAssessment	273
Table 3-10 TRM Algorithm Output Variables - DisplayLogicDetermination	280
Table 3-11 TRM Algorithm Output Variables - GenerateTRMOutput	290
Table B-1 STM Input Variables - Mode C Replies Message	B-1
Table F-1 Parameter Data Item File (PDIF) List	F-1
Table F-2 actions	F-13
Table F-3 display.label270rules (1 of 4)	F-15
Table F-4 display.label270rules (2 of 4)	F-16
Table F-5 display.label270rules (3 of 4)	F-17
Table F-6 display.label270rules (4 of 4)	F-18
Table F-7 modes[n].cost_estimation.online.altitude_inhibit - Xa	F-24
Table F-8 modes[n].cost_estimation.online.altitude_inhibit - Xo	F-24
Table J-1 Julia Functions Used in the ACAS X Algorithms	J-1
Table J-2 Julia Functions Used in the ACAS X Algorithms (cont.)	J-2
Table K-1 Julia Operators	K-6

Table K-2	Example Julia Commands	K-9
Table K-3	(Minimum) Recommended Container Allocations	K-11
Table K-4	(Minimum) Recommended Container Allocations (Continued).....	K-12

TABLE OF FIGURES

	Page
Figure 1-1 Example data processing flow.....	8
Figure 3-1 VerticalTRMUpdate overview.....	150
Figure 3-2 StateAndCostEstimation overview.....	160
Figure 3-3 StateEstimation overview.....	167
Figure 3-4 OnlineUncoordinatedCostEstimation overview.....	190
Figure 3-5 OnlineCostEstimation overview.....	213
Figure 3-6 IndividualCostEstimation overview.....	247
Figure 3-7 ActionSelection overview.....	249
Figure 3-8 CoordinationSelection overview.....	268
Figure 3-9 TrackThreatAssessment overview.....	273
Figure 3-10 DisplayLogicDetermination overview.....	279
Figure 3-11 GenerateTRMOutput overview.....	290

TABLE OF ALGORITHMS

	Page
Algorithm 1 ReceiveDF0	15
Algorithm 2 InitializeModeSTrackFile	16
Algorithm 3 AdvanceModeSTrackFile	17
Algorithm 4 AssociateICAOtoTarget	18
Algorithm 5 TrackExists	18
Algorithm 6 ConvertMeasurements	19
Algorithm 7 AssociateDF0toTarget	19
Algorithm 8 AddModeSTrackToDB	20
Algorithm 9 AddToDB	20
Algorithm 10 RetrieveWithID	20
Algorithm 11 ReceiveModeCReply	22
Algorithm 12 AssociateModeCExternalIDtoTrack	22
Algorithm 13 AddModeCTrackToDB	23
Algorithm 14 InitializeModeCTrack	23
Algorithm 15 UpdateModeCTrack	24
Algorithm 16 GillhamDecode	25
Algorithm 17 GillhamEncode	26
Algorithm 18 BinaryToGray	26
Algorithm 19 InitializeCartesianTracker	28
Algorithm 20 SigmaPointSample	28
Algorithm 21 ConvertToCartesian	29
Algorithm 22 WeightedMeanAndCovariance	29
Algorithm 23 AdvanceCartesianTrack	31
Algorithm 24 LinearTransform	32
Algorithm 25 IsOutlier	32
Algorithm 26 PredictCartesianTracker	33
Algorithm 27 UpdateCartesianTracker	34
Algorithm 28 AugmentedStateToObservation	34
Algorithm 29 InitializeRangeTracker	35
Algorithm 30 GroundRange	35
Algorithm 31 AdvanceRangeTrack	37
Algorithm 32 ReinitializeRangeTracker	38
Algorithm 33 PredictRangeTracker	39
Algorithm 34 UpdateRangeTracker	41
Algorithm 35 UpdateKalmanFilter	41
Algorithm 36 InitializeVerticalTracker	42
Algorithm 37 AdvanceVerticalTrack	44
Algorithm 38 PredictVerticalTracker	45
Algorithm 39 UpdateVerticalTracker	45
Algorithm 40 PredictedAltitude	46
Algorithm 41 ReceiveStateVectorPositionReport	49
Algorithm 42 AssociateADSBReportToTarget	50
Algorithm 43 InitializeADSBTrackFile	51
Algorithm 44 ConvertObservedTargetToENU	51
Algorithm 45 ConvertWGS84ToECEF	52
Algorithm 46 ConvertECEFToWGS84	53

Algorithm 47	ConvertOrthometricToGeodeticHeight	53
Algorithm 48	ConvertGeodeticToOrthometricHeight	54
Algorithm 49	GetEarthRadiusAtLatitude	54
Algorithm 50	GetNormalGravityAtLatitude	54
Algorithm 51	AddADSBTrackToDB	55
Algorithm 52	AddADSRTTrackToDB	55
Algorithm 53	PropagateOwnshipToToa	56
Algorithm 54	RedefineEstimateInRotatedFrame	58
Algorithm 55	RefineENUHeight	59
Algorithm 56	RotateECEFToENU	59
Algorithm 57	RotateENUToECEF	60
Algorithm 58	ReceiveStateVectorVelocityReport	62
Algorithm 59	PropagateIntruderToToa	63
Algorithm 60	ReceiveModeStatusReport	64
Algorithm 61	UpdateAdsBQualityHistory	65
Algorithm 62	InitializeADSBTrackerPosition	66
Algorithm 63	ConvertNACvToSigmaHVA	66
Algorithm 64	AdvanceADSBTrackPosition	67
Algorithm 65	AdvanceADSBTrackVelocity	68
Algorithm 66	PredictADSBTracker	68
Algorithm 67	PredictKalmanFilter	69
Algorithm 68	UpdateADSBTrackerPosition	69
Algorithm 69	UpdateADSBTrackerVelocity	70
Algorithm 70	ReceiveDiscretes	71
Algorithm 71	ReceiveTargetDesignation	73
Algorithm 72	GetDesignatedTargetIDs	73
Algorithm 73	SetPendingTargetDesignation	75
Algorithm 74	ReceiveBaroAltObservation	76
Algorithm 75	UpdateHistory	77
Algorithm 76	InitializeOwnVerticalTracker	77
Algorithm 77	BaroAltAtToa	78
Algorithm 78	SurroundingPoints	79
Algorithm 79	ReceiveRadAltObservation	79
Algorithm 80	ReceiveHeadingObservation	81
Algorithm 81	InitializeHeadingTracker	83
Algorithm 82	RotateHorizontalFrame	84
Algorithm 83	UpdateHeadingTracker	84
Algorithm 84	HeadingAtToa	85
Algorithm 85	ReceiveWgs84Observation	86
Algorithm 86	ReceiveUF16UDS30	88
Algorithm 87	DeleteIntent	88
Algorithm 88	UpdateIntruderVRC	89
Algorithm 89	CoordinationTimeoutCheck	89
Algorithm 90	ReceiveCapabilityReport	90
Algorithm 91	SetCoordination	91
Algorithm 92	SetSensitivityLevel	93
Algorithm 93	SetTransponderData	94
Algorithm 94	UpdateAdvisoryMode	94

Algorithm 95	CorrelationProcessing	95
Algorithm 96	GenerateStmReport	97
Algorithm 97	AddTracksToReport	98
Algorithm 98	HasValidModeCTrack	99
Algorithm 99	FilterTracksForTRM	100
Algorithm 100	CompareIntruderRange	100
Algorithm 101	SlantRange	101
Algorithm 102	AddModeSTrackToReport	102
Algorithm 103	SetIntruderBeliefStates	103
Algorithm 104	SetIntruderDegradedFlags	104
Algorithm 105	AddAltBiasAndSample	105
Algorithm 106	CombineAndSample	105
Algorithm 107	ConvertToAzAndXRangeRate	106
Algorithm 108	SetDisplayDataActive	107
Algorithm 109	VerticalRateArrowUpdate	108
Algorithm 110	AddADSBTrackToReport	110
Algorithm 111	AdsbQualityCheck	111
Algorithm 112	SetDisplayDataPassive	112
Algorithm 113	AddModeCTrackToReport	114
Algorithm 114	RemoveStaleTracks	115
Algorithm 115	RemoveStaleModeSTrack	116
Algorithm 116	RemoveStaleADSBTrack	117
Algorithm 117	RemoveStaleModeCTracks	118
Algorithm 118	RemoveStaleADSRTTrack	118
Algorithm 119	TargetIsEmpty	119
Algorithm 120	ActiveValidationUpdate	121
Algorithm 121	ActiveValidationCheck	122
Algorithm 122	SetTargetDesignation	125
Algorithm 123	UpdateTargetDesignationValidity	127
Algorithm 124	UpdateCSP03000Validity	129
Algorithm 125	AutomaticallyUndesignateCSP03000	130
Algorithm 126	UpdateDNAValidity	131
Algorithm 127	AutomaticallyUndesignateDNA	131
Algorithm 128	UpdateDNABasedOnAltitude	132
Algorithm 129	UpdateDNABasedOnRange	134
Algorithm 130	CheckTargetDesignationTimers	135
Algorithm 131	SetIntruderDesignationData	137
Algorithm 132	SetTargetDesignationInvalid	139
Algorithm 133	IsTargetDesignationActive	141
Algorithm 134	AdjustTargetDesignationValidity	143
Algorithm 135	UpdateDroppedTargetDesignationState	144
Algorithm 136	SetTargetDesignationModeAvailability	145
Algorithm 137	IsCSP03000ModeUnavailableToRun	146
Algorithm 138	IsDNAModeUnavailableToRun	146
Algorithm 139	ProximityEstimation	147
Algorithm 140	VerticalTRMUpdate	152
Algorithm 141	GetVTRMOwnData	153
Algorithm 142	UpdateAltitudeInhibitCState	154

Algorithm 143	DetermineLDI	155
Algorithm 144	StateAndCostEstimation	161
Algorithm 145	IntruderPrep	165
Algorithm 146	StateEstimation	168
Algorithm 147	ConvertHorizontal	169
Algorithm 148	HorizontalWorstCase	170
Algorithm 149	DetermineHorizontalWorstCaseNearMeanFactor	171
Algorithm 150	DetermineHorizontalWorstCasePhiSpreadFactor	172
Algorithm 151	TauEstimation	174
Algorithm 152	GetEntryDistributionTables	175
Algorithm 153	DetermineTauVerticalDistribution	176
Algorithm 154	LookupEntryDistribution	176
Algorithm 155	InterpolateBlocks	177
Algorithm 156	CombineVerticalBeliefs	178
Algorithm 157	SetBeliefBasedOnlineCostState	179
Algorithm 158	DetermineAltitudeDependentCOCFactor	181
Algorithm 159	DetermineVerticalRateReductionFactor	182
Algorithm 160	DetermineVerticalProximateFactor	183
Algorithm 161	OfflineCostEstimation	184
Algorithm 162	GetOfflineCostTauBeliefs	185
Algorithm 163	OfflineStatesScaleFactor	186
Algorithm 164	GetEquivClassTableMaxCutValues	186
Algorithm 165	GetTableMaxCutValue	187
Algorithm 166	LookupOfflineCost	188
Algorithm 167	OnlineUncoordinatedCostEstimation	191
Algorithm 168	GetModifiedGlobalRates	192
Algorithm 169	OnlineUncoordinatedActionCostEstimation	193
Algorithm 170	AdvisoryRestartCost	194
Algorithm 171	UpdateAdvisoryRestartCState	195
Algorithm 172	AltitudeDependentCOCCost	196
Algorithm 173	AltitudeInhibitCost	197
Algorithm 174	BadTransitionCost	198
Algorithm 175	UpdateBadTransitionCState	199
Algorithm 176	BadMaintainTransitionCost	200
Algorithm 177	CriticalIntervalProtectionCost	201
Algorithm 178	UpdateCriticalIntervalProtectionCState	203
Algorithm 179	IsOutsideCriticalInterval	204
Algorithm 180	CriticalIntervalRequiresVerticalDivergence	205
Algorithm 181	InitializationCost	205
Algorithm 182	PreventEarlyCOCCost	206
Algorithm 183	UpdatePreventEarlyCOCCState	207
Algorithm 184	SafeCrossingRADeferralCost	208
Algorithm 185	UpdateSafeCrossingRADeferralCState	209
Algorithm 186	UpdateIntruderInputs	211
Algorithm 187	OnlineCostEstimation	214
Algorithm 188	OnlineCoordinatedActionCostEstimation	216
Algorithm 189	IntruderResponseEstimation	218
Algorithm 190	OwnResponseEstimation	219

Algorithm 191	DetermineResponse	220
Algorithm 192	CompatibilityCost	223
Algorithm 193	UpdateCompatibilityCState	224
Algorithm 194	CoordinatedRADeferralCost	225
Algorithm 195	UpdateCoordinatedRADeferralCState	226
Algorithm 196	CoordinationDelayCost	227
Algorithm 197	UpdateCoordinationDelayCState	227
Algorithm 198	CrossingNoAlertCost	228
Algorithm 199	UpdateCrossingNoAlertCState	229
Algorithm 200	LowAltitudeParallelRADeferralCost	229
Algorithm 201	UpdateLowAltitudeParallelRADeferralCState	231
Algorithm 202	DetermineLowAltitudeCost	232
Algorithm 203	GetLowAltitudeCostTAParameters	233
Algorithm 204	GetLowAltitudeCostRAParameters	234
Algorithm 205	MaxReversalCost	235
Algorithm 206	UpdateMaxReversalCState	236
Algorithm 207	PreventEarlyWeakeningCost	237
Algorithm 208	RestrictCOCDueToReversal	238
Algorithm 209	UpdateRestrictCOCDueToReversalCState	238
Algorithm 210	SA01Heuristic	239
Algorithm 211	UpdateSA01HeuristicCState	240
Algorithm 212	ShouldReverse	241
Algorithm 213	TimeBasedNonComplianceCost	242
Algorithm 214	UpdateTimeBasedNonComplianceCState	244
Algorithm 215	FindCompliantRateDelta	245
Algorithm 216	DetermineOwnNonComplianceFactor	246
Algorithm 217	IndividualCostEstimation	248
Algorithm 218	ActionSelection	250
Algorithm 219	NoIntrudersAction	251
Algorithm 220	DetermineMinimumCostAction	252
Algorithm 221	DetermineMultiIntruderAction	254
Algorithm 222	IndividualSelectionCostEstimation	255
Algorithm 223	SandwichPreventionCost	257
Algorithm 224	UnequippedCostFusion	258
Algorithm 225	MTLODetermination	259
Algorithm 226	DetermineSenses	260
Algorithm 227	DetermineConflictingSenses	260
Algorithm 228	AllowUnequippedMTLO	261
Algorithm 229	MultithreatCostBalancing	263
Algorithm 230	ActionArbitration	264
Algorithm 231	ArbitrateConflictingSenses	265
Algorithm 232	ArbitrateMatchingSenses	267
Algorithm 233	CoordinationSelection	269
Algorithm 234	Crosslink	270
Algorithm 235	EncodeCVC	271
Algorithm 236	EncodeVRC	272
Algorithm 237	EncodeVSB	272
Algorithm 238	TrackThreatAssessment	273

Algorithm 239	DetermineIntruderAlert	275
Algorithm 240	DetermineCode	276
Algorithm 241	DetermineScore	278
Algorithm 242	DisplayLogicDetermination	280
Algorithm 243	DetermineCrossing	281
Algorithm 244	DetermineDisplayData	283
Algorithm 245	GetMatchingLabel270Rule	285
Algorithm 246	Consistent	286
Algorithm 247	DetermineAuralInhibit	287
Algorithm 248	DetermineTrafficAlert	288
Algorithm 249	GenerateTRMOutput	291
Algorithm 250	GenerateTAOnlyModeOutput	292
Algorithm 251	AdjustTAOnlyModeIntruderOutput	294
Algorithm 252	ConvertAdvisory	296
Algorithm 253	GenerateTARAModeOutput	299
Algorithm 254	AdjustTARAModeIntruderOutput	302
Algorithm 255	UpdateIndivAdjustThreatInfo	304
Algorithm 256	UpdateIndivAdjustCounts	305
Algorithm 257	GenerateStandbyModeOutput	306
Algorithm 258	SetRAMessageOutput	308
Algorithm 259	DetermineRACoordinationData	309
Algorithm 260	DetermineSPI	310
Algorithm 261	IsDesignationInForce	311
Algorithm 262	SetGroundMsgData	312
Algorithm 263	DetermineTTIAndTID	313
Algorithm 264	EncodeTIDAAltitude	314
Algorithm 265	EncodeTIDRange	315
Algorithm 266	EncodeTIDBearing	315
Algorithm 267	DetermineDSI	316
Algorithm 268	SetRABroadcastData	317
Algorithm 269	EncodeCAC	318
Algorithm 270	DroppedIntrudersAdjustment	320
Algorithm 271	SetDroppedIntruder	322
Algorithm 272	SetInvalidIntruder	324
Algorithm 273	ActionToRates	325
Algorithm 274	CalculateThresholdRampDownFactor	326
Algorithm 275	CalculateThresholdRampUpFactor	326
Algorithm 276	ExpectedTau	327
Algorithm 277	GetOwnHeight	327
Algorithm 278	GetProtectionModeIndex	328
Algorithm 279	Interpolants	329
Algorithm 280	InterpolateOneDimension	331
Algorithm 281	IsCOC	332
Algorithm 282	IsCorrective	332
Algorithm 283	IsCrossing	333
Algorithm 284	IsDiverging	333
Algorithm 285	IsDNC	334
Algorithm 286	IsDND	334

Algorithm 287	IsIntruderMaster	334
Algorithm 288	IsMaintain	335
Algorithm 289	IsMasterForcingReversal	335
Algorithm 290	IsMTLO	336
Algorithm 291	IsPreventive	336
Algorithm 292	IsProjectedCrossing	337
Algorithm 293	IsReversal	337
Algorithm 294	IsSlowClosure	338
Algorithm 295	IsStrengthening	338
Algorithm 296	IsVerticallyConverging	339
Algorithm 297	MaintainRates	339
Algorithm 298	MinCostIndex	340
Algorithm 299	MTLOAction	340
Algorithm 300	PersistMTLO	341
Algorithm 301	RatesToAction	342
Algorithm 302	RatesToSense	342
Algorithm 303	TRMIIntruderStateUpdate	344
Algorithm 304	VRCToSense	344
Algorithm 305	StmHousekeeping	A-1
Algorithm 306	StmHousekeepingTargetDesignation	A-2
Algorithm 307	ReceiveModeCReplies	B-2
Algorithm 308	MergeModeCReplies	B-3
Algorithm 309	AssociateModeCtoTarget	B-5
Algorithm 310	DetermineModeCReplyAssociationScore	B-7
Algorithm 311	IsImageTrack	B-8
Algorithm 312	SortByRangeSquared	B-8
Algorithm 313	RangeSquared	B-8
Algorithm 314	PromoteModeCTracks	B-10
Algorithm 315	EstablishModeCTrack	B-11
Algorithm 316	FormHypotheticalTracks	B-12
Algorithm 317	HypotheticalTrackTest	B-14
Algorithm 318	AltCodeAgreement	B-15
Algorithm 319	AltCodeEstimate	B-15
Algorithm 320	IntervalMaintenance	B-16
Algorithm 321	EvaluateOnGroundModeC	B-17
Algorithm 322	EvaluateGroundToAirTransitionModeC	B-18
Algorithm 323	EvaluateAirToGroundTransitionModeC	B-19
Algorithm 324	PredictTrackSummary	C-1
Algorithm 325	GetTracksToExtrapolate	C-1
Algorithm 326	ExtrapolateTrack	C-2
Algorithm 327	ExtrapolateActiveTrack	C-2
Algorithm 328	ExtrapolateADSBTrack	C-3
Algorithm 329	ConvertCartesianToPolar2D	C-4
Algorithm 330	DecorrelateTargets	C-6
Algorithm 331	GetTracksToDecorrelate	C-7
Algorithm 332	CorrelationHistoryUpdate	C-7
Algorithm 333	CorrelationHistoryCheck	C-7
Algorithm 334	AddDecorrelatedTrackToTarget	C-8

Algorithm 335	RemoveDecorrelatedTrackFromTarget	C-8
Algorithm 336	GetTrackID	C-8
Algorithm 337	GenerateLeadTrackList	C-9
Algorithm 338	CorrelateTargets	C-11
Algorithm 339	SortByRange	C-12
Algorithm 340	CorrelateIndividualTracks	C-13
Algorithm 341	MergeTargets	C-14
Algorithm 342	MergeTargetDesignations	C-15
Algorithm 343	CorrelateByType	C-16
Algorithm 344	CorrelateImage	C-17
Algorithm 345	CalculateImageRangeRate	C-18
Algorithm 346	CorrelateID	C-19
Algorithm 347	CorrelatePosition	C-20
Algorithm 348	constants	D-4
Algorithm 349	mahal	H-1
Algorithm 350	uchol	H-2
Algorithm 351	block_diag	H-2
Algorithm 352	AngleDifference	H-2
Algorithm 353	Normalize	H-3
Algorithm 354	WrapTo2Pi	H-3
Algorithm 355	WrapToPi	H-3
Algorithm 356	VerticalTRMUpdatePrep	I-2
Algorithm 357	ToVec	J-3

TABLE OF TYPES

Type 1	GeoUtils	D-1
Type 2	StmReport	E-1
Type 3	StmDisplayStruct	E-1
Type 4	TransponderData	E-2
Type 5	ActiveValidationHistory	E-3
Type 6	TrackFile	E-3
Type 7	ADSBTrackFile	E-4
Type 8	CorrelationHistory	E-5
Type 9	DesignationState	E-6
Type 10	HypotheticalModeCTrackFile	E-7
Type 11	ModeCIntervals	E-7
Type 12	ModeCReply	E-7
Type 13	ModeCTrackFile	E-8
Type 14	ModeSTrackFile	E-9
Type 15	OwnDiscreteData	E-10
Type 16	OwnShipData	E-11
Type 17	OwnHistory	E-12
Type 18	VertHistory	E-12
Type 19	RangeAltitude	E-12
Type 20	ReceivedCoordinationData	E-13
Type 21	Target	E-14
Type 22	Database	E-15
Type 23	TrackMap	E-15
Type 24	TrackSummary	E-16
Type 25	ValueTime	E-16
Type 26	LastUpdateRng	E-17
Type 27	TRMInput	E-18
Type 28	TRMOwnInput	E-18
Type 29	TRMIIntruderInput	E-19
Type 30	IntruderHorizontalBelief	E-20
Type 31	IntruderVerticalBelief	E-20
Type 32	OwnVerticalBelief	E-20
Type 33	TRMReport	E-21
Type 34	TRMDisplayData	E-22
Type 35	TRMIIntruderDisplayData	E-22
Type 36	TRMCoordinationInterrogationData	E-23
Type 37	TRMDesignationData	E-23
Type 38	TRMIIntruderDesignationData	E-24
Type 39	TRMLogicModeData	E-24
Type 40	TRMGroundMsgData	E-25
Type 41	TRMRABroadcastData	E-25
Type 42	TRMRACoordinationData	E-26
Type 43	TID	E-26
Type 44	TIDAddress	E-26
Type 45	TIDAltRngBrg	E-27
Type 46	TRMDebugData	E-27
Type 47	TRMIIntruderDebugData	E-27

Type 48	TRMState	E-28
Type 49	TRMOwnState	E-29
Type 50	TRMIintruderState	E-30
Type 51	AdvisoryBeliefState	E-31
Type 52	CombinedVerticalBelief	E-31
Type 53	DisplayLogic	E-32
Type 54	GlobalAdvisory	E-32
Type 55	IndividualAdvisory	E-33
Type 56	OnlineCostState	E-34
Type 57	RDataTable	E-35
Type 58	TauBelief	E-35
Type 59	TRMIintruderData	E-36
Type 60	TRMIndivAdjustState	E-36
Type 61	VerticalRAOutputState	E-37
Type 62	ActionArbitrationCState	E-38
Type 63	ActionArbitrationGlobalCState	E-38
Type 64	AdvisoryRestartCState	E-39
Type 65	AltitudeDependentCOCCState	E-39
Type 66	AltitudeInhibitCState	E-39
Type 67	BadTransitionCState	E-39
Type 68	CompatibilityCState	E-40
Type 69	CoordinationDelayCState	E-40
Type 70	CoordinatedRADeferralCState	E-40
Type 71	CriticalIntervalProtectionCState	E-40
Type 72	CrossingNoAlertCState	E-41
Type 73	InitializationCState	E-41
Type 74	LowAltitudeParallelRADeferralCState	E-41
Type 75	MaxReversalCState	E-41
Type 76	MultithreatCostBalancingCState	E-42
Type 77	PreventEarlyCOCCState	E-42
Type 78	ResponseEstimationCState	E-42
Type 79	RestrictCOCCState	E-42
Type 80	SA01HeuristicCState	E-43
Type 81	SafeCrossingRADeferralCState	E-43
Type 82	TimeBasedNonComplianceCState	E-43

This Page Intentionally Left Blank

1 **Introduction**

1.1 **Purpose**

This document provides the Algorithm Design Description (ADD) for the Surveillance and Tracking Module (STM) and the Threat Resolution Module (TRM) of the next generation Airborne Collision Avoidance System (ACAS X). The algorithms are described at a sufficiently high level to allow for implementation in a variety of software languages and hardware platforms, thereby providing maximum freedom to manufacturers while ensuring the intended output from the system. This document provides descriptions of the algorithm's operations. Every effort has been made to ensure the accuracy of this text, however, in case of a conflict between the description and the function of the algorithm, the algorithm function is correct.

1.2 **Scope**

This document assumes the reader is familiar with ACAS X, including planned variants of the system software and its modular breakdown. If not, the reader is encouraged to review the ACAS X Concept of Operations (ACAS X CONOPS) [4].

The ACAS X algorithm design consists of two separate modules: (1) the Surveillance and Tracking Module (STM), which detects aircraft in the vicinity and tracks their position, and (2) the Threat Resolution Module (TRM), which identifies threats and provides resolution guidance. The algorithms in this ADD are limited to ACAS Xa and ACAS Xo, the variants of ACAS X intended for installation on aircraft that currently require TCAS II equipage. This document contains modifications informed by operational suitability and performance evaluations conducted via modeling and simulation as well as analysis of flight tests in 2013 and 2015 as well as the Operational Evaluation conducted in 2017.

Although this document provides justification for certain aspects of the TRM algorithms, it is not intended to overview the analyses that contributed to the various design decisions. This document does not discuss the process of optimizing the lookup tables. For such details, see [10].

1.3 **Background**

The Traffic alert and Collision Avoidance System (TCAS), currently mandated worldwide on all large transport and cargo aircraft, has been shown to significantly reduce the risk of mid-air collision. TCAS uses an on-board surveillance system to monitor nearby air traffic. The surveillance information is then fed into the threat resolution logic to determine whether to alert the pilots of a potential collision. If deemed necessary to prevent collision, TCAS will issue a resolution advisory to the pilots to climb or descend at a particular rate.

Developing robust collision avoidance logic that reliably prevents collision without excessive alerting is challenging due to sensor error and uncertainty in pilot response and, consequently, the future paths of the aircraft. The current TCAS logic was the result of many years of development and involved the careful engineering of many heuristic rules. Unfortunately, due to the complexity of the logic, it is difficult to revise it to accommodate the evolution of the airspace and the introduction of new surveillance technologies and procedures.

Over the past few years, research has focused on the use of a computational method known as dynamic programming for producing an optimized decision logic for airborne collision avoidance.

This research has resulted in the establishment of the Airborne Collision Avoidance System X (ACAS X) program and is being targeted at becoming the next international standard for collision avoidance for both manned and unmanned aircraft.

1.4 Context

This document incorporates changes informed by operational suitability and performance evaluations of flight tests in 2013 and 2015 and from the Operational Evaluation in 2017.

1.5 Prescription

Each algorithm in this document falls into one of three categories: prescriptive, suggested, or utility.

Prescriptive algorithms specify required functionality which must be included in any ACAS Xa/Xo system in a manner consistent with the Julia code of this document. The prescriptive algorithms therefore fulfill a similar role to "shall" statements within the ACAS Xa/Xo MOPS Volume I [16]. Optimization, refactoring, and translation of the Julia algorithms when developing a final system are acceptable, so long as verification ensures that outputs and side-effects (i.e. database state) for permissible inputs match those of the Julia code contained in this document.

Suggested algorithms are provided as a reference for the implementer when attempting to achieve particular MOPS functional and performance requirements. The suggested algorithms are not the only acceptable approach and may require adaptations due to specific hardware or other system characteristics (i.e. the suggested algorithms are not necessary and may not be sufficient for all systems). There are limited interactions between suggested algorithms and prescriptive algorithms or data structures. Within those areas of interaction the implementer must take careful consideration of the specified behavior. An implementation not leveraging the provided suggested algorithms may safely omit calls to suggested algorithms or data structure elements set solely by the suggested algorithms without affecting prescriptive functionality.

Utility algorithms consist of math functions, data structure modifiers, and other routines that are often found in standard libraries. Like prescriptive algorithms, any implementation must match the Julia code within this document with respect to outputs and side-effects. Unlike the prescriptive algorithms, the utility algorithms may not be specific to ACAS X.

Table 1-1 details the categorization of all algorithms, listed by section, within this document. The listed categorization applies to the entirety of each listed section.

Table 1-1. Algorithm Prescription Categorization

Algorithms	Categorization
Section 2 (STM)	Prescriptive
Section 3 (TRM)	Prescriptive
Appendix A (STM Housekeeping)	Prescriptive
Appendix B (Mode C)	Suggested
Appendix C (Correlation)	Suggested
Appendix D (Constants)	Prescriptive
Appendix E (Data Structures)	Prescriptive
Appendix H (Math Utilities)	Utility
Appendix I (Data Management)	Prescriptive
Appendix J (Standard Library)	Utility

1.6 Specification Language: Julia

In addition to their English descriptions, the algorithms of the ACAS X system are provided in the Julia programming language [2] (<http://julialang.org>), a high-level programming language with a natural and familiar syntax that is being developed at MIT.

The Julia code blocks in this document will be of *named* algorithm components, and will consist either of *types* or *algorithms* (algorithms also are called *functions*).

1.6.1 Data Types

The data types used in the algorithms are summarized in Table 1-2. The basic types of Julia used are Float64, Int64, Bool, Symbol, and Array. Array indexing starts at 1. The aliases R and Z are defined for the basic Julia types Float64 and Int64, respectively.

The type of a variable in Julia can be specified after a character sequence ‘::’ asserting that the next name is the type of the variable. For example, *dz_min*::*R* declares *dz_min* to be a variable of type *R* (Float64). The character sequence ‘<:’ indicates a parent type during type declaration or indicates if the left-hand side of the expression is a subtype of the right-hand side. For example, *ModeSTrackFile* <: *TrackFile* means *ModeSTrackFile* “is a subtype of” *TrackFile*.

This document also uses *composite types*, which organize other types into collections. Composite types are generally defined when several simpler types may have limited meaning individually but describe a clear conceptual object together. For example, TRMINPUT is a composite type consisting of “fields” *own*, and *intruder*. The “fields” of a composite type are made accessible in Julia code by way of the ‘.’ symbol. For example, if *trm_input* is a variable of composite type TRMINPUT, *trm_input.own* gains access to the input data for the own aircraft.

All of the composite types used in the ACAS X system are described within the body of this document and in the appendices. See the “List of Types” for a complete list.

1.6.2 Built-In Values and Functions

In addition to data types, Julia provides several special key words and functions that are used in this document. Special keywords are summarized in Table 1-3. Julia functions are listed in Appendix J. This document relies heavily on the names (defined in Julia’s standard library) “Inf”, “-Inf”, and

Table 1-2. Data Types

Data Type	Source	Julia Type	Description
Array{T,N}	Julia	Array{T,N}	array of data type T with N dimensions e.g., Array{Symbol,1} is a 1-dimensional array of type Symbol e.g., Array{Float64,2} is 2-dimensional array of type Float64
BitArray{N}	Julia	BitArray{N}	array of bits (1s and 0s) with N dimensions
Bool	Julia	Bool	boolean, contains value of TRUE or FALSE
Dict{K,V}	Julia	Dict{K,V}	dictionary, table with keys of type K and values of type V
Float16	Julia	Float16	16-bit floating point (real) value (e.g., 1.0)
Float64	Julia	Float64	64-bit floating point (real) value (e.g., 1.0)
Int64	Julia	Int64	64-bit integer (e.g., 1)
Matrix{T}	Julia	Array{T,2}	alias for 2-dimensional array of data type T
String	Julia	String	character string (e.g., "this is a string")
Symbol	Julia	Symbol	symbol (e.g., :None)
Uint8	Julia	Uint8	8-bit unsigned integer, minimal size for integer values
Uint16	Julia	Uint16	16-bit unsigned integer
Uint32	Julia	Uint32	32-bit unsigned integer
Union(T1,T2,...)	Julia	Union(T1,T2,...)	any one of the specified data types (T_1, T_2 , etc.)
Vector{T}	Julia	Array{T,1}	alias for 1-dimensional array of data type T
R	Alias	Float64	alias for 64-bit floating point (real)
Z	Alias	Int64	alias for 64-bit integer

“NaN”, which represent positive infinity, negative infinity, and not a number. It also relies heavily on the Julia key words for boolean values: TRUE and FALSE.

Table 1-3. Julia Keywords

Keyword	Description
false	boolean value for FALSE
-Inf	negative infinity stored as Float64
Inf	positive infinity stored as Float64
NaN	not a number, used to represent no value, stored as Float64
nothing	a value containing no data, used when no particular value is needed
pi	value of pi (3.141592...)
true	boolean value for TRUE
:undef	symbol representing an undefined value

1.7 Notation

The notational convention was aimed at making the algorithms as compact as possible while ensuring readability and preventing ambiguity. The convention described in this section is applicable to the TRM only.

Table 1-4 summarizes the units used in this document.

Table 1-4. Fundamental Units

Dimension	Unit	Abbreviation
time	seconds	s
distance	feet	ft
angle	radians	rad
velocity	feet per second	ft/s
acceleration	feet per second per second	ft/s ²

1.7.1 Variables

Many of the important variables have both a symbolic notation as well as a textual notation. For example, time would be represented as t symbolically and t textually. The symbolic notation is used in the reference material, the textual notation is used in the pseudocode, without the use of subscripts, superscripts, Greek, or any special mathematical symbols. Associated with each scalar variable is a unit of measurement.

Many of the variables used by the logic are defined in terms of the own, intruder, and relative perspectives. To disambiguate what perspective is used, a text subscript will be used. For example, variable x from the perspective of the own, intruder, and relative perspectives is written x_{own} , x_{int} , and x_{rel} , respectively. In general, $x_{rel} \equiv x_{int} - x_{own}$.

The standard mathematical notation for the first derivative of any variable x with respect to time is written \dot{x} . If x is in ft, then \dot{x} is in ft/s. The second derivative of any variable x with respect to time is written \ddot{x} . If x is in ft, then \ddot{x} is in ft/s². In textual form, \dot{x} is written dx and \ddot{x} is written ddx.

Commonly used variable prefixes and their meanings are listed in Table 1-5.

Table 1-5. Key to Common Variable Prefixes

Prefix	Description
a_	advisory
b_	belief
C_	cost parameter
D_	distance parameter
ddz_	vertical acceleration
dz_	vertical rate
H_	altitude threshold parameter
h_	altitude above ground
input_	input data structure
is_	boolean result
L_	display logic parameter
N_	number parameter
R_	rate parameter
r_	range
ra_	Resolution Advisory
S_	state parameter (could be name or number)
s_c	a context-specific state data structure
st_	state
T_	time parameter
t_	time
w_	weight
X_	coordination parameter
x_	lateral east-component
y_	lateral north-component
z_	barometric altitude

1.7.2 Physical Quantities

Table 1-6 specifies the fundamental physical quantities used by the logic. Time is relative to some fixed instant in time. The variable dt is used to denote the duration of a time step. In the TRM the value of dt is kept fixed at 1 s, the ACAS X cycle time. In the STM the value of dt is relative to the time of applicability of the input data. The lateral and vertical components are with respect to a fixed global coordinate system. Headings are with respect to true north, increasing clockwise.

Table 1-6. Fundamental Physical Quantities

Symbol	Text	Description	Units
<i>t</i>	t	time	s
<i>x</i>	x	lateral east-component	ft
<i>y</i>	y	lateral north-component	ft
<i>z</i>	z	barometric altitude	ft
<i>h</i>	h	altitude above ground	ft
ψ	psi	heading	rad
<i>q</i>	quant	altitude quantization	ft

There are a number of physical quantities that are derived from the fundamental quantities. These derived quantities are used frequently in the pseudocode and merit their own symbols (Table 1-7). Both *chi_abs* and *chi_rel* increase clockwise.

Table 1-7. Derived Physical Quantities

Symbol	Text	Description	Units	Definition
<i>r_slant</i>	r_slant	slant range	ft	$\sqrt{x_{\text{rel}}^2 + y_{\text{rel}}^2 + z_{\text{rel}}^2}$
<i>r_ground</i>	r_ground	ground (or lateral) range	ft	$\sqrt{x_{\text{rel}}^2 + y_{\text{rel}}^2}$
<i>s_ground</i>	s_ground	magnitude of relative velocity vector	ft/s	$\sqrt{\dot{x}_{\text{rel}}^2 + \dot{y}_{\text{rel}}^2}$
χ_{abs}	chi_abs	bearing of intruder relative to north	rad	$\text{atan2}(x_{\text{rel}}, y_{\text{rel}})$
χ_{rel}	chi_rel	bearing of intruder relative to own heading angle	rad	$\text{WRAPTOPI}(\text{atan2}(x_{\text{rel}}, y_{\text{rel}}) - \psi_{\text{own}})$
ϕ_{rel}	phi_rel	absolute value of difference in the direction of the relative horizontal velocity and the bearing of the intruder	rad	$ \text{WRAPTOPI}(\text{atan2}(\dot{y}_{\text{rel}}, \dot{x}_{\text{rel}}) - \text{atan2}(y_{\text{rel}}, x_{\text{rel}})) $

1.7.3 Statistical Quantities

Central to many of the concepts in this document is the normal (or Gaussian) distribution. Univariate normal distributions are defined by a scalar mean μ and scalar standard deviation σ . Multivariate normal distributions are defined by a vector μ and a covariance matrix Σ . In textual form, μ is represented mu, σ is represented sigma, σ^2 is represented var, and Σ is represented Sigma. To associate one of these parameters with a distribution over a particular variable, the variable can be used as a subscript. For example, the standard deviation of *dx_rel* may be written sigma_dx_rel.

Many of the algorithms involve weighted samples. A set of *m* samples, with each sample being represented using an *n* element vector, is represented by an $n \times m$ matrix, where the columns correspond to the different samples. The set of corresponding weights are represented using an *m*-element vector. Weights are generally represented using *w* or textually as weights. Each weight is a real number between 0 and 1, inclusive.

1.7.4 Functions

This document describes the algorithms that implement the ACAS X system. These algorithms are defined as functions in the Julia language. A function takes values as input, applies an algorithm, and outputs values.

- **Input.** The inputs to functions are passed by reference, and only passed by value if they are scalars.
- **Output.** The outputs of a function are initialized in the body of the function.
- **Constant.** Constant variables keep the same values. The assignment of a constant is specified where it is declared.

In some functions, the same variable name is used in the input and output. The output value is provided to the caller via the return statement. In functions that use state variables, the value of a state variable may be modified by the algorithm but not provided as an output of the function.

1.7.5 Diagrams

Diagrams are used in this document to show data processing flow. These diagrams are composed of data blocks and process blocks. Arrows indicate the flow of data into and out of the processing blocks. Data blocks containing scalar variables are represented with a single border. Blocks representing arrays are indicated using a double border. Figure 1-1 shows an example data processing flow involving arrays a and b as input and scalars x and y as output.

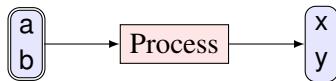


Figure 1-1. Example data processing flow.

The following helper functions are excluded from figures for clarity:

- ACTIONTORATES (Algorithm 273)
- GETOWNHEIGHT (Algorithm 277)
- GETPROTECTIONMODEINDEX (Algorithm 278)
- ISINTRUDERMASTER (Algorithm 287)
- ISPREVENTIVE (Algorithm 291)
- NOINTRUDERSACTION (Algorithm 219)
- RATESTOACTION (Algorithm 301)
- UPDATEINTRUDERVRC (Algorithm 88)
- VERTICALTRMUPDATEPREP (Algorithm 356)

1.8 References

- [1] Dylan M. Asmar, Mykel J. Kochenderfer, and James P. Chryssanthacopoulos. Vertical State Estimation for Aircraft Collision Avoidance with Quantized Measurements. *Journal of Guidance, Control, and Dynamics*, 36(6):1797–1802, October 2013.

-
- [2] Jeff Bezanson, Stefan Karpinski, Viral B. Shah, and Alan Edelman. Julia: A Fast Dynamic Language for Technical Computing. *CoRR*, abs/1209.5145, 2012.
 - [3] Jeffrey Brush and Jessica Lopez. ACAS X: Cost File Compression. AOS-14-0226, March 2014.
 - [4] Mike Castle. Concept of Operations for the Airborne Collision Avoidance System X. ACAS X CONOPS V2 R0, April 2013.
 - [5] Mike Castle. Airborne Collision Avoidance System Xo Systems Requirements Document. ACAS Xo SRD V1 R0, August 2014.
 - [6] Douglas Crockford. The application/json Media Type for JavaScript Object Notation (JSON). Internet informational RFC 4627, July 2006.
 - [7] Zhansheng Duan, Vesselin P. Jilkov, and X. Rong Li. State estimation with quantized measurements: Approximate MMSE approach. In *11th International Conference on Information Fusion*, 2008.
 - [8] Tomas Elder. Intruder Tracker Alternatives Analysis for the Airborne Collision Avoidance System X, Version 1 Revision 0, March 2013.
 - [9] International Civil Aviation Organization. Annex 10 to the Convention on International Civil Aviation, Volume IV, July 2007.
 - [10] Mykel J. Kochenderfer and James P. Chryssanthacopoulos. Robust Airborne Collision Avoidance through Dynamic Programming. Project Report ATC-371, Massachusetts Institute of Technology, Lincoln Laboratory, January 2011.
 - [11] Ed Lorenzo. ACAS X STM Ownship Tracker Alternatives Analysis, Version 2 Revision 1, July 2013.
 - [12] RTCA. Minimum Aviation System Performance Standards (MASPS) for Automatic Dependent Surveillance-Broadcast (ADS-B). DO-242A, June 2002.
 - [13] RTCA. Minimum Operational Performance Standards for 1090 MHz Extended Squitter Automatic Dependent Surveillance - Broadcast (ADS-B) and Traffic Information Services - Broadcast (TIS-B). DO-260B, December 2009.
 - [14] RTCA. Software Considerations in Airborne Systems and Equipment Certification. DO-178C, December 2011.
 - [15] RTCA. Minimum Operational Performance Standards (MOPS) for Aircraft Surveillance Applications (ASA) Systems. DO-317B, June 2014.
 - [16] RTCA. Minimum Operational Performance Standards for Airborne Collision Avoidance System X (ACAS X) (ACAS Xa and ACAS Xo) Volume I. DO-385, September 2018.

This page intentionally left blank.

2 Surveillance and Tracking Module Description

2.1 System Overview

There are multiple entry points to the STM, each associated with a unique type of sensor measurement. A summary of these entry points is given by Table 2-1. The algorithms in this document have been developed to allow the entry points to be accessed in an asynchronous fashion. This design was chosen to allow the STM to accommodate different update rates for the various sensor inputs.

Table 2-1. Summary of Entry Points to the STM

Algorithm	Input Variables
RECEIVEDF0 (Algorithm 1)	Table 2-2
RECEIVEMODECREPLY (Algorithm 11)	Table 2-3
RECEIVEMODECREPLIES (Algorithm 307)	Table B-1
RECEIVESTATEVECTORPOSITIONREPORT (Algorithm 41)	Table 2-5
RECEIVESTATEVECTORVELOCITYREPORT (Algorithm 58)	Table 2-6
RECEIVEMODESTATUSREPORT (Algorithm 60)	Table 2-7
RECEIVEDISCRETES (Algorithm 70)	Table 2-8
RECEIVETARGETDESIGNATION (Algorithm 71)	Table 2-9
RECEIVEBAROALT OBSERVATION (Algorithm 74)	Table 2-10
RECEIVERADALT OBSERVATION (Algorithm 79)	Table 2-11
RECEIVEHEADINGOBSERVATION (Algorithm 80)	Table 2-12
RECEIVEWGS84OBSERVATION (Algorithm 85)	Table 2-13
RECEIVEUF16UDS30 (Algorithm 86)	Table 2-14
RECEIVECAPABILITYREPORT (Algorithm 90)	Table 2-15
GENERATESTMREPORT (Algorithm 96)	N/A

When a sensor measurement is received by the STM, the measurement data is processed by the appropriate algorithms and target or ownship state information is stored in memory. Each measurement is accompanied by a Time of Applicability (TOA) field, thus allowing the STM to receive sensor measurements asynchronously.

The STM stores all information for a single intruding aircraft in a data structure known as a TARGET. A TARGET can contain a MODESTRACKFILE, an ADSBTRACKFILE with ADS-B information, an ADSBTRACKFILE with ADS-R information, and a number of MODECTRACKFILES. Each Track File contains the estimated relative position and velocity of the intruding aircraft as well as any additional information (e.g., reported accuracy or integrity) that has been derived from the corresponding surveillance source. A TARGET can also contain information such as the designation state of an aircraft based on input from the flight crew.

All information related to ownship is stored in a dedicated data structure named *own* of type OWNSHIPDATA. This information includes tracks on ownship heading, barometric altitude, radio altitude and WGS84 position and velocity. Ownship state information is regularly accessed by other functions in the STM to transform intruder measurements from their native coordinates to a common relative coordinate frame.

Upon request, the STM will produce a single data structure that contains all elements needed by the TRM to perform threat evaluation. This is done through a call to GENERATESTMREPORT.

It is expected that such requests will be made on a 1 Hz cycle, however, the algorithms in this document do not preclude requests from being made at other rates. The data structure, known as the STMREPORT, also contains fields that are specifically intended for other external modules such as the pilot display and the aircraft's transponder. The process of report generation iterates through the stored TARGETs and determines which, if any, Track File for a given TARGET is most appropriate to append to the STMREPORT. If necessary, the target's relative position will be extrapolated to the time of the request.

Several reference frames are used throughout the STM. Intruder reports produced by the STM include separate horizontal and vertical estimates of intruder position relative to ownship, each in their own reference frame. Aircraft flying in indicated level flight traverse horizontal planes of constant pressure (i.e. constant pressure altitude, also referred to in this document as barometric altitude), therefore the vertical component of the STM intruder report references the intruder's relative pressure altitude displacement from ownship. The horizontal component of STM intruder reports is delivered in a two-dimension (True North-East) Cartesian reference frame centered about the ownship.

Other reference frames are employed within the tracking portions of the STM specific to each surveillance source. Active surveillance (Mode S and Mode C) provides intruder horizontal observations to the STM in a body-fixed polar (range and bearing) coordinate system. These measurements are converted to the North-up Cartesian coordinate system using the tracked ownship heading value prior to tracking. Passive surveillance (ADS-B) provides intruder observations in geodetic coordinates. Intruder geodetic position and velocity observations are converted to the Cartesian coordinate frame via ownship WGS84 state information prior to tracking. The geodetic-to-Cartesian conversion employed within the STM makes accommodation for the relative rotation of the north-aligned Cartesian reference frame when maneuvering in close proximity to a true pole.

Notice that the Cartesian plane is tangential to the geoid at the ownship position and describes a flat plane in an earth-centered earth-fixed (ECEF) reference frame. The STM horizontal trackers assume that intruder velocities are vectors along this plane. This contrasts with actual intruder trajectories, which typically follow curved surfaces within the ECEF reference frame for level flight maintaining constant orthometric height relative to the geoid. The magnitude of this discrepancy decreases with decreasing intruder range, and is minimal in comparison with other sources of error.

2.2 Active Surveillance

This section defines the algorithms to process surveillance data that have been derived from a transponder based interrogation and reply sequence. Several major functions performed by these algorithms include:

1. The association of a measurement to an existing Track File.
2. The initialization of a new Track File.
3. The update of an existing Track File.
4. The correlation of a Track File to another existing Track File.
5. The update step for active validation.

This section covers reply data that has been elicited from a Mode S or Mode C Transponder. The algorithms in this section do not include the functions that select, prepare and form the intruder information that is provided to the TRM and other external modules. That process is defined in Section 2.6, specifically ADDMODESTRACKTOREPORT (Algorithm 102) and ADDMODECTRACK-



TOREPORT (Algorithm 113).

2.2.1 Input Processing

2.2.1.1 Mode S Surveillance

The entry point to the STM for Mode S surveillance is given by RECEIVEDF0 (Algorithm 1). This algorithm takes as input the variables described in Table 2-2, which are the required measured values and reported data fields associated with the reception of a Downlink Format 0 (DF0) reply message that has been elicited from an intruder's Mode S transponder.

The RECEIVEDF0 algorithm first ensures that an estimate of ownship barometric altitude and ownship heading are available. Next, the provided slant range (r_{slant}), relative bearing (Chi_{rel}), and barometric altitude (z_{baro}) are checked to determine whether all are *NaN*. This special input pattern indicates a request to update the surveillance mode (*surv_mode*) of a Target already in memory without requiring a specific DF0 reply. Under this condition, if a Target is found using ASSOCIATEICAOTOTARGET (Algorithm 4) and a MODESTRACKFILE exists, the surveillance mode is updated and the algorithm terminates.

If any of the provided slant range (r_{slant}), relative bearing (Chi_{rel}), and barometric altitude (z_{baro}) are not *NaN* a check is performed to handle replies from a target under Hybrid Surveillance. When a reply was elicited via a Hybrid Surveillance validation interrogation, and a target is found in memory through a call to ASSOCIATEICAOTOTARGET, a flag (*adsb_qual_override*) is set to override further ADS-B quality checks by the STM for that target. The ADS-B quality check override delegates responsibility for ensuring adequate ADS-B quality of targets under Hybrid Surveillance to the surveillance front end. The ADS-B quality check override will be cleared when a target transitions from the Hybrid Surveillance region into either the Normal or Reduced surveillance regions. No tracks are formed or updated on replies used for Hybrid Surveillance validation.

Standard DF0 processing applies to all other messages received from a target under either Normal or Reduced surveillance. Before association and estimation can be performed, incoming measurements are converted to a different coordinate system in CONVERTMEASUREMENTS (Algorithm 6). If a call to ASSOCIATEDF0TOTARGET is unable to associate the given input data to an existing Target and the measured slant range is valid (i.e., not *NaN*), a new MODESTRACKFILE is created, initialized on the incoming observation data in a call to INITIALIZEMODESTRACKFILE (Algorithm 2), and stored in memory through ADDMODESTRACKTODB (Algorithm 8).

If instead the call to ASSOCIATEDF0TOTARGET successfully associates the input data to an existing Target, the corresponding MODESTRACKFILE is retrieved from memory. If a sufficient time period has passed since the previous update, as dictated by the *min_obs_toa_step* parameter, the retrieved MODESTRACKFILE will be updated through a call to ADVANCEMODESTRACKFILE (Algorithm 3).

A boolean variable *update_track_info* is maintained to indicate when either a new MODESTRACKFILE is created or an existing MODESTRACKFILE is updated. In these cases, additional data from the input DF0 reply is stored within the MODESTRACKFILE, including the reported Mode S address (*mode_s*), altitude quantization (*q_int*), Air-to-Air Reply Information of the DF0 message (*ri*), and track file TOA (*toa*).

This algorithm takes as input the variables described in the following table. This algorithm updates *trk*.

Table 2-2. STM Input Variables - DF0 Message

Variable	Units	Type	Description
r_slant	feet	real	Slant range to intruder
Chi_rel	radian	real	Bearing to intruder (nose is 0, right is $\pi/4$)
z_baro	feet	real	Intruder barometric altitude
mode_s	N/A	uint32	Intruder Mode S address
q_int	feet	uint32	Intruder altitude quantization (25 or 100)
ri	N/A	uint32	Intruder reply information (i.e., equipage)
surv_mode	N/A	uint32	Surveillance Mode
toa	s	real	Time of applicability

Algorithm 1 ReceiveDF0

```

1 function ReceiveDF0(r_slant::R, Chi_rel::R, z_baro::R, mode_s::UInt32, q_int::UInt32, ri::UInt32, surv_mode::  
  UInt32, toa::R)
2   const min_obs_toa_step::R = params().surveillance.min_obs_toa_step
3   if (!isnan(own.toa_h)) && (own.heading_initialized)
4     if (all(isnan([r_slant, Chi_rel, z_baro])))
5       id = AssociateICAOtoTarget(p.18)(mode_s)
6       if (!isnan(id))
7         tgt = RetrieveWithID(p.20)(target_db, id)
8         trk = tgt.modes_track
9         if (TrackExists(p.18)(trk))
10           trk.surv_mode = surv_mode
11         end
12       end
13     elseif (surv_mode == SURVEILLANCE_REGION_HYBRID)
14       id = AssociateICAOtoTarget(p.18)(mode_s)
15       if (!isnan(id))
16         tgt = RetrieveWithID(p.20)(target_db, id)
17         tgt.adsb_qual_override = true
18       end
19     else
20       (Chi_abs, z_rel) = ConvertMeasurements(p.19)(z_baro, Chi_rel, HeadingAtToa(p.85)(toa),  
          BaroAltAtToa(p.78)(toa))
21       id = AssociateDF0toTarget(p.19)(mode_s)
22       update_track_info::Bool = false
23       if (isnan(id) && (!isnan(r_slant)))
24         trk = ModeSTrackFile(p.E-9())
25         InitializeModeSTrackFile(p.16)(trk, r_slant, Chi_abs, z_baro, z_rel, surv_mode, mode_s, toa)
26         AddModeSTrackToDB(p.20)(trk)
27         update_track_info = true
28       elseif (!isnan(id))
29         tgt = RetrieveWithID(p.20)(target_db, id)
30         trk = tgt.modes_track
31         dt = toa - trk.toa
32         if (dt >= min_obs_toa_step)
33           AdvanceModeSTrackFile(p.17)(tgt, trk, r_slant, Chi_abs, z_baro, z_rel, surv_mode, q_int,  
              toa)
34           update_track_info = true
35         end
36       end
37       if (update_track_info)
38         trk.quant = q_int
39         trk.ri = ri
40         trk.toa = toa
41       end
42     end
43   end
44 end

```

When initializing a new MODESTRACKFILE, data sourced from the initiating DF0 reply is provided to the INITIALIZEMODESTRACKFILE (Algorithm 2) algorithm, which ensures that the three tracks of a MODESTRACKFILE are initialized with calls to INITIALIZECARTESIANTRACKER (Algorithm 19), INITIALIZERANGETRACKER (Algorithm 29) and INITIALIZEVERTICALTRACKER (Algorithm 36). The outlier detection count for each of the three tracks (*odc_cart*, *odc_rng* and *odc_vert*) are initialized based on the indicated surveillance region (*surv_mode*). The validity flags for the Cartesian Track and Vertical Track (*valid_cart* and *valid_vert*) are set given the results of the track initializations. Additional data including the mode of surveillance used to elicit the reply (*surv_mode*) and TOA for each track (*toa_rng*, *toa_vert*, *toa_cart*) are saved to the MODESTRACKFILE.

This algorithm takes as input *trk*, *r_slant*, *multithreat*, *Chi_abs*, *z_baro*, *z_rel*, *surv_mode*, *mode_s*, and *toa*. This algorithm updates *trk*.

Algorithm 2 InitializeModeSTrackFile

```

1 function InitializeModeSTrackFile(trk::ModeSTrackFile(p.E-9), r_slant::R, Chi_abs::R, z_baro::R, z_rel::R,
2   surv_mode::UInt32, mode_s::UInt32, toa::R)
3   const mcod_normal::Z = params().surveillance.horizontal.cartfilter.max_outlier_detections
4   const mrod_normal::Z = params().surveillance.horizontal.rangefilter.max_outlier_detections
5   const mvod_normal::Z = params().surveillance.intruder_vertical.max_outlier_detections
6   const mcod_reduced::Z = params().surveillance.horizontal.cartfilter.max_outlier_detections_reduced
7   const mrod_reduced::Z = params().surveillance.horizontal.rangefilter.max_outlier_detections_reduced
8   const mvod_reduced::Z = params().surveillance.intruder_vertical.max_outlier_detections_reduced
9   InitializeCartesianTracker(p.28)(trk, r_slant, Chi_abs, z_rel)
10  InitializeRangeTracker(p.35)(trk, r_slant, z_rel, z_baro)
11  InitializeVerticalTracker(p.42)(trk, z_baro)
12  trk.valid_cart = (trk.updates_cart > 0)
13  trk.valid_vert = (trk.updates_vert > 0)
14  if (surv_mode == SURVEILLANCE_REGION_NORMAL)
15    trk.odc_cart = mcod_normal
16    trk.odc_rng = mrod_normal
17    trk.odc_vert = mvod_normal
18  else
19    trk.odc_cart = mcod_reduced
20    trk.odc_rng = mrod_reduced
21    trk.odc_vert = mvod_reduced
22  end
23  trk.toa_rng = toa
24  trk.toa_vert = toa
25  trk.toa_cart = toa
26  trk.modes = mode_s
27  trk.surv_mode = surv_mode
28  trk.toa = toa
28 end
```

Referenced In: ReceiveDF0(*p.15*)

ADVANCEMODESTRACKFILE (Algorithm 3) begins with a check for a surveillance region transition from Normal to Reduced, where the previous state is indicated by *trk.surv_mode* and the current state is provided as an input (*surv_mode*). When this transition occurs, the outlier detection counts for each tracker are set to the minimum of the current count and the maximum allowed detections in the reduced surveillance region. After completing the surveillance region transition check, the recorded track surveillance region is updated with the new input. The Cartesian Track, Range Track and Vertical Track are then progressed through calls to ADVANCECARTESIANTRACK (Algorithm 23), ADVANCERANGETRACK (Algorithm 31) and ADVANCEVERTICALTRACK (Algorithm 37). A check is performed to ensure that the *adsb_qual_override* is reset if the track is an established MODESTRACKFILE (as indicated by the *trk.valid_rng* flag). Finally, the active validation history is processed. If the range track has at least one valid estimate, the active validation history is updated through ACTIVEVALIDATIONUPDATE (Algorithm 120). Otherwise, the active validation history is reset.

This algorithm takes as input *tgt*, *trk*, *r_slant*, *Chi_abs*, *z_baro*, *z_rel*, *surv_mode*, *q_int*, and *toa*. This algorithm updates *tgt*.

Algorithm 3 AdvanceModeSTrackFile

```

1 function AdvanceModeSTrackFile(tgt::Target(p. E-14), trk::ModeSTrackFile(p. E-9), r_slant::R, Chi_abs::R, z_baro::R,
2   z_rel::R, surv_mode::UInt32, q_int::UInt32, toa::R)
3   const mcod_reduced::Z = params().surveillance.horizontal.cartfilter.max_outlier_detections_reduced
4   const mrod_reduced::Z = params().surveillance.horizontal.rangefilter.max_outlier_detections_reduced
5   const mvod_reduced::Z = params().surveillance.intruder_vertical.max_outlier_detections_reduced
6   if (surv_mode == SURVEILLANCE_REGION_REDUCED) && (trk.surv_mode == SURVEILLANCE_REGION_NORMAL)
7     trk.odc_cart = min(mcod_reduced, trk.odc_cart)
8     trk.odc_rng = min(mrod_reduced, trk.odc_rng)
9     trk.odc_vert = min(mvod_reduced, trk.odc_vert)
10   end
11   trk.surv_mode = surv_mode
12   AdvanceCartesianTrack(p. 31)(trk, r_slant, Chi_abs, z_rel, toa)
13   AdvanceRangeTrack(p. 37)(trk, r_slant, z_rel, z_baro, q_int, toa)
14   AdvanceVerticalTrack(p. 44)(trk, z_baro, q_int, toa)
15   if (trk.valid_rng == true)
16     tgt.adsb_qual_override = false
17   end
18   if (trk.updates_rng > 0)
19     tgt.av_history = ActiveValidationUpdate(p. 121)(tgt, toa)
20   else
21     tgt.av_history = ActiveValidationHistory(p. E-3)()
22     if (tgt.av_state != AV_STATE_INVALID)
23       tgt.av_state = AV_STATE_NOT_VALIDATED
24     end
25   end
25 end

```

Referenced In: ReceiveDF0_(p. 15)

ASSOCIATEICAOTOTARGET (Algorithm 4) searches in the target database for any target containing a track file associated with the given address *icao_address*. If a matching track file is found, the algorithm returns the target ID. If no target is identified found, the algorithm returns *NaN*.

This algorithm takes as input *icao_address*. This algorithm returns *id* or *NaN*.

Algorithm 4 AssociateICAOtoTarget

```

1 function AssociateICAOtoTarget(icao_address::UInt32)
2   for id in keys(target_db)
3     if (TrackExists(p.18)(target_db[id].modes_track))
4       if (target_db[id].modes_track.modes == icao_address)
5         return id::UInt32
6       end
7     end
8     if (TrackExists(p.18)(target_db[id].adsb_track))
9       if (target_db[id].adsb_track.modes == icao_address)
10      return id::UInt32
11     end
12   end
13   if (TrackExists(p.18)(target_db[id].adsr_track))
14     if (target_db[id].adsr_track.modes == icao_address)
15       return id::UInt32
16     end
17   end
18   if (target_db[id].designation_state.is_designated) &&
19     (target_db[id].designation_state.mode_s == icao_address)
20     return id::UInt32
21   end
22 end
23 return NaN
24 end
```

Referenced In: ReceiveTargetDesignation(p.73), AddADSBTrackToDB(p.55), ReceiveCapabilityReport(p.90), ReceiveUF16UDS30(p.88), ReceiveDF0(p.15), AddModeSTrackToDB(p.20), AddADSRTTrackToDB(p.55)

A TARGET can store multiple types of Track Files given different surveillance sources. Multiple Mode C Track Files can be associated to a TARGET, but only a single Mode S Track File, ADS-B Track File, and ADS-R Track File can be stored. Because of this, the TRACKEXISTS (Algorithm 5) can handle different inputs in order to properly check if a given Track exists or not. TRACKEXISTS takes in as input a track, which can either be a vector, a TRACKFILE, or *Nothing*. If the input track is a vector, then a boolean value of TRUE is returned only if the vector of tracks is not empty. Otherwise, a boolean value of TRUE is returned only if the track does not evaluate to *nothing*.

This algorithm takes as input *trk*. This algorithm returns either TRUE or FALSE.

Algorithm 5 TrackExists

```

1 function TrackExists(trk::Union(Vector, TrackFile(p.E-3), Nothing))
2   if typeof(trk) <: Vector
3     return !isempty(trk)
4   else
5     return !is(trk, nothing)
6   end
7 end
```

Referenced In: RemoveDecorrelatedTrackFromTarget(p.C-8), MergeTargets(p.C-15), MergeTargetDesignations(p.C-16), AssociateADSBReportToTarget(p.50), AddADSBTrackToReport(p.110), ReceiveTargetDesignation(p.73), AddTracksToReport(p.98), GetTracksToDecorrelate(p.C-7), AssociateDF0ToTarget(p.19), GetTracksToExtrapolate(p.C-1), AssociateICAOtoTarget(p.18), ReceiveDF0(p.15), RemoveStaleModeCTracks(p.118), CorrelateID(p.C-20), ActiveValidationUpdate(p.121), TargetIsEmpty(p.119), RemoveStaleTracks(p.115), GenerateLeadTrackList(p.C-9), InitializeHeadingTracker(p.83)

CONVERTMEASUREMENTS (Algorithm 6) calculates and returns the absolute bearing of the in-

truder, corrected by the ownship heading, and also the intruder's altitude relative to the ownship. The algorithm takes as inputs z_int , intruder reported altitude; Chi_rel , observed bearing of intruder relative to ownship; Psi_own , ownship heading; and z_own , ownship altitude. The algorithm outputs Chi_abs , intruder absolute bearing; and z_rel , altitude of intruder relative to ownship. When z_int is NaN , z_rel is set to 0.0 as a worst case scenario (representative of a coaltitude intruder).

This algorithm takes as inputs z_int , Chi_rel , Psi_own , and z_own . This algorithm outputs Chi_abs and z_rel .

Algorithm 6 ConvertMeasurements

```

1 function ConvertMeasurements( $z\_int::R$ ,  $Chi\_rel::R$ ,  $Psi\_own::R$ ,  $z\_own::R$ )
2    $Chi\_abs = Chi\_rel + Psi\_own$ 
3   if (isnan( $z\_int$ ))
4      $z\_rel = 0.0$ 
5   else
6      $z\_rel = z\_int - z\_own$ 
7   end
8   return ( $Chi\_abs::R$ ,  $z\_rel::R$ )
9 end
```

Referenced In: InitializeModeCTrack(p. 23), UpdateModeCTrack(p. 24), DetermineModeCReplyAssociationScore(p. B-7), ReceiveDF0(p. 15)

ASSOCIATEDF0TOTARGET (Algorithm 7) attempts to match a Mode S address to an existing intruder track. If a matching track is found, then the id of the track is returned, otherwise NaN is returned. The algorithm takes as input the Mode S address of an intruder ($mode_s$).

This algorithm outputs the index in the target database of the intruder track that matches the provided Mode S address (id).

Algorithm 7 AssociateDF0toTarget

```

1 function AssociateDF0toTarget( $mode\_s::UInt32$ )
2   for  $id$  in keys(target_db)
3     if (TrackExists(p. 18)(target_db[ $id$ ].modes_track))
4       if (target_db[ $id$ ].modes_track.modes == mode_s)
5         return  $id::UInt32$ 
6     end
7   end
8   end
9   return  $NaN$ 
10 end
```

Referenced In: ReceiveDF0(p. 15)

ADDMODESTRACKTODB (Algorithm 8) attempts to correlate a new MODESTRACKFILE (trk) to an existing Target (ASSOCIATEICAOTOTARGET (Algorithm 4)). If the MODESTRACKFILE successfully correlates to a Target then it is added to that Target and the active validation update step is called (ACTIVEVALIDATIONUPDATE (Algorithm 120)). If correlation is not successful, the MODESTRACKFILE is added to a new Target and that Target is stored in memory via a call to ADDTODB (Algorithm 9).

This algorithm takes *trk* as input. The algorithm modifies the *target_db* and may modify *trk* as well.

Algorithm 8 AddModeSTrackToDB

```

1 function AddModeSTrackToDB(trk::ModeSTrackFile(p.E-9))
2   id = AssociateICAtoTarget(p.18)(trk.modes)
3   if (isnan(id))
4     tgt = Target(p.E-14)(trk.modes)
5     tgt.init_time = trk.toa
6     tgt.modes_track = trk
7     AddToDB(p.20)(target_db, tgt)
8   else
9     target_db[id].modes_track = trk
10    target_db[id].av_history = ActiveValidationUpdate(p.121)(target_db[id], trk.toa)
11  end
12 end
```

Referenced In: ReceiveDF0(p.15)
--

The ADDTODB (Algorithm 9) takes a database *db* and a value *v* as input. The database is modified by the addition of the value and incrementing the value counter. The modified database is returned.

Algorithm 9 AddToDB

```

1 function AddToDB{K,V}(db::Database(p.E-15){K,V}, v::V)
2   db.increment += 1
3   while haskey(db, db.increment)
4     db.increment += 1
5   end
6   db[db.increment] = v
7   return db
8 end
```

Referenced In: AddModeCTrackToDB(p.23), AddADSBTrackToDB(p.55), DecorrelateTargets(p.C-6), Hypothetical-TrackTest(p.B-14), AddModeSTrackToDB(p.20), AddADSRTTrackToDB(p.55)
--

The RETRIEVEWITHID (Algorithm 10) searches a given database (input *a*) for an key-value pair with key matching the input *k*. If a key-value pair is identified, then the value is returned (*v*), otherwise *nothing* is returned.

Algorithm 10 RetrieveWithID

```

1 function RetrieveWithID{K,V}(a::Database(p.E-15){K,V}, k::K)
2   v = get(a, k, nothing)
3   return v
4 end
```

Referenced In: ReceiveStateVectorPositionReport(p.49), ReceiveTargetDesignation(p.73), AdjustTargetDesignation-Validity(p.143), SetCoordination(p.91), DeleteIntent(p.88), ReceiveCapabilityReport(p.90), ReceiveStateVectorVelocityReport(p.62), ReceiveModeStatusReport(p.64), ReceiveUF16UDS30(p.88), StmHousekeepingTargetDesignation(p.A-2), ReceiveDF0(p.15)

2.2.1.2 Mode C Surveillance

Mode C surveillance processing is tightly coupled to hardware architecture, design, and performance. As such, no one set of prescriptive Mode C surveillance processing algorithms is appropriate for all ACAS-X implementations. It is the responsibility of the implementer to provide Mode C surveillance processing functions as necessary (such as reply merging, track establishment, on-ground determination, reply association, image track identification, and more) to satisfy the requirements established in the ACAS-X MOPS Vol. I. Singular Mode C replies which have been identified by these functions for use in updating a known track may be provided to the RECEIVEMODECREPLY (Algorithm 11) STM entry point for further STM tracking and processing. RECEIVEMODECREPLY and the algorithms invoked therein form the prescriptive portion of Mode C surveillance processing.

A suite of suggested Mode C surveillance processing algorithms is provided in Appendix B as a recommendation and guide for implementers. This includes an alternative entry point for Mode C surveillance processing, described in RECEIVEMODECREPLIES (Algorithm 307), which ingests a set of Mode C replies received during a whisper-shout sequence and demonstrates a pathway to the functions of reply association, merging, selection, on-ground determination, track establishment, and track update.

Regardless of implementation of the suggested or other entry points, the RECEIVEMODECREPLY algorithm remains a prescriptive requirement in support of the ACAS-X test suite. Furthermore, the STM Mode C algorithms for track initialization (INITIALIZemodectrack (Algorithm 14)) and track update (UPDATemodectrack (Algorithm 15)) must be invoked by any Mode C surveillance processing implementation (see RECEIVEMODECREPLY for reference).

RECEIVEMODECREPLY (Algorithm 11) takes as input a single MODECREPLY, which is described in Table 2-3.

As noted above, this entry point requires that a Surveillance front-end is separately processing Mode C replies and providing a Track ID which associates Mode C replies to Mode C Track Files in memory. A single Mode C reply contains an externally set ID, the measured slant-range, relative bearing, reported altitude code, altitude confidence bits, and TOA.

RECEIVEMODECREPLY ensures that an estimate of ownship barometric altitude and ownship heading are available (i.e., they do not evaluate to *Nan*) as both are needed in the subsequent algorithms. If the coded altitude does not decode to a NAR code and it decodes to an invalid altitude, then no processing is done on this reply and the algorithm terminates.

An attempt to associate the reply to an existing Target is performed through a call to ASSOCIATE-MODECEXTERNALIDTOTRACK (Algorithm 12). This algorithm returns the associated track, or *nothing* if the reply did not associate. If the reply did not associate, then a new MODECTRACKFILE is created, the *external_ID* is set, the track is initialized with a call to INITIALIZemodectrack, and the track is added to the target database through ADDMODECTRACKTODB. Otherwise, if the reply associated with an existing track, that track is updated with a call to UPDATemodectrack.

This algorithm creates or updates a MODECTRACKFILE, *trk*, via calls to INITIALIZemodectrack, ADDMODECTRACKTODB, and UPDATemodectrack.

Table 2-3. STM Input Variables - Mode C Reply Message

Variable	Units	Type	Description
external_ID	N/A	uint32	Optional external ID set by front-end Mode C tracker
coded_alt	ft (once decoded)	integer	Reported Gillham altitude code
conf	N/A	integer	Altitude confidence bits
r_slant	ft	real	Measured slant-range
Chi_rel	rads	real	Relative bearing (nose is 0, right is $\pi/2$)
toa	s	real	Time of applicability

Algorithm 11 ReceiveModeCReply

```

1 function ReceiveModeCReply(reply::ModeCReply(p.E-7))
2   if (isnan(own.toa_h)) || (!own.heading_initialized)
3     return
4   end
5   if (reply.coded_alt != GILLHAM_NAR_CODE)
6     z_baro::R = GillhamDecode(p.25)(reply.coded_alt)
7     if (z_baro == GILLHAM_INVALID_VALUE)
8       return
9     end
10    end
11    trk = AssociateModeCExternalIDtoTrack(p.22)(reply.external_ID)
12    if (trk == nothing)
13      trk = ModeCTrackFile(p.E-8)()
14      trk.external_ID = reply.external_ID
15      trk.on_ground = false
16      InitializeModeCTrack(p.23)(trk, reply)
17      AddModeCTrackToDB(p.23)(trk)
18    else
19      UpdateModeCTrack(p.24)(trk, reply)
20    end
21 end

```

Algorithm 12 AssociateModeCExternalIDtoTrack

```

1 function AssociateModeCExternalIDtoTrack(external_ID::UInt32)
2   for id in keys(target_db)
3     tgt = target_db[id]
4     for modec_track in tgt.modec_tracks
5       if (modec_track.external_ID == external_ID)
6         return modec_track::ModeCTrackFile(p.E-8)
7       end
8     end
9   end
10  return nothing
11 end

```

Referenced In: ReceiveModeCReply(*p.22*)

Algorithm 13 AddModeCTrackToDB

```

1 function AddModeCTrackToDB(trk::ModeCTrackFile(p. E-8))
2   tgt = Target(p. E-14)()
3   tgt.init_time = trk.toa
4   push!(tgt.modec_tracks, trk)
5   AddToDB(p. 20)(target_db, tgt)
6 end
```

Referenced In: EstablishModeCTrack(p. B-11), ReceiveModeCReply(p. 22)
--

Any MODECREPLY that does not associate to an established track is used to initialize new MODECTRACKFILES. INITIALIZEMODECTRACK (Algorithm 14) takes as input a new MODECTRACKFILE and a MODECREPLY to initialize on. Gillham altitude decoding is used to decode the reply's coded altitude (*coded_alt*) into a barometric altitude. The measurement is then transformed into the observation coordinate system through a call to CONVERTMEASUREMENTS and is used to initialize the Cartesian, range, and vertical tracks of the new track through calls to INITIALIZECARTESIANTRACKER, INITIALIZERANGETRACKER, and INITIALIZEVERTICALTRACKER. Additional track information is added to the data structure, which includes outlier detection counts for each tracker, altitude quantization, sensitivity level, and time of applicability values for each tracker.

This algorithm takes as input a MODECTRACKFILE *trk* and a MODECREPLY *reply*. This algorithm updates *trk*.

Algorithm 14 InitializeModeCTrack

```

1 function InitializeModeCTrack(trk::ModeCTrackFile(p. E-8), reply::ModeCReply(p. E-7))
2   const mcod::Z = params().surveillance.horizontal.cartfilter.max_outlier_detections
3   const mrod::Z = params().surveillance.horizontal.rangefilter.max_outlier_detections
4   const mvod::Z = params().surveillance.intruder_vertical.max_outlier_detections
5   toa = reply.toa
6   z_baro::R = GillhamDecode(p. 25)(reply.coded_alt)
7   (Chi_abs, z_rel) = ConvertMeasurements(p. 19)(z_baro, reply.Chi_rel, HeadingAtToa(p. 85)(toa),
     BaroAltAtToa(p. 78)(toa))
8   InitializeCartesianTracker(p. 28)(trk, reply.r_slant, Chi_abs, z_rel)
9   InitializeRangeTracker(p. 35)(trk, reply.r_slant, z_rel, z_baro)
10  InitializeVerticalTracker(p. 42)(trk, z_baro)
11  trk.odc_cart = mcod
12  trk.odc_rng = mrod
13  trk.odc_vert = mvod
14  trk.valid_cart = (trk.updates_cart > 0)
15  trk.valid_vert = (trk.updates_vert > 0)
16  trk.quant = MODE_C_QUANT
17  trk.toa_vert = toa
18  trk.toa_cart = toa
19  trk.toa_rng = toa
20  trk.toa = toa
21 end
```

Referenced In: EstablishModeCTrack(p. B-11), ReceiveModeCReply(p. 22)
--

UPDATEMODECTRACK (Algorithm 15) updates an existing MODECTRACKFILE given a single MODECREPLY. The coded altitude is decoded using GILLHAMDECODE and checked to make sure it returned a valid value. If the time since that last update (*dt*) exceeds *min_obs_toa_step*, then the received measurement is converted to an observation in the track's coordinate system. The

.....

Cartesian, range and vertical tracks are updated using the observations through calls to ADVANCECARTESIANTRACK, ADVANCERANGETRACK, and ADVANCEVERTICALTRACK.

This algorithm takes as input takes as input a MODECTRACKFILE *trk* and a MODECREPLY *reply*. This algorithm updates *trk*.

Algorithm 15 UpdateModeCTrack

```

1 function UpdateModeCTrack(trk::ModeCTrackFile(p.E-8), reply::ModeCReply(p.E-7))
2   const min_obs_toa_step::R = params().surveillance.min_obs_toa_step
3   z_baro::R = GillhamDecode(p.25)(reply.coded_alt)
4   if (z_baro != GILLHAM_INVALID_VALUE)
5     dt = reply.toa - trk.toa
6     if (dt < min_obs_toa_step)
7       return
8   end
9   (Chi_abs, z_rel) = ConvertMeasurements(p.19)(z_baro, reply.Chi_rel, HeadingAtToa(p.85)(reply.toa),
10    BaroAltAtToa(p.78)(reply.toa))
11  AdvanceCartesianTrack(p.31)(trk, reply.r_slant, Chi_abs, z_rel, reply.toa)
12  AdvanceRangeTrack(p.37)(trk, reply.r_slant, z_rel, z_baro, MODE_C_QUANT, reply.toa)
13  AdvanceVerticalTrack(p.44)(trk, z_baro, MODE_C_QUANT, reply.toa)
14  trk.toa = reply.toa
15 end
```

Referenced In: ReceiveModeCReplies(p.B-2), EstablishModeCTrack(p.B-11), ReceiveModeCReply(p.22)
--

GILLHAMDECODE decodes Gillham codes (also known as Gray codes) into altitudes in feet. Prior to any processing, the encoded altitude undergoes a value check. The Gillham code occupies only twelve bits but the encoded altitude is passed to the algorithm as an integer, which stores more than just twelve bits. Any encoded altitude that evaluates to a number greater than or equal to 4096 has therefore been incorrectly processed and should not be used. First, the encoded value is converted to a different code format using a bit ordering defined in CONSTANTS. Next, the new value is checked to see if it is above the maximum possible Gillham encoded value (*MAX_ENCODED_ALT*). If this is true, GILLHAMDECODE returns an indicator that the input was invalid. If *code* is less than the maximum possible value, *code* is checked to see if it is 0. This value indicates that Mode C reply was NAR and so GILLHAMDECODE returns *GILLHAM_NAR_VALUE*. Otherwise, *code* is decoded, beginning with removing the lower three bits of *code* and assigned that value to the variable *FiveHundreds*. The lower three bits are assigned to the variable *OneHundreds*. Next, *FiveHundreds* and *OneHundreds* are converted from Gray codes to binary values. At this point, it should not be possible for *OneHundreds* to have a value of five, six, or zero. If it has any of these values, GILLHAMDECODE returns an indicator that the input was invalid. Otherwise, if *OneHundreds* is equal to seven, it is changed to five. If *FiveHundreds* is odd, *OneHundreds* is set to the difference of six and *OneHundreds*. Finally, the values are converted to feet and the datum offset of 1300 is applied.

This algorithm takes as input *code_format_origin*. This algorithm returns *altitude*.

Algorithm 16 GillhamDecode

```

1 function GillhamDecode(code_format_orig::Z)
2   altitude::R = GILLHAM_INVALID_VALUE
3   if (code_format_orig >= 4096)
4     return altitude::R
5   end
6   code::Z = 0
7   for i in 1:length(GILLHAM_TO_AC)
8     code = code | ((code_format_orig >> (GILLHAM_TO_AC[i]-1)) & 1) << (i-1)
9   end
10  if (code > MAX_ENCODED_ALT)
11    altitude = GILLHAM_INVALID_VALUE
12  elseif (code == 0)
13    altitude = GILLHAM_NAR_VALUE
14  else
15    FiveHundreds::Z = code >> 3
16    OneHundreds::Z = code & 0x07
17    FiveHundreds = FiveHundreds $ (FiveHundreds >> 8)
18    FiveHundreds = FiveHundreds $ (FiveHundreds >> 4)
19    FiveHundreds = FiveHundreds $ (FiveHundreds >> 2)
20    FiveHundreds = FiveHundreds $ (FiveHundreds >> 1)
21    OneHundreds = OneHundreds $ (OneHundreds >> 8)
22    OneHundreds = OneHundreds $ (OneHundreds >> 4)
23    OneHundreds = OneHundreds $ (OneHundreds >> 2)
24    OneHundreds = OneHundreds $ (OneHundreds >> 1)
25    if ((OneHundreds == 5) || (OneHundreds == 6) || (OneHundreds == 0))
26      altitude = GILLHAM_INVALID_VALUE
27    else
28      if (OneHundreds == 7)
29        OneHundreds = 5
30      end
31      if (mod(FiveHundreds, 2) == 1)
32        OneHundreds = 6 - OneHundreds
33      end
34      altitude = ((FiveHundreds * 500) + (OneHundreds * 100)) - 1300
35    end
36  end
37  return altitude::R
38 end

```

Referenced In: AltCodeEstimate(p. B-15), InitializeModeCTrack(p. 23), EvaluateOnGroundModeC(p. B-17), AssociateModeCToTarget(p. B-5), UpdateModeCTrack(p. 24), PromoteModeCTracks(p. B-10), HypotheticalTrackTest(p. B-14), MergeModeCReplies(p. B-3), EstablishModeCTrack(p. B-11), ReceiveModeCReply(p. 22)

GILLHAMENCODE (Algorithm 17) converts altitudes in feet into Gillham codings. It is the same as GILLHAMDECODE except backwards. First, the altitude is rounded to the nearest hundred. Expected values for the encoding are between *NARS_THRESHOLD_MIN* and *NARS_THRESHOLD_MAX* inclusive. If the altitude is not between these values, GILLHAMENCODE returns an indicator that the input was invalid. Otherwise, the altitude is encoded, beginning with the addition of the 1300 foot offset to the altitude. The integer quotient of this value when divided by 500 is calculated, the remainder is then divided by 100. The remainder is not allowed to be zero; if that is the case, one of the "five hundreds" is removed, resulting in five "one hundreds". If *FiveHundreds* is odd, *OneHundreds* is set to the difference of six and *OneHundreds*. If *OneHundreds* is equal to five, it is instead assigned to seven. Next, *FiveHundreds* and *OneHundreds* are converted from binary values to Gray codes (using BINARYTOGRAY (Algorithm 18)). Lastly, the encoded value is converted to a different code format using CONSTANTS.

This algorithm takes as input *altitude*. This algorithm returns *code*.

Algorithm 17 GillhamEncode

```

1 function GillhamEncode(altitude::R)
2   altitude = round(altitude / 100.0) * 100
3   code::Z = convert( Z, GILLHAM_INVALID_VALUE )
4   if (altitude >= NARS_THRESHOLD_MIN) && (altitude <= NARS_THRESHOLD_MAX)
5     value::Z = altitude + 1300
6     FiveHundreds::Z = div(value, 500)
7     OneHundreds::Z = mod(value, 500)/100
8     if (OneHundreds == 0)
9       FiveHundreds = FiveHundreds - 1
10      OneHundreds = 5
11    end
12    if (mod(FiveHundreds, 2) == 1)
13      OneHundreds = 6 - OneHundreds
14    end
15    if (OneHundreds == 5)
16      OneHundreds = 7
17    end
18    FiveHundreds = BinaryToGray(p. 26)(FiveHundreds)
19    OneHundreds = BinaryToGray(p. 26)(OneHundreds)
20    code_format_orig::Z = (FiveHundreds << 3) | OneHundreds
21    code = 0
22    for i in 1:length(AC_TO_GILLHAM)
23      code = code | ((code_format_orig >> (AC_TO_GILLHAM[i]-1)) & 1) << (i-1)
24    end
25  end
26  return code::Z
27 end

```

Referenced In: EncodeCAC(p. 318)

Algorithm 18 BinaryToGray

```

1 function BinaryToGray(value::Z)
2   if(value == 0)
3     return 0
4   end
5   valueOut = value & (1 << (GILLHAM_MAX_BINARY_VALUE - 1))
6   for bitPos in reverse(1:(GILLHAM_MAX_BINARY_VALUE - 1))
7     firstBit::Z = value & (1 << bitPos)
8     secondBit::Z = value & (1 << (bitPos - 1))
9     firstBitShifted::Z = firstBit >> 1
10    valueOut = valueOut | (firstBitShifted $ secondBit)
11  end
12  return valueOut::Z
13 end

```

Referenced In: GillhamEncode(p. 26)

2.2.2 Intruder Horizontal Estimation

The estimation of a target's relative horizontal position and velocity with active reply measurements is performed using two independent tracking algorithms:

- *Cartesian Tracker.* An unscented Kalman filter (UKF) is used to estimate the four Cartesian dimensions of the relative east (positive x) and north (positive y) positions and velocities of the target aircraft. The states are ordered in the state estimate as $[x \ y \ dx \ dy]$. The primary purpose of this tracker is to provide a stable estimate of azimuth and cross ground range rate of a target.

- *Range Tracker:* An unscented Kalman filter is used to estimate the relative ground range and ground range rate of the target aircraft. This filter was designed as a third-order UKF using a range squared motion model to handle the nonlinear property that range is always a positive value while its first and second derivatives, range rate and range acceleration, can be negative or positive values. The states are ordered in the state estimate as $[r \ dr \ ddr]$. While this third-order filter maintains an estimate of range acceleration, only the range and range rate estimates are provided to the TRM.

When an STMREPORT is generated for consumption by the TRM, the azimuth and cross ground range rate components of the Cartesian Track are combined with the Range Track to create a single distribution that represents the relative horizontal state of the target. This process is described in COMBINEANDSAMPLE (Algorithm 106). The approach of separating the problem of tracking the relative horizontal state of a target aircraft into range and azimuth components was found by [8] to yield superior tracker level and system level performance when compared against a stand alone Cartesian UKF.

2.2.2.1 Cartesian Tracker

The Cartesian Tracker is initialized with INITIALIZECARTESIANTRACKER (Algorithm 19). This algorithm takes as input the associated MODESTRACKFILE or MODECTRACKFILE (trk), measured slant-range (r_{slant}), absolute bearing (Chi_abs), and relative altitude (z_{rel}). If the absolute bearing to the intruder is invalid (i.e., evaluates to a NaN), the algorithm produces a no-bearing Track File. This special condition is represented by setting all elements of the tracked state ($trk.mu_cart$) and covariance ($trk.Sigma_cart$) to NaN , resetting the number of track updates ($trk.updates_cart$) to zero and setting the Cartesian Track's validity flag ($trk.valid_cart$) to FALSE.

If the absolute bearing to the intruder is valid, a check is performed to make sure that the range measurement is always at least as big as the relative altitude measurement in magnitude to ensure that certain matrices remain positive semi-definite. The initial position distribution is obtained through the non-linear transformation of the observation from its native coordinate frame to Cartesian coordinates. This process is done in the following steps:

1. The observation distribution represented by the observed values (r_{slant} , Chi_abs and z_{rel}) and the observation noise matrix, R_k , is sampled using SIGMAPONTSAMPLE (Algorithm 20).
2. For each sample, the slant range is verified to be larger than the absolute relative altitude, otherwise the *zero_ground_range* flag is set. Each sample is converted into Cartesian coordinates (O_{xy}) using CONVERTTOCARTESIAN (Algorithm 21).
3. WEIGHTEDMEANANDCOVARIANCE (Algorithm 22) is used to calculate the weighted mean and covariance in the Cartesian coordinate frame (mu_{xy} , $Sigma_{xy}$) from the samples.

In the event that any sample corresponds to states for which the ground range is zero, a constant *gamma* is added to all diagonal elements of the position covariance to ensure numerical stability. The initial velocity distribution is set to a zero-mean Gaussian with covariance (U) read from the parameters file since it cannot be determined from the first measurement alone. Also for numerical stability, *kappa* is added to the diagonal of the full covariance matrix.

This algorithm takes as input the associated MODESTRACKFILE or MODECTRACKFILE (trk), r_{slant} , Chi_abs , and z_{rel} . This algorithm updates trk .

Algorithm 19 InitializeCartesianTracker

```

1 function InitializeCartesianTracker(trk::Union(ModeCTrackFile(p.E-8), ModeSTrackFile(p.E-9)), r_slant::R, Chi_abs::R,
2   z_rel::R)
3   const R_k::Matrix{R} = params().surveillance.horizontal.cartfilter.R
4   const U::Matrix{R} = params().surveillance.horizontal.cartfilter.U
5   const kappa::R = params().surveillance.horizontal.cartfilter.kappa
6   const gamma::Z = params().surveillance.horizontal.cartfilter.gamma
7   if (isnan(Chi_abs))
8     trk.mu_cart = fill(NaN, 4)
9     trk.Sigma_cart = fill(NaN, 4, 4)
10    trk.updates_cart = 0
11    trk.valid_cart = false
12  else
13    if (abs(z_rel) > abs(r_slant))
14      r_slant = abs(z_rel)
15    end
16    (0_pol, w) = SigmaPointSample(p.28)([r_slant, Chi_abs], R_k, 3/2)
17    0_xy = zeros(2, 5)
18    zero_ground_range = false
19    for i in 1:5
20      if (abs(0_pol[1,i]) <= abs(z_rel))
21        zero_ground_range = true
22      end
23      0_xy[:,i] = ConvertToCartesian(p.29)(0_pol[1,i], 0_pol[2,i], z_rel)
24    end
25    (mu_xy,Sigma_xy) = WeightedMeanAndCovariance(p.29)(0_xy,w)
26    if (zero_ground_range)
27      Sigma_xy = Sigma_xy + gamma*eye(2)
28    end
29    trk.mu_cart = [mu_xy, zeros(2)]
30    trk.Sigma_cart = block_diag(p.H-2)(Sigma_xy, U) + kappa*eye(4)
31    trk.updates_cart = 1
32  end
32 end
```

Referenced In: InitializeModeCTrack(p.23), InitializeModeSTrackFile(p.16), AdvanceCartesianTrack(p.31)

Algorithm 20 SigmaPointSample

```

1 function SigmaPointSample(x::Vector{R}, Sigma::Matrix{R}, kappa::R)
2   m = length(x)
3   S = zeros(m,2*m+1)
4   w = zeros(2*m+1)
5   S_u = uchol(p.H-2)(Sigma)
6   S[:,1] = x
7   w[1] = kappa/(m+kappa)
8   for i in 2:m+1
9     S[:,i] = x+sqrt(m+kappa)*S_u[:,i-1]
10    w[i] = 1/(2*(m+kappa))
11    S[:,i+m] = x-sqrt(m+kappa)*S_u[:,i-1]
12    w[i+m] = 1/(2*(m+kappa))
13  end
14  return (S::Matrix{R}, w::Vector{R})
15 end
```

Referenced In: ConvertCartesianToPolar2D(p.C-4), AddADSBTrackToReport(p.110), UpdateCartesianTracker(p.34), CombineAndSample(p.105), AddAltBiasAndSample(p.105), PredictRangeTracker(p.39), ConvertToAzAndXRangeRate(p.106), InitializeCartesianTracker(p.28), UpdateRangeTracker(p.41)

CONVERTTOCARTESIAN (Algorithm 21) takes an intruder position in polar coordinates and calcu-

lates the corresponding relative x and relative y dimensions on a Cartesian plan centered about the ownship.

This algorithm takes as inputs intruder's slant range from ownship (r_{slant}), intruder's bearing from north (Chi_abs), and intruder's altitude relative to ownship (z_{rel}). The algorithm outputs the intruder's x and y coordinates relative to ownship ($cart$).

Algorithm 21 ConvertToCartesian

```

1 function ConvertToCartesian( $r_{slant}:\text{::R}$ ,  $Chi\_abs:\text{::R}$ ,  $z_{rel}:\text{::R}$ )
2    $\rho = \text{GroundRange}_{\text{(p. 35)}}(r_{slant}, z_{rel})$ 
3   if ( $\text{isnan}(Chi\_abs)$ )
4      $cart = [\text{NaN}, \text{NaN}]$ 
5   else
6      $cart = [\rho * \sin(Chi\_abs), \rho * \cos(Chi\_abs)]$ 
7   end
8   return  $cart:\text{::Vector}\{\text{R}\}$ 
9 end
```

Referenced In: UpdateCartesianTracker_(p. 34), AugmentedStateToObservation_(p. 34), DetermineModeCReplyAssociationScore_(p. B-7), InitializeCartesianTracker_(p. 28)

WEIGHTEDMEANANDCOVARIANCE (Algorithm 22) combines sigma point samples and a set of associated weights into a covariance matrix, $Sigma$, and the weighted average, mu . The algorithm takes as inputs a set of samples (z) and a set of weights for those samples (w).

This algorithm outputs the mean of the distribution (mu) and to covariance matrix ($Sigma$).

Algorithm 22 WeightedMeanAndCovariance

```

1 function WeightedMeanAndCovariance( $z:\text{::Array}\{\text{R}\}$ ,  $w:\text{::Vector}\{\text{R}\}$ )
2    $n = \text{size}(z, 1)$ 
3    $p = \text{size}(z, 2)$ 
4    $mu = \text{zeros}(n)$ 
5   for  $i$  in  $1:p$ 
6      $mu += w[i] * z[:, i]$ 
7   end
8    $Sigma = \text{zeros}(n, n)$ 
9   for  $i$  in  $1:p$ 
10     $Sigma += w[i] * (z[:, i] - mu) * (z[:, i] - mu)'$ 
11  end
12  return ( $mu:\text{::Vector}\{\text{R}\}$ ,  $Sigma:\text{::Matrix}\{\text{R}\}$ )
13 end
```

Referenced In: ConvertCartesianToPolar2D_(p. C-4), UpdateCartesianTracker_(p. 34), PredictRangeTracker_(p. 39), ConvertToAzAndXRangeRate_(p. 106), InitializeCartesianTracker_(p. 28), UpdateRangeTracker_(p. 41)

Following initialization, the Cartesian Tracker is advanced through ADVANCECARTESIANTRACK (Algorithm 23). This algorithm checks the validity of the observation data and performs outlier detection to determine how the Cartesian Track should be updated.

Two maximum sequential outlier counts ($max_outliers_normal$ and $max_outliers_reduced$) are retrieved from the parameters file. Two such values are necessary due to the difference in update rates within the Normal and Reduced surveillance regions (applicable to Mode S surveillance only).

.....

The existing Cartesian Track is predicted to the time of the observation through PREDICTCARTESIANTRACKER. Both the prediction and the incoming observation are converted to a state space described by ground range and absolute bearing. The predicted state is converted through a call to LINEARTRANSFORM (where I_{xy} is used as an index to select the X and Y values out of μ_s and Σ_s). The observed slant range and relative altitude is used to determine the ground range of the observation.

When checking whether an observation is an outlier, the ISOUTLIER algorithm computes a difference between its two inputs. In the case of the Cartesian tracker, however, the difference between observed and predicted bearings must be maintained within $-\pi$ to π for mathematical validity. Therefore, this difference is explicitly calculated with ANGLEDIFFERENCE. ISOUTLIER is then invoked with the pre-computed difference as one input and a hardcoded zero value for the comparison input.

The track is coasted or reinitialized if the measured slant-range or absolute bearing are invalid (indicated by evaluating to *Nan*), the relative altitude is significantly greater than the slant-range, the measurement is identified as an outlier (through the call to ISOUTLIER), or if there have been no previous updates to the Cartesian Track. A coast occurs when the outlier detection count (*trk.odc_cart*) permits such an action (i.e., this decrementing counter is greater than zero), there has been more than one update to the Cartesian Track, and if the Cartesian Track is in a valid state. If any of these conditions are false, the track is reinitialized rather than coasted. If this reinitialization occurs on the second update (*trk.updates_cart* is 1 immediately before reinitialization), then the Cartesian track is set to invalid to prevent unwanted behavior during stressing initializations.

If the track is not advanced through either a coast or reinitialization, the track is updated with a call to the update step of the Cartesian Tracker (UPDATECARTESIANTRACKER (Algorithm 27)). The outlier detection count is reset to its maximum value (*max_outliers*) and the update counter for the track is incremented (*trk.updates_cart*). If the update counter exceeds a specified threshold (*detections_to_recover*), the validity flag for the Cartesian Track is reset to TRUE, causing a no-bearing Track File to be marked as having a valid bearing estimate available.

This algorithm takes as input *trk*, *r_slant*, *Chi_abs*, *z_rel*, and *toa*. This algorithm updates *trk*.

Algorithm 23 AdvanceCartesianTrack

```

1 function AdvanceCartesianTrack(trk::Union(ModeCTrackFile(p.E-8), ModeSTrackFile(p.E-9)), r_slant::R, Chi_abs::R,
2   z_rel::R, toa::R)
3   const alt_threshold::R = params().surveillance.horizontal.cartfilter.altitude_threshold
4   const max_outliers_normal::Z = params().surveillance.horizontal.cartfilter.max_outlier_detections
5   const max_outliers_reduced::Z = params().surveillance.horizontal.cartfilter.max_outlier_detections_
6     reduced
7   const outlier_thresh::R = params().surveillance.horizontal.cartfilter.outlier_threshold
8   const detections_to_recover::Z = params().surveillance.horizontal.cartfilter.detections_to_recover
9   const R_q::Matrix{R} = params().surveillance.horizontal.cartfilter.R
10  const I_xy = [1,2]
11  max_outliers::Z = max_outliers_normal
12  if (typeof(trk) == ModeSTrackFile(p.E-9)) && (trk.surv_mode != SURVEILLANCE_REGION_NORMAL)
13    max_outliers = max_outliers_reduced
14  end
15  dt = toa - trk.toa_cart
16  (mu_s, Sigma_s) = PredictCartesianTracker(p.33)(trk.mu_cart, trk.Sigma_cart, dt)
17  (mu_pol, Sigma_pol) = LinearTransform(p.32)(mu_s[I_xy], Sigma_s[I_xy,I_xy])
18  Chi_abs = WrapToPi(p.H-3)(Chi_abs)
19  o_pol = [GroundRange(p.35)(r_slant, z_rel); Chi_abs]
20  d_pol = [o_pol[1] - mu_pol[1]; AngleDifference(p.H-2)(mu_pol[2], o_pol[2])]
21  if (isnan(r_slant) || (isnan(Chi_abs) || (abs(z_rel) - r_slant >= alt_threshold) || (IsOutlier(p.32)(
22    d_pol, 0.0, Sigma_pol + R_q, outlier_thresh)) || (trk.updates_cart == 0)
23  if (trk.odc_cart > 0) && (trk.updates_cart > 1) && (trk.valid_cart)
24    trk.odc_cart -= 1
25  else
26    if (trk.updates_cart == 1)
27      trk.valid_cart = false
28    end
29  InitializeCartesianTracker(p.28)(trk, r_slant, Chi_abs, z_rel)
30  trk.odc_cart = max_outliers
31  trk.toa_cart = toa
32  end
33  else
34    (trk.mu_cart, trk.Sigma_cart) = UpdateCartesianTracker(p.34)(r_slant, Chi_abs, z_rel, mu_s, Sigma_s)
35    trk.odc_cart = max_outliers
36    trk.updates_cart += 1
37    if (trk.updates_cart >= detections_to_recover)
38      trk.valid_cart = true
39    end
40    trk.toa_cart = toa
41  end
42 end
43 end
```

Referenced In: *AdvanceModeSTrackFile*(p.17), *UpdateModeCTrack*(p.24)

LINEARTRANSFORM (Algorithm 24) performs the transformation of a Cartesian state estimate to a polar (range and azimuth) state estimate. The inputs are the mean (*mu_xy*) and covariance matrix (*Sigma_xy*) of the Cartesian state estimate. The outputs are the mean (*mu_ra*) and covariance matrix (*Sigma_ra*) of the polar state estimate. The transformation between Cartesian and polar states is a non-linear process, which is applied directly to the input Cartesian means, resulting in the mean values of *range* and *azimuth* of the polar state. The non-linear transformation cannot be directly applied to the covariance matrix; instead the transformation is linearized about the mean value in the form of the Jacobian matrix *H*, which is applied as a transformation to the input covariance matrix, resulting in the output *Sigma_ra*.

This algorithm inputs *mu_xy* and *Sigma_xy*. This algorithm returns *mu_ra* and *Sigma_ra*.

Algorithm 24 LinearTransform

```

1 function LinearTransform(mu_xy::Vector{R}, Sigma_xy::Matrix{R})
2   range = sqrt(mu_xy[1]^2 + mu_xy[2]^2)
3   azimuth = atan2(mu_xy[1], mu_xy[2])
4   mu_ra = [range, azimuth]
5   H = [mu_xy[1]/mu_ra[1] mu_xy[2]/mu_ra[1]; mu_xy[2]/mu_ra[1]^2 -mu_xy[1]/mu_ra[1]^2]
6   Sigma_ra = H*Sigma_xy*H';
7   return (mu_ra::Vector{R}, Sigma_ra::Matrix{R})
8 end
```

Referenced In: ExtrapolateActiveTrack_(p. C-2), AdvanceCartesianTrack_(p. 31)

Within the STM, outliers are determined by ISOUTLIER (Algorithm 25) by comparing the square of the Mahalanobis distance between an estimate and an observation to a threshold. The outlier threshold for each of the tracks is defined in the parameter file.

This algorithm takes as input *o*, *mu*, *Sigma*, and *thresh*. This algorithm returns TRUE or FALSE.

Algorithm 25 IsOutlier

```

1 function IsOutlier(o::Union(R, Vector{R}), mu::Union(R, Vector{R}), Sigma::Union(R, Matrix{R}), thresh::R)
2   if (thresh > 0) && (!any(isnan(o))) && (!any(isnan(mu))) && (!any(isnan(Sigma)))
3     D = mahal(p.H-1)(o, Sigma, mu, 0)^2
4     return (D >= thresh)::Bool
5   else
6     return false
7   end
8 end
```

Referenced In: ReceiveHeadingObservation_(p. 81), ReceiveBaroAltObservation_(p. 76), AdvanceRangeTrack_(p. 37), ReinitializeRangeTracker_(p. 38), AdvanceADSBTrackPosition_(p. 67), AdvanceVerticalTrack_(p. 44), AdvanceCartesianTrack_(p. 31), AdvanceADSBTrackVelocity_(p. 68)

The UKF assumes a linear, constant-velocity dynamic model, therefore the standard Kalman Filter prediction step is utilized by PREDICTCARTESIANTRACKER (Algorithm 26). The duration of time that has passed since the last successful update to the track (*dt*) is used to formulate the state transition matrix (*F*) and the process noise coupling matrix (*G*). This prediction is not performed if the difference between the TOA and the time of the last track update is less than a defined threshold (*min_extrap_toa_step*). Otherwise, the quantities returned by the algorithm are the predicted state and the state prediction covariance.

This algorithm takes as input *mu*, *Sigma*, and *dt*. This algorithm returns the result of the call to PREDICTKALMANFILTER.

Algorithm 26 PredictCartesianTracker

```

1 function PredictCartesianTracker(mu::Vector{R}, Sigma::Matrix{R}, dt::R)
2   const min_extrap_toa_step::R = params().surveillance.min_extrap_toa_step
3   const Q::R = params().surveillance.horizontal.cartfilter.Q
4   if (dt < min_extrap_toa_step)
5     return (mu::Vector{R}, Sigma::Matrix{R})
6   else
7     F = [1 0 dt 0;
8           0 1 0 dt;
9           0 0 1 0;
10          0 0 0 1]
11    Gdt = 0.5*dt^2
12    G = [Gdt 0 ;
13          0 Gdt;
14          dt 0 ;
15          0 dt ]
16    return PredictKalmanFilter(p. 69)(mu, Sigma, F, G, Q)
17  end
18 end

```

Referenced In: AddADSBTrackToReport(p. 110), AddModeSTrackToReport(p. 102), ExtrapolateActiveTrack(p. C-2), AddModeCTrackToReport(p. 114), AdvanceCartesianTrack(p. 31)

The Cartesian Tracker update step, UPDATECARTESIANTRACKER (Algorithm 27), uses the measured slant range (r_{slant}), absolute bearing (Chi_abs), and relative altitude (z_{rel}), along with the predicted state (mu_s) and state prediction covariance ($Sigma_s$). The predicted observation and the observation prediction covariance are then obtained through the following process:

1. The predicted state (mu_s) and state prediction covariance ($Sigma_s$) are augmented by the measurement noise R_k .
2. The augmented distribution (comprising mu_{aug} and $Sigma_{aug}$) is sampled via SIGMAPONTSAMPLE.
3. Each sample of the augmented distribution (contained in S_{aug}) is transformed by the non-linear function AUGMENTEDSTATETOOBSERVATION, which merges the predicted state with the relative altitude measurement (z_{rel}) and converts the result to Cartesian coordinates. These samples are stored in $gamma$.
4. The transformed samples ($gamma$) are used to calculate the weighted mean (mu_o) and covariance ($Sigma_o$) in the Cartesian coordinate frame.

The Kalman gain (K) is calculated by multiplying the cross covariance of the predicted state (mu_s) and the observation prediction (mu_o) with the inverse of the observation prediction covariance ($Sigma_o$). The observation is then transformed using CONVERTTOCARTESIAN to calculate the observation residual. Finally, the updated state and covariance are calculated using the standard KF equations. For numerical stability purposes, the upper and lower off diagonal terms of ($Sigma$) are averaged.

This algorithm takes as inputs r_{slant} , Chi_abs , z_{rel} , mu_s , and $Sigma_s$. This algorithm outputs mu and $Sigma$.

Algorithm 27 UpdateCartesianTracker

```

1 function UpdateCartesianTracker(r_slant::R, Chi_abs::R, z_rel::R, mu_s::Vector{R}, Sigma_s::Matrix{R})
2   const R_k::Matrix{R} = params().surveillance.horizontal.cartfilter.R
3   (mu_aug, Sigma_aug) = ([mu_s; zeros(2)], block_diag(p.H-2)(Sigma_s, R_k))
4   (S_aug, w_aug) = SigmaPointSample(p.28)(mu_aug, Sigma_aug, 9/2)
5   gamma = zeros(2,13)
6   for i in 1:13
7     gamma[:,i] = AugmentedStateToObservation(p.34)(S_aug[:,i], z_rel)
8   end
9   (mu_o, Sigma_o) = WeightedMeanAndCovariance(p.29)(gamma, w_aug)
10  K::Matrix{R} = zeros(length(mu_s), length(mu_o))
11  for i in 1:length(w_aug)
12    K = K + w_aug[i]*(S_aug[1:4,i] .- mu_s)*(gamma[:,i] .- mu_o)'
13  end
14  K = K * inv(Sigma_o)
15  o = ConvertToCartesian(p.29)(r_slant, Chi_abs, z_rel)
16  mu = mu_s + K*(o .- mu_o)
17  Sigma = Sigma_s .- K*Sigma_o*K'
18  Sigma = (Sigma .+ Sigma') ./ 2
19  return (mu::Vector{R}, Sigma::Matrix{R})
20 end

```

Referenced In: AdvanceCartesianTrack(p.31)

AUGMENTEDSTATETOOBSERVATION (Algorithm 28) merges the predicted state Cartesian Sigma point samples with the measurement and converts the result to Cartesian coordinates.

The algorithm takes as inputs a belief to be augmented (*s_aug*) and intruder's altitude relative to ownship (*z_rel*). The algorithm outputs intruder's x and y coordinates returned from the call to PREDICTKALMANFILTER).

Algorithm 28 AugmentedStateToObservation

```

1 function AugmentedStateToObservation(s_aug::Vector{R}, z_rel::R)
2   (x_rel, y_rel, nothing, nothing, N_r_slant, N_Chi_abs) = s_aug
3   r_slant = sqrt(x_rel2 + y_rel2 + z_rel2) + N_r_slant
4   Chi_abs = atan2(x_rel, y_rel) + N_Chi_abs
5   return ConvertToCartesian(p.29)(r_slant, Chi_abs, z_rel)
6 end

```

Referenced In: UpdateCartesianTracker(p.34)

2.2.2.2 Range Tracker

The Range Tracker is initialized with INITIALIZERANGETRACKER (Algorithm 29). This algorithm takes as input the measured slant-range (*r_slant*) and relative altitude (*z_rel*), which are used to calculate ground range with GROUNDRANGE (Algorithm 30). If the slant-range to the intruder is invalid (i.e., evaluates to a *NaN*), the algorithm produces a no-range Track File that's marked invalid. This special condition is represented by setting all elements of the tracked state (*trk.mu_rng*) and covariance (*trk.Sigma_rng*) to *NaN*, resetting the number of track updates (*trk.updates_rng*) to zero, and setting the Range Track's validity flag (*trk.valid_rng*) to FALSE.

If the slant-range to the intruder is valid, the ground range is set to zero if the magnitude of the relative altitude is greater than the measured slant-range; this is handled by GROUNDRANGE and is done for numerical stability purposes. The initial position distribution is Gaussian centered on

the calculated ground range with variance equivalent to the observation noise (R_k). The initial velocity and acceleration distribution is set to a zero mean Gaussian with covariance U read in from the parameter file.

This algorithm takes as input trk , r_slant , z_rel , and z_baro . This algorithm updates trk .

Algorithm 29 InitializeRangeTracker

```

1 function InitializeRangeTracker( $trk$ :Union(ModeCTrackFile(p. E-8), ModeSTrackFile(p. E-9)),  $r\_slant$ ::R,  $z\_rel$ ::R,
    $z\_baro$ ::R)
2   const  $R_k$ ::R = params().surveillance.horizontal.rangefilter.R
3   const  $U$ ::Matrix{R} = params().surveillance.horizontal.rangefilter.U
4   if (isnan( $r\_slant$ ))
5      $trk.mu\_rng$  = fill(NaN, 3)
6      $trk.Sigma\_rng$  = fill(NaN, 3, 3)
7      $trk.updates\_rng$  = 0
8      $trk.valid\_rng$  = false
9   else
10     $trk.mu\_rng$  = zeros(3)
11     $trk.mu\_rng[1]$  = GroundRange(p. 35)( $r\_slant$ ,  $z\_rel$ )
12     $trk.Sigma\_rng$  = block_diag(p. H-2)(diagm( $R_k$ ),  $U$ )
13     $trk.updates\_rng$  = 1
14  end
15   $trk.is\_rng\_coast$  = false
16   $trk.alt\_not\_avail$  = isnan( $z\_baro$ )
17 end
```

Referenced In: InitializeModeCTrack(p. 23), InitializeModeSTrackFile(p. 16), ReinitializeRangeTracker(p. 38)

GROUND RANGE (Algorithm 30) calculates the ground range of an intruder from ownship by removing the relative vertical component from the slant range. If the intruder's altitude relative to the ownship is NaN , then the slant range is returned. The algorithm takes as inputs intruder's slant range relative to ownship (r_slant) and intruder's altitude relative to ownship (z_rel). The algorithm outputs the intruder's ground range from ownship.

Algorithm 30 GroundRange

```

1 function GroundRange( $r\_slant$ ::R,  $z\_rel$ ::R)
2   if (isnan( $z\_rel$ ))
3     return  $r\_slant$ 
4   elseif ( $abs(z\_rel) < abs(r\_slant)$ )
5     return sqrt( $r\_slant^2 - z\_rel^2$ )
6   else
7     return 0.0
8   end
9 end
```

Referenced In: ConvertToCartesian(p. 29), InitializeRangeTracker(p. 35), AdvanceRangeTrack(p. 37), ReinitializeRangeTracker(p. 38), DetermineModeCReplyAssociationScore(p. B-7), AdvanceCartesianTrack(p. 31), UpdateRangeTracker(p. 41)

Following initialization, the Range Tracker is advanced through ADVANCERANGE TRACK (Algorithm 31). This algorithm checks the validity of the observation data and performs outlier detection to determine how the Range Track should be updated. The first thing done is to have the maximum number of sequential outliers allowed ($max_outliers$) set. This value is dependent on the surveillance region because of the difference in update rate. This is evaluated only for mode-S

tracks because mode-C tracks do not update outside of the Normal surveillance region. If the intruder is outside of the Normal surveillance region, (*max_outliers*) is set to (*max_outliers_reduced*). Afterwards, the existing Range Track is extrapolated to the time of the observation through PREDICTRANGETRACKER. The track is coasted or reinitialized if the measured slant range is invalid (indicated by evaluating to *Nan*), the relative altitude is significantly greater than the slant range by some threshold (*alt_threshold*), if observation is identified as an outlier (through a call to ISOUTLIER), if there have been no previous updates to the Range Track, or if the validity of the altitude from the incoming measurement (*z_baro*) does not match the altitude availability recorded in the track file (*trk.alt_not_avail*). A coast occurs when the outlier detection count (*trk.odc_rng*) permits such an action (i.e., this decrementing counter is greater than zero) and if there has been more than one update to the Range Track. The information for the measurement that was coasted through is stored in *trk.last_update_rng*. Otherwise the track is reinitialized using REINITIALIZERANGETRACKER (Algorithm 32).

If the track is not advanced through either a coast or reinitialization, the algorithm checks to see if the observation is the second measurement received since the last time the track was initialized (indicated by *trk.updates_rng*) and whether the time passed since the last update (*dt*) is large enough (by some threshold *dt_small*). If both of those conditions are met, then the track velocity is replaced by taking the difference in position over the difference in time of the current measurement and the previous track position. Afterwards, the existing track is extrapolated to the time of the observation through PREDICTRANGETRACKER using the new track velocity and the track is updated with a call to the UPDATERANGETRACKER. If the track has been coasted since the last time it was updated (indicated by *trk.is_rng_coast*), then the track velocity is replaced by taking the difference in position over the difference in time of the current measurement and the previous track position (*mu_rng_old*). The outlier detection count is then reset to the maximum value (*max_outliers*), the update counter for the track is incremented (*trk.updates_rng*), the validity flag of the track is enabled (*trk.valid_rng*), and the time of applicability of the track is updated (*trk.toa_rng*).

This algorithm takes as input *trk*, *r_slant*, *z_baro*, *z_rel*, *q_int*, and *toa*. This algorithm updates *trk*.

Algorithm 31 AdvanceRangeTrack

```

1 function AdvanceRangeTrack(trk::Union(ModeCTrackFile(p.E-8), ModeSTrackFile(p.E-9)), r_slant::R, z_rel::R, z_baro
2   ::R, q_int::UInt32, toa::R)
3   const alt_threshold::R = params().surveillance.horizontal.rangefilter.altitude_threshold
4   const max_outliers_normal::Z = params().surveillance.horizontal.rangefilter.max_outlier_detections
5   const max_outliers_reduced::Z = params().surveillance.horizontal.rangefilter.max_outlier_detections_-
6   reduced
7   const outlier_thresh::R = params().surveillance.horizontal.rangefilter.outlier_threshold
8   const dt_small::R = params().surveillance.horizontal.rangefilter.dt_too_small_for_velocity
9   max_outliers::Z = max_outliers_normal
10  if (typeof(trk) == ModeSTrackFile(p.E-9)) && (trk.surv_mode != SURVEILLANCE_REGION_NORMAL)
11    max_outliers = max_outliers_reduced
12  end
13  dt = toa - trk.toa_rng
14  (mu_s,Sigma_s) = PredictRangeTracker(p.39)(trk.mu_rng,trk.Sigma_rng,dt)
15  measurement_invalid::Bool = (isnan(r_slant)) || (abs(z_rel) - r_slant >= alt_threshold)
16  measurement_outlier::Bool = IsOutlier(p.32)(GroundRange(p.35)(r_slant,z_rel),mu_s[1],Sigma_s[1,1],
17    outlier_thresh)
18  alt_avail_mismatch::Bool = trk.alt_not_avail != isnan(z_baro)
19  if (measurement_invalid) || (measurement_outlier) || (trk.updates_rng == 0) || (alt_avail_mismatch)
20    if (trk.odc_rng > 0) && (trk.updates_rng > 1)
21      trk.odc_rng -= 1
22      trk.is_rng_coast = true
23      trk.last_update_rng.r_slant = r_slant
24      trk.last_update_rng.z_rel = z_rel
25      trk.last_update_rng.z_baro = z_baro
26      trk.last_update_rng.toa = toa
27      trk.last_update_rng.measurement_invalid = measurement_invalid
28    else
29      ReinitializeRangeTracker(p.38)(trk, r_slant, z_rel, z_baro, q_int, toa, measurement_invalid)
30      trk.odc_rng = max_outliers
31      trk.toa_rng = toa
32    end
33  else
34    if (trk.updates_rng == 1) && (dt > dt_small)
35      trk.mu_rng[2] = (GroundRange(p.35)(r_slant,z_rel) - trk.mu_rng[1]) / dt
36      (mu_s,Sigma_s) = PredictRangeTracker(p.39)(trk.mu_rng,trk.Sigma_rng,dt)
37    end
38    mu_rng_old::R = trk.mu_rng[1]
39    (trk.mu_rng,trk.Sigma_rng) = UpdateRangeTracker(p.41)(r_slant,z_rel,mu_s,Sigma_s,q_int)
40    if (trk.is_rng_coast)
41      trk.mu_rng[2] = (GroundRange(p.35)(r_slant,z_rel) - mu_rng_old) / dt
42      trk.mu_rng[3] = 0
43    end
44    trk.odc_rng = max_outliers
45    trk.updates_rng += 1
46    trk.valid_rng = true
47    trk.toa_rng = toa
48    trk.is_rng_coast = false
49  end
50 end

```

Referenced In: [AdvanceModeSTrackFile\(p.17\)](#), [UpdateModeCTrack\(p.24\)](#)

Algorithm 32 ReinitializeRangeTracker

```

1 function ReinitializeRangeTracker(trk::Union(ModeCTrackFile(p. E-8), ModeSTrackFile(p. E-9)), r_slant::R, z_rel::R,
2   z_baro::R, q_int::UInt32, toa::R, measurement_invalid::Bool)
3   const outlier_thresh::R = params().surveillance.horizontal.rangefilter.outlier_threshold
4   initialize_on_current_measurement::Bool = true
5   if (trk.odc_rng == 0) && (!trk.last_update_rng.measurement_invalid) && (!measurement_invalid)
6     InitializeRangeTracker(p.35)(trk, trk.last_update_rng.r_slant, trk.last_update_rng.z_rel, trk.
7       last_update_rng.z_baro)
8     dt = toa - trk.last_update_rng.toa
9     (mu_s, Sigma_s) = PredictRangeTracker(p.39)(trk.mu_rng, trk.Sigma_rng, dt)
10    measurement_outlier::Bool = IsOutlier(p.32)(GroundRange(p.35)(r_slant, z_rel), mu_s[1], Sigma_s[1,1],
11      outlier_thresh)
12    alt_avail_mismatch::Bool = trk.alt_not_avail != isnan(z_baro)
13    if (!measurement_outlier) && (!alt_avail_mismatch)
14      (trk.mu_rng, trk.Sigma_rng) = UpdateRangeTracker(p.41)(r_slant, z_rel, mu_s, Sigma_s, q_int)
15      trk.updates_rng += 1
16      trk.valid_rng = true
17      trk.is_rng_coast = false
18      initialize_on_current_measurement = false
19    end
20  end
21  trk.toa_rng = toa
22 end

```

Referenced In: AdvanceRangeTrack(p.37)

The prediction step for the Range Tracker departs from the standard KF paradigm in PREDICTRANGETRACKER (Algorithm 33). To handle the nonlinearity that occurs when a target at close range transitions from converging (negative range rate) to diverging (positive range rate), this algorithm uses a range squared motion model. Range squared tracking exploits the fact that the square of the ground range between two non-accelerating aircraft is always parabolic as a function of time, regardless of the closest point of approach. As a result, a second order motion model more accurately captures the dynamics near the closest point of approach without adding additional process noise.

The prediction is not performed if the difference between the TOA and the time of the last track update is less than a defined threshold (*min_extrap_toa_step*). Otherwise, this process is performed by the following steps:

1. The range state vector and covariance are augmented by the process noise Q .
2. The augmented distribution is sampled using sigma point sampling.
3. Negative accelerations in range cannot occur between two non-accelerating aircraft, and their presence can cause the range squared motion model to become unstable. To prevent this from occurring, all samples are set to a minimum range acceleration of zero. Any previous negative range accelerations are saved in *acc_remainder* for application after the predict step.
4. The state vector of each sample is transformed into range squared coordinates.
5. The range squared position, velocity, and acceleration dimensions of each sample are predicted forward in time by dt .
6. The position dimension of each sample is transformed into range coordinates by taking the square root of the absolute value of the predicted range squared position (*sqrt_r2*). Additional prediction is applied to this sample in range coordinates given the acceleration term of the process noise and the *acc_remainder* stored above.

7. If $\sqrt{r_2}$ is too close to zero, the transformation of predicted velocity and acceleration to range coordinates will be numerically unstable. If $\sqrt{r_2}$ is smaller than $\sqrt{r_2}_{threshold}$, the range squared prediction step is ignored. The velocity and acceleration dimensions are predicted forward in time by dt seconds in range coordinates, while assuming that the closest point of approach has occurred.
8. Otherwise, the predicted range squared velocity and acceleration dimensions of each sample are transformed into range coordinates. They are further predicted forward in time by dt seconds, in range coordinates, using the acceleration term from the process noise and $acc_remainder$.
9. The transformed samples are used to calculate the weighted mean (mu_s) and covariance ($Sigma_s$) of the range position, velocity, and acceleration.

This algorithm takes as input mu , $Sigma$, and dt . This algorithm returns mu_s and $Sigma_s$.

Algorithm 33 PredictRangeTracker

```

1 function PredictRangeTracker(mu::Vector{R}, Sigma::Matrix{R}, dt::R)
2   const Q::R = params().surveillance.horizontal.rangefilter.Q
3   const min_extrap_toa_step::R = params().surveillance.min_extrap_toa_step
4   const sqrt_r2_threshold::R = params().surveillance.horizontal.rangefilter.sqrt_r2_threshold
5   if (dt < min_extrap_toa_step) || (all(isnan(mu)))
6     return (mu::Vector{R}, Sigma::Matrix{R})
7   else
8     mu_aug = [mu; 0.0]
9     Sigma_aug = block_diag(p.H-2)(Sigma, diagm(Q))
10    (S_aug, w_aug) = SigmaPointSample(p.28)(mu_aug, Sigma_aug, 2.0)
11    S_aug_r2 = zeros(4,9)
12    gamma = zeros(3,9)
13    gamma_r2 = zeros(3,9)
14    for i in 1:length(w_aug)
15      acc_remainder::R = 0.0
16      if (S_aug[3,i] < 0.0)
17        acc_remainder = S_aug[3,i]
18        S_aug[3,i] = 0.0
19    end
20    S_aug_r2[1,i] = (S_aug[1,i])^2
21    S_aug_r2[2,i] = 2 * (S_aug[1,i]) * (S_aug[2,i])
22    S_aug_r2[3,i] = 2 * (S_aug[2,i])^2 + 2 * (S_aug[1,i]) * (S_aug[3,i])
23    gamma_r2[1,i] = S_aug_r2[1,i] + S_aug_r2[2,i] * dt + 0.5 * S_aug_r2[3,i] * dt * dt
24    gamma_r2[2,i] = S_aug_r2[2,i] + S_aug_r2[3,i] * dt
25    gamma_r2[3,i] = S_aug_r2[3,i]
26    sqrt_r2::R = sqrt(abs(gamma_r2[1,i]))
27    gamma[1,i] = sqrt_r2 + 0.5 * (S_aug[4,i] + acc_remainder) * dt * dt
28    if (sqrt_r2 < sqrt_r2_threshold)
29      gamma[2,i] = S_aug[4,i] * dt
30      gamma[3,i] = S_aug[3,i] + S_aug[4,i] + acc_remainder
31      if (gamma[3,i] < 0.0)
32        gamma[3,i] = 0.0
33      end
34    else
35      gamma[2,i] = ((gamma_r2[2,i]) / (2 * sqrt_r2)) + (S_aug[4,i] + acc_remainder)*dt
36      gamma[3,i] = ((gamma_r2[3,i] - ((gamma_r2[2,i])^2 / (2 * sqrt_r2^2))) / (2 * sqrt_r2)) + (
37          S_aug[4,i] + acc_remainder)
38    end
39    (mu_s, Sigma_s) = WeightedMeanAndCovariance(p.29)(gamma, w_aug)
40    return (mu_s::Vector{R}, Sigma_s::Matrix{R})
41  end
42 end
```

Referenced In: AdvanceRangeTrack^(p. 37), ReinitializeRangeTracker^(p. 38), AddModeSTrackToReport^(p. 102), ExtrapolateActiveTrack^(p. C-2), AddModeCTrackToReport^(p. 114), ActiveValidationUpdate^(p. 121)

The update step for the Range Tracker is described in UPDATERANGETRACKER (Algorithm 34). This step departs from the standard Kalman filter in that the measurement noise matrix of ground range is not static. Instead, the uncertainty in ground range is a function of the measured slant range (r_{slant}), relative altitude separation (z_{rel}), and the measurement covariances of these quantities (R_{sr} and R_q , respectively), and must be calculated for each measurement.

This algorithm takes as input r_{slant} , z_{rel} , μ_s , $Sigma_s$, and q_{int} . This algorithm returns μ , $Sigma$.

The r_{slant} and z_{rel} distributions are sampled using sigma point samples. For each sample, the resulting ground range ($gamma[1,i]$) is calculated using GROUNDRANGE. These ground range samples are then used to calculate the weighted mean (σ) and covariance (R_k) of the calculated ground range. If the ground range of too many samples evaluates to zero, the calculated covariance of these samples will be too small, and will not correctly reflect the true measurement uncertainty. When the number of samples whose ground range evaluates to zero ($zero_gr_count$) is greater than or equal to $max_zero_gr_count$, an alternative formula is used that directly applies the relative altitude uncertainty due to quantization (Shepphard's correction) to the ground range dimension.

UPDATERANGETRACKER then proceeds as a standard KF update step using UPDATEKALMANFILTER, which is called with σ as the measured ground range and R_k as the measurement covariance, as well as the predicted state (μ_s), the state prediction covariance ($Sigma_s$), and the observation coupling matrix (H). The Kalman filter matrices are identified in the following table:

Table 2-4. Kalman F, G, H, Q, and R Matrix Definitions

Matrix	Definition
F	State Transition Matrix
G	Process Noise Coupling Matrix
H	Observation Coupling Matrix
Q	Process Noise Matrix
R	Measurement Noise Matrix

This algorithm takes as input r_{slant} , z_{rel} , μ_s , $Sigma_s$, and q_{int} . This algorithm returns μ and $Sigma$.

Algorithm 34 UpdateRangeTracker

```

1 function UpdateRangeTracker(r_slant::R, z_rel::R, mu_s::Vector{R}, Sigma_s::Matrix{R}, q_int::UInt32)
2   const R_sr::R = params().surveillance.horizontal.rangefilter.R
3   const max_zero_gr_count::Z = params().surveillance.horizontal.rangefilter.max_zero_gr_count
4   const epsilon::R = params().surveillance.horizontal.rangefilter.epsilon
5   const H::Matrix{R} = [1.0 0.0 0.0]
6   zero_gr_count::Z = 0
7   R_q::R = (q_int2) / 12
8   (mu_aug, Sigma_aug) = ([r_slant; z_rel], block_diag(p.H-2)(diagm(R_sr), diagm(R_q)))
9   (S_aug, w_aug) = SigmaPointSample(p.28)(mu_aug, Sigma_aug, 1.0)
10  gamma = zeros(1,5)
11  for i in 1:5
12    gamma[1,i] = GroundRange(p.35)(S_aug[1,i], S_aug[2,i])
13    if (abs(gamma[1,i]) < epsilon)
14      zero_gr_count += 1
15    end
16  end
17  (o, R_k) = WeightedMeanAndCovariance(p.29)(gamma, w_aug)
18  if (zero_gr_count >= max_zero_gr_count)
19    if (r_slant > sqrt(R_q))
20      R_k = ((r_slant + sqrt(R_q))2 - (r_slant - sqrt(R_q))2)
21    else
22      R_k = (r_slant + sqrt(R_q))2
23    end
24  end
25  (mu, Sigma) = UpdateKalmanFilter(p.41)(o, mu_s, Sigma_s, H, R_k)
26  return (mu::Vector{R}, Sigma::Matrix{Rend

```

Referenced In: AdvanceRangeTrack([p.37](#)), ReinitializeRangeTracker([p.38](#))

Algorithm 35 UpdateKalmanFilter

```

1 function UpdateKalmanFilter(o::Union(R, Vector{R}), mu_s::Vector{R}, Sigma_s::Matrix{R}, H::Matrix{R}, R_k::  
  Union(R, Matrix{R}))
2   S = H*Sigma_s*H' .+ R_k
3   K = Sigma_s * H' * inv(S)
4   mu = mu_s + K*(o .- H*mu_s)
5   I = eye(size(Sigma_s,1))
6   Sigma = (I - K*H)*Sigma_s*(I - K*H)' + K*R_k*K'
7   return (mu::Vector{R}, Sigma::Matrix{Rend

```

Referenced In: ReceiveBaroAltObservation([p.76](#)), UpdateADSBTrackerVelocity([p.70](#)), UpdateADSBTrackerPosition([p.69](#)), UpdateRangeTracker([p.41](#))

2.2.3 Intruder Vertical Estimation

The estimation of a target's vertical position and velocity with ADS-B position messages, DF0, or Mode C reply messages is performed using a linear Kalman filter with some modifications to account for quantized measurements as described in [1]. This tracker is initialized with INITIALIZEVERTICALTRACKER (Algorithm 36). This algorithm produces a NAR track if the input altitude measurement is invalid (i.e., evaluates to a *Nan*). This special condition is represented by setting all elements of the tracked state (*trk.mu_vert*) and covariance (*trk.Sigma_vert*) to *Nan*, resetting the number of track updates (*trk.updates_vert*) to zero, and setting the Vertical Track's validity flag (*trk.valid_vert*) to FALSE.

If the intruder altitude measurement is valid, the initial altitude distribution is Gaussian centered on the input altitude measurement (z_int) while the initial altitude rate distribution is considered to be zero mean Gaussian. The initial variance for these distributions (var_zint and var_dzint respectively) are read in from the parameter file as they cannot be determined on a single measurement alone. These values are independent of quantization, and are selected to allow any track to quickly converge to a steady state covariance estimate. The vertical rate arrow for the display is initialized as level.

This algorithm takes as input trk , and z_int . This algorithm updates trk .

Algorithm 36 InitializeVerticalTracker

```

1 function InitializeVerticalTracker( $trk$ :Union(ModeCTrackFile(p. E-8), ModeSTrackFile(p. E-9), ADSBTrackFile(p. E-4)),  $z\_int$ 
   ::R)
2   const  $var\_zint$ ::R = params().surveillance.intruder_vertical.var_zint
3   const  $var\_dzint$ ::R = params().surveillance.intruder_vertical.var_dzint
4   if (isnan( $z\_int$ )) || ( $z\_int$  > NARS_THRESHOLD_MAX) || ( $z\_int$  < NARS_THRESHOLD_MIN)
5      $trk.mu\_vert$  = fill(NaN, 2)
6      $trk.Sigma\_vert$  = fill(NaN, 2, 2)
7      $trk.updates\_vert$  = 0
8      $trk.valid\_vert$  = false
9   else
10     $trk.mu\_vert$  = [ $z\_int$ , 0]
11     $trk.Sigma\_vert$  = diagm([ $var\_zint$ ,  $var\_dzint$ ])
12     $trk.updates\_vert$  = 1
13     $trk.display\_arrow\_current$  = DISPLAY_ARROW_LEVEL
14     $trk.vert\_arrow\_history$  = Array(Z, 0)
15    unshift!( $trk.vert\_arrow\_history$ , DISPLAY_ARROW_LEVEL)
16  end
17 end
```

Referenced In: InitializeADSBTrackFile(p. 51), InitializeModeCTrack(p. 23), InitializeModeSTrackFile(p. 16), AdvanceVerticalTrack(p. 44)

Following initialization, the Vertical Tracker is advanced through ADVANCEVERTICALTRACK (Algorithm 37). This algorithm checks the validity of the observation data and performs outlier detection to determine how the Vertical Track should be updated. Two maximum sequential outlier counts ($max_outliers_normal$ and $max_outliers_reduced$) are retrieved from the parameters file. Two such values are necessary due to the difference in update rates within the respective surveillance regions. The $max_outliers_reduced$ value is applicable to Mode-S tracks in the reduced surveillance region only. The existing Vertical Track is extrapolated to the time of the observation through PREDICTVERTICALTRACKER. If this is the second update since initialization, the vertical rate observation is compared to a static window specified in the parameter file ($window$). This allows for track startup on high vertical rate intruders without requiring an artificially high initial vertical rate uncertainty. All other updates are subject to the typical Mahalanobis distance test for outlier detection. The effects of altitude quantization on track uncertainty are explicitly addressed in UPDATEVERTICALTRACKER, but these effects must be anticipated in the covariance provided for outlier detection prior to the update, therefore an $altitude_inflation$ derived from Sheppard's correction is added to the $Sigma_s$ used for Mahalanobis distance based outlier detection. The track is coasted or reinitialized if the altitude measurement is invalid (indicated by evaluating to NaN), if the altitude measurement is outside of the allowed bounds (specified by $NARS_THRESHOLD_MAX$ and $NARS_THRESHOLD_MIN$), if the measurement is identified as an outlier (through a call to ISOUTLIER or the static window described above), or if there have been no previous updates to the vertical track ($trk.updates_vert$). A coast occurs when the outlier detection count ($trk.odc_vert$)

permits such an action (i.e., this decrementing counter is greater than zero), there has been more than one update to the Vertical Track, and the Vertical Track is in a valid state. If any of these conditions are false, the track is reinitialized.

If the track is not advanced through either a coast or reinitialization, the track is updated with a call to the update step of the Vertical Tracker (UPDATEVERTICALTRACKER (Algorithm 39)). If this is the second update since track initialization, the vertical rate is seeded based on the observation prior to the call to UPDATEVERTICALTRACKER. The vertical uncertainty of the update (R_q) is selected based on the reported quantization of the observation (q_{int}). The outlier detection count is then reset to its maximum value and the update counter for the track is incremented. If the update counter exceeds a specified threshold ($detections_to_recover$), the validity flag for the Vertical Track is set to TRUE, thus causing a NAR Track File to be marked as having a valid altitude estimate available. The vertical rate arrow for the pilot's display is then set using VERTICALRATEARROWUPDATE (Algorithm 109).

This algorithm takes as input trk , z , q_{int} , and toa . This algorithm updates trk .

Algorithm 37 AdvanceVerticalTrack

```

1 function AdvanceVerticalTrack(trk::Union(ModeCTrackFile(p.E-8), ModeSTrackFile(p.E-9), ADSBTrackFile(p.E-4)), z::R,
2   q_int::UInt32, toa::R)
3   const max_outliers_normal::Z = params().surveillance.intruder_vertical.max_outlier_detections
4   const max_outliers_reduced::Z = params().surveillance.intruder_vertical.max_outlier_detections_reduced
5   const outlier_thresh::R = params().surveillance.intruder_vertical.outlier_threshold
6   const detections_to_recover::Z = params().surveillance.intruder_vertical.detections_to_recover
7   const R_25::R = params().surveillance.intruder_vertical.R_25ft
8   const R_100::R = params().surveillance.intruder_vertical.R_100ft
9   const window::Z = params().surveillance.intruder_vertical.outlier_window
10  max_outliers::Z = max_outliers_normal
11  if (typeof(trk) == ModeSTrackFile(p.E-9)) && (trk.surv_mode != SURVEILLANCE_REGION_NORMAL)
12    max_outliers = max_outliers_reduced
13  end
14  dt = toa - trk.toa_vert
15  (mu_s, Sigma_s) = PredictVerticalTracker(p.45)(trk.mu_vert, trk.Sigma_vert, dt)
16  altitude_inflation::R = (q_int2) / 12
17  if (isnan(z))
18    outlier = true
19  elseif (trk.updates_vert == 1)
20    dz = z - trk.mu_vert[1]
21    outlier = (abs(dz/dt) > window)
22  else
23    outlier = IsOutlier(p.32)(z, mu_s[1], Sigma_s[1,1]+altitude_inflation, outlier_thresh)
24  end
25  if (outlier) || (z > NARS_THRESHOLD_MAX) || (z < NARS_THRESHOLD_MIN) || (trk.updates_vert == 0)
26    if (trk.odc_vert > 0) && (trk.updates_vert > 1) && (trk.valid_vert)
27      trk.odc_vert -= 1
28    else
29      InitializeVerticalTracker(p.42)(trk, z)
30      trk.odc_vert = max_outliers
31      trk.toa_vert = toa
32    end
33  else
34    if (trk.updates_vert == 1)
35      dz = z - trk.mu_vert[1]
36      trk.mu_vert[2] = dz/dt
37      (mu_s, Sigma_s) = PredictVerticalTracker(p.45)(trk.mu_vert, trk.Sigma_vert, dt)
38    end
39    if (q_int == 100)
40      R_q = R_100
41    else
42      R_q = R_25
43    end
44    (trk.mu_vert, trk.Sigma_vert) = UpdateVerticalTracker(p.45)(z, q_int, mu_s, Sigma_s, R_q)
45    trk.odc_vert = max_outliers
46    trk.updates_vert += 1
47    if (trk.updates_vert >= detections_to_recover)
48      trk.valid_vert = true
49    end
50    trk.toa_vert = toa
51  VerticalRateArrowUpdate(p.108)(trk)
52 end

```

Referenced In: ReceiveStateVectorPositionReport(p.49), AdvanceModeSTrackFile(p.17), UpdateModeCTrack(p.24)

The vertical tracker uses the standard Kalman filter prediction step (PREDICTVERTICALTRACKER (Algorithm 38) where the state transition matrix (*F*) and the process noise coupling matrix (*G*) are based on the duration of time that has passed since the last update to the track (*dt*). This prediction, however, is not performed if the difference between the TOA and the time of the last track update is less than a defined threshold (*min_extrap_toa_step*).

This algorithm takes as input μ , Σ , and dt . This algorithm returns the results of the call to PREDICTKALMANFILTER.

Algorithm 38 PredictVerticalTracker

```

1 function PredictVerticalTracker( $\mu$ ::Vector{R},  $\Sigma$ ::Matrix{R},  $dt$ ::R)
2   const Q::R = params().surveillance.intruder_vertical.Q
3   const min_extrap_toa_step::R = params().surveillance.min_extrap_toa_step
4   if ( $dt < \text{min\_extrap\_toa\_step}$ )
5     return ( $\mu$ ::Vector{R},  $\Sigma$ ::Matrix{R})
6   else
7     F = [1.0  $dt$ ; 0.0 1.0]
8     G = [0.5* $dt^2$ ;  $dt$ ]
9     return PredictKalmanFilter(p. 69)( $\mu$ ,  $\Sigma$ , F, G, Q)
10  end
11 end
```

Referenced In: ExtrapolateADSBTrack(p. C-3), BaroAltAtToa(p. 78), EvaluateOnGroundModeC(p. B-17), AddADSB-TrackToReport(p. 110), AddModeSTrackToReport(p. 102), ExtrapolateActiveTrack(p. C-2), GenerateStmReport(p. 97), AdvanceVerticalTrack(p. 44), AddModeCTrackToReport(p. 114), ActiveValidationUpdate(p. 121)

The vertical tracker departs from the standard Kalman filter paradigm in its update step, UPDATERVERTICALTRACKER (Algorithm 39). First, the observation covariance matrix (S) is obtained to calculate the Kalman gain (K) in the standard manner. Then instead of using the standard linearly calculated predicted observation and its covariance, the vertical tracker utilizes PREDICTEDALTITUDE (Algorithm 40) to obtain these quantities to calculate the updated state (μ) and covariance (Σ).

This algorithm takes as input z_int , q_int , μ_s , Σ_s , and R_k . This algorithm returns μ , Σ .

Algorithm 39 UpdateVerticalTracker

```

1 function UpdateVerticalTracker( $z\_int$ ::R,  $q\_int$ ::UInt32,  $\mu_s$ ::Vector{R},  $\Sigma_s$ ::Matrix{R},  $R_k$ ::Union(R,
  Matrix{R}))
2   const H::Matrix{R} = [1.0 0.0]
3   S = H* $\Sigma_s$ *H' .+  $R_k$ 
4   K =  $\Sigma_s$ *H'*inv(S)
5   ( $\mu_o$ ,  $\Sigma_o$ ) = PredictedAltitude(p. 46)( $z\_int$ ,  $\mu_s$ , S[1],  $q\_int$ )
6    $\mu$  =  $\mu_s$  + K*( $\mu_o$  .- H* $\mu_s$ )
7    $\Sigma$  = (eye(2) - K*H)* $\Sigma_s$ *(eye(2) - K*H)' + K*R_k*K' + K* $\Sigma_o$ *K'
8   return ( $\mu$ ::Vector{R},  $\Sigma$ ::Matrix{R})
9 end
```

Referenced In: ReceiveBaroAltObservation(p. 76), AdvanceVerticalTrack(p. 44)

PREDICTEDALTITUDE (Algorithm 40) is the observation prediction step for the intruder vertical tracker. It diverges from the normal Kalman filter framework to handle the quantized nature of reported altitude measurements. The algorithm first initializes a set of partitions (g) of length (L). Next, the denominator for a subsequent step is calculated (d). If the value of d is not a very small value (i.e., nearly zero defined by *stability*), the probability that the observation exists for each of the partition is computed (vector p). This probability density is then used to determine the observation prediction (μ) and the observation prediction covariance (Σ). Otherwise, if d is close to zero

and division by this factor is numerically unstable, the observation prediction (*mu*) is assigned a value that is plus or minus one half of a quantization level from the predicted state. In this case, the observation prediction covariance (*Sigma*) is set to zero. For more information, please see [7].

This algorithm takes as input *z*, *mu_s*, *S*, and *q*. This algorithm returns *mu*, *Sigma*.

Algorithm 40 PredictedAltitude

```

1 function PredictedAltitude(z::R, mu_s::Vector{R}, S::R, q::UInt32)
2   const stability_exponent::Z = params().surveillance.intruder_vertical.stability_exponent
3   const L::Z = params().surveillance.intruder_vertical.L
4   stability::R = 10.0^stability_exponent
5   g = zeros(L)
6   for i in 0:L-1
7     g[i+1] = (1/(2*L))*(((2*i+1)*(z+q/2))+((2*L-2*i-1)*(z-q/2)))
8   end
9   d::R = 0.0
10  for k in 1:L
11    d += exp(-0.5 * (g[k] - mu_s[1]) * inv(S) * (g[k] - mu_s[1]))
12  end
13  p = zeros(L)
14  if (d > stability)
15    for i = 1:L
16      p[i] = (1/d)*exp(-0.5*(g[i]-mu_s[1])*inv(S)*(g[i]-mu_s[1]))
17    end
18    mu::R = 0.0
19    for k in 1:L
20      mu += (p[k] * (g[k] - mu_s[1]))
21    end
22    Sigma::R = 0.0
23    for k in 1:L
24      Sigma += (p[k] * (g[k] - mu_s[1])^2)
25    end
26    Sigma = Sigma - mu^2
27    mu = mu + mu_s[1]
28  else
29    if (abs(q*round(mu_s[1]/q) - (z-q/2)) < abs(q*round(mu_s[1]/q) - (z+q/2)))
30      mu = z - q/2
31    else
32      mu = z + q/2
33    end
34    Sigma = 0.0
35  end
36  return (mu::R, Sigma::R)
37 end
```

Referenced In: UpdateVerticalTracker(p.45)

2.3 Passive Surveillance

This section defines the algorithms to process surveillance data that has been passively received from an ADS-B Report Generator described in [13]. Several major functions performed by these algorithms include:

1. The association of an ADS-B or ADS-R Report to an existing Track File.
2. The initialization of a new ADSBTRACKFILE.
3. The update of an existing ADSBTRACKFILE.
4. The correlation of an ADSBTRACKFILE to another existing Track File.

It is important to note that ADS-R Track Files are treated as ADS-B Track Files with the *rebroadcast*

flag set to TRUE. The algorithms in this section do not include the functions that select, prepare and form the intruder information that is provided to the TRM and other external modules. That process is defined in Section 2.6, specifically ADDADSBTRACKTOREPORT (Algorithm 110).

2.3.1 Input Processing

There are three entry points to the STM for passive surveillance. Each entry point is associated with the reception of a specific ADS-B Report type as generated by an ADS-B Report Generator. As such, each of the entry points require a different set of input parameters as indicated in Table 2-5, Table 2-6 and Table 2-7.

2.3.1.1 State Vector Position Report

RECEIVESTATEVECTORPOSITIONREPORT (Algorithm 41) requires the inputs defined in Table 2-5. Upon the reception of an ADS-B State Vector Position Report, the STM first checks if the ownship altitude track has been established (i.e., the track is not *Nan*), the ownship WGS84 state has been established (i.e., not a *Nan*), that this state information has been updated recently (as determined by *own_wgs84_timeout*), and that the reported latitude and longitude are valid (i.e., not a *Nan*). Next, the predicted location of the ownship is estimated with PROPAGATEOWNSHIPTOTOA (Algorithm 53) to account for ownship motion in the time between the target observation and the most recent ownship WGS84 state input. The position information contained within the report (*lat*, *lon* and *alt*) is then transformed to local East, North, Up (ENU) coordinates, as it is this relative Cartesian coordinate frame that is used for state estimation. The transformation to ENU coordinates is performed with CONVERTOBSERVEDTARGETTOENU (Algorithm 44). Next, the reported Mode S address (*mode_s*), and whether the message came from ADS-R (*rebroadcast*) are used to associate the report to an existing Target (ASSOCIATEADSBREPORTTOTARGET (Algorithm 42)).

If the State Vector Position Report is not associated with an existing Target, a new ADSBTRACKFILE is created and initialized through a call to INITIALIZEADSBTRACKFILE (Algorithm 43). Next, the new ADSBTRACKFILE is stored in memory through ADDADSBTRACKTODB (Algorithm 51) or ADDADSRTTRACKTODB (Algorithm 52) depending on the reception of an ADS-R message. These algorithms attempt to correlate the new ADSBTRACKFILE to an existing Target (ASSOCIATEICAOTOTARGET (Algorithm 4)). If correlation is successful, the ADSBTRACKFILE is added to the existing Target. Otherwise, the ADSBTRACKFILE is added to a new Target. The horizontal track is defined relative to the predicted location of the ownship, and this ownship location is stored as additional data elements in the ADSBTRACKFILE.

If the call to ASSOCIATEADSBREPORTTOTARGET successfully associates to a Target, the corresponding ADSBTRACKFILE is retrieved from memory. If a minimum allowed time (*min_obs_toa_step*) has not passed since the last track update, the ADSBTRACKFILE is not updated with the State Vector Position Report. If the minimum time threshold has passed, temporary copies of the previous horizontal track are created. The relative horizontal track is then redefined in a rotated coordinate frame based on the predicted latitude and longitude of the ownship with REDEFINEESTIMATEINROTATEDFRAME (Algorithm 54). Next, the two tracks in the ADSBTRACKFILE are progressed through calls to ADVANCEADSBTRACKPOSITION and ADVANCEVERTICALTRACK. If the horizontal track was updated inside ADVANCEADSBTRACKPOSITION, the ownship location is stored as additional data elements in the ADSBTRACKFILE; otherwise, the ownship location is not stored and the horizontal track is reset to the previous horizontal track values.

If either a new Track File was created or an existing Track File was updated, the reported Mode S Address (*mode_s*), whether the reported address is a non-standard ICAO address (*non_icao*), the integrity of the reported data (*NIC*), the quantization of the reported altitude (*q_int*), and the time of

applicability (*toa*) are stored in the ADSBTRACKFILE.

This algorithm takes as input the variables in the following table. This algorithm updates *trk*.

Table 2-5. STM Input Variables - State Vector Position Report

Variable	Units	Type	Description
lat	deg	real	Intruder latitude
lon	deg	real	Intruder longitude
alt	feet	real	Intruder barometric altitude
mode_s	N/A	uint32	Intruder Mode S address
nic	N/A	uint32	Intruder NIC
q_int	feet	uint32	Intruder altitude quantization (25 or 100)
rebroadcast	N/A	bool	Flag to indicate message received from ADS-R
non_icao	N/A	bool	Flag to indicate that the address is a non-standard ICAO address (anonymous address)
toa	s	real	Time of applicability

Algorithm 41 ReceiveStateVectorPositionReport

```

1 function ReceiveStateVectorPositionReport(lat::R, lon::R, alt::R, mode_s::UInt32, nic::UInt32, q_int::UInt32,
2   rebroadcast::Bool, non_icao::Bool, toa::R)
3   const min_obs_toa_step::R = params().surveillance.min_obs_toa_step
4   const own_wgs84_timeout::R = params().surveillance.own_wgs84_timeout
5   if (!isnan(own.toa_h)) && (!isnan(lat)) && (!isnan(lon)) && (!isnan(own.wgs84_toa)) && (toa - own.
6     wgs84_toa <= own_wgs84_timeout)
7   update_track_info::Bool = false
8   update_ownership_refs::Bool = false
9   (olat, olon, oalt, nothing, nothing, Xo, Yo, Zo) = PropagateOwnershipToToa(p.56)(toa)
10  (x_rel, y_rel, nothing) = ConvertObservedTargetToENU(p.51)(lat, lon, alt, olat, olon, oalt, Xo, Yo, Zo
11    )
12  id = AssociateADSBReportToTarget(p.50)(mode_s, rebroadcast)
13  if (isnan(id))
14    trk = ADSBTrackFile(p.E-4)()
15    InitializeADSBTrackFile(p.51)(trk, x_rel, y_rel, alt, mode_s, non_icao, rebroadcast, toa)
16    if (!rebroadcast)
17      AddADSBTrackToDB(p.55)(trk)
18    else
19      AddADSRTTrackToDB(p.55)(trk)
20    end
21    update_track_info = true
22    update_ownership_refs = true
23  else
24    tgt = RetrieveWithID(p.20)(target_db, id)
25    if (!rebroadcast)
26      trk = tgt.adsb_track
27    else
28      trk = tgt.adsr_track
29    end
30    dt = toa - trk.toa
31    if (dt >= min_obs_toa_step)
32      mu_hor_temp::Vector{R} = copy(trk.mu_hor)
33      Sigma_hor_temp::Matrix{R} = copy(trk.Sigma_hor)
34      (trk.mu_hor, trk.Sigma_hor) = RedefineEstimateInRotatedFrame(p.58)(trk, olat, olon)
35      AdvanceADSBTrackPosition(p.67)(trk, x_rel, y_rel, toa)
36      AdvanceVerticalTrack(p.44)(trk, alt, q_int, toa)
37      update_track_info = true
38      update_ownership_refs = (trk.toa_hor == toa)
39      if (!update_ownership_refs)
40        trk.mu_hor = mu_hor_temp
41        trk.Sigma_hor = Sigma_hor_temp
42      end
43    end
44    if (update_track_info)
45      trk.nic = nic
46      trk.quant = q_int
47      trk.toa = toa
48    end
49    if (update_ownership_refs)
50      trk.olat_at_hor_toa = olat
51      trk.olon_at_hor_toa = olon
52      trk.oalt_at_hor_toa = oalt
53      trk.Xo_at_hor_toa = Xo
54      trk.Yo_at_hor_toa = Yo
55      trk.Zo_at_hor_toa = Zo
56    end
57  end
58 end

```

ASSOCIATEADSBREPORTTOTARGET (Algorithm 42) attempts to match a provided Mode S address to an existing ADS-B or ADS-R Track File. If a matching Track File is found, the Target ID is

returned; otherwise, a *NaN* is returned. The algorithm takes as inputs a 24-bit address (*mode_s*) and a flag that states whether the source message is an ADS-R rebroadcast (*rebroadcast*). The algorithm outputs the ID of the Target with a matching address.

Algorithm 42 AssociateADSBReportToTarget

```

1 function AssociateADSBReportToTarget(mode_s::UInt32, rebroadcast::Bool)
2   for id in keys(target_db)
3     if (rebroadcast == false) && (TrackExists(p.18)(target_db[id].adsb_track))
4       if (target_db[id].adsb_track.modes == mode_s)
5         return id::UInt32
6       end
7     elseif (rebroadcast == true) && (TrackExists(p.18)(target_db[id].adsr_track))
8       if (target_db[id].adsr_track.modes == mode_s)
9         return id::UInt32
10      end
11    end
12  end
13 return Nan
14 end
```

Referenced In: *ReceiveStateVectorPositionReport*(*p.49*), *ReceiveStateVectorVelocityReport*(*p.62*), *ReceiveModeStatus-Report*(*p.64*)

INITIALIZEADSBTRACKFILE (Algorithm 43) ensures that the horizontal and vertical tracks are initialized with INITIALIZEADSBTRACKERPOSITION (Algorithm 62) and INITIALIZEVERTICALTRACKER (Algorithm 36) respectively. Additional data elements (some of which are not available with an ADS-B Position Report and must be initialized from the parameters file) are stored in the ADSBTRACKFILE. These elements include the accuracy and integrity of the reported position and velocity (*init_NACp*, *init_NACv*, and *init_SIL*), and the time of applicability for the horizontal and vertical tracks, as well as the time of last position report reception (*toa*).

This algorithm takes as input *trk*, *x_rel*, *y_rel*, *alt*, *mode_s*, *non_icao*, and *toa*. This algorithm updates *trk*.

Algorithm 43 InitializeADSBTrackFile

```

1 function InitializeADSBTrackFile(trk::ADSBTrackFile(p. E-4), x_rel::R, y_rel::R, alt::R, mode_s::UInt32, non_icao::Bool, rebroadcast::Bool, toa::R)
2   const init_NACP::UInt32 = params().surveillance.horizontal_adsb.init_nacp
3   const init_NACV::UInt32 = params().surveillance.horizontal_adsb.init_nacv
4   const init_SIL::UInt32 = params().surveillance.horizontal_adsb.init_sil
5   const mhd::Z = params().surveillance.horizontal_adsb.max_outlier_detections
6   const mvod::Z = params().surveillance.intruder_vertical.max_outlier_detections
7   (trk.mu_hor, trk.Sigma_hor) = InitializeADSBTrackerPosition(p.66)(x_rel, y_rel, init_NACV)
8   trk.odc_hor = mhd
9   trk.updates_pos = 1
10  InitializeVerticalTracker(p.42)(trk, alt)
11  trk.odc_vert = mvod
12  trk.valid_vert = (trk.updates_vert > 0)
13  trk.nacp = init_NACP
14  trk.nacv = init_NACV
15  trk.sil = init_SIL
16  trk.toa_hor = toa
17  trk.toa_vert = toa
18  trk.toa_pos_update = toa
19  trk.toa = toa
20  trk.modes = mode_s
21  trk.non_icao = non_icao
22  trk.rebroadcast = rebroadcast
23 end

```

Referenced In: ReceiveStateVectorPositionReport(p.49)

CONVERTOBSERVEDTARGETTOENU (Algorithm 44) transforms an intruder's reported ADS-B position to the local East, North, Up (ENU) coordinate frame of the predicted ownship position. During conversion, if no observed target altitude is available, the target is assumed to be coaltitude.

This algorithm takes as input *tlat*, *tlon*, *talt*, *olat*, *olon*, *oalt*, *Xo*, *Yo*, and *Zo*. This algorithm returns *ew_rel*, *ns_rel*, and *ud_rel*.

Algorithm 44 ConvertObservedTargetToENU

```

1 function ConvertObservedTargetToENU(tlat::R, tlon::R, talt::R, olat::R, olon::R, oalt::R, Xo::R, Yo::R, Zo::R)
    )
2   const min_obs_toa_step::R = params().surveillance.min_obs_toa_step
3   tlat = deg2rad(tlat)
4   tlon = deg2rad(tlon)
5   if (isnan(talt))
6     (Xt, Yt, Zt) = ConvertWGS84ToECEF(p.52)(tlat, tlon, oalt)
7   else
8     (Xt, Yt, Zt) = ConvertWGS84ToECEF(p.52)(tlat, tlon, talt)
9   end
10  DeltaX = Xt - Xo
11  DeltaY = Yt - Yo
12  DeltaZ = Zt - Zo
13  (ew_rel, ns_rel, ud_rel) = RotateECEFtoENU(p.59)(DeltaX, DeltaY, DeltaZ, olat, olon)
14  return (ew_rel::R, ns_rel::R, ud_rel::R)
15 end

```

Referenced In: ReceiveStateVectorPositionReport(p.49)

CONVERTWGS84TOECEF (Algorithm 45) converts a WGS84 position into ECEF coordinates. The calibrated height above mean sea level (MSL) from a barometric sensor (i.e. orthometric height

above MSL) is first converted to WGS84 height above ellipsoid (HAE) to avoid skewing calculations for X/Y positions.

This algorithm takes as input *lat*, *lon*, and *h*. This algorithm returns *X*, *Y*, and *Z*.

Algorithm 45 ConvertWGS84ToECEF

```

1 function ConvertWGS84ToECEF(lat::R, lon::R, h::R)
2   const a::R      = geoutils.earth_semiMajor_axis
3   const e1_sq::R = geoutils.earth_first_eccentricity_sq
4   const m2ft::R = geoutils.meters_to_feet
5   h /= m2ft
6   h = ConvertOrthometricToGeodeticHeight(p. 53)(h, lat)
7   N = a / sqrt(1 - (e1_sq * (sin(lat)^2)))
8   X = (N + h) * cos(lat) * cos(lon)
9   Y = (N + h) * cos(lat) * sin(lon)
10  Z = (N * (1 - e1_sq) + h) * sin(lat)
11  return (X::R, Y::R, Z::R)
12 end
```

Referenced In: [RefineENUHeight\(p. 59\)](#), [PropagateOwnershipToToa\(p. 56\)](#), [ConvertObservedTargetToENU\(p. 51\)](#)

CONVERTECEFToWGS84 (Algorithm 46) converts an ECEF position into WGS84 coordinates. At extreme polar regions of the earth, the variable *inner_square_root* may become less than zero. If the variable *inner_square_root* is greater than or equal to zero, the WGS84 coordinates are calculated in the usual way. If the variable *inner_square_root* is less than or equal to zero, a simplified calculation of the latitude and altitude is performed. If the ECEF position has some offset from the axis of rotation, then the longitude is found in the usual way; otherwise, the longitude is set to zero. The altitude was originally reported as height above mean sea level (MSL) from a barometric sensor (ie orthometric height above MSL), therefore the WGS84 height above ellipsoid (HAE) is transformed back to this coordinate system.

This algorithm takes as input *Xa*, *Ya*, and *Za*. This algorithm returns *lat*, *lon*, and *h*.

Algorithm 46 ConvertECEFToWGS84

```

1 function ConvertECEFToWGS84(Xa::R, Ya::R, Za::R)
2   const e1_sq::R = geoutils.earth_first_eccentricity_sq
3   const e2_sq::R = geoutils.earth_second_eccentricity_sq
4   const m2ft::R = geoutils.meters_to_feet
5   const a::R = geoutils.earth_semmajor_axis
6   const b::R = geoutils.earth_seminor_axis
7   r_ecef = hypot(Xa, Ya)
8   E_sq = a*a - b*b
9   F = 54*(b*b)*(Za*Za)
10  G = r_ecef*r_ecef + (1-e1_sq)*(Za*Za) - e1_sq*E_sq
11  C = (e1_sq*e1_sq)*F*(r_ecef*r_ecef)/(G*G*G)
12  S = (1 + C + sqr(C*C+2*C))^(1.0/3.0)
13  P = F/(3*((S+(1/S)+1)^2)*(G*G))
14  Q = sqr(1+2*(e1_sq*e1_sq)*P)
15  inner_square_root = ((a*a)/2)*(1+(1/Q)) - (P*(1-e1_sq)*(Za*Za))/(Q*(1+Q)) - P*(r_ecef*r_ecef)/2
16  if (inner_square_root >= 0)
17    r_θ = -(P*e1_sq*r_ecef)/(1+Q) + sqr(inner_square_root)
18    U = hypot(r_ecef - e1_sq*r_θ, Za)
19    V = sqr((r_ecef-e1_sq*r_θ)^2 + (1-e1_sq)*(Za*Za))
20    Z_θ = ((b*b)*Za)/(a*V)
21    lat = atan2(Za+e2_sq*Z_θ, r_ecef)
22    lon = atan2(Ya, Xa)
23    h = U*(1 - (b*b)/(a*V))
24  else
25    lat = atan2(Za, r_ecef)
26    if(r_ecef > 0)
27      lon = atan2(Ya, Xa)
28    else
29      lon = 0.0
30    end
31    h = abs(Za) - b
32  end
33  h = ConvertGeodeticToOrthometricHeight(p. 54)(h, lat)
34  h *= m2ft
35  return (lat::R, lon::R, h::R)
36 end

```

Referenced In: RefineENUHeight_(p. 59), PropagateOwnershipToToa_(p. 56), PropagateIntruderToToa_(p. 63)

CONVERTORTHOMETRICTOGEODETICHEIGHT (Algorithm 47) converts an orthometric altitude to a geodetic height based on the provided latitude. The algorithm takes as inputs the orthometric altitude observation of an aircraft (h_o), and the latitude coordinate of an aircraft (ϕ). The algorithm outputs the geodetic height of an aircraft (geo_h).

Algorithm 47 ConvertOrthometricToGeodeticHeight

```

1 function ConvertOrthometricToGeodeticHeight(h_o::R, phi::R)
2   const gamma_s_45::R = geoutils.earth_gravity_lat45
3   R_phi::R      = GetEarthRadiusAtLatitude(p. 54)(phi)
4   gamma_s_phi::R = GetNormalGravityAtLatitude(p. 54)(phi)
5   geo_h::R = R_phi * h_o / ((R_phi * gamma_s_phi / gamma_s_45) - h_o)
6   return geo_h::R
7 end

```

Referenced In: ConvertWGS84ToECEF_(p. 52)

CONVERTGEODETICTOORTHOMETRICHEIGHT (Algorithm 48) converts a geodetic height to an

orthometric altitude based on the provided latitude. The algorithm takes as inputs the geodetic height of an aircraft (h_g), and the latitude coordinate of an aircraft (ϕ). The algorithm outputs the orthometric altitude of an aircraft ($ortho_h$).

Algorithm 48 ConvertGeodeticToOrthometricHeight

```

1 function ConvertGeodeticToOrthometricHeight( $h_g$ ::R,  $\phi$ ::R)
2   const gamma_s_45::R = geoutils.earth_gravity_lat45
3   R_phi::R = GetEarthRadiusAtLatitude(p.54)( $\phi$ )
4   gamma_s_phi::R = GetNormalGravityAtLatitude(p.54)( $\phi$ )
5   ortho_h::R = R_phi * (gamma_s_phi/gamma_s_45) / (R_phi/h_g + 1)
6   return ortho_h::R
7 end
```

Referenced In: ConvertECEFToWGS84(p.53)
--

GETEARTH_RADIUS_AT_LATITUDE (Algorithm 49) uses the provided latitude to calculate and return the radius of the earth at a given latitude, considering the oblate spheroid nature of the earth. The algorithm takes as input the latitude of an aircraft (ϕ). This algorithm outputs the radius of the earth at the provided latitude (r).

Algorithm 49 GetEarthRadiusAtLatitude

```

1 function GetEarthRadiusAtLatitude( $\phi$ ::R)
2   const a::R = geoutils.earth_semiMajor_axis
3   const f::R = geoutils.earth_flattening
4   const m_r::R = geoutils.gravity_ratio
5   r = a / (1 + f + m_r - (2*f*(sin( $\phi$ )^2)))
6   return r::R
7 end
```

Referenced In: ConvertOrthometricToGeodeticHeight(p.53), ConvertGeodeticToOrthometricHeight(p.54)
--

Gravity increases as an object moves away from the equator towards a pole. GETNORMALGRAVITYATLATITUDE (Algorithm 50) calculates the force of gravity at the provided latitude and returns the value. The algorithm takes as input the latitude of an aircraft (ϕ). This algorithm outputs the value of gravity at that given latitude (g).

Algorithm 50 GetNormalGravityAtLatitude

```

1 function GetNormalGravityAtLatitude( $\phi$ ::R)
2   const e1_sq::R = geoutils.earth_first_eccentricity_sq
3   const k_s::R = geoutils.somiglianas_constant
4   const gamma_e::R = geoutils.earth_equatorial_gravity
5   g = gamma_e * (1+(k_s*(sin( $\phi$ )^2))) / sqrt(1 - (e1_sq*(sin( $\phi$ )^2)))
6   return g::R
7 end
```

Referenced In: ConvertOrthometricToGeodeticHeight(p.53), ConvertGeodeticToOrthometricHeight(p.54)
--

ADDADSATTRACKTODB (Algorithm 51) takes an intruder's ADS-B track and adds it to the in-

truder database. Since the database can only have one intruder per id, the algorithm first checks to see if the intruder exists within the database before adding the track. There are no outputs of the algorithm.

Algorithm 51 AddADSBTrackToDB

```

1 function AddADSBTrackToDB(trk::ADSBTrackFile(p. E-4))
2   id = AssociateICAOtoTarget(p. 18)(trk.modes)
3   if (isnan(id))
4     tgt = Target(p. E-14)(trk.modes)
5     tgt.init_time = trk.toa
6     tgt.adsb_track = trk
7     AddToDB(p. 20)(target_db, tgt)
8   else
9     target_db[id].adsb_track = trk
10  end
11 end
```

Referenced In: ReceiveStateVectorPositionReport(<i>p. 49</i>)
--

ADDADSRTTRACKTODB (Algorithm 52) takes an intruder's ADS-R track and adds it to the intruder database. Since the database can only have one intruder per id, the algorithm first checks to see if the intruder exists within the database before adding the track. There are no outputs of the algorithm.

Algorithm 52 AddADSRTTrackToDB

```

1 function AddADSRTTrackToDB(trk::ADSBTrackFile(p. E-4))
2   id = AssociateICAOtoTarget(p. 18)(trk.modes)
3   if (isnan(id))
4     tgt = Target(p. E-14)(trk.modes)
5     tgt.init_time = trk.toa
6     tgt.adsr_track = trk
7     AddToDB(p. 20)(target_db, tgt)
8   else
9     target_db[id].adsr_track = trk
10  end
11 end
```

Referenced In: ReceiveStateVectorPositionReport(<i>p. 49</i>)
--

PROPAGATEOWNSHIPTOTOA (Algorithm 53) estimates the predicted location of the ownship at a specific *toa* to account for ownship motion in the time between the target observation and the most recent ownship WGS84 state input. During propagation, the most recent WGS84 states are converted to ECEF coordinates using CONVERTWGS84TOECEF (Algorithm 45) and ROTATEENU-TOECEF (Algorithm 57), followed by a linear propagation by *dt*. The propagated ECEF position of the ownship is used to calculate a predicted latitude and longitude with CONVERTECEFTOWGS84 (Algorithm 46). These predicted latitude and longitude values are then combined with a separately predicted altitude from BAROALTATTOA (Algorithm 77) to create a more accurate estimate of the ECEF position. Additionally, the predicted latitude and longitude are used to convert the ECEF velocities into ENU coordinates with ROTATEECEFTOENU (Algorithm 56).

This algorithm takes as input *toa*. This algorithm returns *lat_toa*, *lon_toa*, *h_toa*, *vel_ew_toa*, *vel_-*

ns_toa, X_toa, Y_toa, and Z_toa.

Algorithm 53 PropagateOwnshipToToa

```

1 function PropagateOwnshipToToa(toa::R)
2   const kts2mps::R = geoutils.kts_to_mps
3   const m2ft::R = geoutils.meters_to_feet
4   lat = deg2rad(own.wgs84_lat)
5   lon = deg2rad(own.wgs84_lon)
6   h = BaroAltAtToa(p. 78)(own.wgs84_toa)
7   (X, Y, Z) = ConvertWGS84ToECEF(p. 52)(lat, lon, h)
8   vel_ew = own.wgs84_vel_ew * kts2mps
9   vel_ns = own.wgs84_vel_ns * kts2mps
10  vel_ud = own.mu_h[2] / m2ft
11  (vel_X, vel_Y, vel_Z) = RotateENUToECEF(p. 60)(vel_ew, vel_ns, vel_ud, lat, lon)
12  dt = toa - own.wgs84_toa
13  X_toa = X + vel_X*dt
14  Y_toa = Y + vel_Y*dt
15  Z_toa = Z + vel_Z*dt
16  (lat_toa, lon_toa, nothing) = ConvertECEFToWGS84(p. 53)(X_toa, Y_toa, Z_toa)
17  h_toa = BaroAltAtToa(p. 78)(toa)
18  (X_toa, Y_toa, Z_toa) = ConvertWGS84ToECEF(p. 52)(lat_toa, lon_toa, h_toa)
19  (vel_ew_toa, vel_ns_toa, nothing) = RotateECEFToENU(p. 59)(vel_X, vel_Y, vel_Z, lat_toa, lon_toa)
20  return (lat_toa::R, lon_toa::R, h_toa::R, vel_ew_toa::R, vel_ns_toa::R, X_toa::R, Y_toa::R, Z_toa::R)
21 end

```

Referenced In: ReceiveStateVectorPositionReport(*p. 49*), ExtrapolateADSBTrack(*p. C-3*), AddADSBTrackToReport(*p. 110*), ReceiveStateVectorVelocityReport(*p. 62*), ActiveValidationUpdate(*p. 121*)

REDEFINEESTIMATEINROTATEDFRAME (Algorithm 54) redefines the relative horizontal track in a rotated coordinate frame based on the predicted latitude and longitude of the ownship. During redefinition, variables that are defined in association with the predicted location of the ownship are indicated by *_1*, while variables that are defined in association with the location of the ownship at the time the horizontal track was last updated are indicated by *_0*. The predicted latitude and longitude of the ownship are passed into the function as inputs, while the previous latitude and longitude of the ownship are acquired from previously saved ADS-B track data elements.

If the vertical states are valid, the vertical states are predicted to the same time as the horizontal states; otherwise, the intruder is assumed to be coaltitude with the ownship. The relative altitude and relative altitude-rate are then calculated and used as an initial guess for the relative Up/Down position and relative Up/Down rate. This guess is potentially inaccurate because the Earth is spherical and so coaltitude aircraft are not necessarily coplanar. A more accurate relative Up/Down position is estimated in REFINEENUHEIGHT. The improved relative Up/Down position is combined with the original relative horizontal positions to calculate the final relative ECEF position. This final relative ECEF position is then rotated into an ENU coordinate frame that gets its orientation from the predicted latitude and longitude of the ownship. The relative velocities undergo similar rotations to change the relative ENU velocities into the revised ENU coordinate frame. The new relative horizontal components (defined in the rotated ENU coordinate frame) are then combined into a new estimate of the mean states (*mu_hor_1*).

For the second part of redefinition, the covariance matrix is also defined in the new rotated coordinate frame. If the vertical states are not valid, the vertical covariance matrix is set to zero to prevent *NaN* calculations in subsequent steps. The horizontal and vertical covariance matrices are then combined as block matrices along the diagonal of a six-by-six matrix, with zeros everywhere else. This six-by-six matrix is then split up into the four submatrices determined from the position-

position, position-velocity, velocity-position, and velocity-velocity terms. Two rotation matrices are combined, which results in a rotation from ENU at $_0$ to the new ENU at $_1$. This combined rotation matrix and its transpose are pre and post multiplied, respectively, to the four submatrices. The horizontal components from the resulting submatrices are combined into a new covariance matrix (Σ_{hor_l}).

This algorithm takes as input trk , $olat_l$, and $olon_l$. It returns mu_hor_l and Σ_{hor_l} .



Algorithm 54 RedefineEstimateInRotatedFrame

```

1 function RedefineEstimateInRotatedFrame(trk::ADSBTrackFile(p.E-4), olat_1::R, olon_1::R)
2   const m2ft::R = geoutils.meters_to_feet
3   olat_0 = trk.olat_at_hor_toa
4   olon_0 = trkOLON_at_hor_toa
5   ew_rel_0 = trk.mu_hor[1]
6   vel_ew_rel_0 = trk.mu_hor[2]
7   ns_rel_0 = trk.mu_hor[3]
8   vel_ns_rel_0 = trk.mu_hor[4]
9   if(trk.valid_vert)
10    dt_hor_vert = trk.toa_hor - trk.toa_vert
11    talt = trk.mu_vert[1] + trk.mu_vert[2]*dt_hor_vert
12    talt_rate = trk.mu_vert[2]
13  else
14    talt = trk.oalt_at_hor_toa
15    talt_rate = own.mu_h[2]
16  end
17  ud_rel_0 = (talt - trk.oalt_at_hor_toa) / m2ft
18  vel_ud_rel_0 = (talt_rate - own.mu_h[2]) / m2ft
19  ud_rel_0 = RefineENUHeight(p.59)(trk, ew_rel_0, ns_rel_0, ud_rel_0, olat_0, olon_0, talt)
20  (DeltaX, DeltaY, DeltaZ) = RotateENUToECEF(p.60)(ew_rel_0, ns_rel_0, ud_rel_0, olat_0, olon_0)
21  (ew_rel_1, ns_rel_1, nothing) = RotateECEFToENU(p.59)(DeltaX, DeltaY, DeltaZ, olat_1, olon_1)
22  (vel_DeltaX, vel_DeltaY, vel_DeltaZ) = RotateENUToECEF(p.60)(vel_ew_rel_0, vel_ns_rel_0, vel_ud_rel_0, olat_0, olon_0)
23  (vel_ew_rel_1, vel_ns_rel_1, nothing) = RotateECEFToENU(p.59)(vel_DeltaX, vel_DeltaY, vel_DeltaZ, olat_1, olon_1)
24  Sigma_hor_1::Matrix{R} = [ew_rel_1, vel_ew_rel_1, ns_rel_1, vel_ns_rel_1]
25  Sigma_hor_0::Matrix{R} = trk.Sigma_hor
26  Sigma_vert_0::Matrix{R} = trk.Sigma_vert / (m2ft*m2ft)
27  if(trk.valid_vert == false)
28    Sigma_vert_0 = zeros(2,2)
29  end
30  Sigma_0::Matrix{R} = block_diag(p.H-2)(Sigma_hor_0, Sigma_vert_0)
31  const I_xyz::Vector{Z} = [1, 3, 5]
32  const I_dxdydz::Vector{Z} = [2, 4, 6]
33  Sigma_PP_0::Matrix{R} = Sigma_0[I_xy, I_xy]
34  Sigma_PV_0::Matrix{R} = Sigma_0[I_xy, I_dxdydz]
35  Sigma_VP_0::Matrix{R} = Sigma_0[I_dxdydz, I_xy]
36  Sigma_VV_0::Matrix{R} = Sigma_0[I_dxdydz, I_dxdydz]
37  s_phi = sin(olat_0)
38  c_phi = cos(olat_0)
39  s_lambda = sin(olon_0)
40  c_lambda = cos(olon_0)
41  R_0::Matrix{R} = [-s_lambda -s_phi*c_lambda c_phi*c_lambda; c_lambda -s_phi*s_lambda c_phi*s_lambda; 0
      c_phi s_phi]
42  s_phi = sin(olat_1)
43  c_phi = cos(olat_1)
44  s_lambda = sin(olon_1)
45  c_lambda = cos(olon_1)
46  R_1::Matrix{R} = [-s_lambda c_lambda 0; -s_phi*c_lambda -s_phi*s_lambda c_phi; c_phi*c_lambda c_phi*s_lambda
      s_lambda s_phi]
47  R_0_to_1::Matrix{R} = R_1 * R_0
48  Sigma_PP_1::Matrix{R} = R_0_to_1 * Sigma_PP_0 * R_0_to_1'
49  Sigma_PV_1::Matrix{R} = R_0_to_1 * Sigma_PV_0 * R_0_to_1'
50  Sigma_VP_1::Matrix{R} = R_0_to_1 * Sigma_VP_0 * R_0_to_1'
51  Sigma_VV_1::Matrix{R} = R_0_to_1 * Sigma_VV_0 * R_0_to_1'
52  Sigma_1::Matrix{R} = [Sigma_PP_1 Sigma_PV_1; Sigma_VP_1 Sigma_VV_1]
53  const I_xdxdydy::Vector{Z} = [1, 4, 2, 5]
54  Sigma_hor_1::Matrix{R} = Sigma_1[I_xdxdydy, I_xdxdydy]
55  return (mu_hor_1::Vector{R}, Sigma_hor_1::Matrix{Rend

```

Referenced In: ReceiveStateVectorPositionReport(*p.49*), ExtrapolateADSBTrack(*p.C-3*), AddADSBTrackToReport(*p.110*), ReceiveStateVectorVelocityReport(*p.62*), ActiveValidationUpdate(*p.121*)

REFINEENUHEIGHT (Algorithm 55) refines the guess for the relative Up/Down position. The first few steps convert the relative East/West, North/South, and potentially inaccurate Up/Down position of the intruder to an absolute ECEF position, which is needed to find a latitude and longitude. The next steps combine the latitude and longitude with the absolute altitude to calculate an ECEF position that more accurately captures the Up/Down position of the intruder, and ultimately produces the refined guess for the relative Up/Down position.

This algorithm takes as input *trk*, *ew_rel_0*, *ns_rel_0*, *ud_rel_0*, *olat_0*, *olon_0*, and *talt*. This algorithm returns *refined_ud_rel_0*.

Algorithm 55 RefineENUHeight

```

1 function RefineENUHeight(trk::ADSBTrackFile(p. E-4), ew_rel_0::R, ns_rel_0::R, ud_rel_0::R, olat_0::R, olon_0::R
    , talt::R)
2     (DeltaX, DeltaY, DeltaZ) = RotateECEFToECEF(p. 60)(ew_rel_0, ns_rel_0, ud_rel_0, olat_0, olon_0)
3     Xt = DeltaX + trk.Xo_at_hor_toa
4     Yt = DeltaY + trk.Yo_at_hor_toa
5     Zt = DeltaZ + trk.Zo_at_hor_toa
6     (tlat, tlon, nothing) = ConvertECEFToWGS84(p. 53)(Xt, Yt, Zt)
7     (Xt, Yt, Zt) = ConvertWGS84ToECEF(p. 52)(tlat, tlon, talt)
8     DeltaX = Xt - trk.Xo_at_hor_toa
9     DeltaY = Yt - trk.Yo_at_hor_toa
10    DeltaZ = Zt - trk.Zo_at_hor_toa
11    (nothing, nothing, refined_ud_rel_0) = RotateECEFToENU(p. 59)(DeltaX, DeltaY, DeltaZ, olat_0, olon_0)
12    return refined_ud_rel_0::R
13 end
```

Referenced In: RedefineEstimateInRotatedFrame(p. 58)

ROTATEECEFToENU (Algorithm 56) redefines a position from ECEF to ENU by applying a three dimensional rotation based on the provided latitude (*phi*) and longitude (*lambda*).

This algorithm takes as input *X*, *Y*, *Z*, *phi*, and *lambda*. This algorithm returns *ew*, *ns*, and *ud*.

Algorithm 56 RotateECEFToENU

```

1 function RotateECEFToENU(X::R, Y::R, Z::R, phi::R, lambda::R)
2     s_phi = sin(phi)
3     c_phi = cos(phi)
4     s_lambda = sin(lambda)
5     c_lambda = cos(lambda)
6     ew = (-s_lambda*X) + (c_lambda*Y)
7     ns = (-s_phi*c_lambda*X) - (s_phi*s_lambda*Y) + (c_phi*Z)
8     ud = (c_phi*c_lambda*X) + (c_phi*s_lambda*Y) + (s_phi*Z)
9     return (ew::R, ns::R, ud::R)
10 end
```

Referenced In: RedefineEstimateInRotatedFrame(p. 58), RefineENUHeight(p. 59), PropagateOwnershipToToa(p. 56), ReceiveStateVectorVelocityReport(p. 62), ConvertObservedTargetToENU(p. 51)

ROTATEENUToECEF (Algorithm 57) redefines a position from ENU to ECEF by applying a three dimensional rotation based on the provided latitude (*phi*) and longitude (*lambda*).

This algorithm takes as input *ew*, *ns*, *ud*, *phi*, and *lambda*. This algorithm returns *X*, *Y*, and *Z*.

Algorithm 57 RotateENUToECEF

```

1 function RotateENUToECEF(ew::R, ns::R, ud::R, phi::R, lambda::R)
2   s_phi = sin(phi)
3   c_phi = cos(phi)
4   s_lambda = sin(lambda)
5   c_lambda = cos(lambda)
6   X = (-s_lambda*ew) - (s_phi*c_lambda*ns) + (c_phi*c_lambda*ud)
7   Y = (c_lambda*ew) - (s_phi*s_lambda*ns) + (c_phi*s_lambda*ud)
8   Z = (c_phi*ns) + (s_phi*ud)
9   return (X::R, Y::R, Z::R)
10 end
```

Referenced In: RedefineEstimateInRotatedFrame^(p. 58), RefineENUHeight^(p. 59), PropagateOwnshipToToa^(p. 56), ReceiveStateVectorVelocityReport^(p. 62), PropagateIntruderToToa^(p. 63)

2.3.1.2 State Vector Velocity Report

Another entry point to the STM for passive surveillance is defined by RECEIVESTATEVECTORVELOCITYREPORT (Algorithm 58) with inputs defined in Table 2-6. The velocity information included in the report is ground velocity of the broadcasting aircraft (in North/South and East/West components). When a State Vector Velocity Report is received by the STM, the algorithm first ensures that the ownship WGS84 state has been established (i.e., not *NaN*), that this state information has been updated recently (as determined by *own_wgs84_timeout*), and that the reported velocity components are valid (i.e., not a *NaN*). Next, the reported Mode S Address and *rebroadcast* flag are used to associate the report to a Target (ASSOCIATEADSBREPORTTOTARGET (Algorithm 42)). If the association attempt is not successful, the algorithm ends. This implies that the STM will only initialize a new ADSBTRACKFILE using a State Vector Position Report.

If the association attempt is successful, the corresponding ADSBTRACKFILE is retrieved from memory. Next, the predicted location and velocity of the ownship is estimated with PROPAGATEOWNSHIPTOTOA (Algorithm 53) to account for ownship motion in the time between the target observation and the most recent ownship WGS84 state input. The incoming velocity measurements are defined relative to an unknown origin; therefore, an estimated origin is determined through PROPAGATEINTRUDERTOTOA (Algorithm 59). The horizontal velocity measurements are converted into appropriate units. If the vertical track is valid, the Up/Down velocity is estimated as the vertical tracker's velocity; otherwise, it is set to zero. These velocities are rotated out of their ENU coordinate frame (using the estimated latitude and longitude values of the intruder) and into the ECEF coordinate frame. The velocities (now in ECEF) are rotated into the same ENU coordinate frame as the predicted ownship velocities, which results in the calculation of relative velocity measurements in a common coordinate frame. Temporary copies of the previous horizontal track are created. The relative horizontal track is then redefined in a rotated coordinate frame based on the predicted latitude and longitude of the ownship. The flag *trk_advance_allowed* is set to TRUE, allowing ADVANCEADSBTRACKVELOCITY by default. If no velocity estimates have previously been developed for this track and the ownship is at a latitude in nonpolar regions of the earth, the velocity estimates in the ADS-B track are initialized to those derived from the report (note that the indices 2 and 4 correspond to the velocity components of the passive horizontal state estimate). If the ownship is at a latitude in the polar regions of the earth and the velocity estimates in the ADS-B track have not yet been seeded by two position measurements, the flag *trk_advance_allowed* is set to FALSE. If the flag *trk_advance_allowed* is TRUE, the ADS-B Track is progressed through a call to ADVANCEADSBTRACKVELOCITY. Since the State Vector Velocity Report does not include barometric altitude, the vertical track cannot be updated. The additional data elements available in

the report (*NIC* and *toa*) are stored in the ADSBTRACKFILE. If the horizontal track was updated inside ADVANCEADSBTRACKVELOCITY, the ownship location is stored as additional data elements in the ADSBTRACKFILE to reflect the location reference of the updated track estimate. If the horizontal track was not updated, the horizontal track estimate is reset to its original values prior to redefinition in the rotated coordinate frame.

This algorithm takes as input the variables in the following table. This algorithm updates *trk*.

Table 2-6. STM Input Variables - State Vector Velocity Report

Variable	Units	Type	Description
vel_ew	knot	real	Intruder velocity in east-west direction
vel_ns	knot	real	Intruder velocity in north-south direction
mode_s	N/A	uint32	Intruder Mode S address
nic	N/A	uint32	Intruder NIC
rebroadcast	N/A	bool	Flag to indicate message received from ADS-R
non_icao	N/A	bool	Flag to indicate that the address is a non-standard ICAO address (anonymous address)
toa	s	real	Time of applicability

Algorithm 58 ReceiveStateVectorVelocityReport

```

1 function ReceiveStateVectorVelocityReport(vel_ew::R, vel_ns::R, mode_s::UInt32, nic::UInt32, rebroadcast::Bool,
2   non_icao::Bool, toa::R)
3   const own_wgs84_timeout::R = params().surveillance.own_wgs84_timeout
4   const vel_init_lat_thresh::R = params().surveillance.horizontal_adsb.vel_init_lat_thresh
5   const kts2mps::R = geoutils.kts_to_mps
6   const m2ft::R = geoutils.meters_to_feet
7   if (!isnan(own.wgs84_toa)) && (toa - own.wgs84_toa <= own_wgs84_timeout) && (!isnan(vel_ew)) && (!isnan(vel_ns))
8     id = AssociateADSBReportToTarget(p.50)(mode_s, rebroadcast)
9     if (!isnan(id))
10       tgt = RetrieveWithID(p.20)(target_db, id)
11       if (!rebroadcast)
12         trk = tgt.adsb_track
13       else
14         trk = tgt.adsr_track
15       end
16       (olat, olon, oalt, own_vel_ew, own_vel_ns, Xo, Yo, Zo) = PropagateOwnershipToToa(p.56)(toa)
17       (tlat, tlon) = PropagateIntruderToToa(p.63)(trk, toa, oalt, Xo, Yo, Zo)
18       int_vel_ew = vel_ew * kts2mps
19       int_vel_ns = vel_ns * kts2mps
20       int_vel_ud = 0.0
21       if (trk.valid_vert)
22         int_vel_ud = trk.mu_vert[2] / m2ft
23       end
24       (int_vel_X, int_vel_Y, int_vel_Z) = RotateENUToECEF(p.60)(int_vel_ew, int_vel_ns, int_vel_ud,
25         tlat, tlon)
26       (int_vel_ew, int_vel_ns, nothing) = RotateECEFToENU(p.59)(int_vel_X, int_vel_Y, int_vel_Z, olat,
27         olon)
28       dx_rel = int_vel_ew - own_vel_ew
29       dy_rel = int_vel_ns - own_vel_ns
30       (mu_hor_temp::Vector{R}, Sigma_hor_temp::Matrix{R}) = (copy(trk.mu_hor), copy(trk.Sigma_hor))
31       (trk.mu_hor, trk.Sigma_hor) = RedefineEstimateInRotatedFrame(p.58)(trk, olat, olon)
32       trk_advance_allowed::Bool = true
33       trk_updated::Bool = false
34       if (!trk.init_velocity)
35         vel_init_enabled::Bool = (abs(olat) < deg2rad(vel_init_lat_thresh))
36         vel_init_seeded::Bool = (trk.updates_pos > 1)
37         if (vel_init_enabled)
38           trk.mu_hor[2] = dx_rel
39           trk.mu_hor[4] = dy_rel
40           trk.init_velocity = true
41         elseif (vel_init_seeded)
42           trk.init_velocity = true
43         else
44           trk_advance_allowed = false
45         end
46       end
47       if (trk_advance_allowed)
48         AdvanceADSBTrackVelocity(p.68)(trk, dx_rel, dy_rel, toa)
49         (trk.non_icao, trk.nic, trk.toa) = (non_icao, nic, toa)
50         trk_updated = (trk.toa_hor == toa)
51         if (trk_updated)
52           (trk.olat_at_hor_toa, trk.olon_at_hor_toa, trk.oalt_at_hor_toa) = olat, olon, oalt
53           (trk.Xo_at_hor_toa, trk.Yo_at_hor_toa, trk.Zo_at_hor_toa) = (Xo, Yo, Zo)
54         end
55       end
56     end
57   end
58 end

```

PROPAGATEINTRUDERTOTOA (Algorithm 59) estimates the predicted location of the intruder at a specific *toa* to account for intruder motion in the time between the target observation and the most recent track estimate. During intruder propagation, the relative horizontal track is propagated forward to the desired time of applicability in its existing ENU coordinate frame. The vertical track is similarly propagated forward to the desired time of applicability; however, if the vertical track is not valid the altitude is assumed to be coaltitude with the ownship. The propagated altitude is combined with the predicted ownship altitude to form an estimate of the relative Up/Down position. The combined horizontal and vertical predicted position is rotated out of the existing ENU coordinate frame (using latitude and longitude values it is defined with) and into the ECEF coordinate frame. The predicted relative position (now in ECEF) is combined with the predicted location of the ownship to get the absolute location of the intruder, and ultimately the predicted latitude and longitude of the intruder.

This algorithm takes as input *trk*, *toa*, *oalt*, *Xo*, *Yo*, and *Zo*. This algorithm returns *lat_toa* and *lon_toa*.

Algorithm 59 PropagateIntruderToToa

```

1 function PropagateIntruderToToa(trk::ADSBTrackFile(p. E-4), toa::R, oalt::R, Xo::R, Yo::R, Zo::R)
2   const m2ft::R = geoutils.meters_to_feet
3   dt_hor = toa - trk.toa_hor
4   ew_rel = trk.mu_hor[1] + trk.mu_hor[2]*dt_hor
5   ns_rel = trk.mu_hor[3] + trk.mu_hor[4]*dt_hor
6   if(trk.valid_vert)
7     dt_vert = toa - trk.toa_vert
8     talt = trk.mu_vert[1] + trk.mu_vert[2]*dt_vert
9   else
10    talt = oalt
11  end
12  ud_rel = (talt - oalt) / m2ft
13  (DeltaX, DeltaY, DeltaZ) = RotateENUToECEF(p. 60)(ew_rel, ns_rel, ud_rel, trk.olat_at_hor_toa, trk.
14    olon_at_hor_toa)
15  Xt = DeltaX + Xo
16  Yt = DeltaY + Yo
17  Zt = DeltaZ + Zo
18  (lat_toa, lon_toa, nothing) = ConvertECEFToWGS84(p. 53)(Xt, Yt, Zt)
19 return (lat_toa::R, lon_toa::R)
19 end
```

Referenced In: ReceiveStateVectorVelocityReport^(p. 62)

2.3.1.3 Mode Status Report

The third entry point to the STM for passive surveillance is given by RECEIVEMODESTATUSREPORT (Algorithm 60) with inputs defined in Table 2-7. The Mode S address and the *rebroadcast* flag are used to associate the received report with an existing Target. If the call to ASSOCIATEADSBREPORTTOTARGET (Algorithm 42) is unsuccessful, the algorithm will not create a new ADS-BTRACKFILE, the report is discarded, and the algorithm ends. If association is successful, the corresponding ADSBTRACKFILE is retrieved from memory. This ADSBTRACKFILE is then updated with the relevant elements of the Mode Status Report including whether the reported address is a non-standard ICAO address (*non_icao*), the ADS-B version (*adsb_version*), the accuracy of the reported position and velocity data (*nacp* and *nacv*) a measure of data integrity (*sil*), and system design assurance (*sda*).

In the final step of RECEIVEMODESTATUSREPORT, the target's ADS-B data quality history is

updated through a call to UPDATEADSBQUALITYHISTORY (Algorithm 61). The purpose of this algorithm is to track the data quality reported by an intruder so that an M of N check can be performed when the STMREPORT is generated. This is done by first loading the minimum allowable ADS-B quality values from the parameter file. Two quality histories are maintained, one for ADS-B tracks correlated with Mode S surveillance (*tgt.adsb_qual_history*), and one for passive-only ADS-B tracks (*tgt.passive_only_adsb_qual_history*). If the length of either history exceeds the maximum length (*N*), the oldest element is removed from that history. Next, the current reported ADS-B quality values are compared against the minimum values from the parameter file for both normal and passive-only quality requirements. The result of this comparison is stored in the respective history data structure. Preserving the most recent *N* quality values supports a later confirmation in the ADS-BQUALITYCHECK algorithm that at least *M* of those saved values meet the desired threshold. This is referred to as an "*M* of *N*" check throughout this document.

RECEIVEMODESTATUSREPORT takes as input the variables in the following table. This algorithm updates *trk* and *tgt*.

Table 2-7. STM Input Variables - Mode Status Report

Variable	Units	Type	Description
adsb_version	N/A	uint32	Intruder ADS-B version
nacp	N/A	uint32	Intruder NACp
nacv	N/A	uint32	Intruder NACv
sil	N/A	uint32	Intruder SIL
sda	N/A	uint32	Intruder SDA
mode_s	N/A	uint32	Intruder Mode S address
rebroadcast	N/A	bool	Flag to indicate message received from ADS-R
non_icao	N/A	bool	Flag to indicate that the address is a non-standard ICAO address (anonymous address)

Algorithm 60 ReceiveModeStatusReport

```

1 function ReceiveModeStatusReport(adsb_version::UInt32, nacp::UInt32, nacv::UInt32, sil::UInt32, sda::UInt32,
2     mode_s::UInt32, rebroadcast::Bool, non_icao::Bool)
3     id = AssociateADSBReportToTarget(p.50)(mode_s, rebroadcast)
4     if (!isnan(id))
5         tgt = RetrieveWithID(p.20)(target_db, id)
6         if (!rebroadcast)
7             trk = tgt.adsb_track
8         else
9             trk = tgt.adsr_track
10        end
11        trk.non_icao = non_icao
12        trk.adsb_version = adsb_version
13        trk.nacp = nacp
14        trk.nacv = nacv
15        trk.sil = sil
16        trk.sda = sda
17        UpdateAdsbQualityHistory(p.65)(tgt, trk)
18    end
19 end

```

Algorithm 61 UpdateAdsbQualityHistory

```

1 function UpdateAdsbQualityHistory(tgt::Target(p. E-14),trk::ADSBTrackFile(p. E-4))
2   const min_nic::UInt32 = params().surveillance.report_generation.min_adsb_quality.nic
3   const min_nacp::UInt32 = params().surveillance.report_generation.min_adsb_quality.nacp
4   const min_nacv::UInt32 = params().surveillance.report_generation.min_adsb_quality.nacv
5   const min_sil::UInt32 = params().surveillance.report_generation.min_adsb_quality.sil
6   const min_sda_passive_only::UInt32 = params().surveillance.report_generation.min_adsb_quality.passive_-
      only.sda
7   const min_adsb_version::UInt32 = params().surveillance.report_generation.min_adsb_quality.adsb_version
8   const N::UInt32 = params().surveillance.report_generation.min_adsb_quality.N
9   if (length(tgt.adsb_qual_history) >= N)
10    pop!(<i>tgt.adsb_qual_history</i>)
11   end
12   if (length(tgt.passive_only_adsb_qual_history) >= N)
13    pop!(<i>tgt.passive_only_adsb_qual_history</i>)
14   end
15   adsb_quality_good::Bool = false
16   passive_only_quality_good::Bool = false
17   if (trk.nic >= min_nic) && (trk.nacp >= min_nacp) && (trk.nacv >= min_nacv) && (trk.sil >= min_sil) && (
        trk.adsb_version >= min_adsb_version)
18    adsb_quality_good = true
19    if (trk.sda >= min_sda_passive_only)
20     passive_only_quality_good = true
21    end
22   end
23   unshift!(<i>tgt.adsb_qual_history</i>,adsb_quality_good)
24   unshift!(<i>tgt.passive_only_adsb_qual_history</i>,passive_only_quality_good)
25 end

```

Referenced In: ReceiveModeStatusReport(p. 64)

2.3.2 Intruder Horizontal Estimation

The estimation of a target’s horizontal position and velocity using ADS-B reports is performed with a Kalman filter based on the tracking algorithm described in Appendix C of DO-317B [15]. Modifications have been made to this algorithm to make it more suitable for ACAS X. For the purposes of this document, this modified version of the DO-317 algorithm is referred to as the ADS-B Tracker.

In one notable departure from the DO-317 tracking algorithms, the ADS-B Tracker in this document does not estimate the vertical position and velocity of a target aircraft as prescribed in DO-317. The estimation of a target’s vertical state for use by the TRM is done with the same modified Kalman filter as used for active surveillance. The vertical state estimate from the modified Kalman filter is used in conjunction with the latitude and longitude data in a State Vector Position Report to convert observations to ENU coordinates.

The ADS-B Tracker is initialized with INITIALIZEADSBTRACKERPOSITION (Algorithm 62). The algorithm takes as input the relative position of the target aircraft (*x_rel* and *y_rel*) as well as initial accuracy measures of the horizontal velocity measurements (*nacv*). The algorithm sets the initial position distribution to a Gaussian centered on the provided relative position with initial standard deviation (*init_sigma_pos*) read from the parameters file. The initial velocity distribution is considered zero mean Gaussian with a standard deviation derived from the parameter *nacv* (using CONVERT-NACVTOSIGMAHVA). A zero initial velocity is assumed in the absence of an ADS-B velocity report. This velocity estimate will be seeded with either the values received in the next ADS-B velocity report or velocities derived from the next ADS-B position report. A small offset value (*kappa*) is added to the diagonal terms of the covariance matrix *Sigma* to ensure a positive-semidefinite ma-

trix during later Cholesky factorization. The initial mean (*mu*) and covariance (*Sigma*) are returned by the algorithm.

The algorithm takes as input *x_rel*, *y_rel*, and *nacv*. This algorithm returns *mu* and *Sigma*.

Algorithm 62 InitializeADSBTrackerPosition

```

1 function InitializeADSBTrackerPosition(x_rel::R, y_rel::R, nacv::UInt32)
2   const init_sigma_pos::R = params().surveillance.horizontal_adsb.init_sigma_pos
3   const kappa::R = params().surveillance.horizontal_adsb.kappa
4   mu = [x_rel, 0, y_rel, 0]
5   Sigma_tmp = zeros(4,2)
6   Sigma_tmp[1,1] = Sigma_tmp[3,2] = init_sigma_pos
7   Sigma_tmp[2,1] = Sigma_tmp[4,2] = ConvertNACvToSigmaHVA(p. 66)(nacv)
8   Sigma = Sigma_tmp * Sigma_tmp' 
9   Sigma = Sigma + kappa*eye(4)
10  return (mu::Vector{R}, Sigma::Matrix{R})
11 end
```

Referenced In: InitializeADSBTrackFile_(p. 51), AdvanceADSBTrackPosition_(p. 67)

CONVERTNACvTOSIGMAHVA (Algorithm 63) converts an input *NACv* to a velocity uncertainty in a single horizontal dimension. *va* is assigned to the 95 percent radial uncertainty as defined in DO-242A [12]. In order to convert this value into a single standard deviation in one dimension, this value is then divided by 2.448.

This algorithm takes as input *NACv*. This algorithm returns *sigma_hva*.

Algorithm 63 ConvertNACvToSigmaHVA

```

1 function ConvertNACvToSigmaHVA(NACv::UInt32)
2   va = 0
3   if (NACv == 0) || (NACv == 1)
4     va = 10
5   elseif (NACv == 2)
6     va = 3
7   elseif (NACv == 3)
8     va = 1
9   elseif (NACv == 4)
10    va = 0.3
11  end
12  sigma_hva = va/2.448
13  return sigma_hva::R
14 end
```

Referenced In: InitializeADSBTrackerPosition_(p. 66), UpdateADSBTrackerVelocity_(p. 70)



Following initialization, the ADS-B Tracker is advanced based on the ADS-B report type through a call to either ADVANCEADSBTRACKPOSITION (Algorithm 64) or ADVANCEADSBTRACKVELOCITY (Algorithm 65). These algorithms check the validity of the observation data and perform outlier detection to determine how the ADS-B Track should be updated. In both ADVANCEADSBTRACKPOSITION and ADVANCEADSBTRACKVELOCITY, the existing ADS-B Track is first extrapolated to the time of the observation through PREDICTADSBTRACKER. If any of the measured quantities are invalid (indicated by evaluating to *NaN*), the ADS-B Track is not updated and will be

coasted. Otherwise if the observation is identified as an outlier (through a call to IsOUTLIER), the track is not updated (i.e., the track is either coasted or reinitialized). Indices specific to the ADS-B horizontal track are used to select the X and Y position (I_{xy}) or the X and Y velocity (I_{dxdy}) from the μ_s and $Sigma_s$ before calling outlier detection. A coast occurs when the outlier detection count ($trk.odc_rng$) permits such an action (i.e., this decrementing counter is greater than zero). Otherwise the track is reinitialized using INITIALIZEADSBTRACKERPOSITION (Algorithm 62). Finally, if the observation is valid and not an outlier, ADVANCEADSBTRACKPOSITION checks whether an ADS-B velocity report has not been received for this track (indicated by $trk.init_velocity$) and that the incoming update is the second measurement received since the last time the track was initialized (indicated by $trk.updates_pos$). If both of these conditions are true, then the track velocity is replaced by taking the difference in position over the difference in time of the current measurement and the previous track position. Lastly, the track is updated with a call to the update step of the ADS-B Tracker (UPDATEADSBTRACKERPOSITION or UPDATEADSBTRACKERVERLOCITY).

The algorithm takes as input trk , x_rel , y_rel , and toa . This algorithm updates trk .

Algorithm 64 AdvanceADSBTrackPosition

```

1 function AdvanceADSBTrackPosition( $trk::ADSBTrackFile(p.E-4)$ ,  $x\_rel::R$ ,  $y\_rel::R$ ,  $toa::R$ )
2   const max_outliers::Z = params().surveillance.horizontal_adsb.max_outlier_detections
3   const outlier_thresh::R = params().surveillance.horizontal_adsb.outlier_threshold
4   const init_NACv::UInt32 = params().surveillance.horizontal_adsb.init_nacv
5   dt = toa - trk.toa_hor
6   const I_xy::Vector{Z} = [1,3]
7   ( $\mu_s$ ,  $Sigma_s$ ) = PredictADSBTracker(p.68)( $trk.\mu\_hor$ ,  $trk.Sigma\_hor$ , dt)
8   if (isnan(x_rel)) || (isnan(y_rel))
9     return
10  elseif (IsOutlier(p.32)([x_rel,y_rel], $\mu_s[I_{xy}]$ , $Sigma_s[I_{xy},I_{xy}]$ ,outlier_thresh))
11    if ( $trk.odc\_hor > 0$ )
12       $trk.odc\_hor -= 1$ 
13    else
14      ( $trk.\mu\_hor$ , $trk.Sigma\_hor$ ) = InitializeADSBTrackerPosition(p.66)( $x\_rel$ ,  $y\_rel$ , init_NACv)
15       $trk.odc\_hor = max\_outliers$ 
16       $trk.init\_velocity = false$ 
17       $trk.toa\_hor = toa$ 
18       $trk.toa\_pos\_update = toa$ 
19       $trk.updates\_pos = 1$ 
20    end
21  else
22    if (! $trk.init\_velocity$ ) && ( $trk.updates\_pos == 1$ )
23       $trk.\mu\_hor[2] = (x\_rel-trk.\mu\_hor[1])/dt$ 
24       $trk.\mu\_hor[4] = (y\_rel-trk.\mu\_hor[3])/dt$ 
25      ( $\mu_s$ , $Sigma_s$ ) = PredictADSBTracker(p.68)( $trk.\mu\_hor$ ,  $trk.Sigma\_hor$ , dt)
26    end
27    ( $trk.\mu\_hor$ , $trk.Sigma\_hor$ ) = UpdateADSBTrackerPosition(p.69)( $x\_rel$ , $y\_rel$ , $\mu_s$ , $Sigma_s$ )
28     $trk.odc\_hor = max\_outliers$ 
29     $trk.updates\_pos += 1$ 
30     $trk.toa\_hor = toa$ 
31     $trk.toa\_pos\_update = toa$ 
32  end
33 end
```

Referenced In: ReceiveStateVectorPositionReport(p.49)
--

Algorithm 65 AdvanceADSBTrackVelocity

```

1 function AdvanceADSBTrackVelocity(trk::ADSBTrackFile(p. E-4), dx_rel::R, dy_rel::R, toa::R)
2   const max_outliers::Z = params().surveillance.horizontal_adsb.max_outlier_detections
3   const outlier_thresh::R = params().surveillance.horizontal_adsb.outlier_threshold
4   dt = toa - trk.toa_hor
5   const I_dx dy::Vector{Z} = [2,4]
6   (mu_s, Sigma_s) = PredictADSBTracker(p. 68)(trk.mu_hor, trk.Sigma_hor, dt)
7   if (isnan(dx_rel)) || (isnan(dy_rel)) || (IsOutlier(p. 32)([dx_rel,dy_rel],mu_s[I_dx dy],Sigma_s[I_dx dy],
      I_dx dy],outlier_thresh))
8     return
9   else
10    (trk.mu_hor, trk.Sigma_hor) = UpdateADSBTrackerVelocity(p. 70)(dx_rel, dy_rel, trk.nacv, mu_s, Sigma_s)
11    trk.toa_hor = toa
12  end
13 end

```

Referenced In: ReceiveStateVectorVelocityReport^(p. 62)

The ADS-B Tracker utilizes the standard Kalman filter prediction step in PREDICTADSBTRACKER (Algorithm 66). The algorithm will return the existing state estimate unmodified if the requested prediction time is so small (as dictated by *min_extrap_toa_step*) as to lead to numerical instabilities. This algorithm first reads the process noise matrix (*Q*) from the parameter file. Next the duration of time that has passed since the last successful update to the track (*dt*) is used to formulate the state transition matrix (*F*) and the process noise coupling matrix (*G*). These matrices, along with the current state (*mu*) and covariance (*Sigma*), are used to call PREDICKALMANFILTER (Algorithm 67), which returns the predicted state and covariance.

The algorithm takes as input *mu*, *Sigma*, and *dt*. This algorithm returns the output of PREDICKALMANFILTER, namely *mu* and *Sigma*.

Algorithm 66 PredictADSBTracker

```

1 function PredictADSBTracker(mu::Vector{R}, Sigma::Matrix{R}, dt::R)
2   const Q::R = params().surveillance.horizontal_adsb.Q
3   const min_extrap_toa_step::R = params().surveillance.min_extrap_toa_step
4   if (dt < min_extrap_toa_step)
5     return (mu::Vector{R}, Sigma::Matrix{R})
6   else
7     F = eye(4)
8     F[1,2] = F[3,4] = dt
9     G = zeros(4,2)
10    G[1,1] = G[3,2] = 0.5 * dt * dt
11    G[2,1] = G[4,2] = dt
12    return PredictKalmanFilter(p. 69)(mu, Sigma, F, G, Q)
13  end
14 end

```

Referenced In: ExtrapolateADSBTrack^(p. C-3), AddADSBTrackToReport^(p. 110), AdvanceADSBTrackPosition^(p. 67), ActiveValidationUpdate^(p. 121), AdvanceADSBTrackVelocity^(p. 68)

Algorithm 67 PredictKalmanFilter

```

1 function PredictKalmanFilter(mu::Array{R}, Sigma::Matrix{R}, F::Matrix{R}, G::Array{R}, Q::R)
2   mu_s = F*mu
3   Sigma_s = (F*Sigma*F') + (G*Q*G')
4   return (mu_s::Array{R}, Sigma_s::Matrix{R})
5 end
```

Referenced In: ReceiveHeadingObservation(p. 81), PredictVerticalTracker(p. 45), HeadingAtToa(p. 85), ReceiveBaroAltObservation(p. 76), PredictADSBTracker(p. 68), PredictCartesianTracker(p. 33)

The ADS-B Tracker also utilizes the standard Kalman filter update step. However two update algorithms are needed to reflect fundamental differences in State Vector Position Reports (UPDATEADSBTRACKERPOSITION (Algorithm 68)) and State Vector Velocity Reports (UPDATEADSBTRACKERVELOCITY (Algorithm 69)). These two update algorithms both use UPDATEKALMANFILTER but require a different observation coupling matrix (H) and observation noise matrix (R_q or R_k). For position, a NACp of 7 was used to calculate the *update_sigma_pos* parameter that fills R_q . NACp of 7 is used even when a higher value is reported because the error model does not account for uncompensated latency, which dominates the position error for higher NACps. Treating all position measurements at NACp of 7 is a way to inflate the uncertainty for higher NACps to help account for uncompensated latency. For velocity, a call to CONVERTNACvTOSIGMAHVA given the incoming NACv is used when initializing R_q . In either case, the call to UPDATEKALMANFILTER returns the updated state and covariance. In a final step, the updated state covariance ($Sigma$) is scaled by a constant factor (*cov_inflation_factor*) to increase the overall uncertainty. The *cov_inflation_factor* is another way to add uncertainty to help account for uncompensated latency, which the error model does not account for.

This algorithm takes as input *x_rel*, *y_rel*, *mu_s*, and *Sigma_s*. This algorithm returns *mu*, and *Sigma*.

Algorithm 68 UpdateADSBTrackerPosition

```

1 function UpdateADSBTrackerPosition(x_rel::R, y_rel::R, mu_s::Vector{R}, Sigma_s::Matrix{R})
2   const cov_inflation_factor::R = params().surveillance.horizontal_adsb.cov_inflation_factor
3   const update_sigma_pos::R = params().surveillance.horizontal_adsb.update_sigma_pos
4   H = zeros(2,4)
5   H[1,1] = H[2,3] = 1
6   R_q = diagm([update_sigma_pos*update_sigma_pos, update_sigma_pos*update_sigma_pos])
7   (mu, Sigma) = UpdateKalmanFilter(p. 41)([x_rel, y_rel], mu_s, Sigma_s, H, R_q)
8   Sigma = cov_inflation_factor*Sigma
9   return (mu::Vector{R}, Sigma::Matrix{R})
10 end
```

Referenced In: AdvanceADSBTrackPosition(p. 67)

Algorithm 69 UpdateADSBTrackerVelocity

```

1 function UpdateADSBTrackerVelocity(dx_rel::R, dy_rel::R, nacv::UInt32, mu_s::Vector{R}, Sigma_s::Matrix{R})
2   const cov_inflation_factor::R = params().surveillance.horizontal_adsb.cov_inflation_factor
3   H = zeros(2,4)
4   H[1,2] = H[2,4] = 1
5   sigma_hva = ConvertNACvToSigmaHVA(p.66)(nacv)
6   R_k = diagm([sigma_hva*sigma_hva, sigma_hva*sigma_hva])
7   (mu, Sigma) = UpdateKalmanFilter(p.41)([dx_rel, dy_rel], mu_s, Sigma_s, H, R_k)
8   Sigma = cov_inflation_factor*Sigma
9   return (mu::Vector{R}, Sigma::Matrix{R})
10 end

```

Referenced In: AdvanceADSBTrackVelocity(p.68)

2.3.3 Intruder Vertical Estimation

The estimation of a target's vertical position and velocity using ADS-B Reports is performed via the same algorithms that are used for active surveillance. Refer to Section 2.2.3 for a description of these algorithms.

2.4 Ownership Tracking

This section defines the algorithms to process data from sensors that measure the state of ownership. The STM is required to estimate several ownership parameters for two primary purposes:

1. To provide the TRM with an estimate of ownership state.
2. To perform the transformation of intruder surveillance data to the coordinate system used for state estimation.

The algorithms in this section do not include the functions that prepare and form the ownership state information that is provided to the TRM and other external modules. That process is defined in Section 2.6.

2.4.1 Input Processing

There are six unique entry points to the STM related to ownership. Four of the entry points handle the inputs for a unique ownership sensor while the other two entry points provide a means to input discrete values related to ownership. Each entry point requires a different set of input parameters as indicated in Table 2-8, Table 2-9, Table 2-10, Table 2-11, Table 2-12, and Table 2-13. The following subsections define the algorithms that process the data provided by these various entry points.

2.4.2 Ownership Discretes

The STM requires knowledge of certain ownership parameters that change infrequently or only upon initialization. Such parameters are indicated in Table 2-8. The algorithm that handles the assignment of these ownership discrete values is RECEIVEDISCRETES (Algorithm 70).

Table 2-8. STM Input Variables - Ownship Discretes

Variable	Units	Type	Description
address	N/A	uint32	Ownship Mode S address
modeACode	N/A	uint32	Ownship Mode A address
opflg	N/A	bool	System operational flag
manualSL	N/A	uint32	Externally set sensitivity level
own_ground_display_mode_on	N/A	bool	Flag to indicate if the display is allowed on the ground
on_surface	N/A	bool	Flag to indicate whether ownship is operating on the surface or taking-off / airborne
aoto_on	N/A	bool	Flag to indicate if ADS-B Only TA-Only is allowed
is_coarsely_quant	N/A	bool	Flag to indicates if ownship altitude is 100 foot quantized

Algorithm 70 ReceiveDiscretes

```

1 function ReceiveDiscretes(address::UInt32, modeACode::UInt32, opflg::Bool, manualSL::UInt32,
2   own_ground_display_mode_on::Bool, on_surface::Bool, aoto_on::Bool, is_coarsely_quant::Bool)
3   own.modes = address
4   own.discrete.mode_s = address
5   own.discrete.modeACode = modeACode
6   own.discrete.opflg = opflg
7   own.discrete.manualSL = manualSL
8   own.discrete.own_ground_display_mode_on = own_ground_display_mode_on
9   own.discrete.on_surface = on_surface
10  own.discrete.aoto_on = aoto_on
11  own.discrete.is_coarsely_quant = is_coarsely_quant
11 end

```

2.4.3 Target Designation

ACAS X allows the flight crew to assign special processing to a target aircraft by designating that Target (e.g. through the display). Such a command enters ACAS X through the STM via RECEIVE-TARGETDESIGNATION (Algorithm 71) with the input parameters defined in Table 2-9. ACAS X target designation support is provided via the Aircraft Surveillance Applications (ASA) System described in DO-317B [15]. The Aircraft Surveillance and Separation Assurance Processing (ASSAP) function in the ASA System provides the interface to the ACAS X system.

RECEIVETARGETDESIGNATION first checks the value of *designation* to make sure the desired designation is an acceptable value. Currently acceptable designation values include:

- Undesignate all pending designations (*DESIGNATION_NONE*)
- Designate to No Alerts (*DESIGNATION_DESIGNATED_NO_ALERTS*)
- Designate to Protection in a Closely Spaced Parallel Operation (CSPO) with 3000 feet of lateral separation (*DESIGNATION_Xo_CSPO3k*)
- Undesignate from No Alerts (*DESIGNATION_UNDESIGNATE_NO_ALERTS*)
- Undesignate to Xa (*DESIGNATION_UNDESIGNATE_PROTECTION_MODE*)

When an acceptable designation is provided, the reported 24-bit, or Mode S, address (*mode_s*) is used to associate the designation to an existing Target using ASSOCIATEICAOTOTARGET. If asso-

ciation is successful, the corresponding Target is retrieved from memory using RETRIEVEWITHID. Only ICAO compliant 24-bit addresses are eligible for designation; the Target is checked to ensure that the ID used for association was ICAO compliant. If the Target is already designated, its designation is updated using SETPENDINGTARGETDESIGNATION. If the Target was not previously designated, undesignation requests are not honored and designation requests are evaluated to see if they can be honored.

A new Target can be designated whenever the limit on the number of targets designated ($N_{targets_limit}$) has not been reached. The list of targets currently designated is obtained using GETDESIGNATEDTARGETIDS. The number of targets currently designated is the length of that list.

Even if the number of targets currently designated is $N_{targets_limit}$, it may still be possible to designate a new target if one of the currently designated targets is no longer valid. All the currently designated targets are examined to see if *was_dropped* is TRUE and there is no active Resolution Advisory (RA) on that Target. If a Target meets those criteria, it can be undesignated using SETTARGETDESIGNATIONINVALID. The new Target can then be designated in its place. The break command terminates the for loop as soon as the new Target is designated.

SETPENDINGTARGETDESIGNATION is used to update the target designation information.

GETDESIGNATEDTARGETIDS (Algorithm 72) returns a list of all targets in the database with field *is_designated* in DESIGNATIONSTATE set to TRUE.

Table 2-9. STM Input Variables - Target Designation

Variable	Units	Type	Description
mode_s	N/A	uint32	Intruder 24-bit address
designation	N/A	uint32	Selected Xo Designation for Intruder

Algorithm 71 ReceiveTargetDesignation

```

1 function ReceiveTargetDesignation( mode_s::UInt32, designation::UInt32 )
2   const N_targets_limit::Z = params().target_designation.N_targets_limit
3   if (designation == DESIGNATION_NONE) ||
4     (designation == DESIGNATION_DESIGNATED_NO_ALERTS) ||
5     (designation == DESIGNATION_Xo_CSP03k) ||
6     (designation == DESIGNATION_UNDESIGNATE_NO_ALERTS) ||
7     (designation == DESIGNATION_UNDESIGNATE_PROTECTION_MODE)
8     id = AssociateICAOtoTarget(p.18)( mode_s )
9     if (!isnan( id ))
10      tgt::Target(p.E-14) = RetrieveWithID(p.20)( target_db, id )
11      if (!TrackExists(p.18)(tgt.adsr_track)) || (!tgt.adsr_track.non_icao) || (tgt.adsr_track.modes != mode_s)
12        if tgt.designation_state.is_designated
13          SetPendingTargetDesignation(p.75)( designation, tgt.designation_state )
14        elseif (designation != DESIGNATION_NONE) &&
15          (designation != DESIGNATION_UNDESIGNATE_NO_ALERTS) &&
16          (designation != DESIGNATION_UNDESIGNATE_PROTECTION_MODE)
17          designated_target_ids::Vector{UInt32} = GetDesignatedTargetIDs(p.73)()
18          if (length( designated_target_ids ) < N_targets_limit)
19            SetPendingTargetDesignation(p.75)( designation, tgt.designation_state )
20          else
21            for tgtid in designated_target_ids
22              designated_tgt::Target(p.E-14) = RetrieveWithID(p.20)( target_db, tgtid )
23              if (!designated_tgt.designation_state.active_ra) &&
24                (designated_tgt.designation_state.was_dropped)
25                  SetTargetDesignationInvalid(p.139)( designated_tgt.designation_state, true )
26                  SetPendingTargetDesignation(p.75)( designation, tgt.designation_state )
27                  break
28            end
29          end
30        end
31      end
32    end
33  end
34 end
35 end

```

Algorithm 72 GetDesignatedTargetIDs

```

1 function GetDesignatedTargetIDs()
2   designated_target_ids::Vector{UInt32} = Array( UInt32, 0 )
3   for id in keys( target_db )
4     if target_db[id].designation_state.is_designated
5       push!( designated_target_ids, id )
6     end
7   end
8   return designated_target_ids::Vector{UInt32}
9 end

```

Referenced In: ReceiveTargetDesignation(p.73), SetTargetDesignationModeAvailability(p.145)

SETPENDINGTARGETDESIGNATION (Algorithm 73) updates the pending fields in DESIGNATION-STATE for the Target. There are separate pending fields for Designated No Alerts (DNA) (*pending_dna*) and for the protection mode (*pending_protection_mode*). Each time SETPENDINGTARGETDESIGNATION is used with a supported value for *designation*, values in one or more of the pending fields are updated.

If full undesignation is requested (*DESIGNATION_NONE*), DNA and protection mode are both undesignated. When possible, the value of *designated_mode* is set to indicate a specific type of undesignation. The protection mode is undesignated to the value given by *S_standard_protection_mode*; that value should always be *PROTECTION_MODE_Xa* for ACAS Xa/Xo.

If partial undesignation is requested, (*DESIGNATION_UNDESIGNATE_PROTECTION_MODE* or *DESIGNATION_UNDESIGNATE_NO_ALERTS*), only the specified Xo mode is undesignated.

If a specific designation is requested, (*DESIGNATION_Xo_CSPO3k* or *DESIGNATION_DESIGNATED_NO_ALERTS*), the pending fields are set to be compatible with the specified designation. Selection of Xo CSPO-3000 changes the protection mode and deactivates DNA, setting *pending_dna* to FALSE. Selection of DNA, sets *pending_dna* to TRUE and reverts the protection mode to Xa.

Whenever a designation or undesignation request is honored, the Target is marked as being designated and the designated mode, used in the ACAS X output, is set to reflect the input. The field *is_designated* indicates designation processing is active on the Target. It is set to FALSE, only after the STM and TRM processing cycle is complete (see **STMHOUSEKEEPINGTARGETDESIGNATION**). Although the field *designated_mode* is set here, it will be reset if the Target is automatically undesignated by the ACAS X system.

The pending fields are one set of inputs to **SETTARGETDESIGNATION** when the **STMREPORT** is being generated. They specify a designation request, honored by **SETTARGETDESIGNATION** when that request is compatible with the Target surveillance and advisory states. The pending fields are reset whenever the Target is automatically undesignated.

This algorithm takes as input *designation* and *designation_state*. This algorithm updates values in one or more of the pending fields.

Algorithm 73 SetPendingTargetDesignation

```

1 function SetPendingTargetDesignation( designation::UInt32, designation_state::DesignationState(p.E-6) )
2   const S_standard_protection_mode::UInt8 = uint8(params().target_designation.S_standard_protection_mode)
3   if (designation == DESIGNATION_NONE)
4     if (S_standard_protection_mode != designation_state.pending_protection_mode)
5       designation_state.designated_mode      = DESIGNATION_UNDESIGNATE_PROTECTION_MODE
6     elseif designation_state.pending_dna
7       designation_state.designated_mode      = DESIGNATION_UNDESIGNATE_NO_ALERTS
8     else
9       designation_state.designated_mode      = DESIGNATION_NONE
10    end
11   designation_state.is_designated        = true
12   designation_state.pending_dna          = false
13   designation_state.pending_protection_mode = S_standard_protection_mode
14 elseif (designation == DESIGNATION_UNDESIGNATE_NO_ALERTS)
15   designation_state.designated_mode      = designation
16   designation_state.is_designated        = true
17   designation_state.pending_dna          = false
18 elseif (designation == DESIGNATION_UNDESIGNATE_PROTECTION_MODE)
19   designation_state.designated_mode      = designation
20   designation_state.is_designated        = true
21   designation_state.pending_protection_mode = S_standard_protection_mode
22 elseif (designation == DESIGNATION_DESIGNATED_NO_ALERTS)
23   designation_state.designated_mode      = designation
24   designation_state.is_designated        = true
25   designation_state.pending_dna          = true
26   designation_state.pending_protection_mode = S_standard_protection_mode
27 elseif (designation == DESIGNATION_Xo_CSP03k)
28   designation_state.designated_mode      = designation
29   designation_state.is_designated        = true
30   designation_state.pending_dna          = false
31   designation_state.pending_protection_mode = PROTECTION_MODE_Xo_CSP03k
32 end
33 end

```

Referenced In: ReceiveTargetDesignation(p.73)

2.4.4 Barometric Altitude Estimation

The estimation of ownship vertical state is performed with a Kalman filter. This algorithm yields an estimate of ownship altitude as well as ownship vertical rate. The analysis performed to properly tune the observation noise matrix (R_k) and the process noise matrix (Q) is provided by [11].

The entry point to the STM that handles sensor measurements from the onboard barometric altimeter is defined by RECEIVEBAROALTOBSERVATION (Algorithm 74) with input parameters given by Table 2-10. Upon the reception of a sensor measurement, the algorithm first checks whether both inputs are *Nan*. This unique combination is treated as a command from the Surveillance front-end to reset the ownship vertical track. Next the algorithm checks if the ownship vertical track has been initialized. If the TOA associated with the ownship vertical track (*own.toa_h*) is *Nan*, the track is initialized with INITIALIZEOWNVERTICALTRACKER and the TOA is updated. This algorithm sets the initial altitude distribution to a Gaussian centered on the measured barometric altitude (*z_int*) with variance (*var_zint*) read in from the parameters file. The initial velocity distribution is assumed to be zero mean Gaussian with variance (*var_dzint*), which is also read in from the parameters file, as it cannot be determine from a single measurements alone. This algorithm returns the initial mean (*mu*) and covariance (*Sigma*).

A call to UPDATEHISTORY will then update the ownship barometric altitude history to support receiving DF0 messages from the past. If the track is already initialized, and a minimum duration

.....

of time has passed since the last track update (*min_obs_toa_step*), the track is predicted forward to the time of the sensor measurement (PREDICTKALMANFILTER). If the observation is determined to be an outlier (by ISOUTLIER), no update is performed. In the final step, the track is updated with UPDATEKALMANFILTER if no consideration of quantization is necessary (as indicated by *is_coarsely_quant*). Otherwise the track is updated with UPDATEVERTICALTRACKER, which takes ownship altitude quantization into account when estimating vertical state. A call to UPDATEHISTORY will then update the ownship barometric altitude history with the current ownship altitude estimate.

This algorithm takes as input the variables in the following table. This algorithm updates the ownship barometric altitude history and the track.

Table 2-10. STM Input Variables - Barometric Altitude Observation

Variable	Units	Type	Description
h	ft	real	Own barometric altitude
toa	s	real	Time of applicability

Algorithm 74 ReceiveBaroAltObservation

```

1 function ReceiveBaroAltObservation(h::R, toa::R)
2   const own_history_window::Z = params().surveillance.own_history_window
3   const R_k::R = params().surveillance.ownship_vertical.R
4   const min_obs_toa_step::R = params().surveillance.min_obs_toa_step
5   const outlier_thresh::R = params().surveillance.ownship_vertical.outlier_threshold
6   const Q::R = params().surveillance.ownship_vertical.Q
7   const H::Matrix{R} = [1.0 0.0]
8   if (isnan(h)) || (isnan(toa))
9     if (isnan(h)) && (isnan(toa))
10    own.toa_h = NaN
11    own.history.baroalt = ValueTime(p.E-16)()
12  end
13  return
14 end
15 dt = toa - own.toa_h
16 if (isnan(own.toa_h))
17   (own.mu_h, own.Sigma_h) = InitializeOwnVerticalTracker(p.77)(h)
18   own.toa_h = toa
19   UpdateHistory(p.77)(own.history.baroalt, own.mu_h[1], toa, own_history_window)
20 elseif (dt >= min_obs_toa_step)
21   F = [1.0 dt; 0.0 1.0]
22   G = [0.5*dt^2; dt]
23   (mu_s, Sigma_s) = PredictKalmanFilter(p.69)(own.mu_h, own.Sigma_h, F, G, Q)
24   if (!IsOutlier(p.32)(h, mu_s[1], Sigma_s[1,1], outlier_thresh))
25     if (own.discrete.is_coarsely_quant)
26       (own.mu_h, own.Sigma_h) = UpdateVerticalTracker(p.45)(h, MODE_C_QUANT, mu_s, Sigma_s, R_k)
27     else
28       (own.mu_h, own.Sigma_h) = UpdateKalmanFilter(p.41)(h, mu_s, Sigma_s, H, R_k)
29     end
30   own.toa_h = toa
31   UpdateHistory(p.77)(own.history.baroalt, own.mu_h[1], toa, own_history_window)
32 end
33 end
34 end

```

UPDATEHISTORY (Algorithm 75) takes a time-value pair (VALUETIME) that should be stored in the history. A specified time window, *own_history_window*, is used to remove any VALUETIME pairs that are outside this window.

This algorithm takes as input *vt*, *value*, *time*, and *window*. This algorithm updates *vt*.

Algorithm 75 UpdateHistory

```

1 function UpdateHistory(vt::ValueTime(p.E-16), value::R, time::R, window::Z)
2   push!(vt.time,time)
3   push!(vt.value,value)
4   for i in reverse(1:length(vt.time))
5     if (vt.time[i] < time - window)
6       shift!(vt.time)
7       shift!(vt.value)
8     end
9   end
10 end
```

Referenced In: ReceiveHeadingObservation(p.81), EvaluateOnGroundModeC(p.B-17), ReceiveBaroAltObservation(p.76)

Initialization of the ownship vertical state is performed with INITIALIZEOWNVERTICALTRACKER (Algorithm 76).

The ownship vertical tracker uses the standard Kalman filter prediction step, which is also used by the intruder vertical tracker. Refer to section Section 2.2.3 for a description of this algorithm.

The ownship vertical tracker utilizes UPDATEVERTICALTRACKER when ownship altitude quantization is taken into consideration. A description of this algorithm is given in Section 2.2.3. When ownship quantization is not considered, the ownship vertical tracker uses the standard Kalman filter update step as defined in UPDATEKALMANFILTER.

This algorithm takes as input the measured barometric altitude (*z_int*) of the ownship. The output of the algorithm is initialized mean (*own.mu*) and covariance (*own.Sigma*) of the ownship vertical state estimate.

Algorithm 76 InitializeOwnVerticalTracker

```

1 function InitializeOwnVerticalTracker(z_int::R)
2   const var_zint::R = params().surveillance.ownship_vertical.var_zint
3   const var_dzint::R = params().surveillance.ownship_vertical.var_dzint
4   mu = [z_int, 0]
5   Sigma = diagm([var_zint, var_dzint])
6   return (mu::Vector{R}, Sigma::Matrix{R})
7 end
```

Referenced In: ReceiveBaroAltObservation(p.76)

BAROALTATTOA (Algorithm 77) provides the estimated ownship barometric altitude for a specified time of applicability (*t*). This algorithm is used throughout the STM where ownship state is needed to translate intruder measurements into a local coordinate frame. BAROALTATTOA can also handle altitude predictions given time *t* from the past. If the time difference (*dt*) between the requested prediction time (*t*) and the time of the last ownship altitude update (*own.toa_h*) is less than zero

and there are at least two entries of ownship altitude history (*own.history.baroalt*), then the returned prediction will use SURROUNDINGPOINTS to get two altitudes from the history that surround *t* and interpolate between those points (*b1* and *b2* given *time1* and *time2*). The derived *rate* is then used to calculate and return the predicted altitude value from the past. If the request is for a future prediction but a minimum time threshold has not been met (*min_extrap_toa_step*), then it will return the current ownship altitude estimate (*own.mu_h[1]*). Otherwise, the ownship altitude will be extrapolated into the future by a call to PREDICTVERTICALTRACKER.

Algorithm 77 BaroAltAtToa

```

1 function BaroAltAtToa(t::R)
2   const min_extrap_toa_step::R = params().surveillance.min_extrap_toa_step
3   dt = t - own.toa_h
4   if (dt < 0) && (length(own.history.baroalt.time) > 1)
5     (time1, time2, b1, b2) = SurroundingPoints(p. 79)(own.history.baroalt.time, own.history.baroalt.value, t)
6     h_rate::R = (b2 - b1) / (time2 - time1)
7     inter_h::R = b1 + (h_rate * (t - time1))
8     return inter_h::R
9   elseif (dt < min_extrap_toa_step)
10    return own.mu_h[1]::R
11  else
12    (mu_s, Sigma_s) = PredictVerticalTracker(p. 45)(own.mu_h, own.Sigma_h, dt)
13    return mu_s[1]::R
14  end
15 end
```

Referenced In: InitializeModeCTrack(*p. 23*), EvaluateOnGroundModeC(*p. B-17*), AddADSBTrackToReport(*p. 110*), IsC-SPO3000ModeUnavailableToRun(*p. 146*), UpdateModeCTrack(*p. 24*), SetDisplayDataPassive(*p. 112*), AddModeSTrack-ToReport(*p. 102*), PropagateOwnshipToToa(*p. 56*), DetermineModeCReplyAssociationScore(*p. B-7*), RangeSquared(*p. B-8*), AddModeCTrackToReport(*p. 114*), ReceiveDF0(*p. 15*), SetDisplayDataActive(*p. 107*)

SURROUNDINGPOINTS (Algorithm 78) returns two points with corresponding times to be interpolated between, given a time of applicability *toa*. If the *toa* is less than the first time stored in *times*, then the surrounding points and times are the first two entries. Otherwise, the array *times* will be searched to find two points that surround *toa*.

Algorithm 78 SurroundingPoints

```

1 function SurroundingPoints(times::Array{R}, points::Array{R}, toa::R)
2   if (toa < times[1])
3     p1 = points[1]
4     p2 = points[2]
5     t1 = times[1]
6     t2 = times[2]
7   else
8     for i in 1:(length(times) - 1)
9       if (times[i] <= toa) && (times[i+1] > toa)
10         p1 = points[i]
11         p2 = points[i+1]
12         t1 = times[i]
13         t2 = times[i+1]
14       end
15     end
16   end
17   return (t1::R, t2::R, p1::R, p2::R)
18 end

```

Referenced In: BaroAltAtToa(p. 78), HeadingAtToa(p. 85)

2.4.5 Radio Altitude Estimation

The entry point to the STM that handles sensor measurements from the onboard radio altimeter is defined by RECEIVERADALT OBSERVATION (Algorithm 79) with input parameters given by Table 2-11. The algorithm performs no processing on the input radio altitude measurement as this value is stored directly in memory. It is assumed that all credibility checking on the radio altitude is performed external to the STM.

Table 2-11. STM Input Variables - Radio Altitude Observation

Variable	Units	Type	Description
h	ft	real	Own radio altitude

Algorithm 79 ReceiveRadAltObservation

```

1 function ReceiveRadAltObservation(h::R)
2   own.radalt = h
3 end

```

2.4.6 Heading Estimation

The estimation of ownship heading is performed with a Kalman filter that is slightly modified to properly handle the subtraction of angles. This algorithm estimates both ownship heading (used for coordinate conversion with active bearing measurements) and heading rate (to extrapolate the heading estimate). The analysis performed to properly tune the observation noise matrix (R_k) and the process noise matrix (Q) is provided by [11].

The entry point to the STM that handles measurements from the on-board heading sensor is defined by RECEIVEHEADINGOBSERVATION (Algorithm 80) with input parameters given by Table 2-12. When the algorithm receives a measurement (Ψ), WRAPTOPI is called to ensure the heading angle is within the acceptable range of $-\pi$ to π .

An initialized and valid track will only be updated if five conditions are all met: the track has not been updated in the past $\min_obs_toa_step$ period of time; the track has previously been updated within the coast limit duration ($\max_heading_coasts$); the incoming measurement degraded mode ($heading_degraded$) and current track heading state agree; the measurement is valid; and the measurement is not an outlier. The outlier count ($heading_odc$) is decremented only when a measurement is identified as an outlier and all other conditions for updating the track are met. The track will be reset if the outlier count or coast limit has been exceeded.

The heading tracker uses the standard Kalman filter prediction step defined by PREDICTKALMANFILTER (Algorithm 67). This algorithm takes as input the current estimated state ($own.\mu_heading$) and covariance ($own.\Sigma_heading$), the state transition matrix (F), the process noise coupling matrix (G), and the process noise matrix (Q). Since the matrices F and G are time dependent, they are generated based on the duration of time that has passed since the previous track update dt .

The ownership heading track will be initialized if the track is marked as invalid via the $own.heading_state$ flag, or if the track needs to be reset because the outlier count or coast limit has been exceeded. The initialization procedures differ depending on whether the measurement is a Nan or not. In either case, the outlier count is reset to the maximum and the variable $own.heading_initialized$ is set to TRUE; $own.heading_initialized$ remains TRUE for the duration of the system's operation.

If the ownership heading track initializes on a valid measurement (not a Nan), then the $own.heading_state$ is marked as normal or degraded according to the input $heading_degraded$ mode and the tracker TOA is set with the TOA of the measurement. Alternatively, if the ownership heading track initializes on a Nan , the $own.heading_state$ is marked as invalid and the tracker TOA is recorded as Nan .

Finally, the incoming heading measurement is stored in memory via UPDATEHISTORY if the measurement was not Nan and not an outlier.

This algorithm takes as input the variables in the following table. This algorithm updates $own.heading$.

Table 2-12. STM Input Variables - Heading Observation

Variable	Units	Type	Description
Ψ	rad	real	Own heading (north is 0, east is $\pi/2$)
toa	s	real	Time of applicability
$heading_degraded$	NA	boolean	Degraded heading mode

Algorithm 80 ReceiveHeadingObservation

```

1 function ReceiveHeadingObservation(Psi::R, toa::R, heading_degraded::Bool)
2   const max_outliers::Z = params().surveillance.ownship_heading.max_outlier_detections
3   const outlier_thresh::R = params().surveillance.ownship_heading.outlier_threshold
4   const Q::R = params().surveillance.ownship_heading.Q
5   const min_obs_toa_step::R = params().surveillance.min_obs_toa_step
6   const max_heading_coasts::Z = params().surveillance.max_heading_coasts
7   const own_history_window::Z = params().surveillance.own_history_window
8   Psi = WrapToPi(p. H-3)(Psi)
9   store::Bool = false
10  reset_track::Bool = false
11  if (own.heading_state != OWN_HEADING_INVALID)
12    dt = toa - own.toa_heading
13    if (dt >= min_obs_toa_step) && (dt <= max_heading_coasts)
14      degraded_mode_change::Bool = false
15      if (heading_degraded) && (own.heading_state != OWN_HEADING_DEGRADED)
16        degraded_mode_change = true
17      elseif (!heading_degraded) && (own.heading_state == OWN_HEADING_DEGRADED)
18        degraded_mode_change = true
19      end
20      if (!degraded_mode_change)
21        if (!isnan(Psi))
22          F = [1.0 dt; 0.0 1.0]
23          G = [0.5*dt^2; dt]
24          (mu_s, Sigma_s) = PredictKalmanFilter(p. 69)(own.mu_heading, own.Sigma_heading, F, G, Q)
25          if (IsOutlier(p. 32)(AngleDifference(p. H-2)(mu_s[1], Psi), 0.0, Sigma_s[1,1], outlier_thresh))
26            own.heading_ocd -= 1
27          else
28            (own.mu_heading, own.Sigma_heading) = UpdateHeadingTracker(p. 84)(Psi, mu_s, Sigma_s)
29            own.heading_ocd = max_outliers
30            own.toa_heading = toa
31            store = true
32          end
33        end
34      end
35    end
36    if (own.heading_ocd < 0) || (dt > max_heading_coasts)
37      reset_track = true
38    end
39  end
40  if (own.heading_state == OWN_HEADING_INVALID) || (reset_track)
41    (own.mu_heading, own.Sigma_heading) = InitializeHeadingTracker(p. 83)(Psi)
42    own.heading_initialized = true
43    own.heading_ocd = max_outliers
44    if (!isnan(Psi))
45      if heading_degraded
46        own.heading_state = OWN_HEADING_DEGRADED
47      else
48        own.heading_state = OWN_HEADING_NOMINAL
49      end
50      own.toa_heading = toa
51      store = true
52    else
53      own.heading_state = OWN_HEADING_INVALID
54      own.toa_heading = NaN
55    end
56  end
57  if (store)
58    UpdateHistory(p. 77)(own.history.heading, own.mu_heading[1], toa, own.history_window)
59  end
60 end

```

Initialization of the heading tracker is performed with INITIALIZEHEADINGTRACKER (Algorithm 81). This algorithm sets the initial heading distribution to a Gaussian centered on a valid measure-

ment (Ψ_i) with the variance given by the observation noise (R_k). If the measurement is invalid (Nan) then distribution is centered on zero. The initial velocity distribution is set to a zero mean Gaussian with variance (var_dhint) read in from the parameters file as it cannot be determined from a single measurement alone.

When the heading tracker transitions from an invalid track to valid track or from an invalid track to a valid track, the Cartesian tracks of all actively tracked targets are rotated by the current heading estimate through a call to ROTATEHORIZONTALFRAME. This rotation signifies the change from estimating an absolute bearing to the intruder to estimating a relative bearing to the intruder or vice versa. When the track transitions to valid, the incoming heading measurement is added to the Cartesian tracks, switching a relative bearing estimate to an absolute bearing estimate. When the track transitions to invalid, the current heading estimate is subtracted from the Cartesian tracks, switching an absolute bearing estimate to a relative bearing estimate.

An invalid (Nan) measurement is processed at the end of the algorithm so that in the event of a transition to an invalid track, the Cartesian tracks are rotated by the previous heading estimate.

This algorithm takes as input Ψ_i . This algorithm returns μ and Σ .

Algorithm 81 InitializeHeadingTracker

```

1 function InitializeHeadingTracker( $\Psi$ :: $\mathbf{R}$ )
2   const var_dhint:: $\mathbf{R}$  = params().surveillance.ownship_heading.var_dhint
3   const  $R_k$ :: $\mathbf{R}$  = params().surveillance.ownship_heading. $R$ 
4   const  $I_{xdydy}$ :: $\mathbf{Vector}\{Z\}$  = [1, 3, 2, 4]
5   transition_from_headingless = ( $\text{isnan}(\Psi)$ ) && (own.heading_state == OWN_HEADING_INVALID)
6   transition_to_headingless = ( $\text{isnan}(\Psi)$ ) && (own.heading_state != OWN_HEADING_INVALID)
7   if (transition_to_headingless) || (transition_from_headingless)
8     for id in keys(target_db)
9       if (TrackExists(p.18)(target_db[id].modes_track))
10         trk = target_db[id].modes_track
11         if (trk.valid_cart)
12           offset =  $\Psi$ 
13           if (transition_to_headingless)
14             offset = -1.0 * HeadingAtToa(p.85)(trk.toa_cart)
15           end
16           ( $\mu_{hor}$ ,  $\Sigma_{hor}$ ) = RotateHorizontalFrame(p.84)(trk.mu_cart[I_xdydy], trk.Sigma_cart[
17             I_xdydy, I_xdydy], offset)
18           trk.mu_cart =  $\mu_{hor}[I_{xdydy}]$ 
19           trk.Sigma_cart =  $\Sigma_{hor}[I_{xdydy}, I_{xdydy}]$ 
20         end
21       end
22       if (TrackExists(p.18)(target_db[id].modec_tracks))
23         for j in 1:length(target_db[id].modec_tracks)
24           trk = target_db[id].modec_tracks[j]
25           if (trk.valid_cart)
26             offset =  $\Psi$ 
27             offset = -1.0 * HeadingAtToa(p.85)(trk.toa_cart)
28           end
29           ( $\mu_{hor}$ ,  $\Sigma_{hor}$ ) = RotateHorizontalFrame(p.84)(trk.mu_cart[I_xdydy], trk.Sigma_cart[
30             I_xdydy, I_xdydy], offset)
31           trk.mu_cart =  $\mu_{hor}[I_{xdydy}]$ 
32           trk.Sigma_cart =  $\Sigma_{hor}[I_{xdydy}, I_{xdydy}]$ 
33         end
34       end
35     end
36   end
37   if (!isnan( $\Psi$ ))
38     mu = [ $\Psi$ , 0.0]
39     Sigma = diagm([ $R_k$ , var_dhint])
40   else
41     mu = [0.0, 0.0]
42     Sigma = diagm([ $R_k$ , var_dhint])
43   end
44   return ( $\mu$ :: $\mathbf{Vector}\{R\}$ , Sigma:: $\mathbf{Matrix}\{R\}$ )
45 end

```

Referenced In: ReceiveHeadingObservation(p.81)

ROTATEHORIZONTALFRAME (Algorithm 82) offsets a horizontal state estimate and covariance matrix by a specified angle ((az_rad)) in the horizontal plane. The ordering of states in the input and output are expected to be $[xdydy]$. The algorithm breaks the input states into their position and velocity components, applying a rotation matrix to each component. The resulting rotated position and velocities are reassembled into an output state vector ((mu_1)). A similar process is applied to the covariance matrix, with a rotation applied to each 2x2 submatrix. The rotated submatrices are reassembled, and the resulting rotated covariance matrix ((Sigma_1)) is ensured to be symmetric prior to algorithm return.

This algorithm takes as input μ_0 , Σ_0 , and az_rad . It returns μ_1 and Σ_1 .

Algorithm 82 RotateHorizontalFrame

```

1 function RotateHorizontalFrame(mu_0::Vector{R}, Sigma_0::Matrix{R}, az_rad::R)
2   const I_xy_0::Vector{Z} = [1, 3]
3   const I_dxdy_0::Vector{Z} = [2, 4]
4   const I_xdxdy_1::Vector{Z} = [1, 3, 2, 4]
5   R0 = [ cos(az_rad)  sin(az_rad);
6         -sin(az_rad)  cos(az_rad); ]
7   mu_p_0 = mu_0[I_xy_0]
8   mu_v_0 = mu_0[I_dxdy_0]
9   mu_p_1 = R0 * mu_p_0
10  mu_v_1 = R0 * mu_v_0
11  mu_pv_1 = [ mu_p_1, mu_v_1]
12  mu_1 = mu_pv_1[I_xdxdy_1]
13  Sigma_PP_0::Matrix{R} = Sigma_0[I_xy_0, I_xy_0]
14  Sigma_PV_0::Matrix{R} = Sigma_0[I_xy_0, I_dxdy_0]
15  Sigma_VP_0::Matrix{R} = Sigma_0[I_dxdy_0, I_xy_0]
16  Sigma_VV_0::Matrix{R} = Sigma_0[I_dxdy_0, I_dxdy_0]
17  Sigma_PP_1::Matrix{R} = R0 * Sigma_PP_0 * R0'
18  Sigma_PV_1::Matrix{R} = R0 * Sigma_PV_0 * R0'
19  Sigma_VP_1::Matrix{R} = R0 * Sigma_VP_0 * R0'
20  Sigma_VV_1::Matrix{R} = R0 * Sigma_VV_0 * R0'
21  Sigma_PPV_1::Matrix{R} = [Sigma_PP_1 Sigma_PV_1; Sigma_VP_1 Sigma_VV_1]
22  Sigma_1 = Sigma_PPV_1[I_xdxdy_1, I_xdxdy_1]
23  Sigma_1 = (Sigma_1 + Sigma_1')/2
24  return (mu_1::Vector{R}, Sigma_1::Matrix{R})
25 end

```

Referenced In: InitializeHeadingTracker(p.83)

The heading tracker is updated using a slightly modified version of the standard Kalman filter update step as defined in UPDATEHEADINGTRACKER (Algorithm 83). The algorithm departs from standard Kalman filter paradigm as the observation residual is determined through a call to ANGLEDIFFERENCE and WRAPTOPI, which ensures the distance between the measurement and observation prediction is calculated properly. Otherwise, this algorithm calculates the observation covariance matrix (S), the Kalman gain matrix (K) and the updated state (μ) and covariance (Σ) using the standard Kalman filter formulas.

This algorithm takes as input Ψ , μ_s , and Σ_s . This algorithm returns μ and Σ .

Algorithm 83 UpdateHeadingTracker

```

1 function UpdateHeadingTracker(Psi::R, mu_s::Union(Vector{R}, Matrix{R}), Sigma_s::Matrix{R})
2   const R_K::R = params().surveillance.ownship_heading.R
3   const H::Matrix{R} = [1.0 0.0]
4   S = H*Sigma_s*H' .+ R_K
5   K = Sigma_s* H'*inv(S)
6   mu = mu_s + K*WrapToPi(p_h-3)(AngleDifference(p_h-2)(mu_s[1],Psi))
7   mu[1] = WrapToPi(p_h-3)(mu[1])
8   Sigma = Sigma_s - K*S*K'
9   return (mu::Matrix{R}, Sigma::Matrix{R})
10 end

```

Referenced In: ReceiveHeadingObservation(p.81)

HEADINGATTOA (Algorithm 84) provides the estimated ownship heading for a specified time of applicability (t). This algorithm is used throughout the STM where ownship state is needed to trans-

late intruder measurements into a local coordinate frame. It will return NaN if the heading track is uninitialized (as denoted in *own.heading_initialized*), meaning there is no heading track to use in the prediction. If the heading track is initialized but invalid (as denoted in *own.heading_state*) HEADINGATTOA will return zero. This algorithm can also handle heading predictions given time *t* from the past. If the time difference (*dt*) between the requested prediction time (*t*) and the time of the last ownship heading update (*own.toa_heading*) is less than zero and there are at least two entries of ownship heading history (*own.history.heading*), then the returned prediction will use SURROUNDINGPOINTS to get two headings from the history that surround *t* and interpolate between those points (*h1* and *h2* given *time1* and *time2*). The derived *rate* is then used to calculate and return the predicted heading value from the past (constraining the value between $-\pi$ and π using WRAPTOPi). If the request is for a future prediction but a minimum time threshold has not been met (*min_extrap_toa_step*), then it will return the current ownship heading estimate (*own.mu_heading[1]*). Otherwise, the ownship heading will be extrapolated into the future by a call to PREDICTKALMANFILTER. The result that is returned is constrained between $-\pi$ and π due to WRAPTOPi.

This algorithm takes as input *t*. This algorithm returns *heading*.

Algorithm 84 HeadingAtToa

```

1 function HeadingAtToa(t::R)
2   const Q::R = params().surveillance.ownship_heading.Q
3   const min_extrap_toa_step::R = params().surveillance.min_extrap_toa_step
4   heading::R = NaN
5   if (own.heading_initialized)
6     if (length(own.history.heading.time) > 1) && (t < maximum(own.history.heading.time))
7       (time1, time2, h1, h2) = SurroundingPoints(p.79)(own.history.heading.time, own.history.heading.
8         value, t)
9       rate::R = AngleDifference(p.H-2)(h1, h2) / (time2 - time1)
10      heading = WrapToPi(p.H-3)(h1 + rate * (t - time1))
11    elseif (own.heading_state != OWN_HEADING_INVALID)
12      dt = t - own.toa_heading
13      if (dt < min_extrap_toa_step)
14        heading = WrapToPi(p.H-3)(own.mu_heading[1])
15      else
16        F = [1.0 dt; 0.0 1.0]
17        G = [0.5*dt^2; dt]
18        (mu_s, Sigma_s) = PredictKalmanFilter(p.69)(own.mu_heading, own.Sigma_heading, F, G, Q)
19        heading = WrapToPi(p.H-3)(mu_s[1])
20      end
21    else
22      heading = 0.0
23    end
24  end
25  return heading
25 end
```

Referenced In: InitializeModeCTrack(p.23), UpdateModeCTrack(p.24), SetDisplayDataPassive(p.112), GenerateStmReport(p.97), DetermineModeCReplyAssociationScore(p.B-7), ReceiveDF0(p.15), SetDisplayDataActive(p.107), InitializeHeadingTracker(p.83)

2.4.7 WGS84 State Estimation

The entry point to the STM that handles sensor measurements from an onboard WGS84 sensor (i.e., onboard GPS system) is defined by RECEIVEWGS84OBSERVATION (Algorithm 85) with input parameters given by Table 2-13. The current version of this algorithm performs no processing on the input WGS84 data, as these values are stored directly in memory. It is assumed that any

aberrations in ownship WGS84 state will cause active validation of an ADSBTRACKFILE to fail, thus preventing erroneous state information from entering the TRM.

This algorithm takes as input the items in the following table, and stores those values in memory.

Table 2-13. STM Input Variables - WGS84 Observation

Variable	Units	Type	Description
lat	deg	real	Own latitude
lon	deg	real	Own longitude
vel_ew	kn	real	Own velocity in east-west direction
vel_ns	kn	real	Own velocity in north-south direction
toa	s	real	Time of applicability

Algorithm 85 ReceiveWgs84Observation

```

1 function ReceiveWgs84Observation(lat::R, lon::R, vel_ew::R, vel_ns::R, toa::R)
2   own.wgs84_lat = lat
3   own.wgs84_lon = lon
4   own.wgs84_vel_ew = vel_ew
5   own.wgs84_vel_ns = vel_ns
6   own.wgs84_toa = toa
7 end

```

2.5 Coordination Processing

2.5.1 Input Processing

This section defines the algorithms to process coordination data that has been received from a target aircraft. The algorithms in this section do not include the functions that provide the received coordination data to the TRM. That process is defined in Section 2.6.

2.5.1.1 Active Coordination

The entry point to the STM for coordination data that has been received from a target aircraft is given by RECEIVEUF16UDS30 (Algorithm 86). This algorithm takes as input the variables described in Table 2-14, which represent the fields from a UF16-UDS30 coordination interrogation that was transmitted by another aircraft.

RECEIVEUF16UDS30 attempts to associate the input Mode S address to an existing Target through ASSOCIATEICAOTOTARGET. If association is unsuccessful, the coordination data is ignored. If association is successful, the corresponding target is retrieved from memory using RETRIEVEWITHID.

The RECEIVEUF16UDS30 algorithm first runs parity bit checking on the incoming vertical sense bits (vsb). The cancel Vertical Resolution Advisory Complement (cvc) and vertical resolution advisory complement (vrc) in the coordination message are used as an index into the parity table. This is achieved by using the bitwise OR between the cvc bit-shifted to the left by 2, and the vrc (all offset by 1 to adhere to Julia's 1-based indexing because the *parity_table* was defined to be 0-based

indexed per DO-185B). If the parity value is not equal to the *vsb* subfield of the coordination message, then the data received was invalid. Invalid data is not processed further, but a flag is set within the Target to indicate the bad reception. An additional check tests whether the *cvc* or *vrc* are 3, or if the *cvc* equals the *vrc* indicating that the data received is invalid (*cvc*=3 and *vrc*=3 are defined as 'Not assigned'. See MOPS definition of TCAS Resolution Message.). If either *cvc* or *vrc* are 3, then the other aircraft has said that it is increasing and decreasing altitude simultaneously, so the data is nonsensical. If *cvc* and *vrc* are equal, then the other aircraft has said that it is sending and canceling a complement simultaneously, which is also nonsensical.

Messages that are validly formed are processed. First, the received coordination data time stamp (*toa*) is saved. Next, a check is performed to see if the received message contains the vertical complement to cancel (i.e., *cvc* does not equal zero), and that the cancel Vertical Resolution Advisory Complement (*vert_intent[cvc]*) matches the previously saved *vrc*. The map (*vert_intent*) is used to map codes in the coordination message into the vertical intent array given by *own.received_vrcs*. This translates the bit enunciation in the message to what is used internally in the array. The recorded *vrc* (*tgt.coord_data.vrc*) is reset and a call to DELETEINTENT (Algorithm 87) will delete the value in the vertical intent array (*own.received_vrcs*), as long as this *vrc* is not currently being received from another threat. Finally, if the received message contains the vertical complement to add (*vrc*), and the previously saved *vrc* doesn't already indicate desired value, the complement value is saved (*tgt.coord_data*), and the indicated complement in the intent array is set (*own.received_vrcs*).

This algorithm takes as input the variables in the following table. This algorithm updates *tgt.coord_data* and *own.received_vrcs*.

Table 2-14. STM Input Variables - UF16UDS30 Message

Variable	Units	Type	Description
mode_s	N/A	Uint32	Intruder Mode S address
cvc	N/A	Uint32	Cancel Vertical Resolution Advisory Complement
vrc	N/A	Uint32	Vertical Resolution Advisory Complement
vsb	N/A	Uint32	Vertical Sense Bits
toa	s	Float64	Time of Applicability

Algorithm 86 ReceiveUF16UDS30

```

1 function ReceiveUF16UDS30(mode_s::UInt32, cvc::UInt32, vrc::UInt32, vsb::UInt32, toa::R)
2   const parity_table::Vector{Z} = params().coordination.parity_table
3   const vert_intent::Vector{Z} = params().coordination.vert_intent
4   id = AssociateICAOtoTarget(p.18)(mode_s)
5   if (!isnan(id))
6     tgt = RetrieveWithID(p.20)(target_db, id)
7     parity_index::Z = (((cvc << 2) | vrc) + 1)
8     parity::Z = parity_table[parity_index]
9     tgt.bad_uf16uds30 = false
10    if (parity != vsb)
11      tgt.bad_uf16uds30 = true
12    elseif (cvc == 3) || (vrc == 3) || (cvc == vrc)
13      tgt.bad_uf16uds30 = true
14    else
15      tgt.coord_data.toa = toa
16      if (cvc != 0) && (vert_intent[cvc] == tgt.coord_data.vrc)
17        tgt.coord_data.vrc = 0
18        DeleteIntent(p.88)(vert_intent[cvc])
19      end
20      if (vrc != 0) && (tgt.coord_data.vrc != vert_intent[vrc])
21        tgt.coord_data.vrc = vert_intent[vrc]
22        own.received_vrcs[vert_intent[vrc]] = true
23      end
24    end
25  end
26 end

```

Algorithm 87 DeleteIntent

```

1 function DeleteIntent(intent_index::Z)
2   success::Bool = false
3   if (intent_index != 0)
4     success = false
5     for id in keys(target_db)
6       tgt = RetrieveWithID(p.20)(target_db, id)
7       if (intent_index == tgt.coord_data.vrc)
8         success = true
9       end
10    end
11    if (!success)
12      own.received_vrcs[intent_index] = false
13    end
14  end
15 end

```

Referenced In: RemoveStaleModeSTrack(p.116), ReceiveUF16UDS30(p.88), CoordinationTimeoutCheck(p.89)

The UPDATEINTRUDERVRC (Algorithm 88) algorithm takes in a list of all valid TRM *intruders* to be updated with the most recent VRC. The algorithm loops over the *target_db* and all the *intruders* for which the TRM has requested an update to find the corresponding target in the STM database. If the IDs match and the intruder is equipped with EQUIPAGE_TCASRA, the intruder VRC will be updated using the target in the STM database to ensure that the most recent VRC is used by the TRM. The combination of all VRCs received from existing intruders is returned in *received_vrcs* to form the Resolution Advisory Complements (RAC).

This algorithm takes as input *intruders*. This algorithm returns *received_vrcs*.

Algorithm 88 UpdateIntruderVRC

```

1 function UpdateIntruderVRC(intruders::Vector{TRMIintruderInput(p. E-20)})  

2   for id in keys(target_db)  

3     for intruder in intruders  

4       if (intruder.id == id)  

5         if (intruder.equipage == EQUIPAGE_CASRA)  

6           intruder.vrc = target_db[id].coord_data.vrc  

7         end  

8       end  

9     end  

10    end  

11    received_vrcs::Vector{Bool} = copy( own.received_vrcs )  

12    return received_vrcs::Vector{Bool}  

13 end
```

Referenced In: UpdateIntruderInputs(p. 211), VerticalTRMUpdate(p. 152)

COORDINATIONTIMEOUTCHECK (Algorithm 89) takes as input a target retrieved from memory along with the time of the generated report containing coordination data. If the difference in the *report_time* and the time of the last known coordination update is greater than the *timeout*, the coordination data of that target is reset and the vertical intent is removed by calling DELETEINTENT.

This algorithm takes as input *tgt* and *report_time*. This algorithm updates *mu* and *Sigma*, if necessary.

Algorithm 89 CoordinationTimeoutCheck

```

1 function CoordinationTimeoutCheck(tgt::Target(p. E-14), report_time::R)  

2   const timeout::Z = params().coordination.timeout  

3   if (tgt.coord_data.toa > 0) && ((report_time - tgt.coord_data.toa) >= timeout)  

4     cancel_index::Z = tgt.coord_data.vrc  

5     tgt.coord_data = ReceivedCoordinationData(p. E-13)()  

6     DeleteIntent(p. 88)(cancel_index)  

7   end  

8 end
```

Referenced In: RemoveStaleTracks(p. 115)

2.5.1.2 Coordination Capability Report

The STM includes an entry point for coordination capability information in RECEIVECAPABILITYREPORT (Algorithm 90) with inputs defined in Table 2-15. Those inputs marked (reserved) are unused and exist as placeholders to inform future capabilities. This algorithm stores information received from the Collision Avoidance Coordination Capability Bits (CCCB) received via ADS-B. The Mode S address is used to associate the received CCCBs with an existing Target. If an existing Target is found, the bits are stored in the Target structure to determine equipage and coordination protocol during STM report generation.

This algorithm takes as input the variables in the following table. This algorithm updates *tgt*.

Table 2-15. STM Input Variables - Coordination Capability Report

Variable	Units	Type	Description
adsb_version	N/A	uint32	ADS-B version indicator (reserved)
ca_operational	N/A	uint8	Indicates that a collision avoidance system is operational
sense	N/A	uint8	Vertical and/or horizontal sense bits (reserved)
type_capability	N/A	uint8	CAS and type capability
priority	N/A	uint8	Priority flag (reserved)
daa	N/A	uint8	Detect and Avoid subfield that indicates the coordination type to be provided
mode_s	N/A	uint32	Reported Mode S address

Algorithm 90 ReceiveCapabilityReport

```

1 function ReceiveCapabilityReport(adsb_version::UInt32, ca_operational::UInt8, sense::UInt8, type_capability:::
    UInt8, priority::UInt8, daa::UInt8, mode_s::UInt32)
2     id = AssociateICAOtoTarget(p. 18)(mode_s)
3     if (!isnan(id))
4         tgt = RetrieveWithID(p. 20)(target_db, id)
5         tgt.ca_operational = ca_operational
6         tgt.type_capability = type_capability
7         tgt.daa = daa
8     end
9 end

```

During STM report generation, SETCOORDINATION (Algorithm 91) is called to further update the intruder equipage and coordination protocol flags reported for the intruder. Initially, the coordination protocol is defaulted to not use coordination. The algorithm first checks if the intruder has been marked as being capable of vertical or horizontal RAs (DF0 RI = 3 or 4), in which case it may participate in TCAS active coordination. If CA Coordination Capability Bits (CCCB) were received, the intruder may be updated to reflect additional equipages (such as responsive coordination or UAS detect and avoid) and use an appropriate coordination protocol.

This algorithm takes as input *intruder*. This algorithm updates *intruder*.

Algorithm 91 SetCoordination

```

1 function SetCoordination(intruder::TRMIntruderInput(p_E-20))
2   intruder.active_cas_version = ACTIVE_CAS_TCAS
3   intruder.coordination_msg = COORDINATION_NONE
4   tgt = RetrieveWithID(p_20)(target_db, intruder.id)
5   if (intruder.equipage == EQUIPAGE_CASRA)
6     intruder.coordination_msg = COORDINATION_TCAS
7     if (tgt.type_capability == CAS_ACTIVE_NON_TCAS) || (tgt.type_capability == CAS_ACTIVE_NON_TCAS_0CM)
8       intruder.active_cas_version = ACTIVE_CAS_ACASX
9     end
10   else
11     if (tgt.ca_operational == 1)
12       if (tgt.type_capability == CASRESP_ACTIVE) || (tgt.type_capability == CASRESP_PASSV_MODES)
13         intruder.equipage = EQUIPAGE_CASRESP
14         intruder.coordination_msg = COORDINATION_TCAS
15       elseif (tgt.type_capability == CASRESP_PASSV)
16         intruder.equipage = EQUIPAGE_CASRESP
17         intruder.coordination_msg = COORDINATION_0CM
18       end
19     elseif (tgt.daa == DAA_RCV_ACTIVE)
20       intruder.equipage = EQUIPAGE_DAARESP
21       intruder.coordination_msg = COORDINATION_TCAS
22     elseif (tgt.daa == DAA_RCV_PASSV)
23       intruder.equipage = EQUIPAGE_DAARESP
24       intruder.coordination_msg = COORDINATION_0CM
25     end
26   end
27 end

```

Referenced In: AddADSBTrackToReport([p. 110](#)), AddModeSTrackToReport([p. 102](#)), AddModeCTrackToReport([p. 114](#))

2.6 Track Management

This section defines the algorithms that perform maintenance on existing Targets stored in memory and produces reports that are used by other modules in the ACAS X system. The major functions encompassed by these algorithms include:

1. The calculation of ownship sensitivity level using SETSENSITIVITYLEVEL.
2. The determination and packaging of air-to-air data to send to the transponder using SETTRANSPONDERDATA.
3. The update of the ownship advisory mode using UPDATEADVISORYMODE.
4. The extrapolation, decorrelation, and correlation of individual tracks using CORRELATIONPROCESSING.
5. The selection of the surveillance source that is used to generate the STMREPORT.
6. The algorithms for active validation.
7. The generation of the STMREPORT for consumption by the TRM and other external modules.
8. The deletion of deprecated Track Files.

2.6.1 Sensitivity Level Calculation

Before the STMREPORT is generated, the ownship sensitivity level is calculated by SETSENSITIVITYLEVEL (Algorithm 92). The first step is to set whether ownship is on the ground (indicated through *own.on_ground*). The validity of the ownship radio altitude (*own.radalt*) is first checked (i.e., whether or not it evaluates to a NaN) before being used. If the radio altitude is valid, the counter for invalid data is reset (*own.invalid_radalt_cycles*) and the radio altitude is compared against the upper and lower bounds (*ZLIMITU* and *ZLIMITL*) to declare if ownship is on the ground. Other-

wise, if the ownship radio altitude is invalid, the counter for invalid data is incremented. If the data has been invalid for a sufficient number of cycles, then the on ground indication is cleared.

Once the ownship on-ground status is determined, the reported sensitivity level for ownship can be set. To support hysteresis applied to sensitivity level changes, the current sensitivity level (*own.sensitivity_level*) is saved as *prev_sl*. The sensitivity level is then reset to *SENSITIVITY_LEVEL_STANDBY*. To exit the *SENSITIVITY_LEVEL_STANDBY* state, system operation must be permitted (*own.opflg* must be clear) and the pilot-selected *manualSL* value must not be *SENSITIVITY_LEVEL_STANDBY*. In addition, either the *own_ground_display_mode_on* discrete input must be enabled, or the previous ownship on-ground determination must have established the ownship is airborne. If these conditions are met, the sensitivity level is guaranteed to be greater than *SENSITIVITY_LEVEL_STANDBY* and so the previous setting is restored. If the previous setting was *SENSITIVITY_LEVEL_STANDBY*, it is increased to *SENSITIVITY_LEVEL_TA_MODE*.

If a valid radar altitude is available, the sensitivity level is set according to the transition thresholds defined by *ZSL2TO3* (transition from *SENSITIVITY_LEVEL_TA_MODE* to *SENSITIVITY_LEVEL_RA_MODE*) and *ZSL3TO2* (transition from *SENSITIVITY_LEVEL_RA_MODE* to *SENSITIVITY_LEVEL_TA_MODE*). The difference in these thresholds results in hysteresis, preventing rapid oscillations in sensitivity level when operating at a radar altitude near to either threshold.

If no radar altitude has been available for an extended period of time (given by *RADARLOST*) the sensitivity level is set to *SENSITIVITY_LEVEL_RA_MODE*.

Note that ACAS X will never provide an ownship sensitivity level greater than 3 (*SENSITIVITY_LEVEL_RA_MODE*). Note also that a pilot selection of *SENSITIVITY_LEVEL_TA_MODE* for (*manualSL*) intentionally will not prohibit the selection of a *SENSITIVITY_LEVEL_RA_MODE* for sensitivity level.

This algorithm takes no input. This algorithm updates *own.sensitivity_level* and *own.on_ground*.

Algorithm 92 SetSensitivityLevel

```

1 function SetSensitivityLevel()
2   const max_invalid_radalt_cycles::Z = params().surveillance.ownship_vertical.max_invalid_radalt_cycles
3   if (!isnan(own.radalt))
4     own.invalid_radalt_cycles = 0
5     if (own.radalt < ZLIMITL)
6       own.on_ground = true
7     elseif (own.radalt > ZLIMITU)
8       own.on_ground = false
9     end
10   else
11     own.invalid_radalt_cycles += 1
12     if (own.invalid_radalt_cycles > max_invalid_radalt_cycles)
13       own.on_ground = false
14     end
15   end
16   prev_sl = own.sensitivity_level
17   own.sensitivity_level = SENSITIVITY_LEVEL_STANDBY
18   if (own.discrete.opflg == true) && (own.discrete.manualSL != SENSITIVITY_LEVEL_STANDBY)
19     if (own.on_ground == false) || (own.discrete.own_ground_display_mode_on == true)
20       own.sensitivity_level = max(prev_sl, SENSITIVITY_LEVEL_TA_MODE)
21     if (own.invalid_radalt_cycles == 0)
22       if (own.radalt <= ZSL3T02)
23         own.sensitivity_level = SENSITIVITY_LEVEL_TA_MODE
24       elseif (own.radalt >= ZSL2T03)
25         own.sensitivity_level = SENSITIVITY_LEVEL_RA_MODE
26       end
27     elseif (own.invalid_radalt_cycles > RADARLOST)
28       own.sensitivity_level = SENSITIVITY_LEVEL_RA_MODE
29     end
30   end
31 end
32 end

```

Referenced In: UpdateAdvisoryMode(p. 94)

2.6.2 Transponder Mode Management

The SETTRANSPODERRDATA (Algorithm 93) algorithm collects data using the *own.transponder* structure, of type TRANSPODERRDATA. The algorithm begins by resetting all fields to indicate a standby mode of operation. Incrementally these fields are set to indicate higher operational modes as allowed by the ownship sensitivity level and the pilot control selection. Critically, this algorithm determines the ownship Air-to-Air Reply Information (*ri*). If both ownship sensitivity level and the pilot control selection allow the generation of RA alerts as indicated by the *ri* field, the system will be allowed to operate in an RA mode. Finally, the algorithm sets the sensitivity level and additional data link capability report information to be sent to the transponder.

This algorithm takes no input. The algoithm updates the global *own.transponder*.

Algorithm 93 SetTransponderData

```

1 function SetTransponderData()
2   own.transponder.ri = RI_FIELD_NONE
3   own.transponder.bit48 = false
4   own.transponder.bit70 = false
5   if (own.sensitivity_level != SENSITIVITY_LEVEL_STANDBY)
6     own.transponder.bit48 = true
7     own.transponder.ri = RI_FIELD_CASTA
8     if (own.sensitivity_level == SENSITIVITY_LEVEL_RA_MODE) && (own.discrete.manualSL ==
9       SENSITIVITY_LEVEL_AUTOMATIC)
10    own.transponder.ri = RI_FIELD_CASRA
11    own.transponder.bit70 = true
12  end
13 end
14 own.transponder.bit69 = false
15 own.transponder.bit71 = true
16 own.transponder.bit72 = true
17 own.transponder.vi = TCAS_VERSION_INDICATOR_FUTURE
18 own.transponder.sl = own.sensitivity_level
19 end

```

Referenced In: UpdateAdvisoryMode(p. 94)

UPDATEADVISORYMODE (Algorithm 94) invokes the calls to SETSENSITIVITYLEVEL and SETTRANSPONDERDATA which determine the ownship sensitivity level and Air-to-Air reply information. UPDATEADVISORYMODE then resets the TRM operating mode (*trm_opmode*) to *OPMODE-STANDBY*. If and only if the ownship is not operating on the surface, as indicated by the *on_surface* discrete input, is the TRM operating mode set given the Air-to-Air Reply Information determined previously. If the ownship is operating on the surface, the TRM operating mode is maintained in *OPMODE_STANDBY*.

This algorithm takes no input. This algorithm updates *own.trm_opmode*.

Algorithm 94 UpdateAdvisoryMode

```

1 function UpdateAdvisoryMode()
2   SetSensitivityLevel(p. 93)()
3   SetTransponderData(p. 94)()
4   own.trm_opmode = OPMODE_STANDBY
5   if (own.discrete.on_surface == false)
6     if (own.transponder.ri == RI_FIELD_CASTA)
7       own.trm_opmode = OPMODE_TA
8     elseif (own.transponder.ri == RI_FIELD_CASRA)
9       own.trm_opmode = OPMODE_RA
10   end
11 end
12 end

```

Referenced In: GenerateStmReport(p. 97)

2.6.3 Surveillance Source Correlation

Prior to generating the STMREPORT, all targets within the STM must be evaluated in CORRELATIONPROCESSING (Algorithm 95) to determine if any two targets represent a single intruder aircraft or if two intruder aircraft are incorrectly represented by a single target. First, a call to PREDICTTRACKSUMMARY extrapolates all tracks within each target to the report generation time to allow

for spatial correlation tests. Tracks that no longer correlate to their assigned target are moved to new targets in DECORRELATETARGETS. Finally, CORRELATETARGETS iterates through all existing targets and merges track files from any two correlating targets into a single target.

Execution of CORRELATIONPROCESSING modifies the target data base so that each track is assigned to a target identified either by address or spatial matching, as described above. The performance requirements applicable to correlation are specified in §2.2.5.4 of the ACAS Xa/Xo MOPS Volume I [16]. CORRELATIONPROCESSING is a prescriptive algorithm, though the algorithm consists entirely of calls to suggested algorithms provided as guidance in Appendix C. Whether an equipment manufacturer chooses to implement the algorithms of Appendix C, a modification thereof, or provides their own design, the complete system must be shown to meet the requirements ACAS Xa/Xo MOPS Volume I [16] §2.2.5.4. The algorithms of Appendix C have been demonstrated to meet these requirements given specific data sets provided by at least one manufacturer.

Algorithm 95 CorrelationProcessing

```

1 function CorrelationProcessing(T:R)
2   PredictTrackSummary(p. C-1)(T)
3   DecorrelateTargets(p. C-6)(T)
4   CorrelateTargets(p. C-12)(T)
5 end
```

Referenced In: GenerateStmReport(<i>p. 97</i>)

2.6.4 STM Report Generation

Upon request, the STM can produce a data structure called the STMREPORT that contains all elements required by the TRM and other external modules. This process, defined in GENERATESTMREPORT (Algorithm 96), takes a single input (*i*), which is the requested time of applicability for the report. If necessary, tracks are extrapolated by the algorithm so that all information contained in the report is referenced at a common time.

- A call to GENERATESTMREPORT begins with a calculation of ownship sensitivity level, transponder data, and operating mode (UPDATEADVISORYMODE), and is then followed by target correlation (CORRELATIONPROCESSING).
- The algorithm then initializes an empty STMREPORT to be populated with data in subsequent steps.
- The first data element is ownship radio altitude, which is retrieved from memory and added to the report. The ownship heading is added to the report through a call to HEADINGATTOA, which extrapolates ownship heading to the time of the report.
- Then the ownship Mode S address, Mode A code, and TA/RA state are retrieved from memory and added to the report.
- Next, the initialization status of the ownship barometric altitude track is determined through the associated TOA field. If this field is *Nan*, the track is not initialized and the report is populated with empty ownship vertical state information and no intruder state data is added to the report.
- If the ownship barometric altitude track is initialized the degraded surveillance state (*own.degraded_surveillance*) is evaluated and stored.

—,---,---,---,---,---,---,---,---,---,---

- The ownship barometric altitude is then extrapolated to the time of the STMREPORT. This extrapolation, however, is not performed if the difference between the report TOA and the time of the last track update is less than a defined threshold (*min_extrap_toa_step*).
- The ownship vertical belief state is populated with the ownship vertical estimate. Since the TRM expects a single sample to represent the state of ownship altitude, the mean (*mu*) of the ownship vertical track is added to the report with a weight of one.
- Intruder Track Files are then added to the report through a call to ADDTRACKSTOREPORT.

ADJUSTTARGETDESIGNATIONVALIDITY is called to update target designation information for each TRMINTRUDERINPUT in the STMREPORT. It also updates the DESIGNATIONSTATE for each target in the database and adds a special TRMINTRUDERINPUT to the STMREPORT whenever a designated Target was not included in the original STMREPORT. These updates are needed to properly handle cases where the limit on the number of designations was reached or designated targets no longer appear in the the STMREPORT. Xo mode availability information is added to the report using SETTARGETDESIGNATIONMODEAVAILABILITY.

This algorithm takes as input *t*, updates *own.prev_rpt_toa*, and returns *report*.

Algorithm 96 GenerateStmReport

```

1 function GenerateStmReport(t::R)
2   const min_extrap_toa_step::R = params().surveillance.min_extrap_toa_step
3   UpdateAdvisoryMode(p. 94)()
4   CorrelationProcessing(p. 95)(t)
5   report = StmReport(p. E-1)()
6   report.trm_input.own.h = own.radalt
7   report.trm_input.own.psi = HeadingAtToa(p. 85)(t)
8   report.trm_input.own.mode_s = own.modes
9   report.trm_input.own.mode_a = own.discrete.modeACode
10  report.trm_input.own.opmode = own.trm_opmode
11  report.transponder = own.transponder
12  designation_needs_update::Bool = false
13  if (!isnan(own.toa_h))
14    if (own.toa_h < own.prev_rpt_toa)
15      report.trm_input.own.degraded_own_surveillance += DEGRADED_OWN_BAROALT_COAST
16    end
17    if (own.wgs84_toa < own.prev_rpt_toa)
18      report.trm_input.own.degraded_own_surveillance += DEGRADED_OWN_WGS84_COAST
19    end
20    if (own.heading_state == OWN_HEADING_INVALID)
21      report.trm_input.own.degraded_own_surveillance += DEGRADED_OWN_HDG_INVALID
22    elseif (own.toa_heading < own.prev_rpt_toa)
23      report.trm_input.own.degraded_own_surveillance += DEGRADED_OWN_HDG_COAST
24    end
25    dt = t - own.toa_h
26    if (dt < min_extrap_toa_step)
27      (mu, Sigma) = (own.mu_h, own.Sigma_h)
28    else
29      (mu, Sigma) = PredictVerticalTracker(p. 45)(own.mu_h, own.Sigma_h, dt)
30    end
31    resize!(report.trm_input.own.belief_vert, 1)
32    report.trm_input.own.belief_vert[1] = OwnVerticalBelief(p. E-21)()
33    report.trm_input.own.belief_vert[1].z = mu[1]
34    report.trm_input.own.belief_vert[1].dz = mu[2]
35    report.trm_input.own.belief_vert[1].weight = 1.0
36    AddTracksToReport(p. 98)(report, t)
37  else
38    report.trm_input.own.belief_vert = Array(OwnVerticalBelief(p. E-21), 0)
39    report.trm_input.intruder = Array(TRMIntruderInput(p. E-20), 0)
40    report.display = Array(StmDisplayStruct(p. E-1), 0)
41    designation_needs_update = true
42  end
43  AdjustTargetDesignationValidity(p. 143)(report, designation_needs_update)
44  SetTargetDesignationModeAvailability(p. 145)(report, t)
45  own.prev_rpt_toa = t
46  return report::StmReport(p. E-1)
47 end

```

ADDTRACKSTOREPORT (Algorithm 97) iterates through all Targets that are stored in memory and determines which, if any, Track Files for a given Target should be appended to the STMREPORT. Track Files are prioritized per Target based on their underlying surveillance source and applicable quality requirements. Component tracks within each Track File may have discrepant validities. If a Track File is selected with an invalid component track, an appropriate notation will be set in the degraded surveillance flags for that Target (see SETINTRUDERDEGRADEDFLAGS). For example, an invalid vertical track will result in a NAR indication, while an invalid Cartesian track of an active Track File will cause the Target to be reported as bearingless. The selection of Track Files per Target will not select an active surveillance Track File with an invalid range track. Similarly, the algorithm will prefer an active surveillance Track File to a passive surveillance Track File with no valid vertical track. Each Target is processed through a call to REMOVESTALETRACKS, which deletes any

.....

Track Files in the Target that are considered deprecated (i.e., coasted out and not designated). If all Track Files are removed from a Target, no more processing is performed on that Target. If an ADSBTRACKFILE exists for the Target, that Track File passes a quality check and active validation, and the ownership WGS84 state is valid, the relevant data elements in the ADSBTRACKFILE are added to the report through ADDADSBTRACKTOREPORT. If a validated ADSBTRACKFILE is not available and if a MODESTRACKFILE with a valid range track exists for the Target, then the relevant data elements from the MODESTRACKFILE are added to the report through a call to ADDMODESTRACKTOREPORT. Otherwise, the algorithm will check if a MODECTRACKFILE exists for the Target, and if found its relevant data elements are added to the report through a call to ADDMODECTRACKTOREPORT. Still otherwise, if an ADSBTRACKFILE based on non ADS-R data exists, the ADSBTRACKFILE passes the minimum passive-only data quality check (which is overridden for Hybrid Surveillance targets by the *adsb_qual_override* flag), and the ownership WGS84 state is valid, it is added to the report through ADDADSBTRACKTOREPORT. If still no Track File for the Target has been added to the report, then if an ADSBTRACKFILE based on ADS-R data exists, the ADSBTRACKFILE passes the minimum passive-only data quality check, and the ownership WGS84 state is valid, it is added to the report through ADDADSBTRACKTOREPORT. Finally, if the Target contains no valid tracks, SETTARGETDESIGNATIONINVALID is used to update the DESIGNATIONSTATE invalid timers (*timer_cspo3000* and *timer_dna*). If the timers have expired, the Target will be marked for undesignation. Before completing, a call to FILTERTRACKSFORTRM (Algorithm 99) limits the number intruders sent to the TRM and the display.

This algorithm takes as input time t and an STMREPORT *report* and appends any validated track files to the input report.

Algorithm 97 AddTracksToReport

```

1 function AddTracksToReport(report::StmReport(p. E-1), t::R)
2   for id in keys(target_db)
3     if (RemoveStaleTracks(p. 115)(id, t))
4       if (TrackExists(p. 18)(target_db[id].adsb_track)) && (ActiveValidationCheck(p. 122)(target_db[id])) && (
          AdsbQualityCheck(p. 111)(target_db[id], false) && (target_db[id].adsb_track.valid_vert) && (!
          isnan(own.wgs84_toa))
          AddADSBTrackToReport(p. 110)(report, id, t, false, false)
        elseif (TrackExists(p. 18)(target_db[id].modes_track)) && (target_db[id].modes_track.valid_rng)
          AddModeSTrackToReport(p. 102)(report, id, t)
        elseif (TrackExists(p. 18)(target_db[id].modec_tracks)) && HasValidModeCTrack(p. 99)(target_db[id])
          AddModeCTrackToReport(p. 114)(report, id, t)
        elseif (TrackExists(p. 18)(target_db[id].adsb_track)) && ((target_db[id].adsb_qual_override == true
          || (AdsbQualityCheck(p. 111)(target_db[id], true))) && (!isnan(own.wgs84_toa)))
          AddADSBTrackToReport(p. 110)(report, id, t, false, true)
        elseif (TrackExists(p. 18)(target_db[id].adsr_track)) && (AdsbQualityCheck(p. 111)(target_db[id], true)
          ) && (!isnan(own.wgs84_toa))
          AddADSBTrackToReport(p. 110)(report, id, t, true, true)
        else
          SetTargetDesignationInvalid(p. 139)(target_db[id].designation_state, false)
        end
      end
    end
  end
  FilterTracksForTRM(p. 100)(report)
end
```

Referenced In:	GenerateStmReport _(p. 97)
----------------	--------------------------------------

HASVALIDMODECTRACK (Algorithm 98) takes as input a TARGET *tgt* and returns a boolean indicating whether the input contains any non-image MODECTRACKFILE with a valid range track.

Algorithm 98 HasValidModeCTrack

```

1 function HasValidModeCTrack(tgt::Target(p. E-14))
2   for j in 1:length(tgt.modec_tracks)
3     if (!IsImageTrack(p. B-8)(tgt.modec_tracks[j])) && (tgt.modec_tracks[j].valid_rng)
4       return true
5     end
6   end
7   return false
8 end

```

Referenced In: AddTracksToReport(p. 98)

The ACAS X system limits the number of intruders for processing by the Threat Resolution Module (TRM) to *max_intruders*. In the case where there are more intruders than that, some intruders need to be filtered from the STM output.

FILTERTRACKSFORTRM (Algorithm 99) prioritizes intruders for output to the TRM and filters out the lowest priority intruders. Intruders are prioritized based on relative importance with respect to TRM processing.

The intruders are first sorted by slant range via nested calls to COMPAREINTRUDER RANGE (Algorithm 100) and SLANTRANGE (Algorithm 101). The range-sorted intruders are then categorized by the following priorities:

1. There is an active resolution advisory on the intruder
2. An up or down sense VRC was received from the intruder
3. The intruder is designated
4. The intruder is proximate

The first *max_intruders* are chosen for TRM processing from the resulting prioritized and range-ordered lists. The TRM will receive the intruders that are most likely to pose a threat. Because *report.display* and *report.trm_input.intruder* contain information about the same intruders, this filtering also impacts the targets available to the onboard pilot display. Note that while both the *report.display* and *report.trm_input.intruder* vectors are expected to be the same length, defensive programming in this algorithm will handle the alternate case.

This algorithm takes as input *report* and updates *report.display* and *report.trm_input.intruder*.

Algorithm 99 FilterTracksForTRM

```

1 function FilterTracksForTRM( report::StmReport(p.E-1) )
2   const max_intruders::Z = params().surveillance.max_intruders
3   if (max_intruders < length(report.display))
4     display_tmp::Vector{Vector{StmDisplayStruct}} = Array(Vector{StmDisplayStruct(p.E-1)},5)
5     intruder_tmp::Vector{Vector{TRMIIntruderInput}} = Array(Vector{TRMIIntruderInput(p.E-20)},5)
6     for i in 1:length(display_tmp)
7       display_tmp[i] = StmDisplayStruct(p.E-1)[]
8       intruder_tmp[i] = TRMIIntruderInput(p.E-20)[]
9     end
10    sorted_idx = sortperm( report.display, lt = CompareIntruderRange(p.100) )
11    for idx in sorted_idx
12      if (idx <= length(report.trm_input.intruder))
13        target = target_db[report.trm_input.intruder[idx].id]
14        if target.designation_state.active_ra
15          push!( display_tmp[1], report.display[idx] )
16          push!( intruder_tmp[1], report.trm_input.intruder[idx] )
17        elseif (0 != report.trm_input.intruder[idx].vrc)
18          push!( display_tmp[2], report.display[idx] )
19          push!( intruder_tmp[2], report.trm_input.intruder[idx] )
20        elseif target.designation_state.is_designated
21          push!( display_tmp[3], report.display[idx] )
22          push!( intruder_tmp[3], report.trm_input.intruder[idx] )
23        elseif ( report.trm_input.intruder[idx].is_proximate)
24          push!( display_tmp[4], report.display[idx] )
25          push!( intruder_tmp[4], report.trm_input.intruder[idx] )
26        else
27          push!( display_tmp[5], report.display[idx] )
28          push!( intruder_tmp[5], report.trm_input.intruder[idx] )
29        end
30      end
31    end
32    report.display = StmDisplayStruct(p.E-1)[]
33    report.trm_input.intruder = TRMIIntruderInput(p.E-20)[]
34    remaining::Z = max_intruders
35    for i in 1:length(display_tmp)
36      limit::Z = min( length(display_tmp[i]), remaining )
37      if (0 < limit)
38        append!( report.display, display_tmp[i][1:limit] )
39        append!( report.trm_input.intruder, intruder_tmp[i][1:limit] )
40        remaining = remaining - limit
41      end
42    end
43  end
44 end

```

Referenced In: AddTracksToReport(p.98)

Algorithm 100 CompareIntruderRange

```

1 function CompareIntruderRange(intr_a::StmDisplayStruct(p.E-1), intr_b::StmDisplayStruct(p.E-1))
2   return SlantRange(intr_a.r_ground, intr_a.z_rel) < SlantRange(p.101)(intr_b.r_ground, intr_b.z_rel)
3 end

```

Referenced In: FilterTracksForTRM(p.100)

Algorithm 101 SlantRange

```

1 function SlantRange(r_ground::R, z_rel::R)
2   if (isnan(z_rel))
3     return r_ground
4   else
5     return hypot(r_ground, z_rel)
6   end
7 end
  
```

Referenced In: CompareIntruderRange(p. 100)

2.6.4.1 Mode S Track File

ADDMODESTRACKTOREPORT (Algorithm 102) defines the process through which the data in a MODESTRACKFILE is added to an STMREPORT. First, the corresponding Track File is retrieved from memory and a new TRMINTRUDERINPUT (i.e., the data structure in the STMREPORT that represents an intruding aircraft) is initialized. Several of the data elements in the TRMINTRUDERINPUT are then populated with the associated data in the MODESTRACKFILE, including the track id (*id*), the reported Mode S address (*modes* and *is_icao*, which is TRUE by definition), the presence and operational status of TCAS as dictated by the reported RI field (*equipage*), the most recent VRC (*vrc*) that has been received from the intruder, validity of the track for TA/RA processing (*valid*), and the source of the intruder state information (*source*). Targets are marked as invalid for TA/RA processing under some degraded surveillance conditions (when within the reduced surveillance region or when the target is NAR and ownship is above the NAR threshold) and also when ownship is not in a TA mode (as per *own.trm_opmode*). Degraded surveillance flags are set in a call to SETINTRUDERDEGRADEDFLAGS (Algorithm 104).

If necessary, the three tracks contained in the MODESTRACKFILE are extrapolated to the time of the report by calling the corresponding prediction algorithms. This extrapolation is not performed if the time difference between the last track update and the TOA of the report is less than a defined minimum threshold (*min_extrap_toa_step*). The representation of the target aircraft's vertical state is provided as a set of weighted samples given by *vsamples* and *vweights*. These samples are produced using the mean (*mu_vert*) and covariance (*Sigma_vert*) of the vertical track through a call to ADDALTBIASANDSAMPLE. The representation of the target aircraft's relative horizontal state is also provided as a set of weighted samples given by *hsamples* and *hweights*. These samples are produced through a call to COMBINEANDSAMPLE, which merges the Cartesian Track and the Range Track into a single horizontal representation. The vertical and horizontal samples are provided to SETINTRUDERBELIEFSTATES (Algorithm 103) for assignment within the TRMINTRUDERINPUT. In the last steps, the target designation state gets set by SETTARGETDESIGNATION, the STMDISPLAYSTRUCT vector gets filled by SETDISPLAYDATAACTIVE, the proximate state of the intruder is determined by PROXIMITYESTIMATION and the TRMINTRUDERINPUT is appended to STMREPORT.

This algorithm takes as input *report*, *id*, *T*. This algorithm updates *report.trm_input.intruder*.

Algorithm 102 AddModeSTrackToReport

```

1 function AddModeSTrackToReport(report::StmReport(p. E-1), id::UInt32, T::R)
2   const nars_threshold::R = params().display.nars_threshold
3   baro_alt::R = BaroAltAtToa(p. 78)(T)
4   tgt = target_db[id]
5   trk = tgt.modes_track
6   intruder = TRMIIntruderInput(p. E-20)()
7   intruder.id = id
8   intruder.address = trk.modes
9   intruder.is_icao = true
10  if (trk.ri == RI_FIELD_CASRA) || (trk.ri == RI_FIELD_CASRA_H0RZ)
11    intruder.equipage = EQUIPAGE_CASRA
12    intruder.vrc = tgt.coord_data.vrc
13  elseif (trk.ri == RI_FIELD_CASTA)
14    intruder.equipage = EQUIPAGE_CASTA
15  else
16    intruder.equipage = EQUIPAGE_MODES
17  end
18  intruder.source = SOURCE_MODES
19  SetCoordination(p. 91)(intruder)
20  reduced_surveillance = (trk.surv_mode != SURVEILLANCE_REGION_NORMAL)
21  SetIntruderDegradedFlags(p. 104)(intruder, tgt, trk, true, reduced_surveillance, false)
22  intruder.valid = true
23  if (own.trm_opmode == OPMODE_STANDBY) ||
24    (reduced_surveillance) ||
25    (!trk.valid_vert) && (baro_alt >= nars_threshold))
26    intruder.valid = false
27  end
28  dt_vert = T - trk.toa_vert
29  dt_range = T - trk.toa_rng
30  dt_cart = T - trk.toa_cart
31  (mu_vert, Sigma_vert) = PredictVerticalTracker(p. 45)(trk.mu_vert, trk.Sigma_vert, dt_vert)
32  (mu_rng, Sigma_rng) = PredictRangeTracker(p. 39)(trk.mu_rng, trk.Sigma_rng, dt_range)
33  (mu_cart, Sigma_cart) = PredictCartesianTracker(p. 33)(trk.mu_cart, trk.Sigma_cart, dt_cart)
34  (vsamples, vweights) = AddAltBiasAndSample(p. 105)(mu_vert, Sigma_vert, trk.valid_vert)
35  (hsamples, hweights) = CombineAndSample(p. 105)(mu_rng[1:2], Sigma_rng[1:2,1:2], mu_cart, Sigma_cart, trk.
    valid_cart)
36  SetIntruderBeliefStates(p. 103)(intruder, vsamples, vweights, hsamples, hweights)
37  SetTargetDesignation(p. 125)(intruder, T, mu_vert[1], mu_rng[1], tgt.designation_state)
38  SetDisplayDataActive(p. 107)(report, intruder, mu_vert, mu_cart, mu_rng, trk.valid_vert, trk.valid_cart, T,
    trk.display_arrow_current)
39  ProximityEstimation(p. 147)(intruder, baro_alt, mu_vert[1], mu_rng[1], trk.valid_vert, trk.valid_cart)
40  push!(report.trm_input.intruder, intruder)
41 end

```

Referenced In: AddTracksToReport(**p. 98**)

Algorithm 103 SetIntruderBeliefStates

```

1 function SetIntruderBeliefStates(intruder::TRMIntruderInput(p. E-20), vsamples_zdz::Matrix{R}, vweights::Vector{R},
2   hsamples_xydxdy::Matrix{R}, hweights::Vector{R})
3   resize!(intruder.belief_vert, length(vweights))
4   for j in 1:length(vweights)
5     intruder.belief_vert[j] = IntruderVerticalBelief(p. E-21)()
6     intruder.belief_vert[j].z = vsamples_zdz[1,j]
7     intruder.belief_vert[j].dz = vsamples_zdz[2,j]
8     intruder.belief_vert[j].weight = vweights[j]
9   end
10  resize!(intruder.belief_horiz, length(hweights))
11  for j in 1:length(hweights)
12    intruder.belief_horiz[j] = IntruderHorizontalBelief(p. E-21)()
13    intruder.belief_horiz[j].x_rel = hsamples_xydxdy[1,j]
14    intruder.belief_horiz[j].y_rel = hsamples_xydxdy[2,j]
15    intruder.belief_horiz[j].dx_rel = hsamples_xydxdy[3,j]
16    intruder.belief_horiz[j].dy_rel = hsamples_xydxdy[4,j]
17    intruder.belief_horiz[j].weight = hweights[j]
18 end

```

Referenced In: AddADSBTrackToReport^(p. 110), AddModeSTrackToReport^(p. 102), AddModeCTrackToReport^(p. 114)

SETINTRUDERDEGRADEDFLAGS (Algorithm 104) sets the various degraded surveillance flags for each intruder. Processing of the vertical degraded surveillance flags (NAR and VERT_COAST) and INVALID_UF16UDS30 flag are common for all track types. Setting of the REDUCED and NO_BEARING flags are specific to active track types, while the ADSB_ONLY flag is specific to ADS-B track types. Handling of the HORIZ_COAST differs between active and passive track types.

This algorithm takes as input *intruder*, *tgt*, *trk*, *active*, *reduced*, and *passive_only*. This algorithm updates *intruder.degraded_surveillance*.

Algorithm 104 SetIntruderDegradedFlags

```

1 function SetIntruderDegradedFlags(intruder::TRMItruderInput(p. E-20), tgt::Target(p. E-14), trk::Union(  

2   ModeCTrackFile(p. E-8), ModeSTrackFile(p. E-9), ADSBTrackFile(p. E-4)), active::Bool, reduced::Bool, passive_only  

3   ::Bool)  

4   if (!trk.valid_vert)  

5     intruder.degraded_surveillance += DEGRADED_SURVEILLANCE_NAR  

6   end  

7   if (trk.toa_vert < own.prev_rpt_toa)  

8     intruder.degraded_surveillance += DEGRADED_SURVEILLANCE_VERT_COAST  

9   end  

10  if (active)  

11    if (reduced)  

12      intruder.degraded_surveillance += DEGRADED_SURVEILLANCE_REDUCED  

13    end  

14    if (!trk.valid_cart)  

15      intruder.degraded_surveillance += DEGRADED_SURVEILLANCE_NO_BEARING  

16    end  

17    if (trk.toa_rng < own.prev_rpt_toa) || (trk.toa_cart < own.prev_rpt_toa)  

18      intruder.degraded_surveillance += DEGRADED_SURVEILLANCE_HORIZ_COAST  

19    end  

20  else  

21    if (passive_only)  

22      intruder.degraded_surveillance += DEGRADED_SURVEILLANCE_ADSB_ONLY  

23    elseif (reduced)  

24      intruder.degraded_surveillance += DEGRADED_SURVEILLANCE_REDUCED  

25    end  

26    if (trk.toa_hor < own.prev_rpt_toa)  

27      intruder.degraded_surveillance += DEGRADED_SURVEILLANCE_HORIZ_COAST  

28    end  

29  end  

30  if (tgt.bad_uf16uds30)  

31    intruder.degraded_surveillance += DEGRADED_SURVEILLANCE_INVALID_UF16UDS30  

32  tgt.bad_uf16uds30 = false  

33 end

```

Referenced In: AddADSBTrackToReport(p. 110), AddModeSTrackToReport(p. 102), AddModeCTrackToReport(p. 114)

The purpose of ADDALTBIASANDSAMPLE (Algorithm 105) is to add uncertainty to the intruder vertical state distribution to account for altimetry error. This process occurs only if the intruder vertical track is valid. If the track is invalid, the algorithm returns a set of samples and weights with all values set to zero, indicating a NAR intruder. If the vertical is valid, additional uncertainty is added based on the current barometric altitude estimate of ownship. The magnitude of this uncertainty inflation is based on the Standard Altimetry Error Model specified in [9]. This inflated distribution is then converted into five samples through a call to SIGMAPORTSAMPLE.

This algorithm takes as input *mu*, *Sigma*, and *valid*. This algorithm returns values to SIGMAPORTSAMPLE or returns zeros for invalid tracks.

Algorithm 105 AddAltBiasAndSample

```

1 function AddAltBiasAndSample(mu::Vector{R}, Sigma::Matrix{R}, valid::Bool)
2   const asarp_max_alt::Vector{Z} = params().surveillance.intruder_vertical.asarp_max_alt
3   const asarp_sigma::Vector{R} = params().surveillance.intruder_vertical.asarp_sigma
4   if (valid)
5     alt_bias_sigma = 0
6     for i in 1:length(asarp_max_alt)
7       if (mu[1] < asarp_max_alt[i])
8         alt_bias_sigma = asarp_sigma[i]
9         break
10      end
11    end
12    inflated_Sigma = copy(Sigma)
13    inflated_Sigma[1,1] += (alt_bias_sigma * alt_bias_sigma)
14    return SigmaPointSample(p.28)(mu,inflated_Sigma,1.0)
15  else
16    return (zeros(2,5),zeros(5))
17  end
18 end

```

Referenced In: AddADSBTrackToReport(p. 110), AddModeSTrackToReport(p. 102), AddModeCTrackToReport(p. 114)

COMBINEANDSAMPLE (Algorithm 106) takes as input the mean (*mu_rng*) and covariance (*Sigma_rng*) of the Range Track and the mean (*mu_xy*) and covariance (*Sigma_xy*) of the Cartesian Track. These quantities are then provided to CONVERTTOAZANDXRANGERATE, which produces a distribution of the estimated azimuth and cross ground range rate (*mu_az* and *Sigma_az*). The output is merged with the Range Track to yield a distribution of the overall relative horizontal state in polar coordinates (*mu_pol* and *Sigma_pol*). This polar distribution is sampled with a call to SIGMAPONTSAMPLE and the resulting weighted samples are converted back to Cartesian coordinates. The algorithm returns this final set of Cartesian samples (*S_cart*) and their associated weights (*w_pol*).

This algorithm takes as input *mu_rng*, *Sigma_rng*, *mu_xy*, *Sigma_xy*, and *valid_xy*. This algorithm returns *S_cart* and *w_pol*.

Algorithm 106 CombineAndSample

```

1 function CombineAndSample(mu_rng::Vector{R}, Sigma_rng::Matrix{R}, mu_xy::Vector{R}, Sigma_xy::Matrix{R},
                           valid_xy::Bool)
2   mu_pol = zeros(4)
3   Sigma_pol = zeros(4,4)
4   S_cart = zeros(4,9)
5   (mu_az,Sigma_az) = ConvertToAzAndXRangeRate(p.106)(mu_xy,Sigma_xy,valid_xy)
6   mu_pol = [mu_rng, mu_az]
7   Sigma_pol = block_diag(p.H-2)(Sigma_rng,Sigma_az)
8   (S_pol,w_pol) = SigmaPointSample(p.28)(mu_pol,Sigma_pol,2.0)
9   for i in 1:length(w_pol)
10     S_cart[1,i] = S_pol[1,i]*sin(S_pol[3,i])
11     S_cart[2,i] = S_pol[1,i]*cos(S_pol[3,i])
12     S_cart[3,i] = S_pol[2,i]*sin(S_pol[3,i]) + S_pol[4,i]*cos(S_pol[3,i])
13     S_cart[4,i] = S_pol[2,i]*cos(S_pol[3,i]) - S_pol[4,i]*sin(S_pol[3,i])
14   end
15   return (S_cart::Matrix{R}, w_pol::Vector{R})
16 end

```

Referenced In: AddModeSTrackToReport(p. 102), AddModeCTrackToReport(p. 114)

CONVERTTOAZANDXRANGERATE (Algorithm 107) transforms a horizontal distribution from Cartesian coordinates to polar coordinates and extracts the dimensions of azimuth and cross ground range rate. This process only occurs if the Cartesian Track is in a valid state. If the Cartesian Track is invalid, the azimuth and cross ground range rate are set to zero, thus representing a no-bearing intruder to the TRM. If the Cartesian Track is valid, the input Cartesian distribution (*mu_xy* and *Sigma_xy*) is sampled through a call to **SIGMAPONTSAMPLE**. The algorithm then iterates through each weighted sample to perform the nonlinear transformation from Cartesian to polar coordinates. For each sample, the azimuth dimension is first calculated as the arc-tangent of the relative X and Y position. For numerical stability reasons, the calculated azimuth for the central sigma point sample is forced to be in the range of 0 to 2π , while the azimuth for all other sigma points samples is forced to be within π radians from the center point. The velocity vector perpendicular to the range dimension (i.e., the cross ground range rate) is then calculated. The resulting sigma point samples are used to calculate the weighted mean (*mu_az*) and covariance (*Sigma_az*), which are returned by the algorithm.

This algorithm takes as input *mu_xy*, *Sigma_xy*, and *valid*. This algorithm returns *mu_az* and *Sigma_az*.

Algorithm 107 ConvertToAzAndXRangeRate

```

1 function ConvertToAzAndXRangeRate(mu_xy::Vector{R}, Sigma_xy::Matrix{R}, valid::Bool)
2   if (!valid)
3     return (zeros(2),diagm([1e-10,1e-10]))
4   end
5   (S_xy,w_xy) = SigmaPointSample(p.28)(mu_xy,Sigma_xy,2.0)
6   S_az = zeros(2,length(w_xy))
7   for i in 1:length(w_xy)
8     S_az[1,i] = atan2(S_xy[1,i],S_xy[2,i])
9     if (i == 1)
10       if (S_az[1,1] < 0)
11         S_az[1,1] += 2pi
12       end
13     else
14       if (S_az[1,i] - S_az[1,1] > pi)
15         S_az[1,i] -= 2pi
16       elseif (S_az[1,i] - S_az[1,1] < -pi)
17         S_az[1,i] += 2pi
18       end
19     end
20     S_az[2,i] = S_xy[3,i]*cos(S_az[1,i]) - S_xy[4,i]*sin(S_az[1,i])
21   end
22   (mu_az,Sigma_az) = WeightedMeanAndCovariance(p.29)(S_az,w_xy)
23   return (mu_az::Vector{R}, Sigma_az::Matrix{R})
24 end
```

Referenced In: CombineAndSample (p.105)
--

SETDISPLAYDATAACTIVE (Algorithm 108) populates the **STMREPORT** with fields needed to properly display each intruder. These fields include the reported Mode S address (*mode_s*), an indication as to whether the address is ICAO compliant (*is_icao*), the track id of the target (*id*), the vertical state validity (*alt_reporting*), the validity of the bearing data (*bearing_valid*), the direction of the intruder relative vertical velocity (*arrow*), the relative altitude of the intruder (*z_rel*), the bearing relative to ownship (*Chi_rel*), the ground range to the intruder (*r_ground*).

This algorithm takes as input *report*, *intruder*, *mu_vert*, *mu_cart*, *mu_rng*, *valid_vert*, *valid_cart*, *T*,

and *display_arrow_current*. This algorithm updates *intruder.stm_display* and appends that updated information to *report.display*.

Algorithm 108 SetDisplayDataActive

```

1 function SetDisplayDataActive(report::StmReport(p. E-1), intruder::TRMIIntruderInput(p. E-20), mu_vert::Vector{R},
    mu_cart::Vector{R}, mu_rng::Vector{R}, valid_vert::Bool, valid_cart::Bool, T::R, display_arrow_current::
    Z)
2   display::StmDisplayStruct = StmDisplayStruct(p. E-1)()
3   display.mode_s = intruder.address
4   display.is_icao = intruder.is_icao
5   display.id = intruder.id
6   display.z_rel = BaroAltAtToa(p. 78)(T) - mu_vert[1]
7   display.alt_reporting = valid_vert
8   own_heading::R = HeadingAtToa(p. 85)(T)
9   if (valid_cart)
10     display.Chi_rel = WrapToPi(p. H-3)(atan2(mu_cart[1], mu_cart[2]) - own_heading)
11     display.bearing_valid = true
12   else
13     display.Chi_rel = 0.0
14     display.bearing_valid = false
15   end
16   display.r_ground = mu_rng[1]
17   if (valid_vert)
18     display.arrow = display_arrow_current
19   else
20     display.arrow = DISPLAY_ARROW_LEVEL
21   end
22   intruder.stm_display = display
23   push!(report.display, display)
24 end
```

Referenced In: AddModeSTrackToReport(p. 102), AddModeCTrackToReport(p. 114)
--

VERTICALRATEARROWUPDATE (Algorithm 109) updates the vertical rate arrow shown on the pilot's display. It first reads in the following parameters:

- *arrow_N* - the number of past track estimates used in the count.
- *arrow_M* - out of N, the number of past track estimates with the same vertical direction needed to set the arrow on the display
- *level_arrow_threshold* - the vertical speed over which the intruder is no longer considered level.

If the number of past elements exceeds the total used by the comparison (*arrow_N*), the oldest stored values are removed from the vertical arrow history (*trk.vert_arrow_history*) data structure. The history is then updated with the latest vertical direction by comparing the current track's vertical speed to *level_arrow_threshold*. Next, the number of climb, descend, and level arrows are tallied from the history. If any of the tallies are greater than or equal to the parameter *M*, the current vertical arrow (*trk.display_arrow_current*) is set to that direction. If none of the tallies satisfy this condition, the current vertical arrow stays the same.

This algorithm takes as input *trk*. This algorithm updates *trk.display_arrow_current*.

Algorithm 109 VerticalRateArrowUpdate

```

1 function VerticalRateArrowUpdate(trk::TrackFile(p_E-3)
2   const level_arrow_threshold::R = params().display.level_arrow_threshold
3   const M::Z = params().display.arrow_M
4   const N::Z = params().display.arrow_N
5   if (length(trk.vert_arrow_history) >= N)
6     pop!(trk.vert_arrow_history)
7   end
8   mu_dz::R = trk.mu_vert[2]
9   Sigma_dz::R = trk.Sigma_vert[2,2]
10  if (mu_dz > level_arrow_threshold)
11    unshift!(trk.vert_arrow_history, DISPLAY_ARROW_CLIMB)
12  elseif (mu_dz < -level_arrow_threshold)
13    unshift!(trk.vert_arrow_history, DISPLAY_ARROW_DESCEND)
14  else
15    unshift!(trk.vert_arrow_history, DISPLAY_ARROW_LEVEL)
16  end
17  numClimb::Z = 0
18  numDes::Z = 0
19  numLevel::Z = 0
20  for i in 1:length(trk.vert_arrow_history)
21    if (trk.vert_arrow_history[i] == DISPLAY_ARROW_CLIMB)
22      numClimb += 1
23    elseif (trk.vert_arrow_history[i] == DISPLAY_ARROW_DESCEND)
24      numDes += 1
25    else
26      numLevel += 1
27    end
28  end
29  if (numClimb >= M)
30    trk.display_arrow_current = DISPLAY_ARROW_CLIMB
31  elseif (numDes >= M)
32    trk.display_arrow_current = DISPLAY_ARROW_DESCEND
33  elseif (numLevel >= M)
34    trk.display_arrow_current = DISPLAY_ARROW_LEVEL
35  end
36 end

```

Referenced In: AdvanceVerticalTrack(p.44)

2.6.4.2 ADS-B Track File

ADSBTRACKFILEs are added to an STMREPORT through ADDADSBTRACKTOREPORT (Algorithm 110). First, the corresponding Track File is retrieved from memory based on the *rebroadcast* flag, and a new TRMINTRUDERINPUT (i.e., the data structure in the STMREPORT that represents an intruding aircraft) is initialized. Several of the data elements in the TRMINTRUDERINPUT are then populated with the associated data in the ADSBTRACKFILE. This includes the track id (*id*), the reported Mode S address (*modes*), whether the reported Mode S address is an ICAO address (*is_icao*), the presence and operational status of TCAS as dictated by the reported TCAS Operational field of an associated MODESTRACKFILE (*equipage*), the most recent VRC (*vrc*) that has been received from the intruder, the source of the intruder state information (*source*), and the validity of the track data (*valid*) for TA/RA processing by the TRM. Targets are marked as invalid for TA/RA processing under some degraded surveillance conditions: when a target is within the reduced surveillance region; when there is no associated MODESTRACKFILE (as indicated by *passive_only*) and ADS-B-Only TA-Only (AOTO) has been disabled; when ownership is not in a TA mode (as per *own.trm_opmode*); and when the target is NAR while the ownership is above an altitude threshold, *nars_threshold*. Degraded surveillance flags are set in a call to SETINTRUDERDEGRADEDFLAGS.

The latitude and longitude of the ownship is estimated with PROPAGATEOWNSHIPTOTOA (Algorithm 53) to account for ownship motion in the time between the report time and the most recent ownship WGS84 state input. The relative horizontal track is then redefined in a rotated coordinate frame based on the predicted latitude and longitude of the ownship. If necessary, the two tracks contained in the ADSBTRACKFILE are extrapolated to the time of the report by calling the corresponding prediction algorithms. This extrapolation is not performed if the time difference between the last track update and the TOA of the report is less than a defined minimum threshold (*min_extrap_toa_step*). The representation of the target aircraft's vertical state is provided as a set of weighted samples given by *vsamples* and *vweights*. These samples are produced using the mean (*mu_vert*) and covariance (*Sigma_vert*) of the vertical track through a call to ADDALTBIASTANDSAMPLE. The representation of the target aircraft's relative horizontal state is also provided as a set of weighted samples given by *hsamples* and *hweights*. These samples are produced using the mean (*mu_hor*) and covariance (*Sigma_hor*) of the horizontal track that has been converted from metric to English units. The vertical and horizontal samples are provided to SETINTRUDERBELIEFSTATES for assignment within the TRMINTRUDERINPUT. In the last steps, the target designation state gets set by SETTARGETDESIGNATION, the STMDISPLAYSTRUCT vector gets filled by SETDISPLAYDATAPASSIVE, the proximate state of the intruder is determined by PROXIMITYESTIMATION and the TRMINTRUDERINPUT is appended to STMREPORT.

This algorithm takes as input *report*, *id*, *T*, *rebroadcast*, and *passive_only*. This algorithm updates *report.trm_input.intruder*.

Algorithm 110 AddADSBTrackToReport

```

1 function AddADSBTrackToReport(report::StmReport(p.E-1), id::UInt32, T::R, rebroadcast::Bool, passive_only::Bool)
2   const m2ft::R = geoutils.meters_to_feet
3   const min_extrap_toa_step::R = params().surveillance.min_extrap_toa_step
4   const I_xydxdy::Vector{Z} = [1, 3, 2, 4]
5   const nars_threshold::R = params().display.nars_threshold
6   baro_alt::R = BaroAltAtToa(p.78)(T)
7   tgt = target_db[id]
8   if rebroadcast
9     (trk, source) = (tgt.adsr_track, SOURCE_1090ES_ADSR)
10  else
11    (trk, source) = (tgt.adsb_track, SOURCE_1090ES_ADSB)
12  end
13  intruder = TRMIntruderInput(p.E-20)()
14  intruder.id = id
15  intruder.address = trk.modes
16  intruder.is_icao = (trk.non_icao == false)
17  intruder.equipage = EQUIPAGE_MODES
18  modes_trk = ModeSTrackFile(p.E-9)()
19  modes_trk_valid = false
20  if (passive_only == false) && (TrackExists(p.18)(tgt.modes_track))
21    modes_trk = tgt.modes_track
22    modes_trk_valid = true
23    if (modes_trk.ri == RI_FIELD_CASRA) || (modes_trk.ri == RI_FIELD_CASRA_HORZ)
24      intruder.equipage = EQUIPAGE_CASRA
25      intruder.vrc = tgt.coord_data.vrc
26    elseif (modes_trk.ri == RI_FIELD_CASTA)
27      intruder.equipage = EQUIPAGE_CASTA
28    end
29  end
30  intruder.source = source
31  SetCoordination(p.91)(intruder)
32  reduced_surveillance = (modes_trk_valid) && (modes_trk.surv_mode != SURVEILLANCE_REGION_NORMAL)
33  SetIntruderDegradedFlags(p.104)(intruder, tgt, trk, false, reduced_surveillance, passive_only)
34  intruder.valid = true
35  if (own.trm_opmode == OPMODE_STANDBY) ||
36    ((passive_only) && (!own.discrete.auto_on)) ||
37    (reduced_surveillance) ||
38    (!trk.valid_vert) && (baro_alt >= nars_threshold))
39    intruder.valid = false
40  end
41  dt_vert = T - trk.toa_vert
42  dt_hor = T - trk.toa_hor
43  dt_cart = T - modes_trk.toa_cart
44  (mu_vert, Sigma_vert) = PredictVerticalTracker(p.45)(trk.mu_vert, trk.Sigma_vert, dt_vert)
45  if (dt_hor < min_extrap_toa_step)
46    (mu_hor, Sigma_hor) = (trk.mu_hor, trk.Sigma_hor)
47  else
48    (olat, olon, nothing, nothing, nothing, nothing, nothing) = PropagateOwnershipToToa(p.56)(T)
49    (mu_hor_s, Sigma_hor_s) = RedefineEstimateInRotatedFrame(p.58)(trk, olat, olon)
50    (mu_hor, Sigma_hor) = PredictADSBTracker(p.68)(mu_hor_s, Sigma_hor_s, dt_hor)
51  end
52  (mu_cart, Sigma_cart) = PredictCartesianTracker(p.33)(modes_trk.mu_cart, modes_trk.Sigma_cart, dt_cart)
53  (vsamples, vweights) = AddAltBiasAndSample(p.105)(mu_vert, Sigma_vert, trk.valid_vert)
54  (mu_xydxdy, Sigma_xydxdy) = (mu_hor[I_xydxdy] * m2ft, Sigma_hor[I_xydxdy, I_xydxdy] * m2ft^2)
55  (hsamples, hweights) = SigmaPointSample(p.28)(mu_xydxdy, Sigma_xydxdy, 2.0)
56  SetIntruderBeliefStates(p.103)(intruder, vsamples, vweights, hsamples, hweights)
57  r_ground::R = hypot(mu_hor[1], mu_hor[3]) * m2ft
58  SetTargetDesignation(p.125)(intruder, T, mu_vert[1], r_ground, tgt.designation_state)
59  SetDisplayDataPassive(p.112)(report, intruder, mu_vert, mu_hor * m2ft, mu_cart, trk.valid_vert, modes_trk,
60                                valid_cart, T, trk.display_arrow_current)
61  ProximityEstimation(p.147)(intruder, baro_alt, mu_vert[1], r_ground, trk.valid_vert, true)
62 push!(report.trm_input.intruder, intruder)
62 end

```

Referenced In: AddTracksToReport(p.98)

ADSBQUALITYCHECK (Algorithm 111) checks both *tgt.adsb_qual_history* and *tgt.passive_only_adsb_qual_history* using an M of N test. Both *tgt.adsb_qual_history* and *tgt.passive_only_adsb_qual_history* are updated with UPDATEADSBQUALITYHISTORY (Algorithm 61), which checks the quality indicators reported by the ADS-B against minimum values defined in the parameters. If *N* updates have been added to *tgt.adsb_qual_history* and the active validation state is valid, then the target is transitioned to an invalid active validation state. Also, if the active validation state is provisional, the target is transitioned to an invalid active validation state. The input parameter *passive_only* determines which history's M of N test is returned from each call to the algorithm. If *M* of the tests passed in the history, ADSBQUALITYCHECK (Algorithm 111) returns TRUE. Otherwise, it returns FALSE.

This algorithm takes as input *tgt* and *passive_only*. This algorithm returns either *passive_only-quality_good* or *adsb_quality_good*. This algorithm may update *tgt.av_state*.

Algorithm 111 AdsbsQualityCheck

```

1 function AdsbsQualityCheck(tgt::Target(p. E-14), passive_only::Bool)
2   const M::UInt32 = params().surveillance.report_generation.min_adsb_quality.M
3   const N::UInt32 = params().surveillance.report_generation.min_adsb_quality.N
4   adsb_quality_good::Bool = false
5   passive_only_quality_good::Bool = false
6   if (sum(tgt.adsb_qual_history) >= M)
7     adsb_quality_good = true
8   elseif (length(tgt.adsb_qual_history) == N) && ((tgt.av_state == AV_STATE_VALID) || (tgt.av_state ==
9     AV_STATE_PROVISIONAL))
10    tgt.av_state = AV_STATE_INVALID
11  end
12  if (sum(tgt.passive_only_adsb_qual_history) >= M)
13    passive_only_quality_good = true
14  end
15  if (passive_only == true)
16    return passive_only_quality_good::Bool
17  else
18    return adsb_quality_good::Bool
19 end

```

Referenced In: AddTracksToReport ^(p. 98)
--

SETDISPLAYDATAPASSIVE (Algorithm 112) populates the STMREPORT with fields needed to properly display each intruder, similar to SETDISPLAYDATAACTIVE. These fields include the reported Mode S address (*mode_s*), an indication as to whether the address is ICAO compliant (*is_icao*), the track id of the target (*id*), the vertical state validity (*alt_reporting*), the bearing state validity (*bearing_valid*), the direction of the intruder relative vertical velocity (*arrow*), the relative altitude of the intruder (*z_rel*), the bearing relative to ownship (*Chi_rel*), the ground range to the intruder (*r_ground*).

Distinct from SETDISPLAYDATAACTIVE, SETDISPLAYDATAPASSIVE receives as input both the horizontal track of the passive track file (*mu_hor*) as well as the Cartesian track of a correlated active track file (*mu_cart*). The bearing information derived from the passive horizontal track is provided to the display whenever the ownship heading state (*own.heading_state*) is nominal. In the event of a degraded or invalid ownship heading state, bearing information from a valid correlated active Cartesian track file is displayed. This prevents mixing bearing data from different sources on the display under degraded conditions, where each source may be impacted differently by the degraded

heading mode. If no valid correlating active track is available for a passive target under degraded ownship heading conditions, the target is displayed as bearingless.

This algorithm takes as input *report*, *intruder*, *mu_vert*, *mu_hor*, *mu_cart*, *valid_vert*, *valid_cart*, *T*, and *display_arrow_current*. This algorithm updates *intruder.stm_display* and appends that updated information to *report.display*.

Algorithm 112 SetDisplayDataPassive

```

1 function SetDisplayDataPassive(report::StmReport(p.E-1), intruder::TRMIntruderInput(p.E-20), mu_vert::Vector{R},
2   mu_hor::Vector{R}, mu_cart::Vector{R}, valid_vert::Bool, valid_cart::Bool, T::R, display_arrow_current::
3   Z)
4   display::StmDisplayStruct = StmDisplayStruct(p.E-1)()
5   display.mode_s = intruder.address
6   display.is_icao = intruder.is_icao
7   display.id = intruder.id
8   display.z_rel = BaroAltAtToa(p.78)(T) - mu_vert[1]
9   display.alt_reporting = valid_vert
10  own_heading::R = HeadingAtToa(p.85)(T)
11  if (own.heading_state == OWN_HEADING_NOMINAL)
12    display.Chi_rel = WrapToPi(p.H-3)(atan2(mu_hor[1], mu_hor[3]) - own_heading)
13    display.bearing_valid = true
14  elseif (valid_cart)
15    display.Chi_rel = WrapToPi(p.H-3)(atan2(mu_cart[1], mu_cart[2]) - own_heading)
16    display.bearing_valid = true
17  else
18    display.Chi_rel = 0.0
19    display.bearing_valid = false
20  end
21  display.r_ground = hypot(mu_hor[1], mu_hor[3])
22  if (valid_vert)
23    display.arrow = display_arrow_current
24  else
25    display.arrow = DISPLAY_ARROW_LEVEL
26  end
27  intruder.stm_display = display
28  push!(report.display, display)
29 end
```

Referenced In: AddADSBTrackToReport(p.110)

2.6.4.3 Mode C Track File

ADDMODECTRACKTOREPORT (Algorithm 113) defines the process through which the data in a MODECTRACKFILE is added to an STMREPORT. First, the corresponding Track File is retrieved from memory. If the Track File is an image track, then the algorithm will look through all of the Mode C Track Files for the given target and select a non-image Track File. (An image track is a multipath Mode C track that is produced when the interrogation and/or the response bounces off the ground and forms a track.) Once a non-image Track File is selected, a new TRMINTRUDERINPUT (i.e., the data structure in the STMREPORT that represents an intruding aircraft) is initialized. If the selected Track File has been determined to be on the ground (indicated by *trk.on_ground*), it is not added to the report. The ownship barometric altitude for the time of the report is retrieved for later use (*baro_alt*). Several of the data elements in the TRMINTRUDERINPUT are then populated with the associated data in the MODECTRACKFILE, including the track id (*id*), an indication of no reported Mode S address (represented as zero), an indication that the address is not an ICAO address (*is_icao*), the equipage type of ATCRBS (*equipage*), the source of the intruder state information (*source*), and the most recent VRC (*vrc*) that has been received from the intruder.

The validity of the track data for TA/RA processing (*valid*) is based on the validity of the vertical track (*trk.valid_vert*) and the ownship altitude threshold for NAR intruders (*nars_threshold*), as well as the ownship TA mode (*own.trm_opmode*). Degraded surveillance flags are set in a call to SETINTRUDERDEGRADEDFLAGS. Lastly, a call to SETCOORDINATION will set the coordination message to *COORDINATION_NONE* for a Mode C intruder.

If necessary, the three tracks contained in the MODECTRACKFILE are extrapolated to the time of the report by calling the corresponding prediction algorithms. This extrapolation is not performed if the time difference between the last track update and the TOA of the report is less than a defined minimum threshold (*min_extrap_toa_step*). The representation of the target aircraft's vertical state is provided as a set of weighted samples given by *vsamples* and *vweights*. These samples are produced using the mean (*mu_vert*) and covariance (*Sigma_vert*) of the vertical track through a call to ADDALTBIASANDSAMPLE. The representation of the target aircraft's relative horizontal state is also provided as a set of weighted samples given by *hsamples* and *hweights*. These samples are produced through a call to COMBINEANDSAMPLE, which merges the Cartesian Track and the Range Track into a single horizontal representation. The vertical and horizontal samples are provided to SETINTRUDERBELIEFSTATES for assignment within the TRMINTRUDERINPUT. In the last steps, the target designation state gets set by SETTARGETDESIGNATION, the STMDISPLAYSTRUCT vector gets filled by SETDISPLAYDATAACTIVE, the proximate state of the intruder is determined by PROXIMITYESTIMATION and the TRMINTRUDERINPUT is appended to STMREPORT. SETTARGETDESIGNATION is called so the designation validity and status settings will be set appropriately in the TRMINTRUDERINPUT output. These settings will prevent an undesignated Target with only a Mode C track from being designated. Also, if the Target was originally designated when a Mode S or ADS-B track was associated with the Target, the timer remains active until either a Mode S or an ADS-B track is correlated to this Target or the designation times out, whichever comes first. During this interval, SETTARGETDESIGNATION must be called to update the internal designation state history, which includes incrementing the timer values and updating the ground range history.

This algorithm takes as input *report*, *id*, and *T*. This algorithm appends updated information to *report.trm_input.intruder*.

Algorithm 113 AddModeCTrackToReport

```

1 function AddModeCTrackToReport(report::StmReport(p. E-1), id::UInt32, T::R)
2   const nars_threshold::R = params().display.nars_threshold
3   tgt = target_db[id]
4   trk = tgt.modec_tracks[1]
5   if (IsImageTrack(p. B-8)(trk))
6     for j in 1:length(tgt.modec_tracks)
7       if (IsImageTrack(p. B-8)(tgt.modec_tracks[j]) == false) && (tgt.modec_tracks[j].valid_rng)
8         trk = tgt.modec_tracks[j]
9         break
10      end
11    end
12  end
13  if (trk.on_ground)
14    return
15  end
16  baro_alt::R = BaroAltAtToa(p. 78)(T)
17  intruder = TRMIntruderInput(p. E-20)()
18  intruder.id = id
19  intruder.address = 0
20  intruder.is_icao = false
21  intruder.equipage = EQUIPAGE_ATCRBS
22  intruder.vrc = tgt.coord_data.vrc
23  intruder.source = SOURCE_MODEC
24  SetCoordination(p. 91)(intruder)
25  SetIntruderDegradedFlags(p. 104)(intruder, tgt, trk, true, false, false)
26  intruder.valid = true
27  if (own.trm_opmode == OPMODE_STANDBY) ||
28    (!trk.valid_vert) && (baro_alt >= nars_threshold))
29    intruder.valid = false
30  end
31  dt_vert = T - trk.toa_vert
32  dt_range = T - trk.toa_rng
33  dt_cart = T - trk.toa_cart
34  (mu_vert, Sigma_vert) = PredictVerticalTracker(p. 45)(trk.mu_vert, trk.Sigma_vert, dt_vert)
35  (mu_rng, Sigma_rng) = PredictRangeTracker(p. 39)(trk.mu_rng, trk.Sigma_rng, dt_range)
36  (mu_cart, Sigma_cart) = PredictCartesianTracker(p. 33)(trk.mu_cart, trk.Sigma_cart, dt_cart)
37  (vsamples, vweights) = AddAltBiasAndSample(p. 105)(mu_vert, Sigma_vert, trk.valid_vert)
38  (hsamples, hweights) = CombineAndSample(p. 105)(mu_rng[1:2], Sigma_rng[1:2,1:2], mu_cart, Sigma_cart, trk.
39    valid_cart)
40  SetIntruderBeliefStates(p. 103)(intruder, vsamples, vweights, hsamples, hweights)
41  SetTargetDesignation(p. 125)(intruder, T, mu_vert[1], mu_rng[1], tgt.designation_state)
42  SetDisplayDataActive(p. 107)(report, intruder, mu_vert, mu_cart, mu_rng, trk.valid_vert, trk.valid_cart, T,
43    trk.display_arrow_current)
44  ProximityEstimation(p. 147)(intruder, baro_alt, mu_vert[1], mu_rng[1], trk.valid_vert, trk.valid_cart)
45  push!(report.trm_input.intruder, intruder)
46 end

```

Referenced In: AddTracksToReport(<i>p. 98</i>)

2.6.5 Track Deletion

A Track File is deleted from memory if it has not been updated for a period of time that depends on the surveillance source and the expected update rate. The function that handles the deletion of deprecated tracks is REMOVESTALETRACKS (Algorithm 114). This algorithm takes as input the ID, *id*, of the TARGET to be examined and a time value, *T*, needed for comparison to determine if a Track File should be deleted. The coast limits for each surveillance region are read in from the parameters file. REMOVESTALETRACKS begins with determining if the TARGET contains a MODESTRACKFILE with a call to TRACKEXISTS. If a MODESTRACKFILE exists, the appropriate maximum coast period is determined given the surveillance mode of the MODESTRACKFILE. The REMOVESTALEMODESTRACK (Algorithm 115) algorithm is then called on the MODESTRACK-

FILE.

Next, the algorithm checks for the existence of a ADSBTRACKFILE. If both an ADSBTRACKFILE and a MODESTRACKFILE exist within the TARGET the appropriate maximum coast period for the ADSBTRACKFILE is determined given the surveillance mode of the MODESTRACKFILE. If no MODESTRACKFILE exists, the coast limit for the normal surveillance region is used. A call is made to REMOVESTALEADSBTRACK (Algorithm 116) to complete the process of removing stale ADSBTRACKFILE or component tracks.

Calls are then made to REMOVESTALEMODECTRACKS (Algorithm 117) and REMOVESTALEADSRSRTRACK (Algorithm 118). Besides tracks, stale coordination data is deleted in this algorithm; COORDINATIONTIMEOUTCHECK provides that functionality.

At the end of this process, if no Track Files remain for a TARGET and the TARGET is not designated, the TARGET is deleted from memory and the algorithm returns as FALSE. If Track Files do remain for the TARGET or the TARGET is designated, the algorithm returns as TRUE.

This algorithm takes as input *id*, and *T*. This algorithm returns TRUE or FALSE.

Algorithm 114 RemoveStaleTracks

```

1 function RemoveStaleTracks(id::UInt32, T::R)
2   const max_normal_coasts::Z = params().surveillance.max_normal_coasts
3   const max_reduced_coasts::Z = params().surveillance.max_reduced_coasts
4   tgt = target_db[id]
5   if (TrackExists(p.18)(tgt.modes_track))
6     if (tgt.modes_track.surv_mode != SURVEILLANCE_REGION_NORMAL)
7       modes_max_coasts = max_reduced_coasts
8     else
9       modes_max_coasts = max_normal_coasts
10    end
11    RemoveStaleModeSTrack(p.116)(tgt, T, modes_max_coasts)
12  end
13  if (TrackExists(p.18)(tgt.adsb_track))
14    if (TrackExists(p.18)(tgt.modes_track)) && (tgt.modes_track.surv_mode != SURVEILLANCE_REGION_NORMAL)
15      adsb_max_coasts = max_reduced_coasts
16    else
17      adsb_max_coasts = max_normal_coasts
18    end
19    RemoveStaleADSBTrack(p.117)(tgt, T, adsb_max_coasts)
20  end
21  if (TrackExists(p.18)(tgt.modec_tracks))
22    RemoveStaleModeCTracks(p.118)(tgt, T, max_normal_coasts)
23  end
24  if (TrackExists(p.18)(tgt.adsr_track))
25    RemoveStaleADSRTTrack(p.118)(tgt, T, max_normal_coasts)
26  end
27  CoordinationTimeoutCheck(p.89)(tgt, T)
28  if (TargetIsEmpty(p.119)(tgt))
29    delete!(target_db, id)
30    return false
31  else
32    return true
33  end
34 end
```

Referenced In: AddTracksToReport(p. 98)
--

REMOVESTALEMODESTRACK (Algorithm 115) deletes a MODESTRACKFILE if it has not been

updated within the specified maximum coast limit for the given surveillance mode. If the MODESTRACKFILE is deleted, the received coordination *vrc* of the Target will be deleted from the vertical intent array (*own.received_vrcs*) through a call to DELETEINTENT as long as no other threat is simultaneously providing the same *vrc*.

The algorithm will flag as invalid any one of the individual tracks (range, Cartesian, or vertical) within the MODESTRACKFILE if that track has not received a valid update within the prescribed coast limit. This prevents any one track within the Track File from being extensively coasted while the remaining tracks receive valid updates. The Target's ACTIVEVALIDATIONHISTORY is reset whenever the MODESTRACKFILE is to be deleted. This ensures that any future MODESTRACKFILES correlated to this Target do not recycle previous *tgt.av_history* information, and instead a correlated ADSBTRACKFILE must validate anew against the MODESTRACKFILE. The Target's ACTIVEVALIDATIONHISTORY is also reset whenever the Mode-S range track or vertical track are invalidated, thus ensuring that a correlated ADSBTRACKFILE is not considered validated (AV_STATE_VALID) when the the MODESTRACKFILE has either invalid range or vertical tracks. Resetting the ACTIVEVALIDATIONHISTORY involves clearing *tgt.av_history* and the changing *tgt.av_state* to AV_STATE_NOT_VALIDATED if it had not been AV_STATE_NOT_VALIDATED.

This algorithm takes as input *tgt*, *T*, and *max_coast*. This algorithm updates several fields within the *tgt*, including *tgt.modes_track*, *tgt.av_state*, *tgt.av_history*, and *tgt.coord_data*.

Algorithm 115 RemoveStaleModeSTrack

```

1 function RemoveStaleModeSTrack(tgt::Target(p. E-14), T::R, max_coasts::Z)
2   dt = T - tgt.modes_track.toa
3   dt_vert = T - tgt.modes_track.toa_vert
4   dt_cart = T - tgt.modes_track.toa_cart
5   dt_rng = T - tgt.modes_track.toa_rng
6   reset_validation_history::Bool = false
7   if (dt >= max_coasts)
8     tgt.modes_track = nothing
9     reset_validation_history = true
10    cancel_index::Z = tgt.coord_data.vrc
11    tgt.coord_data = ReceivedCoordinationData(p. E-13)()
12    DeleteIntent(p. 88)(cancel_index)
13  else
14    if (dt_vert >= max_coasts)
15      tgt.modes_track.valid_vert = false
16      tgt.modes_track.updates_vert = 0
17      reset_validation_history = true
18    end
19    if (dt_cart >= max_coasts)
20      tgt.modes_track.valid_cart = false
21      tgt.modes_track.updates_cart = 0
22    end
23    if (dt_rng >= max_coasts)
24      tgt.modes_track.valid_rng = false
25      tgt.modes_track.updates_rng = 0
26      reset_validation_history = true
27    end
28  end
29  if (reset_validation_history)
30    tgt.av_history = ActiveValidationHistory(p. E-3)()
31    if (tgt.av_state != AV_STATE_INVALID)
32      tgt.av_state = AV_STATE_NOT_VALIDATED
33    end
34  end
35 end
```

Referenced In: RemoveStaleTracks(p. 115)
--

REMOVESTALEADSBTRACK (Algorithm 116) deletes an ADSBTRACKFILE if the horizontal track has not been updated with a position update within the specified maximum coast limit for the given surveillance mode. The vertical track of an ADSBTRACKFILE may be invalidated independently if it has reached the coast limit without receiving a valid update. If an ADSBTRACKFILE is removed, the ACTIVEVALIDATIONHISTORY for that target will be reset and the active validation state reset to *AV_STATE_NOT_VALIDATED*, but only if Active Validation had not failed.

This algorithm takes as input *tgt*, *T*, and *max_coast*. This algorithm updates many fields within *tgt*, including *tgt.adsb_track*, *tgt.adsb_qual_history*, *tgt.av_state*, and *tgt.av_history*.

Algorithm 116 RemoveStaleADSBTrack

```

1 function RemoveStaleADSBTrack(tgt::Target(p. E-14), T::R, max_coasts::Z)
2   dt = T - tgt.adsb_track.toa
3   dt_vert = T - tgt.adsb_track.toa_vert
4   dt_pos = T - tgt.adsb_track.toa_pos_update
5   reset_validation_history = false
6   if (dt >= max_coasts) || (dt_pos >= max_coasts)
7     tgt.adsb_track = nothing
8     reset_validation_history = true
9     tgt.adsb_qual_history = Array(Bool,0)
10    else
11      if (dt_vert >= max_coasts)
12        tgt.adsb_track.valid_vert = false
13        tgt.adsb_track.updates_vert = 0
14        reset_validation_history = true
15      end
16    end
17    if reset_validation_history
18      tgt.av_history = ActiveValidationHistory(p. E-3)()
19      if (tgt.av_state != AV_STATE_INVALID)
20        tgt.av_state = AV_STATE_NOT_VALIDATED
21      end
22    end
23 end
```

Referenced In: RemoveStaleTracks(p. 115)

REMOVESTALEMODECTRACKS (Algorithm 117) iterates through all MODECTRACKFILES and will delete a Track File if it has not been updated within the appropriate coast limit, or if the vertical track reaches the coast limit. The Cartesian and range tracks of a MODECTRACKFILE may be independently invalidated if they reach the coast limit. The vertical track does not independently coast out. Each MODECTRACKFILE is checked to see whether it is an image track (using ISIMAGETRACK). If all of the MODECTRACKFILES correlated to the Target are marked as image tracks, the first MODECTRACKFILE is selected as the non-image track.

This algorithm takes as input *tgt*, *T*, and *max_coast*. This algorithm updates *tgt.modec_tracks*.

Algorithm 117 RemoveStaleModeCTracks

```

1 function RemoveStaleModeCTracks(tgt::Target(p.E-14), T::R, max_coasts::Z)
2   const max_duplicate_coasts::Z = params().surveillance.max_duplicate_coasts
3   only_image_tracks = true
4   for i in reverse(1:length(tgt.modec_tracks))
5     dt = T - tgt.modec_tracks[i].toa
6     dt_vert = T - tgt.modec_tracks[i].toa_vert
7     dt_cart = T - tgt.modec_tracks[i].toa_cart
8     dt_rng = T - tgt.modec_tracks[i].toa_rng
9     if (dt >= max_coasts) || (dt_vert >= max_coasts)
10      splice!(tgt.modec_tracks, i)
11    elseif (length(tgt.modec_tracks) > 1) && (dt >= max_duplicate_coasts)
12      splice!(tgt.modec_tracks, i)
13    else
14      if (!IsImageTrack(p.B-8)(tgt.modec_tracks[i]))
15        only_image_tracks = false
16      end
17      if (dt_cart >= max_coasts)
18        tgt.modec_tracks[i].valid_cart = false
19        tgt.modec_tracks[i].updates_cart = 0
20      end
21      if (dt_rng >= max_coasts)
22        tgt.modec_tracks[i].valid_rng = false
23        tgt.modec_tracks[i].updates_rng = 0
24      end
25    end
26  end
27  if (only_image_tracks) && (TrackExists(p.18)(tgt.modec_tracks))
28    tgt.modec_tracks[1].is_image = false
29  end
30 end

```

Referenced In: RemoveStaleTracks(p.115)

REMOVESTALEADSRTTRACK (Algorithm 118) will delete an ADS-R Track File if the horizontal track has not been updated with a position update within the maximum coast limit. The vertical track within an ADS-R Track File may be flagged as invalid upon reaching the coast limit without a valid update.

This algorithm takes as input *tgt*, *T*, and *max_coast*. This algorithm updates *tgt.adsr_track*.

Algorithm 118 RemoveStaleADSRTrack

```

1 function RemoveStaleADSRTrack(tgt::Target(p.E-14), T::R, max_coasts::Z)
2   dt = T - tgt.adsr_track.toa
3   dt_vert = T - tgt.adsr_track.toa_vert
4   dt_hor = T - tgt.adsr_track.toa_hor
5   if (dt >= max_coasts) || (dt_hor >= max_coasts)
6     tgt.adsr_track = nothing
7   else
8     if (dt_vert >= max_coasts)
9       tgt.adsr_track.valid_vert = false
10      tgt.adsr_track.updates_vert = 0
11    end
12  end
13 end

```

Referenced In: RemoveStaleTracks(p.115)

TARGETISEMPTY (Algorithm 119) takes as input *tgt* and updates *tgt.modec_tracks*. This algorithm returns TRUE if the Target, *tgt*, is empty of any Mode S, Mode C, ADS-B, and ADS-R Track Files, and is not designated. Otherwise, FALSE is returned.

This algorithm takes as input *tgt* and returns a boolean.

Algorithm 119 TargetIsEmpty

```

1 function TargetIsEmpty( tgt::Target(p. E-14) )
2     return ( !TrackExists(p.18)(tgt.modes_track) ) &&
3         ( !TrackExists(p.18)(tgt.adsb_track) ) &&
4         ( !TrackExists(p.18)(tgt.modec_tracks) ) &&
5         ( !TrackExists(p.18)(tgt.adsr_track) ) &&
6         (tgt.designation_state.is_designated == false) ::Bool
7 end
```

Referenced In: MergeTargets(p. C-15), RemoveStaleTracks(p. 115)
--

2.6.6 Active Validation

The STM will provide the TRM (via the STMREPORT) with the estimated relative position and velocity of a target aircraft that has been derived from ADS-B surveillance data, but only if the ADS-B Track passes a number of checks. One such check is Active Validation, which compares the ADS-B track with the Mode S Track on the same target. If the two tracks do not match within a specified threshold, the ADS-B Track is considered invalid and the Mode S Track is provided to the TRM.

The active validation state of a TARGET is stored in its *av_state* field. The available state values are defined in CONSTANTS (Appendix D) as follows:

- **NOT_VALIDATED:** The target's active validation state has not yet been determined. This state can transition to all other states.
- **PROVISIONAL:** The target has passed an initial M of N test and is likely to become fully validated. A target in this state transitions to VALID once the full M of N tests have passed, or back to NOT_VALIDATED if any test fails. No transition is available directly to INVALID.
- **VALID:** The target has been successfully validated. This state can transition to INVALID if later validations fail the M of N test, or to NOT_VALIDATED if a track file coasts out.
- **INVALID:** The target has failed more than (N-M) of the previous N tests. Once marked as INVALID, a target cannot transition back to VALID.

Active Validation is a two step process. The first step (ACTIVEVALIDATIONUPDATE (Algorithm 120)) updates and stores the data structures that contain the track elements that are relevant for the Active Validation comparison. The second step (ACTIVEVALIDATIONCHECK (Algorithm 121)) compares the stored data structures and determines the validity of the ADS-B Track. This two step process is necessary as Active Validation utilizes track data at the time of a Mode S Track update, while the comparison itself is not necessary until the STMREPORT is generated.

Within ACTIVEVALIDATIONUPDATE, active validation will not be performed if there is no MODESTRACKFILE for the given target, no ADSBTRACKFILE for the given target, no established range estimate for the Mode S Track, or no established velocity estimate for the ADS-B Track. Next, if

the number of past elements exceeds the total used by the comparison (N), the oldest stored values are removed from the active validation history (v) data structure.

The active portion of the validation history data structure is then updated with the MODESTRACKFILE. Before the validation history data structure is updated, the MODESTRACKFILE is extrapolated to the requested time, t . If the vertical track of the MODESTRACKFILE is invalid, a value of zero is substituted for the vertical dimension. Next, the passive portion of the validation history data structure is updated with the Target's ADSBTRACKFILE, which is extrapolated to the requested time, t . The extrapolation of the horizontal estimate involves three steps. First, the latitude and longitude of the ownship is estimated with PROPAGATEOWNSHIPTOA (Algorithm 53) to account for ownship motion in the time between the active validation update and the most recent ownship WGS84 state input. Second, the relative horizontal track is redefined in a rotated coordinate frame based on the predicted latitude and longitude of the ownship. Third and finally, the rotated estimate is predicted forward. Like in the history of the active track, the altitude is recorded as zero if the vertical track is invalid.

This algorithm takes as input tgt and t . This algorithm returns the updated ACTIVEVALIDATIONHISTORY v .

Algorithm 120 ActiveValidationUpdate

```

1 function ActiveValidationUpdate(tgt::Target(p. E-14), t::R
2   const m2ft::R = geoutils.meters_to_feet
3   const N::Z = params().surveillance.av_threshold.N
4   v = tgt.av_history
5   if (!TrackExists(tgt.modes_track)) || (!TrackExists(p. 18)(tgt.adsb_track)) || (!tgt.modes_track.valid_rng)
6     return v
7   end
8   if (length(v.active.range) >= N)
9     pop!(<v.active.range)
10    pop!(<v.active.altitude)
11    pop!(<v.passive.range)
12    pop!(<v.passive.altitude)
13  end
14  modes_trk = tgt.modes_track
15  dt_vert = t - modes_trk.toa_vert
16  dt_range = t - modes_trk.toa_rng
17  (mu_vert,Sigma_vert) = PredictVerticalTracker(p. 45)(modes_trk.mu_vert,modes_trk.Sigma_vert,dt_vert)
18  (mu_rng,Sigma_rng) = PredictRangeTracker(p. 39)(modes_trk.mu_rng, modes_trk.Sigma_rng, dt_range)
19  unshift!(<v.active.range, mu_rng[1])
20  if (modes_trk.valid_vert)
21    unshift!(<v.active.altitude, mu_vert[1])
22  else
23    unshift!(<v.active.altitude, 0.0)
24  end
25  adsb_trk = tgt.adsb_track
26  dt_pos = t - adsb_trk.toa_hor
27  (olat, olon, nothing, nothing, nothing, nothing, nothing) = PropagateOwnershipToToa(p. 56)(t)
28  (mu_hor_s, Sigma_hor_s) = RedefineEstimateInRotatedFrame(p. 58)(adsb_trk, olat, olon)
29  (mu_s, Sigma_s) = PredictADSBTracker(p. 68)(mu_hor_s, Sigma_hor_s, dt_pos)
30  unshift!(<v.passive.range, hypot(mu_s[1],mu_s[3])*m2ft)
31  if (adsb_trk.valid_vert)
32    dt_vert = t - adsb_trk.toa_vert
33    (mu_s, Sigma_s) = PredictVerticalTracker(p. 45)(adsb_trk.mu_vert, adsb_trk.Sigma_vert, dt_vert)
34    unshift!(<v.passive.altitude, mu_s[1])
35  else
36    unshift!(<v.passive.altitude, 0.0)
37  end
38  return v::ActiveValidationHistory(p. E-3)
39 end

```

Referenced In: AdvanceModeSTrackFile(p. 17), AddModeSTrackToDB(p. 20)

The second step of the active validation process, ACTIVEVALIDATIONCHECK, first reads in from the parameters file the values that are used to determine ADS-B Track validity. These parameters include:

- N - the number of past track estimates used in the comparison.
- M - out of N , the number of track estimates that must be within the validation thresholds for the ADS-B Track to be considered valid.
- r - the range threshold.
- a - the altitude threshold.

The active validation history data structure (*av*) that is input to the algorithm is broken into its various components: active range (*ar*), active altitude (*aa*), passive range (*pr*), and passive altitude (*pa*). Each of these components is a vector of equivalent length that is no longer than N elements long. This vector length is iterated over and a validation comparison is performed at each iteration.

The validation comparison checks if the range and altitude of the passive track are within the defined thresholds (r and a respectively) of the active track, and if so, the running count $test$ is incremented by 1. Once all comparisons are made, if the running count is greater than or equal to the parameter M , the ADS-B Track is marked valid: the output, $valid$, is set to TRUE and the active validation state of the target is set to AV_STATE_VALID . If, however, the active validation state was invalid, $AV_STATE_INVALID$, then the state is not updated; a track that was identified as invalid is never again marked valid. A target is allowed to provisionally pass validation (entering $AV_STATE_PROVISIONAL$) prior to completing the full N number of active validation checks by instead satisfying an $init_M$ of an $init_N$ number of validation checks. The provisional validation state provides track source hysteresis during transitions between Hybrid and ACAS X surveillance regions. The returned boolean value is used along with other checks in ADDTRACKSTOREPORT to determine if the ADS-B track can be provided to the TRM for threat resolution.

This algorithm takes as input tgt . This algorithm updates $tgt.av_state$ and returns the Boolean $valid$.

Algorithm 121 ActiveValidationCheck

```

1 function ActiveValidationCheck(tgt::Target(p. E-14))
2   const M::Z = params().surveillance.av_threshold.M
3   const init_N::Z = params().surveillance.av_threshold.init_N
4   const init_M::Z = params().surveillance.av_threshold.init_M
5   const r::R = params().surveillance.av_threshold.r
6   const a::R = params().surveillance.av_threshold.a
7   av = tgt.av_history
8   ar = av.active.range
9   aa = av.active.altitude
10  pr = av.passive.range
11  pa = av.passive.altitude
12  test = 0
13  for i in 1:length(ar)
14    if (abs(ar[i] - pr[i]) <= r) && (abs(aa[i] - pa[i]) <= a)
15      test += 1
16    end
17  end
18  valid::Bool = false
19  if (test >= M)
20    if (tgt.av_state != AV_STATE_INVALID)
21      tgt.av_state = AV_STATE_VALID
22      valid = true
23    end
24  elseif (length(ar) == init_N) && (test == init_M)
25    if (tgt.av_state != AV_STATE_INVALID)
26      tgt.av_state = AV_STATE_PROVISIONAL
27      valid = true
28    end
29  else
30    if (tgt.av_state == AV_STATE_VALID)
31      tgt.av_state = AV_STATE_INVALID
32    elseif (tgt.av_state == AV_STATE_PROVISIONAL)
33      tgt.av_state = AV_STATE_NOT_VALIDATED
34    end
35  end
36  return valid::Bool
37 end
```

Referenced In: AddTracksToReport(<small>p. 98</small>)

2.6.7 Target Designation

ACAS X traffic designation support is provided via the Aircraft Surveillance Applications (ASA) System described in DO-317B [15]. The Aircraft Surveillance and Separation Assurance Processing (ASSAP) function in the ASA System provides the interface to the ACAS X system. Two Xo designation modes are currently defined for ACAS: Designated No Alerts (DNA) and CSPO-3000. The ASSAP function places requirements on the types of outputs from ACAS X. The ASSAP function will allow designation to Designated No Alerts (DNA) for any traffic that is flagged as valid for DNA designation (*intruder.xo_valid[Xo_IDX_DNA]* is TRUE). The ASSAP function will also allow designation to CSPO-3000 for any traffic that is flagged as valid for CSPO-3000 designation (*intruder.xo_valid[Xo_IDX_CSPO3k]* is TRUE). It will undesignate any targets that are marked as invalid. Consequently, whenever the Target should remain designated, the selected designation must be marked as valid.

The algorithms in this section describe the management of designation state DESIGNATIONSTATE and preparation of designation output for the TRM.

SETTARGETDESIGNATION (Algorithm 122) populates the fields in TRMINTRUDERINPUT associated with ACAS Xo processing. Those fields are the requested designation (*designated_mode*), the desired protection mode (*protection_mode*), the desired Designated No Alerts (DNA) state (*dna*), the validity for each Xo mode (*xo_valid*), and the Xo status (*xo_status*).

It first uses UPDATETARGETDESIGNATIONVALIDITY to determine whether Xo processing is valid for the Target and to update the target designation invalid timers. In certain cases, UPDATETARGETDESIGNATIONVALIDITY will automatically undesignate the Target. The Target DESIGNATIONSTATE (*designation_state*) is updated as a result. The *active_dna* flag is returned to reflect a change in the active DNA state due to automatic undesignation.

Next, SETTARGETDESIGNATION determines which type of Xo processing should be used based on validity, the timers, and the resolution advisory (RA) state. For each Xo mode, there is a stored active setting and a stored pending setting. These settings are stored in the Target DESIGNATIONSTATE. The active fields (*designation_state.active_dna* and *designation_state.active_protection_mode*) store the target designation used for TRM processing of this Target on the previous processing cycle. The pending fields (*designation_state.pending_dna* and *designation_state.pending_protection_mode*) store the requested target designation for this Target. The pending settings are originally stored in RECEIVETARGETDESIGNATION but are updated in UPDATETARGETDESIGNATIONVALIDITY whenever the Target is automatically undesignated. Otherwise, SETTARGETDESIGNATION leaves the stored active and pending settings unmodified. Instead, it uses local variables (*active_dna*, *pending_dna*, and *pending_protection_mode*) to configure the target designation setting to be provided to the TRM. The bulk of SETTARGETDESIGNATION involves setting those local variables and that will be described next.

The local value of pending DNA (*pending_dna*) is initialized to the stored setting of pending DNA (*designation_state.pending_dna*). The final value of *pending_dna* will be stored in *intruder.dna* in the TRMINTRUDERINPUT output. A pending designation for DNA is not honored (is suspended) when the Target is not valid for TRM processing or there are multiple threats and this Target is one of the threats. A threat is a Target for which the TRM has generated an RA. In those cases, DNA is deactivated (*pending_dna* is set to FALSE) and *designation_state.status* is set to reflect the reason for the suspension. When the Target is one of multiple threats, *active_dna* is set to FALSE to reflect the temporary suspension of DNA processing.

SETTARGETDESIGNATION next determines if CSPO-3000 can be used, independent of the status of



DNA (*ok_cspo3000*). First, the local value of pending protection mode (*pending_protection_mode*) is initialized. If the pending designation for DNA was honored, *pending_protection_mode* is initialized to the stored setting of pending protection mode (*designation_state.pending_protection_mode*). Otherwise, the stored setting of active protection mode (*designation_state.active_protection_mode*) is used to initialize the local value of pending protection mode (*pending_protection_mode*). Doing so allows an active CSPO-3000 designation to be maintained, if possible. The final value of *pending_protection_mode* will be stored in *intruder.protection_mode*.

A pending designation for protection mode is not honored (is suspended) when the Target is not valid for TRM RA processing. A pending designation for protection mode is also not honored (is delayed) when the protection mode used for TRM processing would change and there is an active RA on the Target (*designation_state.active_ra* is TRUE). In either case, *designation_state.status* is set to reflect the reason for the suspension or delay.

Designation to DNA requires the protection mode to be *S_standard_protection_mode*. SETTARGETDESIGNATION ensures that if the Target will be processed for DNA, the pending protection mode is *S_standard_protection_mode* and not *PROTECTION_MODE_Xo_CSPO3k*. Similarly, if the Target will be processed for CSPO-3000, SETTARGETDESIGNATION ensures the flag for pending DNA will be FALSE. If the protection mode change is delayed, the flag for pending DNA is set to the value of *active_dna*. Using the value of *active_dna* preserves the active DNA designation until the Xo mode is switched to CSPO-3000.

SETTARGETDESIGNATION finishes by calling SETINTRUDERDESIGNATIONDATA to populate fields in TRMINTRUDERINPUT (*intruder*) based on the local values of *pending_dna* and *pending_protection_mode* and on the *valid* and *status* fields in DESIGNATIONSTATE (*designation_state*).

This algorithm takes as input *intruder*, *t*, *mu_vert*, *r_ground*, and *designation_state*. This algorithm updates SETINTRUDERDESIGNATIONDATA.

Algorithm 122 SetTargetDesignation

```

1 function SetTargetDesignation( intruder::TRMIntruderInput(p.E-20), t::R, mu_vert::R, r_ground::R,
2   designation_state::DesignationState(p.E-6) )
3   const S_standard_protection_mode::UInt8 = uint8(params().target_designation.S_standard_protection_mode)
4   const T_invalid_limit::Z           = params().target_designation.T_invalid_limit
5   active_dna::Bool =
6     UpdateTargetDesignationValidity(p.127)( designation_state, intruder.id, t, mu_vert, r_ground,
7                                         intruder.degraded_surveillance, intruder.valid,
8                                         intruder.is_icao )
9   pending_dna::Bool = designation_state.pending_dna
10  if pending_dna && !designation_state.valid[Xo_IDX_DNA] && (0 == designation_state.timer_dna)
11    pending_dna = false
12    if (Xo_STATUS_NONE == designation_state.status)
13      designation_state.status = Xo_STATUS_SUSPENDED_INVALID
14    end
15  elseif designation_state.multithreat_ra
16    if pending_dna
17      pending_dna = false
18      if (Xo_STATUS_NONE == designation_state.status)
19        designation_state.status = Xo_STATUS_SUSPENDED_MULTITHREAT
20      end
21    end
22    active_dna = false
23  end
24  ok_cspo3000::Bool = false
25  if designation_state.valid[Xo_IDX_CSP03k] ||
26    ((designation_state.active_protection_mode == PROTECTION_MODE_Xo_CSP03k) &&
27     (0 < designation_state.timer_protection_mode <= T_invalid_limit))
28    ok_cspo3000 = true
29  end
30  pending_protection_mode::UInt8 = designation_state.pending_protection_mode
31  if (pending_dna != designation_state.pending_dna)
32    pending_protection_mode = designation_state.active_protection_mode
33  end
34  if !ok_cspo3000 && (pending_protection_mode == PROTECTION_MODE_Xo_CSP03k)
35    pending_protection_mode = S_standard_protection_mode
36    if (Xo_STATUS_NONE == designation_state.status)
37      designation_state.status = Xo_STATUS_SUSPENDED_INVALID
38    end
39  end
40  if designation_state.active_ra &&
41    (pending_protection_mode != designation_state.active_protection_mode)
42    pending_protection_mode = designation_state.active_protection_mode
43    if (S_standard_protection_mode != pending_protection_mode)
44      pending_dna = false
45    else
46      pending_dna = active_dna
47    end
48    if (Xo_STATUS_NONE == designation_state.status)
49      designation_state.status = Xo_STATUS_DELAYED_ACTIVE_RA
50    end
51  end
52  SetIntruderDesignationData(p.137)(designation_state, pending_dna, pending_protection_mode, intruder)
52 end

```

Referenced In: AddADSBTrackToReport(p.110), AddModeSTrackToReport(p.102), AddModeCTrackToReport(p.114)

UPDATETARGETDESIGNATIONVALIDITY (Algorithm 123) updates the validity, status, pending, and timer information in the Target DESIGNATIONSTATE for each of the Xo modes. The target id (*id*), report time (*t*), and estimated altitude of the Target (*mu_vert*) are used to update the state information for CSPO-3000. The relative ground range from ownship to the Target (*r_ground*) is used to update the state information for Designated No Alerts (DNA). The degraded surveillance status

(*degraded_surveillance*), intruder valid flag (*valid_input*), and flag to indicate ICAO compliance (*is_icao*) are used to update the state information for both CSPO-3000 and DNA.

The *valid_input* flag is used to indicate to the TRM this Target is valid for producing advisories. The TRM will not produce traffic advisories (TAs) or resolution advisories (RAs) for the Target when *valid_input* is FALSE. Xo designation for DNA and CSPO-3000 for a Target is available only when advisories can be produced by the TRM for that Target. Also, Xo designation for DNA and CSPO-3000 for a Target is available only when that Target has an ICAO compliant 24-bit address. Consequently, designation to DNA and CSPO-3000 are marked as invalid when *is_icao* is FALSE and/or *valid_input* is FALSE.

UPDATCSPO3000VALIDITY is called to update the validity, status, and pending information for CSPO-3000 in *designation_state*. It also implements automatic undesignation requirements for CSPO-3000. The CSPO-3000 active flag in *designation_state* remains unchanged.

UPDATEDNAVALIDITY is called to update the validity, status, and pending information for Designated No Alerts (DNA) in *designation_state*. It also implements automatic undesignation requirements for DNA and returns a flag indicating whether DNA remains active (*active_dna*). The DNA active flag in *designation_state* remains unchanged.

After the validity, status, and pending information have been updated for both DNA and CSPO-3000, the timers for both Xo modes are updated. If the Target is designated, the timers are incremented. Otherwise, the timers are reset. For designated targets, the Xo status is updated to *Xo_STATUS_SUSPENDED_INVALID* when there is no other status information, the Xo mode is marked as invalid for designation, and the timer is being started.

Once the timers have been updated, CHECKTARGETDESIGNATIONTIMERS is called to automatically undesignate the Target if the timers have expired. Automatic undesignation of the Target results in updates to *designation_state*. It also results in the *active_dna* flag being set to FALSE.

In the cases where CSPO-3000 is automatically undesignated, the automatic designation will take effect on this processing cycle, or as soon as there is no active resolution advisory. In the cases where DNA is automatically undesignated, the automatic designation will take effect on this processing cycle.

This algorithm takes as input *designation_state*, *id*, *t*, *mu_vert*, *r_ground*, *degraded_surveillance*, *valid_input*, and *is_icao*. This algorithm updates *designation_state* and returns *active_dna*.

Algorithm 123 UpdateTargetDesignationValidity

```

1 function UpdateTargetDesignationValidity( designation_state::DesignationState(p.E-6), id::UInt32, t::R,
2                                         mu_vert::R, r_ground::R, degraded_surveillance::UInt16,
3                                         valid_input::Bool, is_icao::Bool )
4   const S_standard_protection_mode::UInt8 = uint8(params().target_designation.S_standard_protection_mode)
5   designation_state.status = Xo_STATUS_NONE
6   if valid_input && is_icao
7     designation_state.valid[Xo_IDX_DNA] = true
8     designation_state.valid[Xo_IDX_CSP03k] = true
9   else
10    designation_state.valid[Xo_IDX_DNA] = false
11    designation_state.valid[Xo_IDX_CSP03k] = false
12  end
13  UpdateCSP03000Validity(p.129)( id, t, mu_vert, degraded_surveillance, valid_input, designation_state )
14  active_dna::Bool = UpdateDNAValidity(p.131)( r_ground, degraded_surveillance, designation_state )
15  if designation_state.is_designated
16    if designation_state.pending_dna
17      if !designation_state.valid[Xo_IDX_DNA]
18        if (Xo_STATUS_NONE == designation_state.status) &&
19          (0 == designation_state.timer_dna)
20          designation_state.status = Xo_STATUS_SUSPENDED_INVALID
21      end
22      designation_state.timer_dna = designation_state.timer_dna + 1
23    end
24  end
25  if (S_standard_protection_mode != designation_state.pending_protection_mode)
26    if !designation_state.valid[Xo_IDX_CSP03k]
27      if (Xo_STATUS_NONE == designation_state.status) &&
28        (0 == designation_state.timer_protection_mode)
29        designation_state.status = Xo_STATUS_SUSPENDED_INVALID
30      end
31      designation_state.timer_protection_mode = designation_state.timer_protection_mode + 1
32    end
33  end
34 else
35   designation_state.timer_dna = 0
36   designation_state.timer_protection_mode = 0
37 end
38 CheckTargetDesignationTimers(p.135)( designation_state )
39 if !designation_state.valid[Xo_IDX_DNA]
40   active_dna = false
41 end
42 return active_dna::Bool
43 end

```

Referenced In: SetTargetDesignation(p.125)

UPDATECSPO3000VALIDITY (Algorithm 124) updates the CSPO-3000 validity, status, and pending information in the Target DESIGNATIONSTATE (*designation_state*). It also implements automatic undesignation requirements for CSPO-3000.

Xo CSPO-3000 is valid for use only when the TRM can produce resolution advisories (RAs) on the Target and the CSPO-3000 geometric criterion is met.

First, the surveillance quality of the Target is checked to see if RAs can be produced. The TRM will not produce RAs when the surveillance for the Target (*degraded_surveillance*) is ADS-B only, without active validation, (*DEGRADED_SURVEILLANCE_ADSB_ONLY*) or NAR (*DEGRADED_SURVEILLANCE_NAR*) and is invalid for use. The *valid_surv* flag indicates whether the surveillance is of sufficient quality to be used to produce RAs.

Next, IsCSPO3000MODEUNAVAILABLETORUN is called to determine if CSPO-3000 is available

for use based on ownership state. If the Target was designated for CSPO-3000 and CSPO-3000 Xo mode is unavailable, CSPO-3000 is an invalid designation mode and AUTOMATICALLYUNDESIGNATECSPO3000 is used to automatically undesignate the Target from CSPO-3000 mode and update the designation state. Automatic undesignation under this condition results in a status of *Xo_STATUS_UNDESIGNATED_UNAVAILABLE*.

Since there can be no RAs when ownership is not in TA/RA operational mode (*own.trm_opmode* has a value other than *OPMODE_RA*), the designation state *active_ra* value is set to FALSE to allow the protection mode to be switched.

The input argument *valid_input* is the value of the valid flag used to indicate to the TRM this Target is valid for producing RAs. It is used here, along with *own.trm_opmode* and *valid_surv*, as a gate for evaluation of the geometric validity criteria. When the TRM is able to produce RAs, the final validity check is based on geometry. Xo CSPO-3000 is valid only when the Target is below an altitude threshold. If the Target is not designated for CSPO-3000, the initial threshold is used (*H_threshold_lo_cspo3000*). Otherwise, hysteresis is provided if Xo CSPO-3000 is currently designated (*H_threshold_hi_cspo3000*). AUTOMATICALLYUNDESIGNATECSPO3000 is used to automatically undesignate the Target from CSPO-3000 mode when the Target altitude (*mu_vert*) is above the altitude threshold (*alt_threshold*). It also updates the designation state. Automatic undesignation under this condition results in a status of *Xo_STATUS_UNDESIGNATED_GEOMETRIC*.

In the cases where CSPO-3000 is automatically undesignated, the automatic undesignation will take effect on the current processing cycle unless there is an active RA (*designation_state.active_ra* is TRUE). When there is an active RA, the undesignation will take effect as soon as there is no RA.

This algorithm takes as input *id*, *t*, *mu_vert*, *degraded_surveillance*, *valid_input*, and *designation_state*. This algorithm updates *designation_state* and calls AUTOMATICALLYUNDESIGNATECSPO3000.



Algorithm 124 UpdateCSPO3000Validity

```

1 function UpdateCSPO3000Validity( id::UInt32, t::R, mu_vert::R, degraded_surveillance::UInt16,
2                                     valid_input::Bool, designation_state::DesignationState(p.E-6) )
3   const H_threshold_lo::R = params().target_designation.td_cspo3000.H_threshold_lo
4   const H_threshold_hi::R = params().target_designation.td_cspo3000.H_threshold_hi
5   valid_surv::Bool = true
6   if bool(degraded_surveillance & DEGRADED_SURVEILLANCE_ADSB_ONLY) ||
7     bool(degraded_surveillance & DEGRADED_SURVEILLANCE_NAR)
8     valid_surv = false
9     designation_state.valid[Xo_IDX_CSP03k] = false
10    end
11   if IsCSPO3000ModeUnavailableToRun(p.146)( [id], t )
12     designation_state.valid[Xo_IDX_CSP03k] = false
13     AutomaticallyUndesignateCSPO3000(p.130)( designation_state, Xo_STATUS_UNDESIGNATED_UNAVAILABLE )
14     valid_input = false
15   end
16   if (own.trm_opmode != OPMODE_RA)
17     designation_state.active_ra = false
18   elseif valid_surv && valid_input
19     alt_threshold::R = H_threshold_lo
20     if (designation_state.active_protection_mode == PROTECTION_MODE_Xo_CSP03k) ||
21       (designation_state.pending_protection_mode == PROTECTION_MODE_Xo_CSP03k)
22       alt_threshold = H_threshold_hi
23     end
24     if (alt_threshold < mu_vert)
25       designation_state.valid[Xo_IDX_CSP03k] = false
26       AutomaticallyUndesignateCSPO3000(p.130)( designation_state, Xo_STATUS_UNDESIGNATED_GEOMETRIC )
27     end
28   end
29 end

```

Referenced In: UpdateTargetDesignationValidity(p.127)

AUTOMATICALLYUNDESIGNATECSPO3000 (Algorithm 125) automatically undesignates the Target from CSPO-3000 Xo mode. It also updates the Target DESIGNATIONSTATE (*designation_state*).

Automatic undesignation will occur only when the Target is designated for a protection mode. The Target is considered designated to a protection mode whenever *pending_protection_mode* is something other than the default (*S_standard_protection_mode*). Automatic undesignation results in the designation state status set to the value of the input variable *status*, the designation state pending flag (*pending_protection_mode*) set to the default protection mode, and the designation state *designated_mode* set to *DESIGNATE_UNDESIGNATE_PROTECTION_MODE*.

If the Target is not designated for a protection mode, there is no change to the designation state.

Because the designation state pending flag is set to *S_standard_protection_mode*, the automatic undesignation will go into effect on this cycle unless there is an active RA on this Target.

This algorithm takes as input *designation_state* and *status*. This algorithm updates *designation_state*.

Algorithm 125 AutomaticallyUndesignateCSPO3000

```

1 function AutomaticallyUndesignateCSPO3000( designation_state::DesignationState(p.E-6), status::UInt8 )
2   const S_standard_protection_mode::UInt8 = uint8(params().target_designation.S_standard_protection_mode)
3   if (S_standard_protection_mode != designation_state.pending_protection_mode)
4     designation_state.status           = status
5     designation_state.pending_protection_mode = S_standard_protection_mode
6     designation_state.designated_mode      = DESIGNATION_UNDESIGNATE_PROTECTION_MODE
7   end
8 end

```

Referenced In: UpdateCSPO3000Validity(p. 129)

UPDATEDNAVALIDITY (Algorithm 126) updates the Designated No Alerts (DNA) validity, status, and pending information in the Target DESIGNATIONSTATE (*designation_state*). It also implements automatic undesignation requirements for DNA.

Xo DNA mode is valid whenever the TRM can produce Traffic Advisories (TAs) on the Target, the DNA Xo mode is available, there is a valid bearing estimate for the Target, and the DNA geometric criteria for altitude and range are satisfied. The *active_dna* flag is initialized to the value of *designation_state.active_dna*, which reflects whether or not the DNA mode for this Target was active in the TRM on the previous cycle. The *active_dna* flag is set to FALSE in this algorithm if the Target is automatically undesignated; *designation_state.active_dna* remains unchanged.

Determination of DNA validity based on whether the TRM can produce TAs is performed in UPDATEREGARDDESIGNATIONVALIDITY.

IsDNAMODEUNAVAILABLETORUN determines whether DNA Xo mode is available for use based on ownership state. If DNA Xo mode is unavailable, DNA is an invalid designation mode and AUTOMATICALLYUNDESIGNATEDDNA is used to automatically undesignate the Target. It also updates the value of *active_dna* and the designation state. Automatic undesignation under this condition results in a status of *Xo_STATUS_UNDESIGNATED_UNAVAILABLE*.

The validity of the bearing estimate output to the TRM is examined next. When *degraded_surveillance* indicates there is no bearing estimate, DNA is an invalid designation mode and AUTOMATICALLYUNDESIGNATEDDNA is used to automatically undesignate the Target. It also updates the value of *active_dna* and the designation state. Automatic undesignation under this condition results in a status of *Xo_STATUS_UNDESIGNATED_NO_BEARING*.

UPDATEDNABASEDONALTITUDE is used to determine whether the Target should be automatically undesignated based on altitude criteria. UPDATEDNABASEDONRANGE is used to determine whether the Target should be automatically undesignated based on range criteria. Both algorithms update the *active_dna* flag and the designation state when the Target is automatically undesignated.

An automatic undesignation will take effect on this processing cycle.

This algorithm takes as input *r_ground*, *degraded_surveillance*, and *designation_state*. This algorithm updates *designation_state* and returns *active_dna*.

Algorithm 126 UpdateDNAValidity

```

1 function UpdateDNAValidity( r_ground::R, degraded_surveillance::UInt16,
2                               designation_state::DesignationState(p_E-6 ) )
3   active_dna::Bool = designation_state.active_dna
4   if IsDNAModeUnavailableToRun(p_146)()
5     designation_state.valid[Xo_IDX_DNA] = false
6     active_dna = AutomaticallyUndesignateDNA(p_131)( designation_state, Xo_STATUS_UNDESIGNATED_UNAVAILABLE )
7   end
8   if (0 != (degraded_surveillance & DEGRADED_SURVEILLANCE_NO_BEARING))
9     designation_state.valid[Xo_IDX_DNA] = false
10    active_dna = AutomaticallyUndesignateDNA(p_131)( designation_state, Xo_STATUS_UNDESIGNATED_NO_BEARING )
11  end
12 active_dna = UpdateDNABasedOnAltitude(p_132)( designation_state, active_dna )
13 active_dna = UpdateDNABasedOnRange(p_134)( r_ground, designation_state, active_dna )
14 return active_dna::Bool
15 end
```

Referenced In: UpdateTargetDesignationValidity(*p_127*)

AUTOMATICALLYUNDESIGNATEDDNA (Algorithm 127) automatically undesignates the Target from Designated No Alerts (DNA) Xo mode. It also updates the Target DESIGNATIONSTATE (*designation_state*) and sets the active flag (*active_dna*) to FALSE.

Automatic undesignation will occur only when the Target is designated to DNA mode. The Target is considered designated to DNA mode when *pending_dna* is TRUE. Automatic undesignation results in the designation state status set to the value of the input variable *status*, the designation state pending flag (*pending_dna*) set to FALSE, and the designation state *designated_mode* set to DESIGNATE_UNDESIGNATE_NO_ALERTS.

If the Target is not designated to DNA mode, there is no change to the designation state.

Whenever the designation state *active_dna* is TRUE, the *active_dna* argument is set to FALSE to indicate the target designation is no longer in force. Otherwise, it retains the value it had at input.

Because the designation state pending flag is set to FALSE, the automatic undesignation will go into effect on this cycle.

This algorithm takes as input *designation_state* and *status*. This algorithm updates *designation_state* and returns *active_dna*.

Algorithm 127 AutomaticallyUndesignateDNA

```

1 function AutomaticallyUndesignateDNA( designation_state::DesignationState(p_E-6), status::UInt8 )
2   if designation_state.pending_dna
3     designation_state.status      = status
4     designation_state.pending_dna = false
5     designation_state.designated_mode = DESIGNATION_UNDESIGNATE_NO_ALERTS
6   end
7   active_dna::Bool = false
8   return active_dna::Bool
9 end
```

Referenced In: UpdateDNAValidity(*p_131*), UpdateDNABasedOnRange(*p_134*), UpdateDNABasedOnAltitude(*p_132*)

UPDATEDNABASEDONALTITUDE (Algorithm 128) updates the Designated No Alerts (DNA) sta-

tus and pending information in the Target DESIGNATIONSTATE (*designation_state*) based on altitude criteria. It also implements altitude-based automatic undesignation requirements for DNA.

There are two separate criteria for automatic undesignation based on altitude:

1. Ownship has landed after being airborne
2. Ownship is going around

Designation to DNA is permitted at any time; ownship may be on the ground or airborne. Automatic undesignation can occur only after ownship has become airborne.

When the Target DESIGNATIONSTATE is first created, ownship state is neither airborne nor landing. Ownship is determined to be airborne (*is_airborne*) when the radar altimeter is inactive (*Nan*) or the radar altimeter reading is above the upper aural inhibit threshold (*hnoauralhi*). Ownship is determined to be landing (*is_landing*) when the radar altimeter is active (not *Nan*) and its reading is below the lower aural inhibit threshold (*hnoaurallo*).

Ownship is considered to have landed when it was airborne and now is on the ground (*own.on_ground*). When ownship has landed, AUTOMATICALLYUNDESIGNATEDDNA is used to automatically undesignate the Target from DNA mode. *is_airborne* and *is_landing* are both reset.

Ownship is considered to be going around when it goes above the upper aural inhibit threshold (airborne again) after going below the lower aural inhibit threshold (*is_landing* is TRUE). When ownship is going around, AUTOMATICALLYUNDESIGNATEDDNA is used to automatically undesignate the Target from DNA mode. Once that happens, ownship is no longer considered to be landing but it is still considered airborne.

Automatic undesignation due to these geometric constraints always results in a status of *Xo_STATUS_UNDESIGNATED_GEOMETRIC* and the *active_dna* flag is set to FALSE.

An automatic undesignation will take effect on this processing cycle.

This algorithm takes as input *designation_state* and *active_dna*. This algorithm updates *designation_state* and returns *active_dna*.

Algorithm 128 UpdateDNABasedOnAltitude

```

1 function UpdateDNABasedOnAltitude( designation_state::DesignationState(p. E-6), active_dna::Bool )
2   const hnoaurallo::R = params().display.hnoaurallo
3   const hnoauralhi::R = params().display.hnoauralhi
4   if designation_state.is_airborne && own.on_ground
5     active_dna = AutomaticallyUndesignateDNA(p.131)( designation_state, Xo_STATUS_UNDESIGNATED_GEOMETRIC )
6     designation_state.is_landing = false
7     designation_state.is_airborne = false
8   elseif designation_state.is_airborne && !isnan( own.radalt ) && (own.radalt <= hnoaurallo)
9     designation_state.is_landing = true
10  elseif isnan( own.radalt ) || (hnoauralhi < own.radalt)
11    if designation_state.is_landing
12      active_dna = AutomaticallyUndesignateDNA(p.131)( designation_state, Xo_STATUS_UNDESIGNATED_GEOMETRIC
13        )
14    end
15    designation_state.is_airborne = true
16    designation_state.is_landing = false
17  end
18 return active_dna::Bool

```

Referenced In: UpdateDNAValidity(p.131)

UPDATEDNABASEDONRANGE (Algorithm 129) updates the Designated No Alerts (DNA) status and pending information in the Target DESIGNATIONSTATE (*designation_state*) based on range criteria. It also implements automatic undesignation requirements for DNA.

There are two separate criteria for automatic undesignation based on range:

1. The Target is no longer proximate
2. The track of the Target is diverging from the ownship track

Designation to DNA is permitted at any time; the Target may be proximate at the time of designation or not. Automatic undesignation can occur only if the Target track is diverging from the ownship track over a set of position estimates.

When the Target is designated for DNA and is within the proximate traffic range of ownship (*D_rng_threshold_lo*), it is considered proximate (*was_proximate*). AUTOMATICALLYUNDESIGNATEDNA is used to automatically undesignate the Target from DNA mode as soon as it loses proximity (exceeds *D_rng_threshold_hi*) after being proximate. Hysteresis is applied for undesignation so the Target doesn't become undesigned immediately after losing proximity.

The Target may be designated for DNA while outside the proximate traffic range. In this case, it may never become proximate but it must be automatically undesigned when it diverges. To determine divergence when the Target never becomes proximate, an M of N range test is applied. When the Target is designated for DNA, the relative ground range (*r_ground*) is recorded each processing cycle. The Target is determined to be diverging when *N* entries exist in the history (*r_ground_history*) and the current relative ground range is larger than *M* of the *N* relative ranges in the history. When that happens, AUTOMATICALLYUNDESIGNATEDNA is used to automatically undesignate the Target from DNA mode.

If the Target is not designated for DNA, the relative ground range history is cleared and *was_proximate* is set to FALSE.

The Julia built-in function *deleteat!* is used to remove the oldest entries from the list. The Julia built-in function *sort* is used to order the list from smallest to largest range. The current relative ground range is pushed onto the history using the Julia built-in function *push!*.

Automatic undesignation due to these geometric constraints always results in a status of *Xo_STATUS_UNDESIGNED_GEOMETRIC* and the *active_dna* flag is set to FALSE.

An automatic undesignation will take effect on this processing cycle.

This algorithm takes as input *r_ground*, *designation_state* and *active_dna*. This algorithm updates *designation_state* and returns *active_dna*.

Algorithm 129 UpdateDNABasedOnRange

```

1 function UpdateDNABasedOnRange( r_ground::R, designation_state::DesignationState(p. E-6), active_dna::Bool )
2   const D_rng_threshold_lo::R = params().target_designation.td_dna.D_rng_threshold_lo
3   const D_rng_threshold_hi::R = params().target_designation.td_dna.D_rng_threshold_hi
4   const M::Z               = params().target_designation.td_dna.M
5   const N::Z               = params().target_designation.td_dna.N
6   M_of_N::Z = min( M, N )
7   if designation_state.active_dna || designation_state.pending_dna
8     count::Z = length( designation_state.r_ground_history )
9     if (N < count)
10      deleteat!( designation_state.r_ground_history, 1:(count-N) )
11    end
12    if (r_ground <= D_rng_threshold_lo)
13      designation_state.was_proximate = true
14    elseif designation_state.pending_dna || active_dna
15      if designation_state.was_proximate
16        if (D_rng_threshold_hi < r_ground)
17          active_dna = AutomaticallyUndesignateDNA(p.131)( designation_state,
18                                              Xo_STATUS_UNDESIGNATED_GEOMETRIC )
19        end
20        elseif (N == length( designation_state.r_ground_history ))
21          range_ordered::Vector{R} = sort( designation_state.r_ground_history )
22          if (range_ordered[M_of_N] < r_ground)
23            active_dna = AutomaticallyUndesignateDNA(p.131)( designation_state,
24                                              Xo_STATUS_UNDESIGNATED_GEOMETRIC )
25          end
26        end
27      else
28        designation_state.was_proximate = false
29        designation_state.r_ground_history = Array( R, 0 )
30      end
31    return active_dna::Bool
32 end

```

Referenced In: UpdateDNAValidity(p.131)

CHECKTARGETDESIGNATIONTIMERS (Algorithm 130) automatically undesignates the Target if the timers have expired. Automatic undesignation of the Target results in updates to the fields associated with each of the Xo modes in the Target DESIGNATIONSTATE (*designation_state*).

Each of the Xo modes is examined separately to determine if the designation should be automatically timed out due to expiration of the associated timer.

The timer for DNA (*timer_dna*) is checked when the Target is designated for DNA (*pending_dna* is TRUE) and it is invalid (*valid[Xo_IDX_DNA]* is FALSE).

The timer for CSPO-3000 (*timer_protection_mode*) is checked when the Target is designated for CSPO-3000 (*pending_protection_mode* is the CSPO-3000 protection mode) and it is invalid (*valid[Xo_IDX_CSPO3k]* is FALSE).

In both cases, the mode is automatically undesignated when the timer has exceeded the time limit (*T_invalid_limit*). Status is set to indicate the undesignation is due to a timeout (*Xo_STATUS_UNDESIGNATED_TIMEOUT*). If the timer is running but no status is provided to indicate a reason, status is set to indicate the timer is running (*Xo_STATUS_SUSPENDED_TIMER*).

Automatic undesignation of DNA is effected by setting the pending DNA (*pending_dna*) to FALSE and setting *designated_mode* to *DESIGNATE_UNDESIGNATE_NO_ALERTS*.

...
...

Automatic undesignation of CSPO-3000 is effected by setting the pending protection mode (*pending_protection_mode*) to the default (*S_standard_protection_mode*) and setting *designated_mode* to *DESIGNATE_UNDESIGNATE_PROTECTION_MODE*.

An automatic designation from an invalid timeout will take effect on this processing cycle.

Whenever there is no active designation or pending designation for an Xo mode at the end of the algorithm, the associated timer is reset.

This algorithm takes as input *designation_state*. This algorithm updates *designation_state*.

Algorithm 130 CheckTargetDesignationTimers

```

1 function CheckTargetDesignationTimers( designation_state::DesignationState(p. E-6) )
2   const T_invalid_limit::Z = params().target_designation.T_invalid_limit
3   const S_standard_protection_mode::Uint8 = uint8(params().target_designation.S_standard_protection_mode)
4   if designation_state.pending_dna && !designation_state.valid[Xo_IDX_DNA]
5     if (T_invalid_limit < designation_state.timer_dna)
6       designation_state.pending_dna = false
7       designation_state.designated_mode = DESIGNATION_UNDESIGNATE_NO_ALERTS
8       designation_state.status = Xo_STATUS_UNDESIGNATED_TIMEOUT
9     elseif (0 < designation_state.timer_dna)
10      if (Xo_STATUS_NONE == designation_state.status)
11        designation_state.status = Xo_STATUS_SUSPENDED_TIMER
12      end
13    end
14  end
15  if (PROTECTION_MODE_Xo_CSP03k == designation_state.pending_protection_mode) &&
16  !designation_state.valid[Xo_IDX_CSP03k]
17    if (T_invalid_limit < designation_state.timer_protection_mode)
18      designation_state.pending_protection_mode = S_standard_protection_mode
19      designation_state.designated_mode = DESIGNATION_UNDESIGNATE_PROTECTION_MODE
20      designation_state.status = Xo_STATUS_UNDESIGNATED_TIMEOUT
21    elseif (0 < designation_state.timer_protection_mode)
22      if (Xo_STATUS_NONE == designation_state.status)
23        designation_state.status = Xo_STATUS_SUSPENDED_TIMER
24      end
25    end
26  end
27  if !designation_state.active_dna && !designation_state.pending_dna
28    designation_state.timer_dna = 0
29  end
30  if (S_standard_protection_mode == designation_state.active_protection_mode) &&
31  (S_standard_protection_mode == designation_state.pending_protection_mode)
32    designation_state.timer_protection_mode = 0
33  end
34 end
```

Referenced In: UpdateTargetDesignationValidity(^{p. 127}), UpdateDroppedTargetDesignationState(^{p. 144})
--

SETINTRUDERDESIGNATIONDATA (Algorithm 131) populates the fields in TRMINTRUDERINPUT associated with ACAS Xo processing. Those fields are the requested designation (*designated_mode*), the desired protection mode (*protection_mode*), the desired Designated No Alerts (DNA) state (*dna*), the validity for each Xo mode (*xo_valid*), and the Xo status (*xo_status*).

SETINTRUDERDESIGNATIONDATA first uses the settings for DNA (*pending_dna*) and protection mode (*pending_protection_mode*) selected in SETTARGETDESIGNATION or ADJUSTTARGETDESIGNATIONVALIDITY to set the *dna* and *protection_mode* fields in TRMINTRUDERINPUT.

Next, it uses the stored pending values (*pending_dna* and *pending_protection_mode*) and the stored

designated mode value (*designated_mode*) in DESIGNATIONSTATE to set the *designated_mode* field in TRMINTRUDERINPUT. If the stored pending values indicate Xo processing is active, the designated mode in TRMINTRUDERINPUT is set accordingly. Otherwise, the designated mode for output is the same as the designated mode stored in DESIGNATIONSTATE.

The valid state to be output to the ASSAP function via the TRM (*xo_valid* in TRMINTRUDERINPUT) is set next. Validity values are associated with each of the two Xo designation types. Both *valid* in DESIGNATIONSTATE and *xo_valid* in TRMINTRUDERINPUT are vectors that contain the validity state for each of the Xo modes. Whenever the Target should remain designated, the selected designation must be marked as valid. The protection mode validity is considered and set first. The DNA validity is considered and set second. In both cases, the rules are the same:

- If the Xo mode is valid, it will stay valid
- If the Xo mode timer is active, the Xo mode will be valid
- If the Xo mode is selected for undesignation processing, the Xo mode will be valid
- Otherwise, the Xo mode will be invalid

Finally, status information is prepared for output to the ASSAP function via the TRM (*xo_status* in TRMINTRUDERINPUT). Generally, the status information from (*status* in DESIGNATIONSTATE) is preserved in the TRMINTRUDERINPUT output. The exception being when either Xo mode is active and running on the timer (*status* in DESIGNATIONSTATE equals *Xo_STATUS_SUSPENDED_TIMER*). In that case, the information about the timer is suppressed. Except in special cases, e.g., a multiple threat RA or active RA, the status information from *xo_status* in TRMINTRUDERINPUT is preserved by the TRM for output to the ASSAP function.

This algorithm takes as input *designation_state*, *pending_dna*, *pending_protection_mode*, and *intruder*. This algorithm updates numerous fields within the *intruder* structure.



Algorithm 131 SetIntruderDesignationData

```

1 function SetIntruderDesignationData(designation_state::DesignationState(p. E-6), pending_dna::Bool,
2                                     pending_protection_mode::UInt8, intruder::TRMIntruderInput(p. E-20))
3   const T_invalid_limit::Z           = params().target_designation.T_invalid_limit
4   const S_standard_protection_mode::UInt8 = UInt8(params().target_designation.S_standard_protection_mode)
5   intruder.dna          = pending_dna
6   intruder.protection_mode = pending_protection_mode
7   if designation_state.pending_dna
8     intruder.designated_mode = DESIGNATION_DESIGNATED_NO_ALERTS
9   elseif (PROTECTION_MODE_Xo_CSP03k == designation_state.pending_protection_mode)
10    intruder.designated_mode = DESIGNATION_Xo_CSP03k
11  else
12    intruder.designated_mode = designation_state.designated_mode
13  end
14  if designation_state.valid[Xo_IDX_CSP03k]
15    intruder.xo_valid[Xo_IDX_CSP03k] = true
16  elseif (0 < designation_state.timer_protection_mode <= T_invalid_limit)
17    intruder.xo_valid[Xo_IDX_CSP03k] = true
18  elseif (DESIGNATION_UNDESIGNATE_PROTECTION_MODE == intruder.designated_mode)
19    intruder.xo_valid[Xo_IDX_CSP03k] = true
20  else
21    intruder.xo_valid[Xo_IDX_CSP03k] = false
22  end
23  if designation_state.valid[Xo_IDX_DNA]
24    intruder.xo_valid[Xo_IDX_DNA] = true
25  elseif (0 < designation_state.timer_dna <= T_invalid_limit)
26    intruder.xo_valid[Xo_IDX_DNA] = true
27  elseif (DESIGNATION_UNDESIGNATE_NO_ALERTS == intruder.designated_mode)
28    intruder.xo_valid[Xo_IDX_DNA] = true
29  else
30    intruder.xo_valid[Xo_IDX_DNA] = false
31  end
32  if !intruder.valid
33    intruder.xo_status = designation_state.status
34  elseif (Xo_STATUS_SUSPENDED_TIMER != designation_state.status)
35    intruder.xo_status = designation_state.status
36  elseif intruder.dna
37    intruder.xo_status = Xo_STATUS_NONE
38  elseif (S_standard_protection_mode != intruder.protection_mode)
39    intruder.xo_status = Xo_STATUS_NONE
40  else
41    intruder.xo_status = designation_state.status
42  end
43 end

```

Referenced In: SetTargetDesignation(p. 125), AdjustTargetDesignationValidity(p. 143)

SETTARGETDESIGNATIONINVALID (Algorithm 132) resets fields in DESIGNATIONSTATE when the Target has been dropped. The inputs are the designation state to be updated (*designation_state*) and whether a timeout should be forced (*force_timeout*). If there is no forced timeout, the Target will remain designated until the timers expire.

Whenever SETTARGETDESIGNATIONINVALID is called, both Xo modes (*valid[Xo_IDX_DNA]* and *valid[Xo_IDX_CSP03k]*) are set to invalid and the status (*status*) is reset. The designated mode (*designated_mode*) is updated only if the Target is not designated or its designation status is changing. The majority of SETTARGETDESIGNATIONINVALID is used only when the Target is designated.

When the Target is designated, the timers are updated as specified by *force_timeout* and additional values are updated depending on whether the designation is active. When a timeout is forced, both

timers are updated. Otherwise, only the timer associated with the designated mode is updated. IS-TARGETDESIGNATIONACTIVE is used to determine whether the Target designation is still active. It is called with *check_timer* set to TRUE so the target designation will be flagged as inactive if the timers have expired. Note that the timers are updated before the call to ISTARGETDESIGNATIONACTIVE.

If the Target is designated and the target designation is no longer active, target designation state fields are reset as follows:

- When the designated mode (*designated_mode*) is a designation and not an undesignation, the status for output to the ASSAP function is set to *Xo_STATUS_UNDESIGNATED_TIMEOUT*.
- When there is a pending DNA, DNA is undesignated (*dna* is set to FALSE and *designated_mode* is set to *DESIGNATION_UNDESIGNATE_NO_ALERTS*).
- When there is a pending protection mode (i.e., CSPO-3000), protection mode is undesignated (*protection_mode* is set to *S_standard_protection_mode* and *designated_mode* is set to *DESIGNATION_UNDESIGNATE_PROTECTION_MODE*).
- When the designated mode (*designated_mode*) is a designation and not an undesignation, the designated mode is set to *DESIGNATION_NONE*.
- The timers (*timer_dna* and *timer_protection_mode*) are reset and state variables for active DNA (*is_landing*, *is_proximate*, *r_ground_history*) are reset.

If the Target is designated and the target designation is still active, there are a couple of minor adjustments depending on the designated mode as follow:

- When the designated mode (*designated_mode*) is a designation and not an undesignation, the status for output to the ASSAP function is set to *Xo_STATUS_SUSPENDED_INVALID*.
- When DNA designation is active or pending, the ground history (*r_ground_history*) is updated so there is an entry for every processing cycle.

If the Target is not designated, the designated mode is set to *DESIGNATION_NONE* as a precaution.

This algorithm takes as input *designation_state* and *force_timeout*. This algorithm updates *designation_state*.

Algorithm 132 SetTargetDesignationInvalid

```

1 function SetTargetDesignationInvalid( designation_state::DesignationState(p. E-6), force_timeout::Bool )
2   const T_invalid_limit::Z = params().target_designation.T_invalid_limit
3   const S_standard_protection_mode::UInt8 = uint8(params().target_designation.S_standard_protection_mode)
4   designation_state.valid[Xo_IDX_DNA]      = false
5   designation_state.valid[Xo_IDX_CSP03K] = false
6   designation_state.status           = Xo_STATUS_NONE
7   if designation_state.is_designated
8     if force_timeout
9       designation_state.timer_dna          = T_invalid_limit + 1
10      designation_state.timer_protection_mode = T_invalid_limit + 1
11    else
12      if designation_state.pending_dna
13        designation_state.timer_dna = designation_state.timer_dna + 1
14      else
15        designation_state.timer_dna = T_invalid_limit + 1
16      end
17      if (S_standard_protection_mode != designation_state.pending_protection_mode)
18        designation_state.timer_protection_mode = designation_state.timer_protection_mode + 1
19      else
20        designation_state.timer_protection_mode = T_invalid_limit + 1
21      end
22    end
23    if !IsTargetDesignationActive(p. 141)( designation_state, true )
24      if (DESIGNATION_UNDESIGNATE_NO_ALERTS      != designation_state.designated_mode) &&
25        (DESIGNATION_UNDESIGNATE_PROTECTION_MODE != designation_state.designated_mode)
26        designation_state.status           = Xo_STATUS_UNDESIGNATED_TIMEOUT
27        designation_state.designated_mode = DESIGNATION_NONE
28      end
29      if designation_state.pending_dna
30        designation_state.pending_dna      = false
31        designation_state.designated_mode = DESIGNATION_UNDESIGNATE_NO_ALERTS
32      end
33      if (S_standard_protection_mode != designation_state.pending_protection_mode)
34        designation_state.pending_protection_mode = S_standard_protection_mode
35        designation_state.designated_mode      = DESIGNATION_UNDESIGNATE_PROTECTION_MODE
36      end
37      designation_state.timer_dna          = 0
38      designation_state.timer_protection_mode = 0
39      designation_state.is_landing        = false
40      designation_state.was_proximate    = false
41      designation_state.r_ground_history = Array( R, 0 )
42    else
43      if (DESIGNATION_UNDESIGNATE_NO_ALERTS      != designation_state.designated_mode) &&
44        (DESIGNATION_UNDESIGNATE_PROTECTION_MODE != designation_state.designated_mode)
45        designation_state.status = Xo_STATUS_SUSPENDED_INVALID
46      end
47      if (designation_state.active_dna || designation_state.pending_dna)
48        push!( designation_state.r_ground_history, NaN )
49      end
50    end
51  else
52    designation_state.designated_mode = DESIGNATION_NONE
53  end
54 end

```

Referenced In: ReceiveTargetDesignation(p. 73), AddTracksToReport(p. 98)

ISTARGETDESIGNATIONACTIVE (Algorithm 133) returns TRUE or FALSE to indicate whether any target designations are active for the associated Target. The *designation_state*, associated with a specific Target, is examined to determine if designation is active. If *check_timer* is TRUE, the timer status is included in the determination.

By default, all local variables associated with the timer are initialized to FALSE. Each timer can be in one of the following states:

- A timer value between 1 and the time limit ($T_{invalid_limit}$), inclusive, indicates the timer is running
- A timer value greater than the time limit, indicates the timer is expired and not running
- A timer value of 0 indicates the timer is not running and not expired

When the timers are to be checked, the local variables associated with the timers are set based on the timer state. If either timer is running, *is_timer_running* is set to TRUE. If the DNA timer is expired, *is_timer_expired_dna* is set to TRUE. If the protection mode timer is expired, *is_timer_expired_protection_mode* is set to TRUE.

After the check timer block, the active flag (*is_active*) is set based on the timer flag settings, the designation active states (*active_dna* and *active_protection_mode*), and the designation pending states (*pending_dna* and *pending_protection_mode*) as follows:

- If the timer is running, designation is active
- If DNA is active or pending and not expired, designation is active
- If protection mode is active or pending and not expired, designation is active
- Otherwise, designation is not active

This algorithm takes as input *designation_state* and *check_timer*. This algorithm returns *is_active*.

Algorithm 133 IsTargetDesignationActive

```

1 function IsTargetDesignationActive( designation_state::DesignationState(p.E-6), check_timer::Bool )
2   const T_invalid_limit::Z           = params().target_designation.T_invalid_limit
3   const S_standard_protection_mode::UInt8 = uint8(params().target_designation.S_standard_protection_mode)
4   is_timer_running::Bool            = false
5   is_timer_expired_dna::Bool       = false
6   is_timer_expired_protection_mode::Bool = false
7   is_active::Bool = false
8   if check_timer
9     if (0 < designation_state.timer_dna      <= T_invalid_limit) ||
10    (0 < designation_state.timer_protection_mode <= T_invalid_limit)
11    is_timer_running = true
12  else
13    is_timer_running = false
14    if (T_invalid_limit < designation_state.timer_dna)
15      is_timer_expired_dna = true
16    end
17    if (T_invalid_limit < designation_state.timer_protection_mode)
18      is_timer_expired_protection_mode = true
19    end
20  end
21 end
22 if is_timer_running
23   is_active = true
24 elseif !is_timer_expired_dna &&
25   (designation_state.active_dna || designation_state.pending_dna)
26   is_active = true
27 elseif !is_timer_expired_protection_mode &&
28   ((designation_state.active_protection_mode != S_standard_protection_mode) ||
29   (designation_state.pending_protection_mode != S_standard_protection_mode))
30   is_active = true
31 else
32   is_active = false
33 end
34 return is_active::Bool
35 end

```

Referenced In: MergeTargetDesignations(p.C-16), StmHousekeepingTargetDesignation(p.A-2), SetTargetDesignationInvalid(p.139)

ADJUSTTARGETDESIGNATIONVALIDITY (Algorithm 134) updates the Target DESIGNATIONSTATE for every intruder in the target database and conditionally updates the validity information in TR-MINTRUDERINPUT for each intruder in the output to the TRM.

This algorithm ensures that the valid flags are set appropriately when a Target has been dropped from the output to the TRM.

First, a lookup table (*intruder_dict*) is created that contains all of the intruders being provided to the TRM for processing. The entries in the table are referenced by Target id. The remainder of ADJUSTTARGETDESIGNATIONVALIDITY uses this lookup table as a reference as it updates the designation state for each Target in the target database.

In the loop over all Targets in the target database, if the Target id is not found in *intruder_dict*, the Target was omitted from the output to the TRM and is considered dropped. Reasons for the Target being omitted from the output include:

1. The Target track was dropped
2. The Target was filtered from the outputs to the TRM
3. AddTracksToReport was not called (*designation_needs_update* is TRUE)

For each dropped Target, UPDATEDDROPPEDTARGETDESIGNATIONSTATE updates the designation state appropriately. Omission of a Target from the output to the TRM is indicated by setting DESIGNATIONSTATE *valid* to FALSE and *was_dropped* to TRUE.

In ACAS X, designation on the Target is maintained for a period of time after all tracks on the Target are lost and the Target would otherwise be considered "dropped". To prevent the ASSAP function from undesignating this dropped Target, the ACAS X intruder output to the ASSAP function must include designation information for that Target. The special invalid intruder is used to achieve this objective, a TRMINTRUDERINPUT with *DEGRADED_SURVEILLANCE_ALL* is added to the *report*. The DROPPEDINTRUDERSADJUSTMENT algorithm in the TRM uses the TRMINTRUDER-INPUT designated mode, validity, and status information in the special invalid intruder to produce designation outputs for dropped targets that will prevent the ASSAP function from undesignating the traffic.

The dropped designated traffic cannot be displayed. In that case, if a new Target is designated, this undisplayable designated Target becomes undesignated and the information about both Targets will be output to the TRM. When the designation timer expires on the undisplayable designated Target, it will also be undesignated and that information will be output to the TRM. While its designation timer is running, the Target remains designated and valid in the TRM designation output to the ASSAP function, as long as another Target is not designated.

Whenever the Target is included in the original output to the TRM, the Target was not dropped. That condition is indicated by setting *was_dropped* in the DESIGNATIONSTATE to FALSE. The DESIGNATIONSTATE timers are reset to 0 when Target designation is valid.

This algorithm takes as input *report* and *designation_needs_update*. It updates *tgt.designation_state*.

Algorithm 134 AdjustTargetDesignationValidity

```

1 function AdjustTargetDesignationValidity( report::StmReport(p.E-1), designation_needs_update::Bool )
2   const S_standard_protection_mode::UInt8 = uint8(params().target_designation.S_standard_protection_mode)
3   intruder_dict::Dict<UInt32,TRMIntruderInput> = Dict<UInt32,TRMIntruderInput(p.E-20)>()
4   for i::Z in 1:length( report.trm_input.intruder )
5     intruder_dict[report.trm_input.intruder[i].id] = report.trm_input.intruder[i]
6   end
7   for id::UInt32 in keys( target_db )
8     tgt::Target(p.E-14) = RetrieveWithID(p.20)( target_db, id )
9     if !haskey( intruder_dict, id )
10      if designation_needs_update
11        tgt.designation_state.status = Xo_STATUS_NONE
12      end
13      UpdateDroppedTargetDesignationState(p.144)( designation_needs_update, tgt.designation_state )
14      tgt.designation_state.valid[Xo_IDX_DNA] = false
15      tgt.designation_state.valid[Xo_IDX_CSP03k] = false
16      tgt.designation_state.was_dropped = true
17      if tgt.designation_state.is_designated
18        intruder::TRMIntruderInput(p.E-20) =
19          TRMIntruderInput(p.E-20)( id, tgt.designation_state.mode_s, DEGRADED_SURVEILLANCE_ALL )
20        SetIntruderDesignationData(p.137)( tgt.designation_state, false,
21          S_standard_protection_mode, intruder )
22        push!( report.trm_input.intruder, intruder )
23      end
24    else
25      if tgt.designation_state.valid[Xo_IDX_DNA]
26        tgt.designation_state.timer_dna = 0
27      end
28      if tgt.designation_state.valid[Xo_IDX_CSP03k]
29        tgt.designation_state.timer_protection_mode = 0
30      end
31      tgt.designation_state.was_dropped = false
32    end
33  end
34 end

```

Referenced In: GenerateStmReport(p.97)

UPDATEDROPPEDTARGETDESIGNATIONSTATE (Algorithm 135) updates the DESIGNATIONSTATE of a Target in the target database that is designated to an Xo mode and is being omitted from the list of intruders for TRM processing.

If the Target is not designated to an Xo mode, this algorithm has no effect.

The input variable *designation_needs_update* indicates whether or not the designation state was already updated in this processing cycle. If a designation state valid flag is TRUE when the Target is designated, the Target may have been filtered from the output to the TRM and the designation state timer would not have been updated in this processing cycle.

When the Target is designated, the following actions are taken:

- The DESIGNATIONSTATE timers are updated, if they wouldn't have been updated already
- The DESIGNATIONSTATE ground history (*r_ground_history*) is updated, if it wouldn't have been updated already
- The DESIGNATIONSTATE status is updated, if the status information didn't indicate undesignation already

CHECKTARGETDESIGNATIONTIMERS automatically undesignates the Target if the timers have

expired. Once the timers have expired, the ground history is cleared. Otherwise, a *Nan* entry is added to the ground history so there is an entry for every time step. The Julia built-in function *deleteat!* is used to remove the oldest entries from the ground history.

If the Target is still flagged as designated after the first set of updates, the designation state status is updated when appropriate. The Target is flagged as designated when it is in the processing of being undesignated. If the Target is being undesignated, the designation state status is updated to indicate the Target was dropped.

This algorithm takes as input *designation_needs_update* and *designation_state*. This algorithm updates *designation_state*.

Algorithm 135 UpdateDroppedTargetDesignationState

```

1 function UpdateDroppedTargetDesignationState( designation_needs_update::Bool,
2                                     designation_state::DesignationState(p.E-6) )
3   const S_standard_protection_mode::UInt8 = UInt8(params().target_designation.S_standard_protection_mode)
4   const N::Z = params().target_designation.td_dna.N
5   if designation_state.is_designated
6     if designation_needs_update || designation_state.valid[Xo_IDX_DNA]
7       designation_state.timer_dna = designation_state.timer_dna + 1
8     end
9     if designation_needs_update || designation_state.valid[Xo_IDX_CSP03k]
10      designation_state.timer_protection_mode = designation_state.timer_protection_mode + 1
11    end
12    if designation_needs_update
13      CheckTargetDesignationTimers(p.135)( designation_state )
14      if !designation_state.pending_dna &&
15        (S_standard_protection_mode == designation_state.pending_protection_mode)
16        designation_state.r_ground_history = Array( R, 0 )
17      else
18        push!( designation_state.r_ground_history, NaN )
19      end
20    end
21    count::Z = length( designation_state.r_ground_history )
22    if (N < count)
23      deleteat!( designation_state.r_ground_history, 1:(count-N) )
24    end
25  end
26  if designation_state.is_designated
27    if (DESIGNATION_UNDESIGNATE_NO_ALERTS == designation_state.designated_mode) ||
28      (DESIGNATION_UNDESIGNATE_PROTECTION_MODE == designation_state.designated_mode)
29    if (Xo_STATUS_UNDESIGNATED_TIMEOUT != designation_state.status)
30      designation_state.status = Xo_STATUS_UNDESIGNATED_DROPPED
31    end
32    designation_state.r_ground_history = Array( R, 0 )
33    elseif (Xo_STATUS_UNDESIGNATED_TIMEOUT != designation_state.status) &&
34      (Xo_STATUS_UNDESIGNATED_DROPPED != designation_state.status)
35      designation_state.status = Xo_STATUS_SUSPENDED_DROPPED
36    end
37  end
38 end
```

Referenced In: AdjustTargetDesignationValidity(p.143)
--

SETTARGETDESIGNATIONMODEAVAILABILITY (Algorithm 136) sets the values in the Xo mode availability field in the TRMOWNINPUT data structure in the STM Report. The inputs are the STM Report (*report*) and the report time (*t*).

The algorithm first sets the Xo mode availability to 'Available to Run' for both Designated No Alerts

—*—*—*—*—*—*—*—*—*—*—*—

(DNA) and CSPO-3000 Xo modes. It then determines the Xo mode for every Target designated to an Xo mode and sets the appropriate Xo mode availability to 'On'. Finally, it calls IsDNAMODEUNAVAILABLETORUN and IsCSPO3000MODEUNAVAILABLETORUN, setting the appropriate Xo mode availability to 'Unavailable to Run' when the Xo mode is unavailable.

SETTARGETDESIGNATIONMODEAVAILABILITY outputs the STM Report (*report*) with the Xo mode availability set appropriately.

This algorithm takes as input *report* and *t*. This algorithm updates *report.trm_input.own*.

Algorithm 136 SetTargetDesignationModeAvailability

```

1 function SetTargetDesignationModeAvailability(report:::StmReport(p. E-1), t:::R)
2   report.trm_input.own.xo_availability[Xo_IDX_CSP03k] = Xo_AVAILABILITY_AVAILABLE_TO_RUN
3   report.trm_input.own.xo_availability[Xo_IDX_DNA] = Xo_AVAILABILITY_AVAILABLE_TO_RUN
4   designated_target_ids::Vector{UInt32} = GetDesignatedTargetIDs(p. 73)()
5   for id::UInt32 in designated_target_ids
6     designation_state::DesignationState(p. E-6) = target_db[id].designation_state
7     if (designation_state.active_protection_mode == PROTECTION_MODE_Xo_CSP03k) ||
8       (designation_state.pending_protection_mode == PROTECTION_MODE_Xo_CSP03k)
9       report.trm_input.own.xo_availability[Xo_IDX_CSP03k] = Xo_AVAILABILITY_ON
10    end
11    if designation_state.active_dna || designation_state.pending_dna
12      report.trm_input.own.xo_availability[Xo_IDX_DNA] = Xo_AVAILABILITY_ON
13    end
14  end
15  if IsDNAModeUnavailableToRun(p. 146)()
16    report.trm_input.own.xo_availability[Xo_IDX_DNA] = Xo_AVAILABILITY_UNAVAILABLE_TO_RUN
17  end
18  if IsCSPO3000ModeUnavailableToRun(p. 146)( designated_target_ids, t )
19    report.trm_input.own.xo_availability[Xo_IDX_CSP03k] = Xo_AVAILABILITY_UNAVAILABLE_TO_RUN
20  end
21 end
```

Referenced In: GenerateStmReport(p. 97)
--

IsCSPO3000MODEUNAVAILABLETORUN (Algorithm 137) returns whether or not the CSPO-3000 Xo mode is unavailable to run. It uses a vector of ids for Targets designated to any Xo mode (*designated_target_ids*) and the report time (*t*) as inputs.

CSPO-3000 Xo mode is unavailable when resolution advisories (RAs) cannot be produced or ownership is above an altitude threshold.

The barometric altitude in feet (*baroalt*) is obtained from BAROALTATTOA. If ownership is in TA/RA operational mode (*own.trm_opmode* is *OPMODE_RA*) and has a valid barometric altitude (*baroalt* is not *Nan*), then the geometric criterion is evaluated. By default, the low altitude threshold, *H_threshold_lo_cspo3000*, is used as the altitude threshold. However, if any of the Targets in *designated_target_ids* are designated to the CSPO-3000 Xo mode, hysteresis is provided and the altitude threshold is raised to *H_threshold_hi_cspo3000*. Ownership barometric altitude is then compared with the altitude threshold to set the unavailable flag (*is_unavailable_to_run*).

If ownership is in an operational mode other than TA/RA or has an invalid barometric altitude, *is_unavailable_to_run* is set to TRUE.

IsCSPO3000MODEUNAVAILABLETORUN returns the value of *is_unavailable_to_run*.

This algorithm takes as input *designated_target_ids* and *t*. This algorithm returns *is_unavailable_to_run*.

to_run.

Algorithm 137 IsCSPO3000ModeUnavailableToRun

```

1 function IsCSPO3000ModeUnavailableToRun( designated_target_ids::Vector{UInt32}, t::R )
2   const H_threshold_lo::R = params().target_designation.td_cspo3000.H_threshold_lo
3   const H_threshold_hi::R = params().target_designation.td_cspo3000.H_threshold_hi
4   is_unavailable_to_run::Bool = false
5   alt_threshold::R = H_threshold_lo
6   baroalt::R = BaroAltAtToa(p.78)( t )
7   if (own.trm_opmode == OPMODE_RA) && !isnan( baroalt )
8     for id::UInt32 in designated_target_ids
9       designation_state::DesignationState(p.E-6) = target_db[id].designation_state
10      if (designation_state.active_protection_mode == PROTECTION_MODE_Xo_CSP03k) ||
11        (designation_state.pending_protection_mode == PROTECTION_MODE_Xo_CSP03k)
12        alt_threshold = H_threshold_hi
13      end
14    end
15    if (alt_threshold < baroalt)
16      is_unavailable_to_run = true
17    end
18  else
19    is_unavailable_to_run = true
20  end
21  return is_unavailable_to_run::Bool
22 end

```

Referenced In: SetTargetDesignationModeAvailability(p. 145), UpdateCSPO3000Validity(p. 129)
--

IsDNAMODEUNAVAILABLETORUN (Algorithm 138) returns whether or not the Designated No Alerts (DNA) Xo mode is unavailable to run. No criteria have been identified for making DNA Xo mode unavailable at the console. IsDNAMODEUNAVAILABLETORUN always returns FALSE. This algorithm provides an interface to support maturation of the Xo operational concept.

Algorithm 138 IsDNAModeUnavailableToRun

```

1 function IsDNAModeUnavailableToRun()
2   is_unavailable_to_run::Bool = false
3   return is_unavailable_to_run::Bool
4 end

```

Referenced In: UpdateDNAValidity(p. 131), SetTargetDesignationModeAvailability(p. 145)

2.6.8 Proximity Estimation

The PROXIMITYESTIMATION (Algorithm 139) algorithm determines whether the intruder is within the range and altitude thresholds for issuing a proximity advisory. PROXIMITYESTIMATION checks for NAR using the *valid_vert* flag from the given track. It also checks for a bearingless target by using the *valid_cart* flag from a given track, when available. The algorithm does not test for a valid range value because the Surveillance and Tracking Module (STM) filters out tracks with no valid range.

This algorithm takes as input *intruder*, *z_own*, *z_int*, *r_ground*, *valid_vert*, and *valid_cart*. This algorithm updates *intruder.is_proximate*.

Algorithm 139 ProximityEstimation

```
1 function ProximityEstimation( intruder::TRMIIntruderInput(p.E-20), z_own::R, z_int::R, r_ground::R, valid_vert::  
    Bool, valid_cart::Bool )  
2     const D_proximity_range_threshold::R = params().display.proximity_range_threshold  
3     const H_proximity_altitude_threshold = params().display.proximity_altitude_threshold  
4     z_rel::R = abs(z_own - z_int)  
5     intruder.is_proximate = false  
6     if (r_ground <= D_proximity_range_threshold) && (z_rel < H_proximity_altitude_threshold)  
7         if (valid_vert) || (valid_cart)  
8             intruder.is_proximate = true  
9         end  
10    end  
11 end
```

Referenced In: AddADSBTrackToReport(p. 110), AddModeSTrackToReport(p. 102), AddModeCTrackToReport(p. 114)

This page intentionally left blank.

3 Threat Resolution Module Description

3.1 System Overview

The main data and processing components of the Threat Resolution Module (TRM) are shown in Figure 3-1. Each processing component in the figure is described in a separate section of the document and is invoked from VERTICALTRMUPDATE (Algorithm 140). The variables in the figure, and their minimal types, are described in Table 3-1. The TRM inputs and outputs are discussed in Sections 3.1.1 and 3.1.2.

VERTICALTRMUPDATE performs some initial data preparation prior to commencing threat processing. GETVTRMOWNDATA is used to update ownship state information used for TRM processing.

Threat processing through the TRM is linear.

The STATEANDCOSTESTIMATION component (Section 3.2) initializes intruder state to account for special processing modes and intruder coordination capabilities and evaluates the probabilistic threat represented by each intruder in isolation.

The resulting state and cost information for all intruders are then supplied to the ACTIONSELECTION component (Section 3.3), which selects a single global action for own aircraft and individual actions for each intruder aircraft.

The resulting global action, as well as the individual actions for each intruder, are supplied to the COORDINATIONSELECTION component (Section 3.4) and TRACKTHREATASSESSMENT component (Section 3.5), which produce data for each individual intruder to be used for coordination and on the onboard pilot display.

The DISPLAYLOGICDETERMINATION component (Section 3.6) uses these results, along with state information, to produce data used for visual and aural alerts on the onboard pilot display.

Finally, the GENERATETRMOUTPUT component (Section 3.7) adjusts the data used for coordination and the onboard pilot display to account for degraded surveillance, TA-Only Operational Mode, Standby Operational Mode, and Designated No Alerts (DNA) Xo mode processing. GENERATETRMOUTPUT also does housekeeping on dropped and invalid intruders, updates the TRM state variables, and packages the data for output from the TRM in the TRMREPORT.

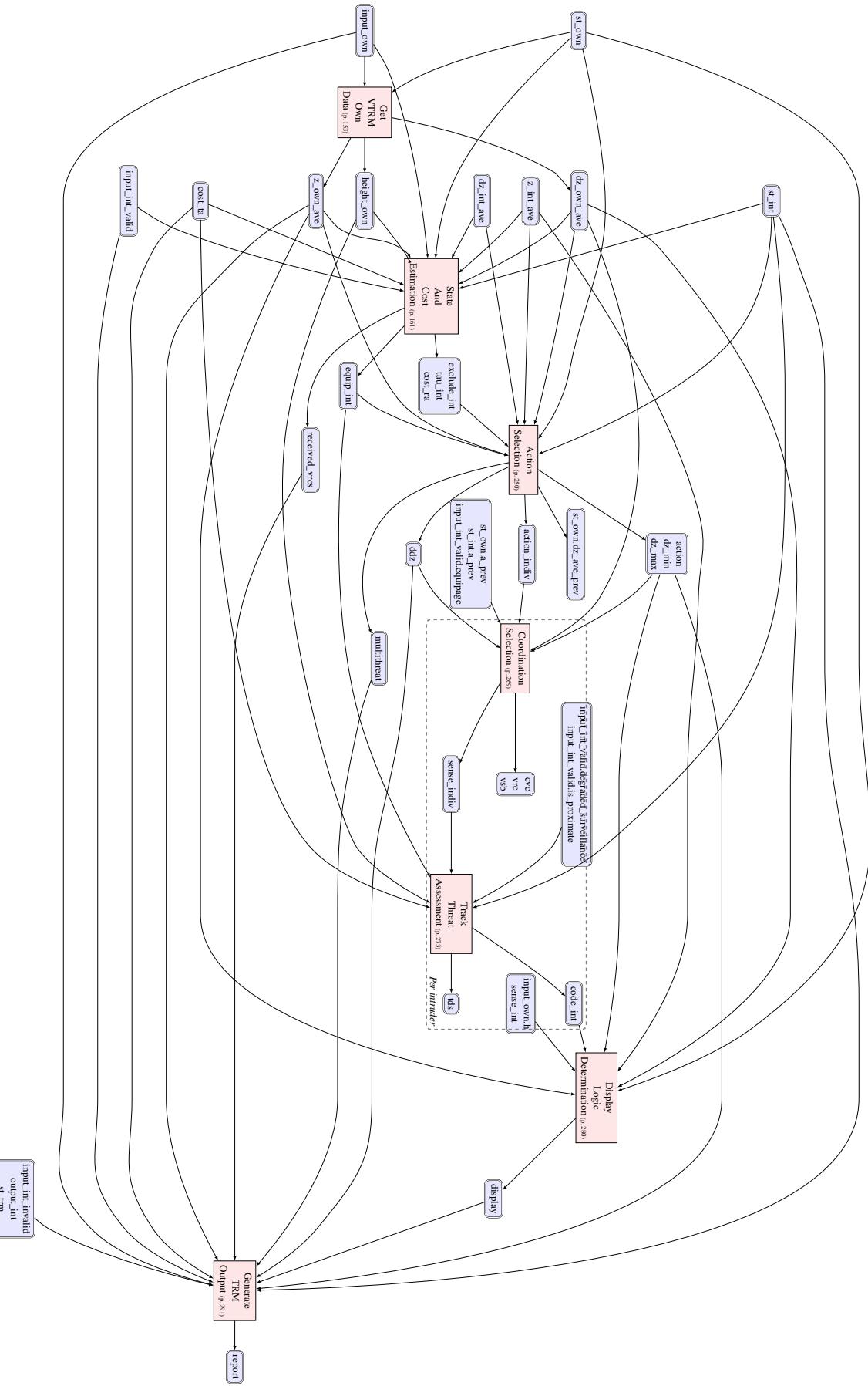


Figure 3-1. VerticalTRMUpdate overview.

Table 3-1. TRM Internal Variables - VerticalTRMUpdate (Figure 3-1)

Variable	Units	Type	Description
action	N/A	integer	Selected global action (Resolution Advisory (RA)) for own ship
action_indiv	N/A	integer	Array of current advisory action (RA) for each individual intruder
code_int	N/A	uint8	Pilot Display value: Code indicating type of advisory for each intruder
cost_ra	N/A	real	Matrix of combined RA costs of each action and each intruder
cost_ta	N/A	real	Matrix of combined TA costs of each action and each intruder
cvc	N/A	uint32	Coordination value: Cancel Vertical Complement
ddz	ft/s ²	real	Vertical acceleration associated with the selected global action
display	N/A	DisplayLogic(p. E-35)	Information for the onboard display
dz_int_ave	ft/s	real	Vector of the weighted average of vertical rate samples for each intruder
dz_max	ft/s	real	Maximum vertical rate associated with the selected global action
dz_min	ft/s	real	Minimum vertical rate associated with the selected global action
dz_own_ave	ft/s	real	Weighted average of vertical rate samples for own aircraft
equip_int	N/A	bool	Array of whether each intruder is capable of coordination
exclude_int	N/A	bool	Array of whether each intruder should be excluded from action selection
height_own	ft	real	Height of own aircraft; either radar altimeter or barometric altitude
input_int_invalid	N/A	TRMIIntruderInput(p. E-20)	Array of input data for each intruder not valid for advisory processing
input_int_valid	N/A	TRMIIntruderInput(p. E-20)	Array of input data for each intruder valid for advisory processing
input_own	N/A	TRMOwnInput(p. E-19)	Input data for own aircraft
mode_int	N/A	integer	Array of protection mode index for each intruder (not shown)
multithreat	N/A	bool	Flag indicating whether or not there are multiple threats (RAs)
output_int	N/A	TRMIIntruderData(p. E-39)	Array of coordination and display data for each intruder
received_vrcs	N/A	bool	Coordination intent vector from STM
report	N/A	TRMReport(p. E-23)	Report containing advisories for coordination and the onboard pilot display
sense_indiv	N/A	symbol	Up or down RA sense selected for an individual intruder
st_int	N/A	TRMIIntruderState(p. E-33)	Array of state information for each individual intruder
st_own	N/A	TRMOwnState(p. E-32)	State information for own aircraft
st_trm	N/A	TRMState(p. E-31)	State information for own aircraft and all intruders processed by the TRM
tau_int	s	real	Vector of expected tau (time of closest point of approach) for each intruder
tds	N/A	real	Pilot Display value: Track Display Score
vrc	N/A	uint32	Coordination value: Vertical Resolution Advisory Complement
vsb	N/A	uint32	Coordination value: Vertical Sense Bits
z_int_ave	ft	real	Vector of the weighted average of vertical samples for each intruder
z_own_ave	ft	real	Weighted average of vertical samples for own aircraft

Algorithm 140 VerticalTRMUpdate

```

1 function VerticalTRMUpdate( st_trm::TRMState(p.E-31), input::TRMInput(p.E-19) )
2   input_own::TRMOwnInput(p.E-19) = input.own
3   st_own::TRMOwnState(p.E-32)    = st_trm.st_own
4   action::Z = 0
5   dz_min::R = 0.0
6   dz_max::R = 0.0
7   ddz::R    = 0.0
8   multithreat::Bool    = false
9   received_vrcs::Vector{Bool} = zeros( Bool, 4 )
10  (st_int::Vector{TRMIIntruderState(p.E-33)}, input_int_invalid::Vector{TRMIIntruderInput(p.E-20)},
11   input_int_valid::Vector{TRMIIntruderInput(p.E-20)}, num_intruders::Z,
12   z_int_ave::Vector{R}, dz_int_ave::Vector{R},
13   mode_int::Vector{Z}, code_int::Vector{UInt8}, sense_int::Vector{Symbol},
14   cost_ta::Matrix{R}, output_int::Vector{TRMIIntruderData(p.E-39)}) =
15    VerticalTRMUpdatePrep(p.I-2)( st_trm, input )
16  (z_own_ave::R, dz_own_ave::R, height_own::R) =
17    GetVTRMOwnData(p.153)( input_own::TRMOwnInput(p.E-19), st_own::TRMOwnState(p.E-32) )
18  if (0 < num_intruders)
19    (equip_int::Vector{Bool}, exclude_int::Vector{Bool},
20     tau_int::Vector{R}, cost_ra::Matrix{R}, received_vrcs) =
21      StateAndCostEstimation(p.161)( height_own, z_own_ave, dz_own_ave, input_own,
22                                     input_int_valid, mode_int, z_int_ave, dz_int_ave, st_own, st_int, cost_ta )
23  (action, dz_min, dz_max, ddz, action_indiv::Vector{Z}, st_own.dz_ave_prev, multithreat) =
24    ActionSelection(p.250)( mode_int, cost_ra, z_int_ave, dz_int_ave, tau_int, z_own_ave, dz_own_ave,
25                           equip_int, exclude_int, st_own, st_int )
26  for j in 1:num_intruders
27    (cvc::UInt32, vrc::UInt32, vsb::UInt32, sense_indiv::Symbol) =
28      CoordinationSelection(p.269)( action_indiv[j], action, dz_min, dz_max, ddz, dz_own_ave,
29                                    st_own.a_prev, st_int[j].a_prev, input_int_valid[j].equipage )
30    (code_int[j], tds::R) =
31      TrackThreatAssessment(p.273)( mode_int[j], height_own, ToVec(p.J-3)(cost_ta[j,:]), sense_indiv,
32                                    input_int_valid[j].degraded_surveillance, input_int_valid[j].is_proximate,
33                                    (DESIGNATION_NONE != input_int_valid[j].designated_mode),
34                                    equip_int[j], st_int[j] )
35    (output_int[j]) =
36      TRMIIntruderData(p.E-39)( input_int_valid[j].id, sense_indiv, input_own.mode_s,
37                                input_int_valid[j].address, cvc, vrc, vsb, multithreat,
38                                input_int_valid[j].coordination_msg, tds, code_int[j],
39                                input_int_valid[j].designated_mode, st_int[j].processing,
40                                st_int[j].no_alerts, st_int[j].protection_mode_curr,
41                                (code_int[j] == TACODE_RA), input_int_valid[j].address, input_int_valid[j].is_icao )
42    sense_int[j] = sense_indiv
43  end
44 else
45   received_vrcs = UpdateIntruderVRC(p.89)( input_int_valid )
46   (action, dz_min, dz_max, ddz, multithreat) =
47     NoIntrudersAction(p.251)( dz_own_ave, st_own )
48 end
49 display::DisplayLogic(p.E-35) =
50   DisplayLogicDetermination(p.280)( input_own.h, z_own_ave, dz_own_ave, z_int_ave,
51                                     action, dz_min, dz_max, sense_int, code_int, st_own )
52 report::TRMReport(p.E-23) =
53   GenerateTRMOutput(p.291)( input_own, display, multithreat, action,
54                             dz_min, dz_max, ddz, z_own_ave, input_int_invalid,
55                             input_int_valid, output_int, cost_ta, received_vrcs, st_int, st_trm )
56 return report::TRMReport(p.E-23)
57 end

```

GETVTRMOWNDATA (Algorithm 141) returns the ownship barometric altitude estimate in feet, the ownship vertical rate estimate in feet/second, and the value to use for ownship height above ground level in feet. It also updates the state variables associated with low altitude resolution advisory inhibits.

GETVTRMOWNDATA first computes the weighted means for ownship barometric altitude (*z_own_ave*) and ownship vertical rate (*dz_own_ave*). It then determines the value to be used for ownship height AGL (*height_own*). The algorithm finishes by updating the setting of *inhibited* in ALTITUDEINHIBITCSTATE (*st_own.st_altitude_inhibit*) based on the current protection mode (Xa or Xo).

This algorithm takes as input *input_own* and *st_own*. The algorithm returns *z_own_ave*, *dz_own_ave*, and *height_own*. It modifies *st_own*, which is passed by reference.

Algorithm 141 GetVTRMOwnData

```

1 function GetVTRMOwnData( input_own::TRMOwnInput(p. E-19), st_own::TRMOwnState(p. E-32) )
2   z_own_ave::R = 0.0
3   dz_own_ave::R = 0.0
4   for b::OwnVerticalBelief(p. E-21) in input_own.belief_vert
5     z_own_ave = z_own_ave + (b.z * b.weight)
6     dz_own_ave = dz_own_ave + (b.dz * b.weight)
7   end
8   height_own::R = GetOwnHeight(p. 327)( input_own.h, z_own_ave )
9   for mode_idx::Z in 1:length( st_own.st_alt_inhibit )
10    UpdateAltitudeInhibitCState(p. 154)( mode_idx, input_own.h,
11                                         st_own.st_alt_inhibit[mode_idx] )
12  end
13  return ( z_own_ave::R, dz_own_ave::R, height_own::R)
14 end
```

Referenced In: VerticalTRMUpdate(p. 152)

UPDATEALTITUDEINHIBITCSTATE (Algorithm 142) is used to update the state variables used in ALTITUDEINHIBITCOST once per cycle. On the first processing cycle after start-up, the values in inhibit vector (*s_c.inhibited*) are initialized using the parameter file. The algorithm then iterates over each inhibit and compares the altitude of the own aircraft (*h_own*) to parameter defined thresholds. If the own aircraft altitude is undefined, the inhibit is disabled. If the own aircraft altitude is below the low threshold for a given inhibit (*H_lo[i]*), the inhibit is enabled. If the own aircraft altitude is above the high threshold for a given inhibit (*H_hi[i]*), the inhibit is disabled.

DETERMINELDI is used to obtain the low-level descend inhibit value (*s_c.ldi*) based on the values in *s_c.inhibited*.

This algorithm takes as input *mode_idx*, *h_own*, and *s_c*. It updates *s_c::AltitudeInhibitCState*, which is passed by reference.

Algorithm 142 UpdateAltitudeInhibitCState

```

1 function UpdateAltitudeInhibitCState( mode_idx::Z, h_own::R, s_c::AltitudeInhibitCState(p. E-42) )
2   const H_lo::Vector{R} = params().modes[mode_idx].cost_estimation.online.altitude_inhibit.H_lo
3   const H_hi::Vector{R} = params().modes[mode_idx].cost_estimation.online.altitude_inhibit.H_hi
4   B_init::Vector{Bool} = bool( params().modes[mode_idx].cost_estimation.online.altitude_inhibit.B_init )
5   if !s_c.initialized
6     for i = 1:length( B_init )
7       s_c.inhibited[i] = B_init[i]
8     end
9   s_c.initialized = true
10  end
11  for i = 1:length( B_init )
12    if isnan( h_own )
13      s_c.inhibited[i] = false
14    elseif (h_own < H_lo[i])
15      s_c.inhibited[i] = true
16    elseif (H_hi[i] < h_own)
17      s_c.inhibited[i] = false
18    end
19  end
20  s_c.ldi = DetermineLDI(p. 155)( mode_idx, s_c.inhibited )
21 end

```

Referenced In: GetVTRMOwnData(p. 153)

DETERMINELDI (Algorithm 143) returns the Low-level Descend Inhibit (LDI) value based on the values in the *inhibited* vector.

The algorithm first obtains the altitude inhibit rules from the *cost_estimation.online.altitude_inhibit* parameter structure. It also obtains the vertical rate for climb and descend RAs from the parameter *actions.corrective_rate*

The algorithm iterates over each entry in *inhibited*. There is an entry in *inhibited* to correspond with each altitude inhibit rule. In order for an altitude inhibit to be active, the value of *inhibited* for the current entry must be TRUE and a penalty is to be applied. The altitude inhibit rules use minimum and maximum vertical rates to represent the RAs pertaining to each rule.

If the maximum vertical rate is negative, indicating a descend RA, a decision is made as to whether the inhibit is for all positive descend RAs or whether it is for increased rate descend RAs only. That determination is made by comparing the maximum vertical rate in the altitude inhibit rule ($R_{dz_hi}[i]$) against the corrective rate ($R_{corrective}$). If the vertical rate is greater than the corrective rate, the altitude inhibit is for an increased rate descend RA.

Otherwise, if the maximum vertical rate is +Infinity and the minimum vertical rate is -Infinity, all RAs will be inhibited.

ldi is then set based on which active altitude inhibit encompasses the most RAs.

This algorithm takes as input *mode_idx* and *inhibited*. The algorithm returns *ldi*.

Algorithm 143 DetermineLDI

```

1 function DetermineLDI( mode_idx::Z, inhibited::Vector{Bool} )
2   const C_inhibit::Vector{R} =
3     params().modes[mode_idx].cost_estimation.online.altitude_inhibit.C_inhibit
4   const R_dz_hi::Vector{R} =
5     params().modes[mode_idx].cost_estimation.online.altitude_inhibit.R_dz_hi
6   const R_dz_lo::Vector{R} =
7     params().modes[mode_idx].cost_estimation.online.altitude_inhibit.R_dz_lo
8   const R_corrective::R = params().actions.corrective_rate
9   ldi::UInt8 = LDI_NONE
10  inhibited_inc_descend::Bool = false
11  inhibited_descends::Bool = false
12  inhibited_all::Bool = false
13  for i = 1:length( inhibited )
14    if inhibited[i] && (0 < C_inhibit[i])
15      if (R_dz_hi[i] < 0)
16        if (abs( R_dz_hi[i] ) <= R_corrective)
17          inhibited_descends = true
18        else
19          inhibited_inc_descend = true
20        end
21      elseif (Inf == R_dz_hi[i]) && (-Inf == R_dz_lo[i])
22        inhibited_all = true
23      end
24    end
25  end
26  if inhibited_all
27    ldi = LDI_ALL
28  elseif inhibited_descends
29    ldi = LDI_DESCENDS
30  elseif inhibited_inc_descend
31    ldi = LDI_INCREASE_DESCEND
32  end
33  return ldi::UInt8
34 end

```

Referenced In: UpdateAltitudeInhibitCState(p.154)

3.1.1 Inputs and Outputs to the TRM

One of the primary ways in which ACAS X improves on previous collision avoidance efforts is in explicitly considering uncertainty about the conditions of the local airspace. Instead of single-sample measurements of various physical quantities, the system works with collections of weighted samples (called beliefs), where every sample of inputs is assigned a weight representing the probability that the actual values are close to the measurements in the sample.

Among the inputs to the TRM, there are three such beliefs: INTRUDERHORIZONTALBELIEF (p.E-21), INTRUDERVERTICALBELIEF (p.E-21), and OWNVERTICALBELIEF (p.E-21). INTRUDERHORIZONTALBELIEF contains relative horizontal distance and rate beliefs from own aircraft to an intruder aircraft. INTRUDERVERTICALBELIEF and OWNVERTICALBELIEF contain altitude and vertical rate beliefs; they are used for intruder aircraft and own aircraft, respectively.

The entire input to the TRM, which includes the own and intruder beliefs, is defined by a nested type TRMINPUT (p.E-19). TRMINPUT contains own aircraft data, TRMOWNINPUT (p.E-19), and an array of intruder aircraft data of type TRMINTRUDERINPUT (p.E-20), one per known intruder.

Most TRM input is produced by the STM as output from GENERATESTMREPORT (Algorithm 96). However, in order to utilize the most recent coordination inputs to the STM, the TRM calls back to the STM using UPDATEINTRUDERVRC (Algorithm 88). This call updates the *vrc* field in

TRMINTRUDERINPUT for all intruders being processed by the TRM and returns the coordination *received_vrcs*, for output as the Resolution Advisory Complements (RAC).

The output from the TRM is defined by another nested type TRMREPORT (p.E-23). TRMREPORT contains information for an onboard pilot display, TRMDISPLAYDATA (p.E-24); an array of per-intruder data for coordination with other ACAS X-equipped aircraft, TRMCOORDINATIONINTERROGATIONDATA (p.E-25); and information about conformant own vertical rates and acceleration and internal threat processing TRMDEBUGDATA (p.E-29). It also contains information for air-to-air broadcast, TRMRABROADCASTDATA (p.E-27) and information for transmission to a ground station, TRMGROUNDMMSGDATA (p.E-27).

In addition, the TRM inputs and outputs state information, TRMSTATE, details of which are provided in Section 3.1.2. VERTICALTRMUPDATE uses a helper function, VERTICALTRMUPDATEPREP (Algorithm 356), to check the validity of the inputs, create the intruder state structures for new intruders, and create arrays for processing intruder data. VERTICALTRMUPDATEPREP is not described in this document but is included in Appendix I.

3.1.2 System Architecture

An effort was made to make the data flow in the diagrams explicit. However, the TRM has some state, or memory, which is updated each time the main algorithm loop is run.

This memory is captured by TRMSTATE, which is passed as an argument into the main VERTICALTRMUPDATE method. TRMSTATE holds state information on previous advisories for own aircraft (TRMOWNSTATE), state information on previous advisories and online cost settings for each intruder (TRMINTRUDERSTATE), and configuration parameters (PARAMSFILE_TYPE). Within TRMINTRUDERSTATE, state information for each online cost is stored in a data structure of type ONLINECOSTSTATE. These online cost settings are described more fully in Appendix E.6.

The states, and consequently the online cost settings and best individual action with respect to own aircraft, are different for each intruder aircraft. Thus, updates to the state variables for one intruder should not affect those of another intruder. To that end, the system maintains a vector of TRMINTRUDERSTATE, one for each individual intruder. The associated ONLINECOSTSTATE information is also specific to an individual intruder. To ensure separation of online cost settings for each intruder, the state for each online cost is initialized and stored separately by ONLINECOSTSTATE for each TRMINTRUDERSTATE.

The previous global advisory of the TRMOWNSTATE is represented by *st_own.a_prev*. The actions and rates pertaining to the global advisory issued at the previous time step include: *action*, *dz_min*, *dz_max*, and *ddz*. An additional variable pertaining to the global advisory, *multithreat*, is a boolean that tracks whether the previous issued advisory was due to multiple intruders. These variables are all stored in *st_own.a_prev*. Additionally, the state variable *dz_ave_prev* is used to persist the vertical rates for a Maintain advisory.

The display logic values pertaining to the global advisory issued at the previous time step are: *action_prev*, *word_prev*, *turn_off_aurals_prev*, *crossing_prev*, and *ra_suppressed_prev*, which are stored in *st_own*. They are used and set in DISPLAYLOGICDETERMINATION.

Several TRMOWNSTATE variables are needed for setting fields in the TRMRABROADCASTDATA and TRMGROUNDMMSGDATA outputs. *st_own.ra_broadcast_prev* is a copy of the TRMRABROADCASTDATA output from the previous cycle. Information about the most recent threat for the TRMGROUNDMMSGDATA output is stored in *st_own.tid_latest*, *st_own.tti_latest*, and *st_own.tgtid_latest*.

The individual intruder action and corresponding rates from the previous time step are also needed for use within the per-intruder processing blocks. The previous advisory of the TRMINTRUDER-STATE is represented by *st_intruder.a_prev*. The state variables pertaining to individual intruder advisories include: *action*, *dz_min*, *dz_max*, and *sense_prev*, which are stored in *st_intruder.a_prev*. The intruder advisory state variable *st_intruder.a_prev.ra_prev* records whether the previous action was a resolution advisory (RA). The intruder advisory belief state variable *b_prev* stores the vertical probabilities, *b_prev.w_vert_prev*, along with a flag, *b_prev.need_init*, indicating whether the probabilities were stored on the previous cycle. State is also maintained for coordination with the intruder and for the type TRM processing performed for the intruder. State variables related to coordination include: *vrc_prev*, *cvc_prev*, and *equipage_prev*. The intruder state variables *address*, *is_icao*, and *coordination_msg_prev* are stored input values used for setting fields in TRMCOORDINATIONINTERROGATIONDATA for dropped intruders. The *st_intruder.ra_found* state variable is used for setting the id of the most recent threat, TID, in TRMGROUNDMMSGDATA. The intruder state variable *st_intruder.time_since_ta* stores the traffic advisory timer for this intruder. State variables related to the processing state for the intruder are: *protection_mode_prev*, *protection_mode_curr*, *processing*, and *status*, which are stored in *st_intruder*.

The intruder state variable *id* is the target identifier supplied by the STM. It is used to associate the TRMINTRUDERINPUT structure for an individual intruder with the appropriate intruder state.

All of the rates are set to *NaN* at initialization. Also at initialization, the actions are set to 0 and *st_own.a_prev.multithreat* is set to FALSE. After each cycle, the state variables associated with the global advisory, *st_own.a_prev*, are set to *action*, *dz_min*, *dz_max*, and *multithreat* as output by the ACTIONSELECTION component. The individual action and rate variables, *st_intruder.a_prev*, are set from the *act_indiv*, *dz_min_indiv*, *dz_max_indiv*, and *sense_indiv*, values within the COORDINATIONSELECTION component.

Some of the state values are used for maintaining intermediate information for cost estimation and action selection. For TRMINTRUDERSTATE, *st_cost_on* and *st_arbitrate* maintain state information that is associated with the individual intruder state. *st_int.st_arbitrate* stores context-specific state information for ACTIONARBITRATION. *st_int.st_cost_on* stores context-specific state information for ONLINEUNCOORDINATEDACTIONCOSTESTIMATION and ONLINECOORDINATEDACTIONCOSTESTIMATION. For TRMOWNSTATE, *st_multithreat_cost_balancing* and *st_arbitrate* maintain state information that is associated with the global state. *st_own.st_multithreat_cost_balancing* stores context-specific state information for MULTITHREATCOSTBALANCING. *st_own.st_arbitrate* stores context-specific state information for ACTIONARBITRATION.

The state variables associated with processing are set at initialization, in INTRUDERPREP, or as part of output processing as described in Section 3.7. The state variables associated with coordination are set at initialization, in INTRUDERPREP, in UPDATEINTRUDERINPUTS, or as part of output processing as described in Section 3.7.

The parameters contained in *params*, of type PARAMSFILE_TYPE are referenced by name in the algorithms. Their values are stored in a parameter file written in JavaScript Object Notation (JSON). The parameters in that JSON file are described in Appendix F.

3.1.3 Intruder Designation and Protection Modes

Like its predecessor, TCAS, ACAS X is designed to prevent collision with multiple simultaneous intruders. ACAS X supports Operational Modes that pertain across intruders: Traffic Advisory/Resolution Advisory (TA/RA), Traffic Advisory Only (TA-Only), and Standby. While in TA/RA mode,

ACAS X allows different intruders to be treated differently and allows the same intruder to be treated differently at different times. An intruder can be designated for special processing or it can be designated for standard processing with different parameters and lookup tables, used in STATE-ANDCOSTESTIMATION. When different parameters and lookup tables are used they are referred to collectively as a protection mode. The parameters that depend on protection mode can be found in Appendix F.3, under the “modes” section in the JSON parameter specification.

Two general classes of ACAS X TA/RA processing are described in this document: ‘Xa’ is the class of a general intruder, representing the primary focus of development of ACAS X. ‘Xo’ is the class tailored for intruders in a special operational mode, such as Closely Spaced Parallel Operations (CSPO).

For Xo, an intruder can be designated for special processing, Designated No Alerts (‘DNA’), or for standard processing with different parameters and lookup tables, ‘CSPO 3000’. Xo operational use and requirements are described in the ACAS Xo System Requirements Document (SRD) [5].

ACAS X traffic designation support is provided via the Aircraft Surveillance Applications (ASA) System described in DO-317B [15]. The Aircraft Surveillance and Separation Assurance Processing (ASSAP) function in the ASA System provides the interface to the ACAS X system.

In the TRM, Xo ‘DNA’ operation is independent of protection mode and is indicated by the *dna* flag in TRMINTRUDERINPUT. DNA suppresses Resolution Advisories and Traffic Advisories for the designated intruder only. TA-Only Operational Mode, Standby Operational Mode, and DNA processing are handled by the TRM as special cases that call for adjustments to the outgoing co-ordination messages and/or the pilot display. All three are discussed in GENERATE TRM OUTPUT.

Xo ‘CSPO 3000’ is the protection mode tailored for an intruder aircraft paired with own aircraft in Closely Spaced Parallel Operations (CSPO). The currently defined operational standard is CSPO 3000.

The codes for identifying the two currently supported protection modes are given in CONSTANTS (Appendix D):

1 = Xa
2 = Xo CSPO3k

Other protection modes may be defined in the future.

Switching from one protection mode to another is prohibited when a Resolution Advisory is active.

GETPROTECTIONMODEINDEX (Algorithm 278) takes the protection mode code associated with an individual intruder (*input_int_valid.protection_mode*), and returns the index for the parameters associated with that mode in the “modes” section of the JSON parameter file. The index is stored in the *mode_int* array for quick reference.

This algorithm takes as input *input_int_valid.protection_mode*. This algorithm returns the index for the parameters associated with that mode in the “modes” section of the JSON parameter file.

3.2 State and Cost Estimation

An overview of the STATEANDCOSTESTIMATION component of the Threat Resolution Module is shown in Figure 3-2. This component evaluates the probabilistic threat represented by each intruder in isolation.

STATEANDCOSTESTIMATION (Algorithm 144) has four general processing elements.

- Initialize the variables.
- Loop over intruders to compute values not dependent on resolution advisory coordination inputs. This first loop calls **INTRUDERPREP** (Section 3.2.1), **STATEESTIMATION** (Section 3.2.2), **OFFLINECOSTESTIMATION** (Section 3.2.3). and **ONLINEUNCOORDINATEDCOSTESTIMATION** (Section 3.2.4).
 - **INTRUDERPREP** preprocesses the inputs for each intruder to account for ACAS X operational modes, such as Traffic Advisory Only (TA-Only).
 - **STATEESTIMATION** produces vertical samples and tau beliefs for estimating offline and online costs.
 - **OFFLINECOSTESTIMATION** involves multilinear interpolation of a large data file and outputs.
 - **ONLINEUNCOORDINATEDCOSTESTIMATION** estimates online coordination-independent costs for each action based on the current and previous dynamic and advisory states of own aircraft and the individual intruder aircraft.
- Acquire the most recent VRC values for the known intruders using **UPDATEINTRUDERINPUTS** (Section 3.2.5).
- Loop over intruders to compute cost values dependent on resolution advisory coordination inputs and then compute the cost totals. This final loop calls **ONLINECOSTESTIMATION** (Section 3.2.6) and **INDIVIDUALCOSTESTIMATION** (Section 3.2.7).
 - **ONLINECOSTESTIMATION** estimates online coordination-dependent costs for each action based on the current and previous dynamic and advisory states of own aircraft and the individual intruder aircraft.
 - **INDIVIDUALCOSTESTIMATION** adjusts and fuses the offline and online costs for the individual intruder aircraft.

Each of these algorithms is described in detail below.

Descriptions of the **STATEANDCOSTESTIMATION** output variables are found in Table 3-2.

Table 3-2. TRM Algorithm Output Variables - StateAndCostEstimation

Variable	Units	Type	Description
cost_ra	N/A	real	Matrix of intruder-specific costs to be used for issuing RAs
cost_ta	N/A	real	Matrix of intruder-specific costs to be used for issuing TAs
dz_int_ave	ft/s	real	Vector of averaged vertical rate for each intruder
equip_int	N/A	bool	Vector of whether each intruder can produce RA coordination messages
exclude_int	N/A	bool	Vector of whether each intruder should be excluded from RA processing
mode_int	N/A	integer	Vector of protection mode index for each intruder
received_vrcs	N/A	bool	Union of all VRCs received by STM; used as RAC
st_int	N/A	TRMIintruderState(p.E-33)	Vector of state information for each intruder
tau_int	s	real	Vector of expected time of closest point of approach for each intruder
z_int_ave	ft	real	Vector of averaged barometric altitude for each intruder

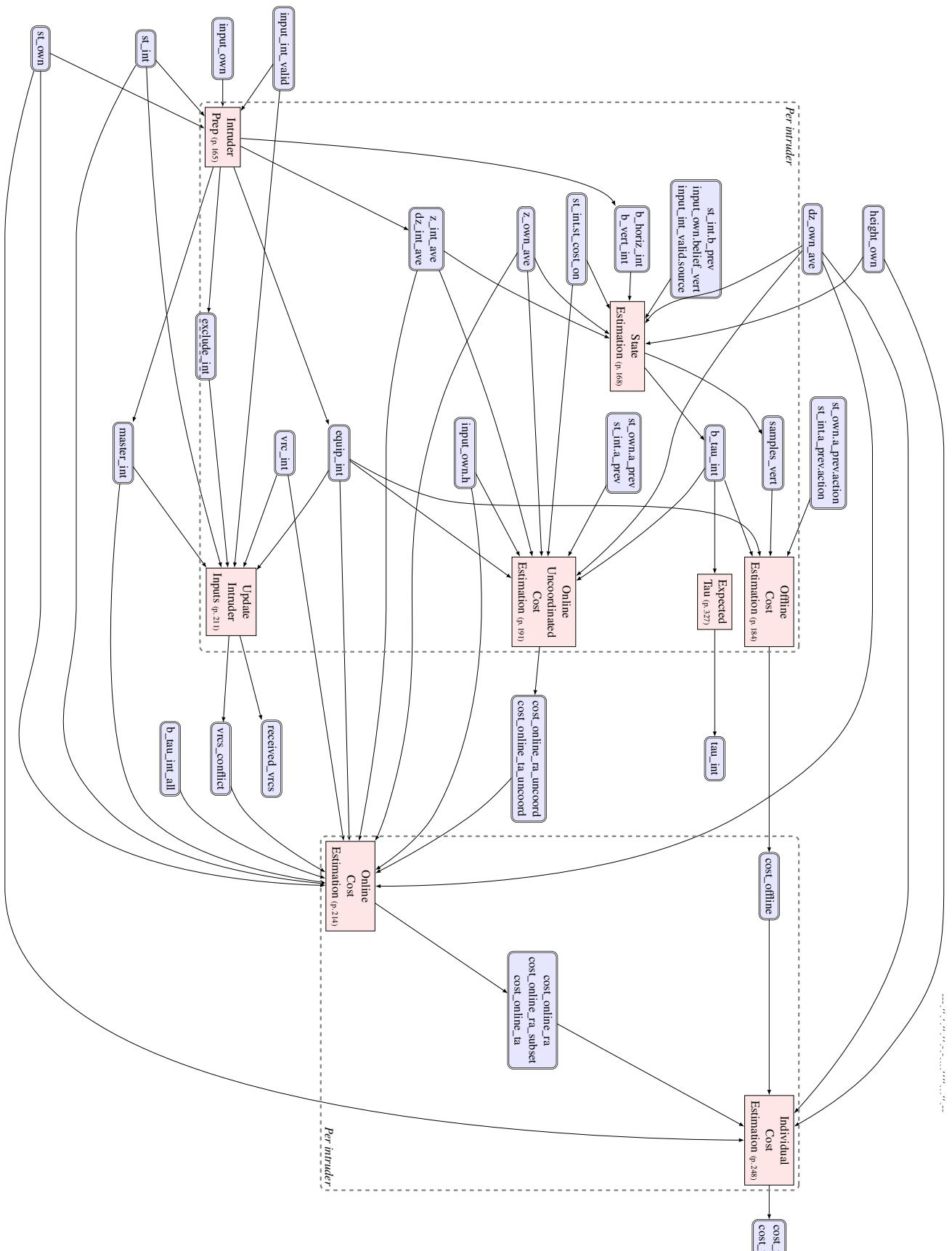


Figure 3-2. StateAndCostEstimation overview.

Algorithm 144 StateAndCostEstimation

```

1 function StateAndCostEstimation( height_own::R, z_own_ave::R, dz_own_ave::R, input_own::TRMOwnInput(p.E-19),
2                               input_int_valid::Vector{TRMIIntruderInput(p.E-20)},
3                               mode_int::Vector{Z}, z_int_ave::Vector{R}, dz_int_ave::Vector{R},
4                               st_own::TRMOwnState(p.E-32), st_int::Vector{TRMIIntruderState(p.E-33)},
5                               cost_ta::Matrix{R} )
6   const N_actions::Z = params().actions.num_actions
7   const N_intruders::Z = length( input_int_valid )
8   cost_offline::Matrix{R} = zeros( R, N_intruders, N_actions )
9   cost_online_ta_uncoord::Matrix{R} = zeros( R, N_intruders, N_actions )
10  cost_online_ra_uncoord::Matrix{R} = zeros( R, N_intruders, N_actions )
11  cost_ra::Matrix{R} = zeros( R, N_intruders, N_actions )
12  equip_int::Vector{Bool} = zeros( Bool, N_intruders )
13  master_int::Vector{Bool} = zeros( Bool, N_intruders )
14  exclude_int::Vector{Bool} = zeros( Bool, N_intruders )
15  vrc_int::Vector{UInt32} = zeros( UInt32, N_intruders )
16  tau_int::Vector{R} = zeros( R, N_intruders )
17  b_tau_int_all::Vector{Vector{TauBelief(p.E-38)}} = []
18#
19  for j in 1:N_intruders
20    (mode_int[j], equip_int[j], master_int[j], exclude_int[j],
21     b_horiz_int::Vector{IntruderHorizontalBelief(p.E-21)}, b_vert_int::Vector{IntruderVerticalBelief(p.E-21)},
22     z_int_ave[j], dz_int_ave[j]) =
23      IntruderPrep(p.165)( input_own, input_int_valid[j], st_own, st_int[j] )
24    (samples_vert::Vector{CombinedVerticalBelief(p.E-34)}, b_tau_int::Vector{TauBelief(p.E-38)}) =
25      StateEstimation(p.168)( mode_int[j], st_int[j].b_prev,
26                             input_own.belief_vert, b_vert_int, b_horiz_int,
27                             height_own, z_own_ave, dz_own_ave, z_int_ave[j], dz_int_ave[j],
28                             input_int_valid[j].source, st_int[j].st_cost_on )
29    (cost_offline[j,:]) =
30      OfflineCostEstimation(p.184)( mode_int[j], samples_vert, b_tau_int, equip_int[j],
31                                   st_own.a_prev.action, st_int[j].a.prev.action )
32    (cost_online_ra_uncoord[j,:], cost_online_ta_uncoord[j,:]) =
33      OnlineUncoordinatedCostEstimation(p.191)( mode_int[j], input_own.h, z_own_ave, dz_own_ave,
34                                                z_int_ave[j], dz_int_ave[j], b_tau_int, st_own.a_prev, st_int[j].a_prev,
35                                                equip_int[j], st_int[j].st_cost_on )
36    tau_int[j] = ExpectedTau(p.327)( b_tau_int, false )
37    push!( b_tau_int_all, b_tau_int )
38  end
39#
40# IMPORTANT: Input any UF16UDS30 messages via ReceiveUF16UDS30() by this point
41#
42 (received_vrcs::Vector{Bool}, vrcs_conflict::Bool) =
43   UpdateIntruderInputs(p.211)( input_int_valid, equip_int, master_int, exclude_int, st_int, vrc_int )
44#
45  for j in 1:N_intruders
46    (cost_online_ra::Vector{R}, cost_online_ra_subset::Vector{R}, cost_online_ta::Vector{R}) =
47      OnlineCostEstimation(p.214)( mode_int[j], input_own.h, z_own_ave, dz_own_ave,
48                                   z_int_ave[j], dz_int_ave[j], b_tau_int_all[j], st_own, st_int[j], master_int[j],
49                                   vrc_int[j], equip_int[j], vrcs_conflict, ToVec(p.J-3)(cost_online_ra_uncoord[j,:]),
50                                   ToVec(p.J-3)(cost_online_ta_uncoord[j,:]) )
51    (cost_ra[j,:], cost_ta[j,:]) =
52      IndividualCostEstimation(p.248)( mode_int[j], height_own, dz_own_ave, st_own,
53                                       ToVec(p.J-3)(cost_offline[j,:]), cost_online_ra, cost_online_ra_subset,
54                                       cost_online_ta )
55  end
56  return (equip_int::Vector{Bool}, exclude_int::Vector{Bool},
57          tau_int::Vector{R}, cost_ra::Matrix{R}, received_vrcs::Vector{Bool})
58 end

```

Referenced In: VerticalTRMUpdate(p.152)

3.2.1 Intruder Prep

INTRUDERPREP (Algorithm 145) is used to prepare the intruder inputs for processing. It sets fields in the intruder state (*st_int*) and returns values to be stored in processing arrays.

The value of *st_int.no_alerts* is set to reflect whether the Designated No Alerts (DNA) Xo mode is active for this intruder. Since DNA suppresses traffic advisories as well as resolution advisories, it is not registered as a processing mode. Instead, the DNA setting (*st_int.no_alerts*) is checked whenever special processing is required for DNA.

Next the value of *st_int.is_threat* is cleared to indicate that this intruder is not recognized as a threat for this cycle. The output variable *belief_vert_int* is initialized with the contents of *input_int.belief_vert*.

The first conditional block sets the processing mode based on the quality of the surveillance for this intruder and the ACAS X operational mode. The TRM processing is adjusted if the quality of the surveillance is inadequate for producing resolution advisories. The quality of the surveillance from the STM is indicated by *input_int.degraded_surveillance*, which is a bit vector. Each bit represents a specific type of degraded surveillance. If no bits are set, the value of *input_int.degraded_surveillance* is 0 (*DEGRADED_SURVEILLANCE_NONE*) indicating no degradation. TRM processing for this intruder is based on the setting of the *degraded_surveillance* bits as follows:

- A non-zero value for the *DEGRADED_SURVEILLANCE_NAR* bit indicates Non-Altitude Reporting (NAR) surveillance. This type of surveillance is inadequate for producing resolution advisories (RAs); it is adequate for producing traffic advisories (TAs) only. The intruder is flagged by INTRUDERPREP for degraded surveillance processing: (*st_int.processing* is set to *RA_PROCESSING_DEGRADED_SURVEILLANCE*). The vertical beliefs for own aircraft are used to populate *belief_vert_int*, replacing the intruder vertical beliefs missing in the input. An intruder with this type of surveillance degradation will be excluded from determination of the global RA.
- A non-zero value for the *DEGRADED_SURVEILLANCE_ADSB_ONLY* bit indicates ADS-B only surveillance, with no active validation. This type of surveillance has been deemed inadequate for producing resolution advisories (RAs); it is adequate for producing traffic advisories (TAs) only. The intruder is flagged by INTRUDERPREP for degraded surveillance processing: (*st_int.processing* is set to *RA_PROCESSING_DEGRADED_SURVEILLANCE*). An intruder with this type of surveillance degradation will be excluded from determination of the global RA.
- A non-zero value for the *DEGRADED_SURVEILLANCE_NO_BEARING* bit indicates surveillance with no valid bearing estimate. This type of surveillance is adequate for producing both resolution advisories (RAs) and traffic advisories (TAs). There is no special processing in INTRUDERPREP or the rest of the TRM for this type of surveillance degradation. Because no special processing is required, this case is not called out in the conditional.
- A non-zero value for the *DEGRADED_SURVEILLANCE_REDUCED* bit indicates surveillance with an update rate of less than 1 Hz. This type of surveillance is inadequate for producing vertical resolution advisories (RAs) and traffic advisories (TAs). The intruder is invalid for vertical TRM processing. An intruder with this type of degraded surveillance **must** be omitted from the inputs to INTRUDERPREP. For that reason, this value for *degraded_surveillance* is not called out in the conditional.
- A value of *DEGRADED_SURVEILLANCE_ALL* indicates every degraded surveillance bit is set. This setting flags the input for dropped intruder processing in the TRM when the intruder

is designated for Xo processing. The intruder is invalid for TRM processing. An intruder with this degraded surveillance setting **must** be omitted from the inputs to INTRUDERPREP. For that reason, this value for *degraded_surveillance* is not called out in the conditional.

When the surveillance quality is adequate for producing both RAs and TAs, the ownship ACAS X operational mode setting (*input_own.opmode*) is used to determine the type of TRM processing to be used for the intruder as follows:

- When *input_own.opmode* is *OPMODE_RA*, own aircraft is in ACAS X TA/RA operational mode. *st_int.processing* is set to *RA_PROCESSING_GLOBAL_TARA*. If the previous cycle used degraded surveillance processing for this intruder, the full cost state is reset. A full reset forces a reset of the initialization counter in INITIALIZATIONCSTATE to enforce the initial delay for track stabilization. It fully clears the online cost state for selecting resolution advisories and determining traffic advisories. Otherwise, there is no reset of the online cost state in this algorithm. In the case of an intruder that was invalid on the previous cycle, the entire intruder online cost state would have been reset on the previous cycle.
- When *input_own.opmode* is *OPMODE_TA*, own aircraft is in ACAS X TA-Only operational mode. *st_int.processing* is set to *RA_PROCESSING_GLOBAL_TA_Only*. The advisory state was partially reset on the previous cycle. The online cost state is not reset so RAs will be produced for conversion to TAs. Since the processing mode is *RA_PROCESSING_GLOBAL_TA_Only*, no VRCs will be output to any intruder.
- When *input_own.opmode* is *OPMODE_STANDBY*, own aircraft is in Standby ACAS X operational mode. In Standby mode, proximity advisories can be produced, but no RAs or TAs are produced. When ownship is in Standby ACAS X operational mode, all intruders are invalid for vertical TRM processing. All intruders **must** be omitted from inputs to INTRUDERPREP. As a result, *input_own.opmode* will never be set to *OPMODE_STANDBY* in this algorithm. For that reason, this setting is not called out in the conditional.

Next, the values to be used for the intruder altitude (*z_int_ave*) and vertical rate (*dz_int_ave*) are determined by taking the weighted mean of the vertical beliefs for each of those values.

In the second conditional block, the selected protection mode is checked and validated. A change of protection mode is allowed only if there is either no active resolution advisory or Global TA-only processing is in effect. Any rules regarding use of Designated No Alerts (DNA) with a specific protection mode are enforced by the STM.

The values of the current protection mode (*st_int.protection_mode_curr*), protection mode index (*mode_int*), whether the intruder is excluded from determination of the global RA (*exclude_int*), and the online cost state are set based on the results of the processing above.

mode_int is an array index used throughout the TRM to retrieve either Xa or Xo CSPO-3000 parameters for use. The GETPROTECTIONMODEINDEX helper algorithm returns the index into the parameter array *params().modes*. The value of *protect_mode_int* must be validated prior to running the TRM. *exclude_int* is set to TRUE when the intruder is marked as having certain types of degraded surveillance.

The values of the designated mode state variable (*st_int.designated_mode*), whether the intruder is CAS-equipped (*equip_int*), and whether the CAS-equipped intruder is a master to own aircraft (*master_int*) are set based on the inputs only. Note that for an intruder to be processed as CAS-equipped (*equip_int* is TRUE), both the intruder and own aircraft must be in TA/RA operational mode and have coordination capability.

The altitude inhibit cost state (ALTITUDEINHIBITCSTATE) for the intruder is assigned based on the protection mode selected for this cycle. Finally, if the ownship operational mode transitioned from TA-Only on the last cycle to TA/RA on this cycle, COORDINATIONDELAYCOST is activated by resetting COORDINATIONDELAYCSTATE with *is_count_enabled* initialized to TRUE. Otherwise, COORDINATIONDELAYCSTATE settings from the previous cycle are retained.

This algorithm takes as input *input_own*, *input_int*, *st_own*, and *st_int*. This algorithm returns *mode_int*, *equip_int*, *master_int*, *exclude_int*, *input_int.belief_horiz*, *input_int.belief_vert*, *z_int_ave*, and *dz_int_ave*. It also sets fields in the intruder state *st_int*.

Algorithm 145 IntruderPrep

```

1 function IntruderPrep( input_own::TRMOwnInput(p. E-19), input_int::TRMIIntruderInput(p. E-20),
2           st_own::TRMOwnState(p. E-32), st_int::TRMIIntruderState(p. E-33) )
3   protect_mode_int::UInt8 = 0
4   st_int.no_alerts      = input_int.dna
5   st_int.is_threat      = false
6   belief_vert_int::Vector{IntruderVerticalBelief(p. E-21)} = deepcopy( input_int.belief_vert )
7   if (0 != (input_int.degraded_surveillance & DEGRADED_SURVEILLANCE_NAR))
8     st_int.processing = RA_PROCESSING_DEGRADED_SURVEILLANCE
9     belief_vert_int = Array( IntruderVerticalBelief(p. E-21), length( input_own.belief_vert ) )
10    for i in 1:length( input_own.belief_vert )
11      belief_vert_int[i] = IntruderVerticalBelief(p. E-21)( input_own.belief_vert[i].z,
12                                         input_own.belief_vert[i].dz,
13                                         input_own.belief_vert[i].weight )
14    end
15   elseif (0 != (input_int.degraded_surveillance & DEGRADED_SURVEILLANCE_ADSB_ONLY))
16     st_int.processing = RA_PROCESSING_DEGRADED_SURVEILLANCE
17   elseif (OPMODE_RA == input_own.opmode)
18     if (RA_PROCESSING_DEGRADED_SURVEILLANCE == st_int.processing)
19       st_int.st_cost_on = OnlineCostState(p. E-37)()
20     end
21     st_int.processing = RA_PROCESSING_GLOBAL_TARA
22   else
23     st_int.processing = RA_PROCESSING_GLOBAL_TA_Only
24   end
25   z_int_ave::R = 0.0
26   dz_int_ave::R = 0.0
27   for b::IntruderVerticalBelief(p. E-21) in belief_vert_int
28     z_int_ave = z_int_ave + (b.z * b.weight)
29     dz_int_ave = dz_int_ave + (b.dz * b.weight)
30   end
31   if (st_int.protection_mode_prev == input_int.protection_mode) ||
32     (st_int.processing == RA_PROCESSING_GLOBAL_TA_Only) ||
33     (COC == st_int.a_prev.action)
34     protect_mode_int = input_int.protection_mode
35   else
36     protect_mode_int = st_int.protection_mode_prev
37   end
38   st_int.protection_mode_curr = protect_mode_int
39   st_int.designated_mode      = input_int.designated_mode
40   mode_int::Z    = GetProtectionModeIndex(p. 328)( protect_mode_int )
41   exclude_int::Bool = (RA_PROCESSING_DEGRADED_SURVEILLANCE == st_int.processing)
42   equip_int::Bool  = (EQUIPAGE_CASRA == input_int.equipage) &&
43                      (OPMODE_RA == input_own.opmode)
44   master_int::Bool = IsIntruderMaster(p. 334)( input_own, input_int )
45   st_int.st_cost_on.alt_inhibit = st_own.st_alt_inhibit[mode_int]
46   if (OPMODE_TA == st_own.opmode_prev) && (OPMODE_RA == input_own.opmode)
47     st_int.st_cost_on.coord_delay = CoordinationDelayCState(p. E-43)( true )
48   end
49   return (mode_int::Z, equip_int::Bool, master_int::Bool, exclude_int::Bool,
50          input_int.belief_horiz, belief_vert_int::Vector{IntruderVerticalBelief(p. E-21)},
51          z_int_ave::R, dz_int_ave::R)
52 end

```

Referenced In: StateAndCostEstimation(p. 161)

3.2.2 State Estimation

An overview of the STATEESTIMATION component in STATEANDCOSTESTIMATION is shown in Figure 3-3.

STATEESTIMATION (Algorithm 146) produces own aircraft and intruder state variables for use in estimating offline and online costs. It inputs horizontal and vertical beliefs for a single intruder

aircraft (b_int_prev , b_vert_int , b_horiz_int , z_int_ave , and $source_int$) and vertical beliefs for own aircraft (b_vert_own and $height_own$). These beliefs are received directly as generated by the STM.

The STATEESTIMATION algorithm always performs its evaluation with regard to a single intruder, independent of any other intruders.

CONVERTHORIZONTAL (Algorithm 147) is used to convert the intruder horizontal beliefs from b_horiz_int into beliefs for relative range in feet (r_ground), relative speed in feet/second (s_ground), and relative bearing in radians (phi_rel) with corresponding weights (w_int_horiz).

HORIZONTALWORSTCASE (Algorithm 148) is used next to compute an additional belief for the worst case condition in the horizontal plane. If a non-zero weight belief is produced by HORIZONTALWORSTCASE, it is added to the horizontal beliefs produced by CONVERTHORIZONTAL. That will result in the worst case condition being included in the belief structure produced by TAUESTIMATION. NORMALIZE (Algorithm 353) is used to ensure the resulting weights add to 1.0. A horizontal worst case belief is added whenever the possible relative position and velocity beliefs for the intruding aircraft could include a zero horizontal miss distance trajectory.

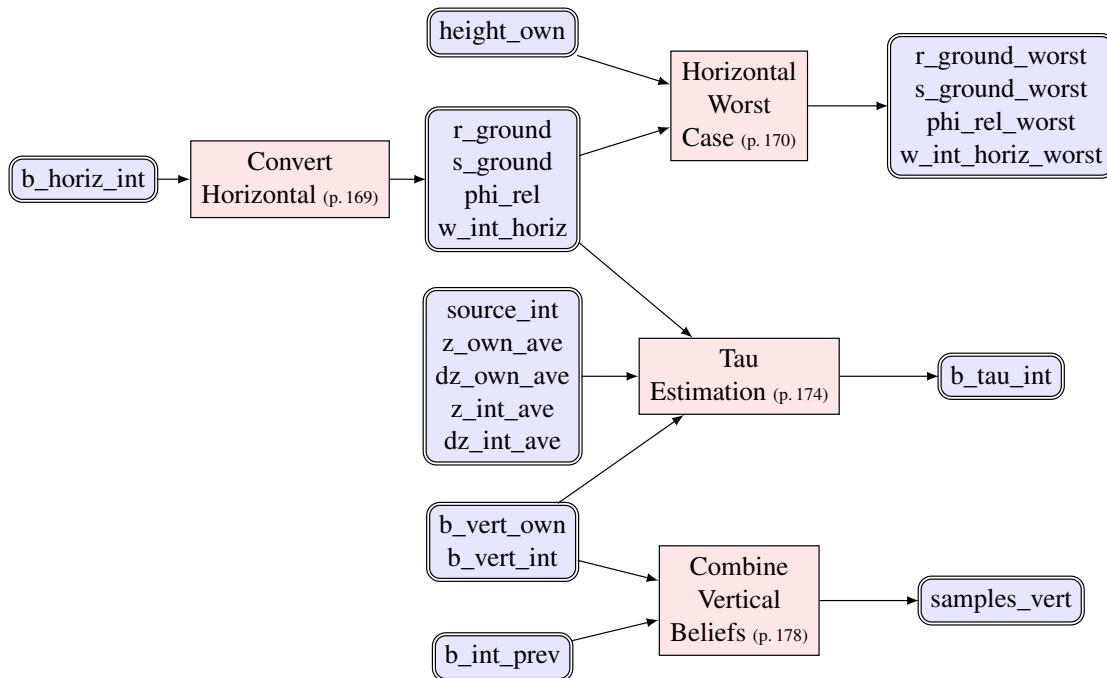
WRAPTOPI (Algorithm 355) ensures the magnitude of the relative bearing angle is no less than zero nor greater than π radians.

TAUESTIMATION (Algorithm 151) estimates the time until horizontal conflict. It produces a set of tau beliefs (b_tau_int).

COMBINEVERTICALBELIEFS (Algorithm 156) produces a set of vertical estimates for own aircraft and the intruder, assuming no pilot response. The estimates include the vertical separation and the estimated vertical rate of each aircraft. These beliefs are stored in $samples_vert$.

SETBELIEFBASEDONLINECOSTSTATE (Algorithm 157) is used last to set online cost state variables that are dependent on horizontal and vertical state information. Those state variables are stored in the ONLINECOSTSTATE (Type 56) data structure (s_c).

This algorithm takes as input $mode_int$, b_int_prev , b_vert_int , b_horiz_int , z_int_ave , $source_int$, b_vert_own , $height_own$, and s_c . Descriptions of the STATEESTIMATION output variables are found in Table 3-3.

**Figure 3-3. StateEstimation overview.****Table 3-3. TRM Algorithm Output Variables - StateEstimation**

Variable	Units	Type	Description
b_tau_int	(s,N/A)	(real,real)	Vector of vertical beliefs for each time until horizontal conflict
samples_vert	(ft,ft/s,ft/s,N/A)	(real,real,real,real)	Vector of times until horizontal conflict and their weights
s_c	N/A	OnlineCostState(p.E-37)	State variables for estimating online costs

Algorithm 146 StateEstimation

```

1 function StateEstimation( mode_int::Z,
2           b_int_prev::AdvisoryBeliefState(p.E-34),
3           b_vert_own::Vector{OwnVerticalBelief(p.E-21)},
4           b_vert_int::Vector{IntruderVerticalBelief(p.E-21)},
5           b_horiz_int::Vector{IntruderHorizontalBelief(p.E-21)},
6           height_own::R, z_own_ave::R, dz_own_ave::R,
7           z_int_ave::R, dz_int_ave::R,
8           source_int::Z, s_c::OnlineCostState(p.E-37) )
9   (r_ground::Vector{R}, s_ground::Vector{R}, phi_rel::Vector{R}, w_int_horiz::Vector{R}) =
10    ConvertHorizontal(p.169)( b_horiz_int )
11   (r_ground_worst::R, s_ground_worst::R, phi_rel_worst::R, w_int_horiz_worst::R) =
12    HorizontalWorstCase(p.170)( mode_int, r_ground, s_ground, phi_rel, w_int_horiz, height_own )
13   if (0.0 < w_int_horiz_worst)
14     push!( r_ground, r_ground_worst )
15     push!( s_ground, s_ground_worst )
16     push!( phi_rel, phi_rel_worst )
17     push!( w_int_horiz, w_int_horiz_worst )
18     w_int_horiz = Normalize(p.H-3)( w_int_horiz )
19   end
20   for i = 1:length(phi_rel)
21     phi_rel[i] = abs( WrapToPi(p.H-3)( phi_rel[i] ) )
22   end
23   b_tau_int::Vector{TauBelief(p.E-38)} =
24    TauEstimation(p.174)( mode_int, source_int, z_own_ave, dz_own_ave, z_int_ave, dz_int_ave,
25                           r_ground, s_ground, phi_rel, w_int_horiz, b_vert_own, b_vert_int )
26   samples_vert::Vector{CombinedVerticalBelief(p.E-34)} =
27    CombineVerticalBeliefs(p.178)( b_vert_own, b_vert_int, b_int_prev )
28   SetBeliefBasedOnlineCostState(p.179)( mode_int, height_own, z_own_ave, dz_own_ave,
29                                         r_ground, s_ground, phi_rel, w_int_horiz, b_vert_int, b_tau_int, s_c )
30   return (samples_vert::Vector{CombinedVerticalBelief(p.E-34)}, b_tau_int::Vector{TauBelief(p.E-38)})
31 end

```

Referenced In: StateAndCostEstimation(p.161)

CONVERTHORIZONTAL (Algorithm 147) converts the relative horizontal position and relative horizontal velocity samples of the intruder into a three-dimensional coordinate system, which is described in Section 5.3 of [10]. An important step not introduced in [10] is taking the absolute value of angle phi, which exploits symmetry of the distribution of tau to reduce file sizes by about half.

This algorithm takes as input b_horiz_int . This algorithm returns r_ground , s_ground , phi_rel , and w_int_horiz .

Algorithm 147 ConvertHorizontal

```

1 function ConvertHorizontal( b_horiz_int::Vector{IntruderHorizontalBelief(p.E-21)} )
2   const N_beliefs::Z      = length( b_horiz_int )
3   r_ground::Vector{R}     = zeros( R, N_beliefs )
4   s_ground::Vector{R}     = zeros( R, N_beliefs )
5   phi_rel::Vector{R}      = zeros( R, N_beliefs )
6   w_int_horiz::Vector{R}  = zeros( R, N_beliefs )
7   for i = 1:N_beliefs
8     b::IntruderHorizontalBelief(p.E-21) = b_horiz_int[i]
9     r_ground[i]    = hypot( b.x_rel, b.y_rel )
10    s_ground[i]   = hypot( b.dx_rel, b.dy_rel )
11    phi_rel[i]    = atan2( b.dy_rel, b.dx_rel ) - atan2( b.y_rel, b.x_rel )
12    w_int_horiz[i] = b.weight
13  end
14  return (r_ground::Vector{R}, s_ground::Vector{R}, phi_rel::Vector{R}, w_int_horiz::Vector{R})
15 end

```

Referenced In: StateEstimation(p.168)

HORIZONTALWORSTCASE (Algorithm 148) computes an additional horizontal belief with a zero horizontal miss distance trajectory for an individual intruder. This additional belief ensures the cost computations will account for the worst case condition in the horizontal plane.

All the relative bearing estimates are checked to identify the maximum relative bearing (*phi_max*), the minimum relative bearing (*phi_min*), and the mean relative bearing (*phi_mean*). All relative bearing estimates are wrapped between zero and 2π to properly deal with angle measures beyond a full circle. The max and min relative bearing estimates are used to determine if the zero horizontal miss distance trajectory falls within the span of the relative bearing estimates. Additionally, the mean relative bearing must indicate convergence for the worst case belief to be added. The relevance of the head-on trajectory is indicated by the returned weight (*w_int_horiz_worst*). It is calculated by interpolating the weights of the mean estimate and either the min or max estimate, whichever is closer.

If the worst case belief is determined to be necessary, a complete horizontal belief estimate is constructed. The mean range and speed are computed to be used in the worst case belief state and worst case relative bearing is set as π . Two additional algorithms, DETERMINEHORIZONTALWORSTCASEPHISREADFACTOR and DETERMINEHORIZONTALWORSTCASENEARMEANFACTOR, are used to reduce or increase the weight of the worst case belief in special conditions. The range, speed, relative bearing, and weight of the worst case belief are returned to STATEESTIMATION where, if a new belief is added, the weights are renormalized.

This algorithm takes as input *mode_int*, *r_ground*, *s_ground*, *phi_rel*, *w_int_horiz*, and *height_own*. This algorithm returns *r_ground_worst*, *s_ground_worst*, *phi_rel_worst*, and *w_int_horiz_worst*.

Algorithm 148 HorizontalWorstCase

```

1 function HorizontalWorstCase( mode_int::Z, r_ground::Vector{R}, s_ground::Vector{R},
2                               phi_rel::Vector{R}, w_int_horiz::Vector{R}, height_own::R )
3     r_ground_worst::R = 0.0
4     s_ground_worst::R = 0.0
5     phi_rel_worst::R = 0.0
6     w_int_horiz_worst::R = 0.0
7     phi_max::R = -Inf
8     phi_min::R = Inf
9     phi_mean::R = Inf
10    w_phi_max::R = 0.0
11    w_phi_min::R = 0.0
12    w_phi_mean::R = 0.0
13    for i = 1:length( w_int_horiz )
14        phi::R = WrapTo2Pi(p.H-3)( phi_rel[i] )
15        if (phi_max < phi)
16            phi_max = phi
17            w_phi_max = w_int_horiz[i]
18        end
19        if (phi < phi_min)
20            phi_min = phi
21            w_phi_min = w_int_horiz[i]
22        end
23        if (w_phi_mean < w_int_horiz[i])
24            phi_mean = phi
25            w_phi_mean = w_int_horiz[i]
26        end
27    end
28    if (pi/2 < phi_mean < 3pi/2)
29        w_worst_case::R = 0.0
30        if (phi_mean <= pi) && (pi < phi_max)
31            w_worst_case = w_phi_mean + ((w_phi_max - w_phi_mean) * ((pi - phi_mean) / (phi_max - phi_mean)))
32        elseif (phi_min < pi) && (pi <= phi_mean)
33            w_worst_case = w_phi_min + ((w_phi_mean - w_phi_min) * ((pi - phi_min) / (phi_mean - phi_min)))
34        end
35        if (0.0 < w_worst_case)
36            for i = 1:length( w_int_horiz )
37                r_ground_worst = r_ground_worst + (r_ground[i] * w_int_horiz[i])
38                s_ground_worst = s_ground_worst + (s_ground[i] * w_int_horiz[i])
39            end
40            phi_rel_worst = pi
41            phi_mean_wc_delta::R = abs( phi_mean - pi )
42            near_mean_scale_factor::R =
43                DetermineHorizontalWorstCaseNearMeanFactor(p.171)( mode_int, phi_mean_wc_delta, r_ground_worst )
44            phi_spread::R = phi_max - phi_min
45            phi_spread_scale_factor::R =
46                DetermineHorizontalWorstCasePhiSpreadFactor(p.172)( mode_int, phi_spread, height_own )
47            w_int_horiz_worst = w_worst_case * near_mean_scale_factor * phi_spread_scale_factor
48        end
49    end
50    return (r_ground_worst::R, s_ground_worst::R, phi_rel_worst::R, w_int_horiz_worst::R)
51 end

```

Referenced In: StateEstimation(p.168)

DETERMINEHORIZONTALWORSTCASENEARMEANFACTOR (Algorithm 149) is used to increase the weight of the worst case belief state according the proximity of the mean relative bearing estimate to the worst case. Although in most instances the uncertainty associated with belief states improves the performance, it can also lead to delayed action in some unique geometries and surveillances. The most prevalent of these is fast-closing, zero horizontal miss distance geometries with range dependent bearing uncertainty in the surveillance.

Two sub-factors control the overall amount of increase (*near_mean_factor*) to apply to the worst case belief state. The angle factor (*angle_factor*) is computed using the angle difference between the worst case relative bearing and the mean relative bearing estimate. As the difference between them increases, the value of the factor decreases. The range factor (*range_factor*) is computed using the worst case range. As the range decreases, the value of the factor decreases. The overall factor uses the minimum of the two sub-factors so that either factor can prevent the worst case weight from being increased.

This algorithm takes as input *mode_int*, *phi_mean_wc_delta*, and *r_ground_worst*. This algorithm returns *near_mean_factor*.

Algorithm 149 DetermineHorizontalWorstCaseNearMeanFactor

```

1 function DetermineHorizontalWorstCaseNearMeanFactor( mode_int::Z, phi_mean_wc_delta::R, r_ground_worst::R )
2   const X_near_mean_factor_max::R = =
3     params().modes[mode_int].state_estimation.tau.horiz_worst_case.X_near_mean_factor_max
4   const X_near_mean_angle_threshold_lo::R =
5     params().modes[mode_int].state_estimation.tau.horiz_worst_case.X_near_mean_angle_threshold_lo
6   const X_near_mean_angle_threshold_hi::R =
7     params().modes[mode_int].state_estimation.tau.horiz_worst_case.X_near_mean_angle_threshold_hi
8   const D_near_mean_range_threshold_lo::R =
9     params().modes[mode_int].state_estimation.tau.horiz_worst_case.D_near_mean_range_threshold_lo
10  const D_near_mean_range_threshold_hi::R =
11    params().modes[mode_int].state_estimation.tau.horiz_worst_case.D_near_mean_range_threshold_hi
12  angle_factor::R = 0.0
13  range_factor::R = 0.0
14  rolloff::R = 0.0
15  rolloff = X_near_mean_angle_threshold_hi - X_near_mean_angle_threshold_lo
16  angle_factor = CalculateThresholdRampDownFactor(p. 326)( phi_mean_wc_delta, X_near_mean_angle_threshold_lo,
17    rolloff )
17  rolloff = D_near_mean_range_threshold_hi - D_near_mean_range_threshold_lo
18  range_factor = CalculateThresholdRampUpFactor(p. 326)( r_ground_worst, D_near_mean_range_threshold_lo,
19    rolloff )
20  near_mean_factor::R = ( X_near_mean_factor_max - 1.0 ) * min( angle_factor, range_factor ) + 1.0
21 return (near_mean_factor::R)
21 end
```

Referenced In: HorizontalWorstCase(p. 170)
--

DETERMINEHORIZONTALWORSTCASEPHISPREADFACTOR (Algorithm 150) is used to reduce or increase the weight of the worst case belief state according the spread of the relative bearing estimates. In slow closure cases, such as parallel geometries, the spread of the relative bearing estimates can increase dramatically, leading to an improper application of the worst case belief.

A number of conditions are checked to determine the value of the scaling factor to be applied (*spread_scale_factor*). Nominally the weight is changed by the high factor (*X_phi_spread_factor_hi*). However, if the own aircraft height is below a threshold (*H_phi_spread_threshold*) and the spread of the relative bearing estimates is larger than the minimum threshold (*X_phi_spread_threshold_lo*), the scaling factor is changed. If the spread is greater than the high threshold (*X_phi_spread_threshold_hi*), the scale factor is reduced to the lowest possible threshold (*X_phi_spread_factor_lo*). Otherwise, the scale factor is calculated by interpolating the high and low factors based on the spread of the relative bearing estimates.

This algorithm takes as input *mode_int*, *phi_spread*, and *height_own*. This algorithm returns *spread_scale_factor*.

Algorithm 150 DetermineHorizontalWorstCasePhiSpreadFactor

```

1 function DetermineHorizontalWorstCasePhiSpreadFactor( mode_int::Z, phi_spread::R, height_own::R )
2   const X_phi_spread_factor_lo::R      =
3     params().modes[mode_int].state_estimation.tau.horiz_worst_case.X_phi_spread_factor_lo
4   const X_phi_spread_factor_hi::R    =
5     params().modes[mode_int].state_estimation.tau.horiz_worst_case.X_phi_spread_factor_hi
6   const H_phi_spread_threshold::R   =
7     params().modes[mode_int].state_estimation.tau.horiz_worst_case.H_phi_spread_threshold
8   const X_phi_spread_threshold_lo::R =
9     params().modes[mode_int].state.estimation.tau.horiz_worst_case.X_phi_spread_threshold_lo
10  const X_phi_spread_threshold_hi::R =
11    params().modes[mode_int].state.estimation.tau.horiz_worst_case.X_phi_spread_threshold_hi
12  spread_scale_factor::R = X_phi_spread_factor_hi
13  if (height_own <= H_phi_spread_threshold) && (X_phi_spread_threshold_lo <= phi_spread)
14    if (X_phi_spread_threshold_hi <= phi_spread)
15      spread_scale_factor = X_phi_spread_factor_lo
16    else
17      rolloff::R = X_phi_spread_threshold_hi - X_phi_spread_threshold_lo
18      ramp_factor = CalculateThresholdRampDownFactor(p. 326)( phi_spread, X_phi_spread_threshold_lo,
19                                rolloff )
20      spread_scale_factor = X_phi_spread_factor_lo + ((X_phi_spread_factor_hi - X_phi_spread_factor_lo)
21                                         * ramp_factor)
22    end
23  return (spread_scale_factor::R)
24 end

```

Referenced In: HorizontalWorstCase(p. 170)

3.2.2.1 Tau Estimation

TAUESTIMATION (Algorithm 151) takes three-dimensional samples for the observed state of the intruder. It produces a probability distribution over the values of tau (in seconds) given by *T_tau_values*.

Tau is the time until the intruder comes within a lateral conflict distance of own aircraft. Since tau is a quantity that is defined in the future and there is stochasticity in the horizontal dynamics, it is not possible to know this value with absolute certainty. It is important to distinguish the tau in this logic from the tau that is used by the legacy TCAS logic. In the legacy logic, tau is interpreted to mean the time of closest approach and the logic uses simple models to provide a point estimate of tau.

The estimation process involves averaging the probability distributions associated with the interpolants of all the three-dimensional samples. Determining the probability distribution of an interpolant involves looking up distributions organized in the horizontal entry distribution file (*T_horizontal_table*). For Xo, the horizontal entry distribution file used depends on whether the surveillance source is ADS-B (*SOURCE_1090ES_ADSB*) or ADS-R (*SOURCE_1090ES_ADSR*) or whether it is from active surveillance only. The same horizontal entry distribution file is used for ADS-R and for ADS-B surveillance, whether or not the ADS-B surveillance is validated by active surveillance. For Xa, the same horizontal entry distribution file is used for all surveillance sources. The state samples are derived from intruder aircraft horizontal belief states.

TAUESTIMATION takes as input *mode_int*, *source_int*, *z_own_ave*, *dz_own_ave*, *z_int_ave*, *dz_int_ave*, *r_ground*, *s_ground*, *phi_rel*, *w_horiz*, *b_vert_own*, and *b_vert_int*. This algorithm returns *b_tau_int*.

...
...

GETENTRYDISTRIBUTIONTABLES determines which horizontal and vertical entry distribution files should be used based on the protection mode index (*mode_int*), and surveillance source (*source_int*).

To address vertically converging, slow closure encounters, the probability distribution associated with the interpolants is based on vertical state and distribution information, as opposed to horizontal. A slow closure encounter is defined to be one where the sampled relative range and relative ground speed for the intruder are below parameterized thresholds, *D_range_thres* and *R_speed_thres*. Additionally, the vertical entry table must have a large enough vertical separation (*H_alt_rel_thres*) and enough weight above the weight threshold (*W_thres_vert*), to be considered for use. Conditions must then be satisfied based on the horizontal convergence or divergence of the intruder, based on *phi_rel*. In the case that the intruder is horizontally converging, the horizontal distribution must have sufficient weight below a tau threshold (*T_thresh_horiz*), and more weight than the vertical distribution below the same tau threshold. In the case that the intruder is horizontally diverging, the intruder must be within *D_range_diverge_thres* horizontally. In these instances, a vertical entry distribution file (*T_vertical_table*), is used. The state samples, in this case, are derived from own aircraft and intruder aircraft vertical belief states.

DETERMINETAUVERTICALDISTRIBUTION is used to calculate the probability distribution associated with the interpolants based on the vertical entry distribution file. It is called only in the case of a vertically converging slow closure encounter. It is called at most once per intruder per cycle; it is independent of the horizontal beliefs.

Algorithm 151 TauEstimation

```

1 function TauEstimation( mode_int::Z, source_int::Z,
2           z_own_ave::R, dz_own_ave::R, z_int_ave::R, dz_int_ave::R,
3           r_ground::Vector{R}, s_ground::Vector{R}, phi_rel::Vector{R}, w_horiz::Vector{R},
4           b_vert_own::Vector{OwnVerticalBelief(p.E-21)},
5           b_vert_int::Vector{IntruderVerticalBelief(p.E-21)} )
6   const T_thres_horiz::R      = params().modes[mode_int].state_estimation.tau.T_thres_horiz
7   const W_thres_horiz::R      = params().modes[mode_int].state_estimation.tau.W_thres_horiz
8   const W_thres_vert::R      = params().modes[mode_int].state_estimation.tau.W_thres_vert
9   const D_range_diverge_thres::R= params().modes[mode_int].state_estimation.tau.D_range_diverge_thres
10  const X_phi_converge_thres::R = params().modes[mode_int].state_estimation.tau.X_phi_converge_thres
11  (T_horizontal_table::RDataTable(p.E-38),
12   T_vertical_table::RDataTable(p.E-38),
13   T_tau_values::Vector{R}) =
14   GetEntryDistributionTables(p.175)( mode_int, source_int )
15   t::Z                      = length( T_tau_values )
16   num_tau_blocks::Z          = t - 1
17   w_out::Vector{R}           = zeros( R, t )
18   w_vert::Vector{R}           = zeros( R, t )
19   w_below_thres_vert::R      = -Inf
20   is_vert_initialized::Bool  = false
21   is_vertically_converging::Bool =
22     IsVerticallyConverging(p.339)( mode_int, z_own_ave, dz_own_ave, z_int_ave, dz_int_ave )
23   for i = 1:length( w_horiz )
24     w_temp::Vector{R} = zeros( R, t )
25     w_below_thres_horiz::R = 0.0
26     datum = R[ r_ground[i], s_ground[i], phi_rel[i] ]
27     w_temp = LookupEntryDistribution(p.176)( mode_int, datum, T_horizontal_table, num_tau_blocks )
28     for j = 1:t
29       if (T_tau_values[j] <= T_thres_horiz)
30         w_below_thres_horiz = w_below_thres_horiz + w_temp[j]
31       end
32     end
33     if is_vertically_converging &&
34       IsSlowClosure(p.338)( mode_int, r_ground[i], s_ground[i], (z_int_ave - z_own_ave) )
35       if !is_vert_initialized
36         (w_vert, w_below_thres_vert) =
37           DetermineTauVerticalDistribution(p.176)( mode_int, b_vert_own, b_vert_int,
38                                         T_vertical_table, T_tau_values )
39         is_vert_initialized = true
40       end
41       if (0.0 <= w_below_thres_vert < W_thres_vert)
42         if (r_ground[i] < D_range_diverge_thres) ||
43           ((phi_rel[i] < X_phi_converge_thres) &&
44            (W_thres_horiz < w_below_thres_horiz) &&
45            (w_below_thres_vert < w_below_thres_horiz))
46           w_temp = w_vert
47         end
48       end
49     end
50     w_out = w_out + (w_horiz[i] * w_temp)
51   end
52   w_out[t] = max( 0.0, 1.0 - sum( w_out[1:num_tau_blocks] ) )
53   b_tau_int::Vector{TauBelief} = Array( TauBelief(p.E-38), 0 )
54   for i = 1:t
55     if (0.0 < w_out[i]) || (i == t)
56       push!( b_tau_int, TauBelief(p.E-38)( T_tau_values[i], w_out[i] ) )
57     end
58   end
59   return b_tau_int::Vector{TauBelief(p.E-38)}
60 end
```

Referenced In: StateEstimation(p.168)

GETENTRYDISTRIBUTIONTABLES (Algorithm 152) determines which horizontal and vertical entry distribution files should be used based on the protection mode index, *mode_int*, and surveillance source, *source_int*. While this algorithm would support completely unique entry distribution tables for each surveillance and mode combination, only Xo utilizes this functionality.

GETENTRYDISTRIBUTIONTABLES takes as input *mode_int* and *source_int*. This algorithm returns *t_horizontal_table*, *t_vertical_table*, and *t_tau_values*.

Algorithm 152 GetEntryDistributionTables

```

1 function GetEntryDistributionTables( mode_int::Z, source_int::Z )
2   const T_tau_values::Vector{R} =
3     params().modes[mode_int].state_estimation.tau.entry_dist.T_tau_values
4   const T_vertical_table::RDataTable(p.E-38) =
5     params().modes[mode_int].state_estimation.tau.entry_dist.vertical_table_content
6   t_horizontal_table::RDataTable(p.E-38)
7   if (SOURCE_1090ES_ADSB == source_int) || (SOURCE_1090ES_ADSR == source_int)
8     t_horizontal_table =
9       params().modes[mode_int].state_estimation.tau.entry_dist.horizontal_table_content
10   else
11     t_horizontal_table =
12       params().modes[mode_int].state_estimation.tau.entry_dist.horizontal_active_table_content
13   end
14   return (t_horizontal_table::RDataTable, T_vertical_table::RDataTable(p.E-38), T_tau_values::Vector{R})
15 end
```

Referenced In: TauEstimation(<i>p. 174</i>)
--

DETERMINETAUVERTICALDISTRIBUTION (Algorithm 153) returns the probability distribution associated with the interpolants based on the vertical entry distribution file. The vertical entry distribution file is used only under certain conditions as described in TAUESTIMATION. The probability distribution based on the vertical entry distribution file is calculated at most once per intruder each cycle, if needed.

DETERMINETAUVERTICALDISTRIBUTION takes as input *mode_int*, *b_vert_own*, *b_vert_int*, *T_vertical_table*, and *T_tau_values*. This algorithm returns *w_vert*, and *w_below_thres_vert*.

Algorithm 153 DetermineTauVerticalDistribution

```

1 function DetermineTauVerticalDistribution( mode_int::Z, b_vert_own::Vector{OwnVerticalBelief(p.E-21)},  

2                                         b_vert_int::Vector{IntruderVerticalBelief(p.E-21)},  

3                                         T_vertical_table::RDataTable(p.E-38), T_tau_values::Vector{R} )  

4   const T_thres_horiz::R = params().modes[mode_int].state_estimation.tau.T_thres_horiz  

5   n::Z = length( b_vert_own )  

6   m::Z = length( b_vert_int )  

7   t::Z = length( T_tau_values )  

8   num_tau_blocks::Z = t - 1  

9   w_vert::Vector{R} = zeros( R, t )  

10  w_below_thres_vert::R = 0.0  

11  for j in 1:n  

12    for k in 1:m  

13      datum::Vector{R} = R[ (b_vert_int[k].z - b_vert_own[j].z), b_vert_own[j].dz, b_vert_int[k].dz ]  

14      w_samp::Vector{R} =  

15        LookupEntryDistribution(p.176)( mode_int, datum, T_vertical_table, num_tau_blocks )  

16      w_vert = w_vert + (b_vert_own[j].weight * b_vert_int[k].weight * w_samp)  

17    end  

18  end  

19  for j in 1:t  

20    if (T_tau_values[j] <= T_thres_horiz)  

21      w_below_thres_vert = w_below_thres_vert + w_vert[j]  

22    end  

23  end  

24  return (w_vert::Vector{R}, w_below_thres_vert::R)
25 end
```

Referenced In: TauEstimation(p.174)

LOOKUPENTRYDISTRIBUTION (Algorithm 154) determines the probability distribution of an interpolant by looking up distributions organized in the horizontal entry distribution file. The state samples are derived from intruder horizontal belief states. In the slow closure case, LOOKUPENTRYDISTRIBUTION uses a vertical entry distribution file. The state samples are derived from own aircraft and intruder aircraft vertical belief states.

The algorithm uses INTERPOLATEBLOCKS to generate the distributions that are to be returned. The horizontal and vertical entry distribution files are both fixed block dictionaries that follow the specification in Appendix G. The value of *w_samp* returned by LOOKUPENTRYDISTRIBUTION is the average of all the probability distributions associated with the interpolants.

This algorithm takes as input *mode_int*, *datum*, *table*, and *num_blocks*. It returns *w_samp*.

Algorithm 154 LookupEntryDistribution

```

1 function LookupEntryDistribution( mode_int::Z, datum::Vector{R}, table::RDataTable(p.E-38), num_blocks::Z )  

2   const T_threshold_factor::R = params().modes[mode_int].state_estimation.tau.threshold_factor  

3   w_samp::Vector{R} = zeros( R, num_blocks + 1 )  

4   w_interp::Vector{R} = InterpolateBlocks(p.177)( datum, table, num_blocks )  

5   threshold::R = T_threshold_factor * max( maximum( w_interp ), 1.0 - sum( w_interp ) )  

6   for j = 1:num_blocks  

7     if (threshold <= w_interp[j])  

8       w_samp[j] = w_interp[j]  

9     end  

10   end  

11   w_samp[end] = 1.0 - sum( w_samp )  

12   return w_samp::Vector{R}
13 end
```

Referenced In: DetermineTauVerticalDistribution(p.176), TauEstimation(p.174)

INTERPOLATEBLOCKS (Algorithm 155) interpolates the input values to produce the weighted entries that can be looked up in the table format. This is done through a call to INTERPOLANTS which returns the table indices to use and the weight associated with each index. These indices are used to create the blocks containing the weighted values that represent the distributions.

This algorithm takes as input x , *table*, and *block_size*. This algorithm returns *block*.

Algorithm 155 InterpolateBlocks

```

1 function InterpolateBlocks( x::Vector{R}, table::RDataTable(p. E-38), block_size::Z )
2   const D::Z = length( x )
3   block::Vector{R} = zeros( R, block_size )
4   (indices::Vector{Z}, weights::Vector{R}) =
5     Interpolants(p. 329)( x, table.cut_counts, 1, D, table.cuts )
6   for j = 1:length( indices )
7     offset::Z = table.index[ indices[j] ]
8     for k = 1:block_size
9       block[k] = block[k] + (weights[j] * table.data[offset+k])
10    end
11  end
12  return block::Vector{R}
13 end
```

Referenced In: <i>LookupEntryDistribution</i> (<i>p. 176</i>)
--

3.2.2.2 Combine Vertical Beliefs

COMBINEVERTICALBELIEFS (Algorithm 156) combines the intruder and own aircraft vertical beliefs to produce a set of vertical samples. The vertical samples are used as inputs to OFFLINECOST-ESTIMATION (Algorithm 161).

This algorithm takes as input *b_vert_own*, *b_vert_int*, and *b_int*. It returns *samples_vert*.

Algorithm 156 CombineVerticalBeliefs

```

1 function CombineVerticalBeliefs( b_vert_own::Vector{OwnVerticalBelief(p.E-21)},
2                               b_vert_int::Vector{IntruderVerticalBelief(p.E-21)},
3                               b_int::AdvisoryBeliefState(p.E-34) )
4   w_vert::R = 0.0
5   if !b_int.need_init
6     w_vert_prev_sum::R = sum( b_int.w_vert_prev )
7   end
8   samples_vert::Vector{CombinedVerticalBelief} = CombinedVerticalBelief(p.E-34)[ ]
9   b_int.w_vert_prev = R[]
10  for i in 1:length( b_vert_own )
11    for j in 1:length( b_vert_int )
12      if !b_int.need_init
13        w_vert = w_vert_prev_sum * b_vert_own[i].weight * b_vert_int[j].weight
14      else
15        w_vert = b_vert_own[i].weight * b_vert_int[j].weight
16      end
17      push!( samples_vert,
18               CombinedVerticalBelief(p.E-34)( b_vert_int[j].z - b_vert_own[i].z,
19                                         b_vert_own[i].dz, b_vert_int[j].dz,
20                                         w_vert ) )
21    )
22    push!( b_int.w_vert_prev, w_vert )
23  end
24 end
25 b_int.w_vert_prev = Normalize(p.H-3)( b_int.w_vert_prev )
26 b_int.need_init = (length( b_int.w_vert_prev ) == 0)
27 for i in 1:length( samples_vert )
28   samples_vert[i].weight = b_int.w_vert_prev[i]
29 end
30 return samples_vert::Vector{CombinedVerticalBelief(p.E-34)}
31 end

```

Referenced In: StateEstimation(p.168)

3.2.2.3 Set Belief-Based Online Cost

SETBELIEFBASEDONLINECOSTSTATE (Algorithm 157) is used to set state variables for online costs that are dependent on details of the intruder aircraft and own aircraft dynamic state.

Several online cost algorithms have state variables dependent on vertical and/or horizontal beliefs:

- ALTITUDEDEPENDENTCOCCOST (Algorithm 172)
- CRITICALINTERVALPROTECTIONCOST (Algorithm 177)
- LOWALTITUDEPARALLELRADEFERRALCOST (Algorithm 200)
- PREVENTEARLYCOCCOST (Algorithm 182)
- SA01HEURISTIC (Algorithm 210)

The state variable values are set here to confine the dynamic state information they use for inputs to STATEESTIMATION and to closely associate the calculated values with the appropriate online cost algorithms. Setting the state variables within STATEESTIMATION also provides more options for adjusting the method of calculating the values.

The calculations of values for the CRITICALINTERVALPROTECTIONCSTATE (Type 71), LOWALTITUDEPARALLELRADEFERRALCSTATE (Type 74), PREVENTEARLYCOCCSTATE (Type 77), and SA01HEURISTICCSTATE (Type 80) variables are performed within this algorithm. There are two instances of LOWALTITUDEPARALLELRADEFERRALCSTATE. The first (*s_c.low_alt_par-*

allel_ra_deferral) contains state settings used for selecting resolution advisories. The second (*s_c.ta_low_alt_parallel_ra_deferral*) contains state settings used for determining traffic advisories. Both instances are set with the same relative range and speed values. The calculations for ALTITUDEDEPENDENTCOCCSTATE (Type 65) are performed in DETERMINEALTITUDEDEPENDENTCOFCFACTOR (Algorithm 158). DETERMINEALTITUDEDEPENDENTCOFCFACTOR is called twice: once to calculate the factor used to set *altitude_dependent_coc.scaled_cost_coc_ra*; the second time to calculate the factor used to set *altitude_dependent_coc.scaled_cost_factor_ta*.

This algorithm takes as input *mode_int*, *height_own*, *z_own_ave*, *dz_own_ave*, *r_ground*, *s_ground*, *phi_rel*, *w_int_horiz*, *b_vert_int*, *b_tau_int*, and *s_c*. This algorithm updates *s_c*::*OnlineCostState* which contains state variables for online costs.

Algorithm 157 SetBeliefBasedOnlineCostState

```

1 function SetBeliefBasedOnlineCostState( mode_int::Z, height_own::R, z_own_ave::R, dz_own_ave::R,
2                                     r_ground::Vector{R}, s_ground::Vector{R}, phi_rel::Vector{R}, w_int_horiz::Vector{R},
3                                     b_vert_int::Vector{IntruderVerticalBelief(p.E-21)}, b_tau_int::Vector{TauBelief(p.E-38)},
4                                     s_c::OnlineCostState(p.E-37) )
5   r_ground_int_ave::R = 0.0
6   s_ground_int_ave::R = 0.0
7   phi_rel_ave::R      = 0.0
8   for i in 1:length(phi_rel)
9     r_ground_int_ave = r_ground_int_ave + (w_int_horiz[i] * r_ground[i])
10    s_ground_int_ave = s_ground_int_ave + (w_int_horiz[i] * s_ground[i])
11    phi_rel_ave      = phi_rel_ave + (w_int_horiz[i] * phi_rel[i])
12  end
13  s_c.low_alt_parallel_ra_deferral.range      = r_ground_int_ave
14  s_c.low_alt_parallel_ra_deferral.speed_rel = s_ground_int_ave
15  s_c.ta_low_alt_parallel_ra_deferral.range      = r_ground_int_ave
16  s_c.ta_low_alt_parallel_ra_deferral.speed_rel = s_ground_int_ave
17  s_c.critical_interval_protection.angle = phi_rel_ave
18  s_c.critical_interval_protection.range = r_ground_int_ave
19  s_c.critical_interval_protection.speed = s_ground_int_ave
20  s_c.sa01_heuristic.range = r_ground_int_ave
21  s_c.prevent_early_coc.range = r_ground_int_ave
22  h_rel_threshold_proximate::R =
23    params().modes[mode_int].cost_estimation.online.altitude_dependent_coc.H_rel_threshold_proximate
24  t_threshold::R =
25    params().modes[mode_int].cost_estimation.online.altitude_dependent_coc.T_threshold
26  c_coc::R =
27    params().modes[mode_int].cost_estimation.online.altitude_dependent_coc.C_coc
28  factor::R =
29    DetermineAltitudeDependentCOCFactor(p.181)( mode_int, height_own, z_own_ave, dz_own_ave,
30                                              b_vert_int, b_tau_int,
31                                              h_rel_threshold_proximate, t_threshold )
32  s_c.altitude_dependent_coc.scaled_cost_coc_ra = factor * c_coc
33  h_rel_threshold_proximate =
34    params().modes[mode_int].track_threat.ta_altitude_dependent_coc.H_rel_threshold_proximate
35  t_threshold =
36    params().modes[mode_int].track_threat.ta_altitude_dependent_coc.T_threshold
37  c_coc =
38    params().modes[mode_int].track_threat.ta_altitude_dependent_coc.C_coc
39  factor =
40    DetermineAltitudeDependentCOCFactor(p.181)( mode_int, height_own, z_own_ave, dz_own_ave,
41                                              b_vert_int, b_tau_int,
42                                              h_rel_threshold_proximate, t_threshold )
43  s_c.altitude_dependent_coc.scaled_cost_coc_ta = factor * c_coc
44 end
```

Referenced In: StateEstimation(p.168)

DETERMINEALTITUDEDEPENDENTCOFCOST (Algorithm 158) is used to set state variables for ALTITUDEDEPENDENTCOFCOST. As the traffic advisory determination uses a different factor than the resolution advisory logic, the DETERMINEALTITUDEDEPENDENTCOFCOST is called twice with different parameters for each.

This algorithm determines how much of an additional cost will be applied to clear of conflict at the current state (*factor_coc*). This factor varies with altitude so that as altitude increases the likelihood of having an alert increases proportionally. In this way, the *factor_coc* can be thought of as the amount of additional logic sensitivity to be applied at a given altitude for a given intruder.

At various points in this algorithm the CALCULATETHRESHOLDRAMPUPFACTOR and CALCULATETHRESHOLDRAMPDOWNFACTOR algorithms are used. These methods take an input value, a threshold parameter, and a roll-off parameter and return a value between 0 and 1. The different factors combine to determine the additional sensitivity added to the logic.

The algorithm loops over each vertical sample in turn to determine the necessity of the additional sensitivity. Due to this iterative approach, the intermediate calculations are weighted and summed during the loop. Two factors (*factor_reduction* and *factor_proximate*) are calculated using sub-functions and components of the vertical sample. The reduction factor reduces the amount of sensitivity increase in circumstances where the own aircraft vertical rate is low. The proximate factor is the main discriminator of when to increase the sensitivity based on the vertical sample and a forward projection in time.

Nested within the vertical sample loop is another loop which iterates over the horizontal tau samples. Since many tau samples have zero weight, a check is performed to determine if further calculations are required. Furthermore, the tau value itself must be within the maximum tau applicability defined by the parameters and less than the largest possible tau value, to prevent improper increases in sensitivity. For each valid tau sample a vertical projection of the value of tau forward in time is performed. This projected separation (*z_rel_actual*) represents the expected vertical separation at the time of loss of horizontal separation. It is used to calculate a factor (*factor_actual*) which reduces the applicability of the additional sensitivity in cases where there will be a large vertical separation when horizontal separation is lost. This condition occurs most frequently in geometries where the aircraft are expected to cross altitude very early in an encounter. To allow these well-separated crossings to occur, the additional sensitivity should not be applied. The intermediate calculation of *factor_coc* combines the weight, proximate factor, actual factor, and a final factor computed using the tau sample value. As the value of tau increases, this factor reduces the additional sensitivity to be applied.

Once the algorithm finishes looping over the vertical samples, the *factor_coc* is scaled by the altitude of the own aircraft. This causes the additional sensitivity to be applied in full above the high altitude threshold (*H_threshold_hi*), not at all below the low threshold (*H_threshold_lo*), and proportionally between. Finally, the reduction factor is applied here as well to appropriately reduce the additional sensitivity in the desired cases. The sensitivity due to this factor can be reduced up to a maximum of *X_vrrf*.

This algorithm takes as input *mode_int*, *height_own*, *z_own_ave*, *dz_own_ave*, *b_vert_int*, *b_tau_int*, *H_rel_threshold_proximate*, and *T_threshold*. This algorithm returns *factor_coc*.

Algorithm 158 DetermineAltitudeDependentCOCFactor

```

1 function DetermineAltitudeDependentCOCFactor( mode_int::Z, height_own::R, z_own_ave::R, dz_own_ave::R,
2   b_vert_int::Vector{IntruderVerticalBelief(p.E-21)}, b_tau_int::Vector{TauBelief(p.E-38)},
3   H_rel_threshold_proximate::R, T_threshold::R )
4   const H_rel_rolloff_proximate::R =
5     params().modes[mode_int].cost_estimation.online.altitude_dependent_coc.H_rel_rolloff_proximate
6   const X_proximate_to_actual_factor::R =
7     params().modes[mode_int].cost_estimation.online.altitude_dependent_coc.X_proximate_to_actual_factor
8   const T_rolloff::R =
9     params().modes[mode_int].cost_estimation.online.altitude_dependent_coc.T_rolloff
10  const X_vrrf::R =
11    params().modes[mode_int].cost_estimation.online.altitude_dependent_coc.X_vrrf
12  const H_threshold_lo::R =
13    params().modes[mode_int].cost_estimation.online.altitude_dependent_coc.H_threshold_lo
14  const H_threshold_hi::R =
15    params().modes[mode_int].cost_estimation.online.altitude_dependent_coc.H_threshold_hi
16  const T_tau_values::Vector{R} =
17    params().modes[mode_int].state_estimation.tau.entry_dist.T_tau_values
18  factor_coc::R = 0.0
19  factor_reduction::R = 0.0
20  if (H_threshold_lo <= height_own)
21    for bv_int in b_vert_int
22      z_rel::R = bv_int.z - z_own_ave
23      factor_reduction = factor_reduction + (bv_int.weight *
24        DetermineVerticalRateReductionFactor(p.182)( mode_int, height_own, z_rel, dz_own_ave ))
25      factor_proximate::R = bv_int.weight *
26        DetermineVerticalProximateFactor(p.183)( mode_int, height_own, z_rel, dz_own_ave,
27                                              bv_int.dz, H_rel_threshold_proximate )
28      for bt_int in b_tau_int
29        if (0.0 < bt_int.weight) && ((bt_int.tau <= (T_threshold + T_rolloff)) &&
30          (bt_int.tau < T_tau_values[end]))
31          z_rel_actual::R = abs( z_rel + ((bv_int.dz - dz_own_ave) * bt_int.tau) )
32          threshold::R = H_rel_threshold_proximate * X_proximate_to_actual_factor
33          rolloff::R = H_rel_rolloff_proximate * X_proximate_to_actual_factor
34          factor_actual::R =
35            CalculateThresholdRampDownFactor(p.326)( z_rel_actual, threshold, rolloff )
36          factor_coc = factor_coc + (bt_int.weight * factor_proximate * factor_actual *
37            CalculateThresholdRampDownFactor(p.326)( bt_int.tau, T_threshold, T_rolloff ))
38        end
39      end
40    end
41    factor_coc = factor_coc * CalculateThresholdRampUpFactor(p.326)( height_own, H_threshold_lo,
42                                         (H_threshold_hi - H_threshold_lo) )
43    factor_coc = factor_coc * (X_vrrf + ((1.0 - X_vrrf) * (1.0 - factor_reduction)))
44  end
45  return factor_coc::R
46 end

```

Referenced In: SetBeliefBasedOnlineCostState(p.179)

DETERMINEVERTICALRATEREDUCTIONFACTOR (Algorithm 159) uses two factors to determine when to reduce the additional sensitivity of the logic on near-level aircraft with sufficient separation. The first involves comparing the own aircraft vertical rate to a set of thresholds to determine how close it is to level. When the own aircraft is perfectly level, the reduction will be at a maximum and as the vertical rate increases, the reduction will decrease. The second component is based on the current vertical separation. As the current relative altitude approaches co-altitude, the reduction is decreased. This provides a safe-guard on the vertical rate-reduction factor so that even if an aircraft is level, the additional sensitivity will be applied if separation is lost.

This algorithm takes as input *mode_int*, *height_own*, *z_rel*, and *dz_own_ave*. This algorithm returns *factor_vrrf*.

Algorithm 159 DetermineVerticalRateReductionFactor

```

1 function DetermineVerticalRateReductionFactor( mode_int::Z, height_own::R, z_rel::R, dz_own_ave::R )
2   const R_own_threshold_vrrf::R =
3     params().modes[mode_int].cost_estimation.online.altitude_dependent_coc.R_own_threshold_vrrf
4   const R_own_rolloff_vrrf::R =
5     params().modes[mode_int].cost_estimation.online.altitude_dependent_coc.R_own_rolloff_vrrf
6   const H_threshold_hi::R =
7     params().modes[mode_int].cost_estimation.online.altitude_dependent_coc.H_threshold_hi
8   const H_rel_threshold_vrrf_lo::R =
9     params().modes[mode_int].cost_estimation.online.altitude_dependent_coc.H_rel_threshold_vrrf_lo
10  const H_rel_threshold_vrrf_hi::R =
11    params().modes[mode_int].cost_estimation.online.altitude_dependent_coc.H_rel_threshold_vrrf_hi
12  const H_rel_rolloff_vrrf::R =
13    params().modes[mode_int].cost_estimation.online.altitude_dependent_coc.H_rel_rolloff_vrrf
14  factor_dz_own::R =
15    CalculateThresholdRampDownFactor(p.326)( abs( dz_own_ave ), R_own_threshold_vrrf, R_own_rolloff_vrrf )
16  z_rel_threshold_vrrf::R = H_rel_threshold_vrrf_lo
17  if (H_threshold_hi < height_own)
18    z_rel_threshold_vrrf = H_rel_threshold_vrrf_hi
19  end
20  factor_z_rel =
21  CalculateThresholdRampUpFactor(p.326)( abs( z_rel ), z_rel_threshold_vrrf, H_rel_rolloff_vrrf )
22  factor_vrrf::R = min( factor_z_rel, factor_dz_own )
23  return factor_vrrf::R
24 end

```

Referenced In: DetermineAltitudeDependentCOCFactor(p. 181)

DETERMINEVERTICALPROXIMATEFACTOR (Algorithm 160) uses the vertical state of the own aircraft and intruder to determine how much additional sensitivity is required. The vertical samples are projected forward in time by $T_{proximate}$ seconds to obtain the expected vertical separation in the near future. If the vertical samples indicate the own aircraft and intruder will cross altitude in the next $T_{proximate}$ seconds, the anticipated vertical separation is set to zero. Furthermore, if the projected vertical separation is greater than the current separation, the current separation is used instead. The vertical separation samples compared to their associated threshold parameters to determine the proximate factor (*factor_proximate*). As the expected vertical separation decreases, more sensitivity will be applied.

At high altitudes there is a need to provide greater protection against slow vertical closure encounters due to altimetry error biases. To increase sensitivity in these geometries, a special factor (*factor_z_rel_high_alt*) is calculated using the current altitude separation. As the separation decreases, the sensitivity is increased. If at any time this factor would yield a greater level of sensitivity than the proximate factor, it replaces the proximate factor.

This algorithm takes as input *mode_int*, *height_own*, *z_rel*, *dz_own_ave*, *dz_int*, and *H_rel_threshold_proximate*. This algorithm returns *factor_proximate*.

Algorithm 160 DetermineVerticalProximateFactor

```

1 function DetermineVerticalProximateFactor( mode_int::Z, height_own::R, z_rel::R, dz_own_ave::R,
2                                     dz_int::R, H_rel_threshold_proximate::R )
3   const T_proximate::R =
4     params().modes[mode_int].cost_estimation.online.altitude_dependent_coc.T_proximate
5   const H_threshold_hi::R =
6     params().modes[mode_int].cost_estimation.online.altitude_dependent_coc.H_threshold_hi
7   const H_rel_rolloff_proximate::R =
8     params().modes[mode_int].cost_estimation.online.altitude_dependent_coc.H_rel_rolloff_proximate
9   const H_rel_highalt_threshold::R =
10    params().modes[mode_int].cost_estimation.online.altitude_dependent_coc.H_rel_highalt_threshold
11   const H_rel_highalt_rolloff::R =
12    params().modes[mode_int].cost_estimation.online.altitude_dependent_coc.H_rel_highalt_rolloff
13   z_rel_proximate::R = z_rel + ((dz_int - dz_own_ave) * T_proximate)
14   if IsProjectedCrossing(p.337)( z_rel, z_rel_proximate )
15     z_rel_proximate = 0.0
16   elseif (abs( z_rel ) < abs( z_rel_proximate ))
17     z_rel_proximate = abs( z_rel )
18   else
19     z_rel_proximate = abs( z_rel_proximate )
20   end
21   factor_proximate::R = CalculateThresholdRampDownFactor(p.326)( z_rel_proximate, H_rel_threshold_proximate,
22                 H_rel_rolloff_proximate )
23   factor_z_rel_high_alt::R = 0
24   if (H_threshold_hi < height_own)
25     factor_z_rel_high_alt =
26       CalculateThresholdRampDownFactor(p.326)( abs(z_rel), H_rel_highalt_threshold, H_rel_highalt_rolloff
27         )
28   end
29   if (factor_proximate < factor_z_rel_high_alt)
30     factor_proximate = factor_z_rel_high_alt
31   end
32   return factor_proximate::R
32 end

```

Referenced In: DetermineAltitudeDependentCOCFactor(p.181)

3.2.3 Offline Cost Estimation

OFFLINECOSTESTIMATION (Algorithm 161) outputs a vector containing offline costs associated with each action for an individual intruder based on the dynamic state and the resolution advisory (RA) state. The dynamic state is received as input through the *samples_vert* and *b_tau_int* variables while *action_own_prev* and *action_indiv_prev* contain the RA state. The costs are computed by looking up pre-calculated values stored in tables that were generated and compressed offline. The principles of offline cost table generation can be found in [10]. The compressed offline cost tables are provided as part of the ACAS X system.

The input variable *equip_int* is used to choose either the previous global action or the previous individual action for this intruder to use as the current action when looking up the offline cost values for all possible subsequent actions. If the intruder is equipped, the individual advisory is used to maintain consistency with the last coordination message transmitted. If the intruder is unequipped, the global advisory is used. In this context, the value of *equip_int* is TRUE only when the intruder is able to send resolution advisory coordination messages.

Pre-processing is required to prepare the dynamic states for use by the tables. The tau values in the offline cost table may not align perfectly with the tau values in *b_tau_int* produced by TAUESTIMATION. GETOFFLINECOSTTAUBELIEFS (Algorithm 162) returns a TAUBELIEF structure (*b_tau_ec_int*), with the values of *tau* corresponding to the tau values in the offline cost table and the values

of *weight* based on the weight values in *b_tau_int*. GETEQUIVCLASSTABLEMAXCUTVALUES (Algorithm 164) returns a COMBINEDVERTICALBELIEF structure (*vtable_max*), with the values of *z_rel*, *dz_own*, and *dz_int* set to the edges of the offline cost table. The *weight* value in that data structure is always set to 1.0.

The algorithm loops over the set of vertical samples to produce an offline cost vector of all the vertical actions. Vertical belief samples are extracted one-by-one from *samples_vert* for individual processing. OFFLINESTATESSCALEFACTOR (Algorithm 163) is used to gracefully handle values of *z_rel*, *dz_own*, and *dz_int* that go beyond the edges of the offline cost table. If one or more of the dynamic state values is beyond the edge of the table, as captured in *vtable_max*, then *scale_factor* is used to proportionally adjust all three values to the appropriate table values. In most cases, the value of *scale_factor* will be 1.0 and have no effect on the belief state values.

The dynamic state point and offline tables produce an offline cost vector of all the actions, including those not currently available, through a call to LOOKUPOFFLINECOST (Algorithm 166). OFFLINECOSTESTIMATION sums the weighted, negative of the results from each call to LOOKUPOFFLINECOST to convert from “values” (negative costs) to costs.

This algorithm takes as input *mode_int*, *samples_vert*, *b_tau_int*, *equip_int*, *action_own_prev*, and *action_indiv_prev*. This algorithm returns *cost*.

Algorithm 161 OfflineCostEstimation

```

1 function OfflineCostEstimation( mode_int::Z, samples_vert::Vector{CombinedVerticalBelief(p.E-34)},  

2                                b_tau_int::Vector{TauBelief(p.E-38)}, equip_int::Bool,  

3                                action_own_prev::Z, action_indiv_prev::Z )  

4     const N_actions::Z = params().actions.num_actions  

5     action_prev_real::R = convert( R, COC )  

6     if equip_int  

7         action_prev_real = convert( R, action_indiv_prev )  

8     else  

9         action_prev_real = convert( R, action_own_prev )  

10    end  

11    b_tau_ec_int::Vector{TauBelief(p.E-38)} = GetOfflineCostTauBeliefs(p.185)( mode_int, b_tau_int )  

12    vtable_max::CombinedVerticalBelief(p.E-34) = GetEquivClassTableMaxCutValues(p.186)( mode_int )  

13    cost::Vector{R} = zeros( R, N_actions )  

14    for j in 1:length( samples_vert )  

15        scale_factor::R = OfflineStatesScaleFactor(p.186)( samples_vert[j], vtable_max )  

16        z_rel_scaled::R = scale_factor * samples_vert[j].z_rel  

17        dz_own_scaled::R = scale_factor * samples_vert[j].dz_own  

18        dz_int_scaled::R = scale_factor * samples_vert[j].dz_int  

19        point::Vector{R} = [ z_rel_scaled,  

20                            dz_own_scaled,  

21                            dz_int_scaled,  

22                            action_prev_real  

23                            ]  

24        point_cost::Vector{R} = LookupOfflineCost(p.188)( mode_int, point, N_actions, b_tau_ec_int )  

25        for k = 1:N_actions  

26            cost[k] = cost[k] - (samples_vert[j].weight * point_cost[k])  

27        end  

28    end  

29    return cost::Vector{R}  

30 end
```

Referenced In: StateAndCostEstimation(p.161)
--

GETOFFLINECOSTTAUBELIEFS (Algorithm 162) maps the tau beliefs from TAUESTIMATION into the tau values in the equivalence class table to create an array of tau beliefs that can be used in

LOOKUPOFFLINECOST. The tau values used in the equivalence class table are read directly from that table into T_{tau_values} . The tau values in the output tau belief vector ($b_{tau_ec_int}$) are set to the tau values in T_{tau_values} . The weight associated with each tau value in $b_{tau_ec_int}$ is the sum of the weights for the associated range of tau values in b_{tau_int} , the tau beliefs from TAUESTIMATION. The weight for the last tau value in $b_{tau_ec_int}$ is the sum of all the remaining weights in b_{tau_int} .

This algorithm takes as input *mode_int* and *b_tau_int*. This algorithm returns *b_tau_ec_int*.

Algorithm 162 GetOfflineCostTauBeliefs

```

1 function GetOfflineCostTauBeliefs( mode_int::Z, b_tau_int::Vector{TauBelief(p.E-38)} ) 
2   const equiv_class_table_content::Vector{RDataTable(p.E-38)} =
3     params().modes[mode_int].cost_estimation.offline.origami.equiv_class_table_content
4   const N_tau_values::Z           = equiv_class_table_content[1].cut_counts[1]
5   const T_tau_values::Vector{R} = equiv_class_table_content[1].cuts[1:N_tau_values]
6   t_in::Z = length( b_tau_int )
7   i_in::Z = 1
8   b_tau_ec_int::Vector{TauBelief} = Array( TauBelief(p.E-38), N_tau_values )
9   for i_out = 1:N_tau_values
10    b_tau_ec_int[i_out] = TauBelief(p.E-38)( T_tau_values[i_out], 0.0 )
11    while (i_in <= t_in) && (b_tau_int[i_in].tau <= b_tau_ec_int[i_out].tau)
12      b_tau_ec_int[i_out].weight = b_tau_ec_int[i_out].weight + b_tau_int[i_in].weight
13      i_in = i_in + 1
14    end
15  end
16  while (i_in <= t_in)
17    b_tau_ec_int[N_tau_values].weight = b_tau_ec_int[N_tau_values].weight + b_tau_int[i_in].weight
18    i_in = i_in + 1
19  end
20  return b_tau_ec_int::Vector{TauBelief(p.E-38)}
21 end
```

Referenced In: OfflineCostEstimation(p. 184)

OFFLINESTATESSCALEFACTOR (Algorithm 163) scales values that are beyond the edges of the offline cost table. Two of the table states, the current advisory and horizontal tau, cannot have values outside the prescribed limits. All valid RA states are represented and the final tau state is understood to be infinite.

The remaining three states, relative vertical separation, own aircraft vertical rate, and intruder vertical rate, can all fall outside the bounds of the offline cost table. The case of large relative vertical separation is generally not of interest to the logic and the final entry can be safely used for all values that fall beyond it. The case of large vertical rates, on the other hand, does have performance implications if not handled properly. While not generally expected, high performance aircraft can exceed the vertical rates used in the table.

In order to account for this case, the estimated vertical separation and both vertical rates can be scaled while preserving the time until loss of vertical separation. This is accomplished by computing a *scale_factor*, which is simply the largest magnitude vertical rate outside the table bounds divided by the largest vertical rate in the table. In the likely event that neither vertical rate is outside the table edge, the *scale_factor* is set to 1.0 so as to have no effect. The *scale_factor* can then be used in OFFLINECOSTESTIMATION to alter the vertical belief samples immediately before they are used to look-up costs in the table. The scaling does not persist beyond the table look-ups; all future uses of the vertical states use the original, unscaled values.

This algorithm takes as input *sample_vert* and *vtable_max*. This algorithm returns *scale_factor*.

Algorithm 163 OfflineStatesScaleFactor

```

1 function OfflineStatesScaleFactor( sample_vert::CombinedVerticalBelief, vtable_max::CombinedVerticalBelief(p. E-34) )
2   scale_factor::R = 1.0
3   if (abs( vtable_max.dz_own ) < abs( sample_vert.dz_own ))
4     scale_factor = abs( vtable_max.dz_own / sample_vert.dz_own )
5   end
6   if (abs( vtable_max.dz_int ) < abs( sample_vert.dz_int ))
7     scale_factor = min( scale_factor, abs( vtable_max.dz_int / sample_vert.dz_int ) )
8   end
9   return scale_factor::R
10 end
```

Referenced In: OfflineCostEstimation(p. 184)

GETEQUIVCLASSTABLEMAXCUTVALUES (Algorithm 164) returns the maximum cut value for each dimension in the offline cost table. The values are returned in a COMBINEDVERTICALBELIEF data structure in which the weight value is set to 1.0.

It uses the helper algorithm GETTABLEMAXCUTVALUE to return the maximum value for a specific dimension. The dimensions are referenced by name.

The format of the offline cost table is described in Appendix G, DATA TABLE FORMAT SPECIFICATION.

This algorithm takes as input *mode_int*. This algorithm returns *table_max*.

Algorithm 164 GetEquivClassTableMaxCutValues

```

1 function GetEquivClassTableMaxCutValues( mode_int::Z )
2   const equiv_class_table_content::Vector{RDataTable(p. E-38)} =
3     params().modes[mode_int].cost_estimation.offline.origami.equiv_class_table_content
4   table_max::CombinedVerticalBelief = CombinedVerticalBelief(p. E-34)()
5   table_max.z_rel = GetTableMaxCutValue(p. 187)( equiv_class_table_content[1], "relh" )
6   table_max.dz_own = GetTableMaxCutValue(p. 187)( equiv_class_table_content[1], "dh.0" )
7   table_max.dz_int = GetTableMaxCutValue(p. 187)( equiv_class_table_content[1], "dh.1" )
8   table_max.weight = 1.0
9   return table_max::CombinedVerticalBelief(p. E-34)
10 end
```

Referenced In: OfflineCostEstimation(p. 184)

GETTABLEMAXCUTVALUE (Algorithm 165) returns the maximum value for a specific dimension in the table. This is done by first finding the index in the *table_data.cut_counts* array corresponding to the specified dimension. It then determines the indices for the start and end of the cuts associated with that dimension. Finally, it goes through that list of cuts to find the maximum value, which it returns.

This algorithm takes as input *table_data* and *cut_name*. This algorithm returns *max_cut_value*.

Algorithm 165 GetTableMaxCutValue

```

1 function GetTableMaxCutValue( table_data::RDataTable(p. E-38) , cut_name::String )
2   max_cut_value::R = 0.0
3   cut_start::Z = 1
4   cut_index::Z = 1
5   for i in 1:length(table_data.cut_names)
6     if (table_data.cut_names[i] == cut_name)
7       cut_index = i
8       break
9     end
10   end
11   for i in 1:cut_index-1
12     cut_start = cut_start + table_data.cut_counts[i]
13   end
14   cut_end::Z = cut_start + table_data.cut_counts[cut_index] - 1
15   for i in cut_start:cut_end
16     max_cut_value = max( max_cut_value, table_data.cuts[i] )
17   end
18   return max_cut_value::R
19 end

```

Referenced In: GetEquivClassTableMaxCutValues(*p. 186*)

LOOKUPOFFLINECOST (Algorithm 166) performs the cost table lookups for a given state point across all the tau beliefs. The generated tables are compressed using a technique called Data Origami [3]. Data Origami reformats the original offline table into a collection of smaller tables referred to as splits, where each split corresponds to one of the allowable state transitions from the current action (resolution advisory state). Each of these split tables is normalized by subtracting out a set of MinBlock costs, which are the minimum costs across the tau dimension. Lastly, the normalized split tables are compared against one another and similar splits are identified and grouped into equivalence classes. The resulting MinBlocks and equivalence class tables are fixed sized block dictionaries as specified in Appendix G.

Due to the discrete nature of the data tables, most input states cannot be looked up directly in the table. Instead, input states must be represented using a collection of valid table indices that have been weighted appropriately. This mapping from input states to table indices is performed using multi-linear interpolation through the call to INTERPOLANTS (Algorithm 279). After this mapping, LOOKUPOFFLINECOST loops over the allowable actions, the mapped table indices, and the tau beliefs to extract the appropriate values in the cost table, weighting and summing as it goes along. Unavailable actions are given infinite cost as a result of the initialization of *cost*. Upon completion, the cost vector for all the actions is returned.

This algorithm takes as input *mode_int*, *point*, *num_actions*, and *b_tau_int*. This algorithm returns *cost*.

Algorithm 166 LookupOfflineCost

```

1 function LookupOfflineCost( mode_int::Z, point::Vector{R}, num_actions::Z, b_tau_int::Vector{TauBelief(p.E-38)} )
2   const action_pattern_number::Vector{Z} =
3     params().modes[mode_int].cost_estimation.offline.origami.action_pattern_number
4   const N_allowable_actions::Vector{Z} =
5     params().modes[mode_int].cost_estimation.offline.origami.N_allowable_actions
6   const allowable_actions::Array{Z,2} =
7     params().modes[mode_int].cost_estimation.offline.origami.allowable_actions
8   const equiv_class_number::Array{Z,2} =
9     params().modes[mode_int].cost_estimation.offline.origami.equiv_class_number
10  const minblocks_index::Array{Z,2} =
11    params().modes[mode_int].cost_estimation.offline.origami.minblocks_index
12  const minblocks_table_content::RDataTable(p.E-38) =
13    params().modes[mode_int].cost_estimation.offline.origami.minblocks_table_content
14  const equiv_class_table_content::Vector{RDataTable(p.E-38)} =
15    params().modes[mode_int].cost_estimation.offline.origami.equiv_class_table_content
16  cost::Vector{R} = fill( convert( R, -Inf ), num_actions )
17  st_trans::Z = convert( Z, point[4] )
18  action_pattern::Z = action_pattern_number[st_trans]
19  N_allowable_action_count::Z = N_allowable_actions[action_pattern]
20  (indices::Vector{Z}, weights::Vector{R}, block_size::Z) =
21    Interpolants(p.329)( [0, point], minblocks_table_content.cut_counts,
22                          2, 3, minblocks_table_content.cuts )
23  tau_value_count::Z = equiv_class_table_content[1].cut_counts[1]
24  split_num_count::Z = minblocks_table_content.cut_counts[1]
25  index_ec_max::Z = (block_size * tau_value_count) + 1
26  index_ec_pos::Vector{Z} = Array( Z, length( indices ) )
27  index_ec_neg::Vector{Z} = Array( Z, length( indices ) )
28  index_mb::Vector{Z} = Array( Z, length( indices ) )
29  for j = 1:length( indices )
30    index_ec_pos[j] = ((indices[j] - 1) * tau_value_count) + 1
31    index_ec_neg[j] = index_ec_max - index_ec_pos[j] - tau_value_count + 1
32    index_mb[j] = ((indices[j] - 1) * split_num_count) + 1
33  end
34  for k = 1:N_allowable_action_count
35    action::Z = allowable_actions[ action_pattern, k ]
36    equiv_class::Z = equiv_class_number[ action, st_trans ]
37    split_num::Z = minblocks_index[ action, st_trans ]
38    index_ec::Vector{Z}
39    if (0 <= equiv_class)
40      index_ec = index_ec_pos
41    else
42      equiv_class = -equiv_class
43      index_ec = index_ec_neg
44    end
45    data_ec::Vector{Float16} = equiv_class_table_content[equiv_class].data
46    data_mb::Vector{Float16} = minblocks_table_content.data
47    cost_act::R = -Inf
48    for j = 1:length( indices )
49      cost_point::R = 0.0
50      offset_ec::UInt32 = index_ec[j] - 1
51      for i = 1:length( b_tau_int )
52        cost_point = cost_point + (b_tau_int[i].weight * convert( R, data_ec[offset_ec + i] ))
53      end
54      offset_mb::UInt32 = index_mb[j] + split_num - 1
55      if (cost_act == -Inf)
56        cost_act = weights[j] * (cost_point + convert( R, data_mb[offset_mb] ))
57      else
58        cost_act = cost_act + (weights[j] * (cost_point + convert( R, data_mb[offset_mb] )))
59      end
60    end
61    cost[action] = cost_act
62  end
63  return (cost::Vector{R})
64 end

```

Referenced In: OfflineCostEstimation(p.184)

3.2.4 Online Cost Estimation Independent of Coordination

An overview of the ONLINEUNCOORDINATEDCOSTESTIMATION component in STATEANDCOSTESTIMATION is shown in Figure 3-4.

ONLINEUNCOORDINATEDCOSTESTIMATION (Algorithm 167) determines the values of the online costs that are not dependent on coordination with intruding aircraft. Online costs capture characteristics of the current state, such as altitudes, that cannot be suitably reflected in the offline cost tables. The online costs determined in this algorithm are independent of the resolution advisory coordination state of own aircraft and the intruder aircraft.

ONLINEUNCOORDINATEDCOSTESTIMATION always performs its evaluation with regard to a single intruder, independent of any other intruders.

It first computes values needed by the various online cost algorithms.

GETMODIFIEDGLOBALRATES (Algorithm 168) determines the vertical rate limits for the previous cycle (dz_min_prev and dz_max_prev) to be used for computing online costs.

EXPECTEDTAU (Algorithm 276) is called twice to provide a point estimate for time until loss of separation under different assumptions. The weighted average value for tau including the horizon ($tau_expected$) and excluding the horizon ($tau_expected_no_horizon$) are used for computing online costs. See section 3.2.2.1 for a discussion of tau.

The cost for each action in the output cost vectors, $cost_ra$ and $cost_ta$, is initialized to 0.0.

The *update* variable is initialized to TRUE. The *update* variable is used to flag the first iteration so each online cost algorithm will know to update the state variables in s_c . That set of updates will occur only once for each intruder on each cycle.

ONLINEUNCOORDINATEDCOSTESTIMATION then loops over all the possible actions to compute the online costs for each action. Each action (*act*) is represented as a minimum and maximum vertical rate (dz_min and dz_max).

ACTIONTORATES (Algorithm 273) is used to obtain the minimum and maximum vertical rates for the action under evaluation. ACTIONTORATES is a frequently used helper algorithm that is intentionally omitted from the flow diagrams.

For each action under evaluation, ONLINEUNCOORDINATEDACTIONCOSTESTIMATION (Algorithm 169) computes the sum of the online costs to be used for selecting resolution advisories ($cost_ra[act]$). It also computes the sum of the online costs to be used for determining traffic advisories ($cost_ta[act]$).

Later, in ONLINECOSTESTIMATION (Section 3.2.6), the coordination-independent online costs for each action, in $cost_ra$ and $cost_ta$, will be summed with the associated coordination-dependent online costs.

ONLINEUNCOORDINATEDCOSTESTIMATION requires knowledge of numerous variables to calculate the various online costs. The required inputs include the previous global resolution advisory issued by the system (a_global_prev), the previous resolution advisory for the intruder considered in isolation (a_indiv_prev), and the online cost state (s_c). The weighted average of the own aircraft and intruder's vertical states (h_own , z_own_ave , dz_own_ave , z_int_ave , and dz_int_ave) are also needed along with the samples and corresponding weights for tau (b_tau_int) and whether the intruder is able to send resolution advisory coordination messages ($equip_int$).

Descriptions of the **ONLINEUNCOORDINATEDCOSTESTIMATION** output variables are found in Table 3-4.

This algorithm takes as input *mode_int*, *h_own*, *a_global_prev*, *a_indiv_prev*, *z_own_ave*, *dz_own_ave*, *z_int_ave*, *dz_int_ave*, *b_tau_int*, *equip_int*, and *s_c*. This algorithm outputs *cost_ra* and *cost_ta*.

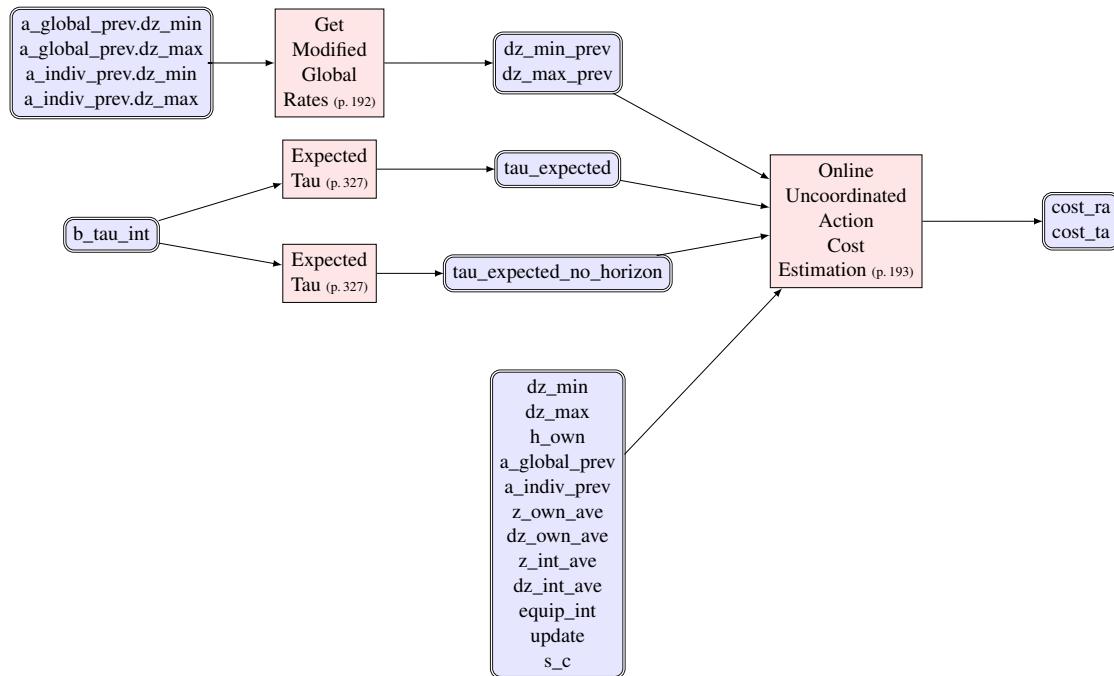


Figure 3-4. OnlineUncoordinatedCostEstimation overview.

Table 3-4. TRM Algorithm Output Variables - OnlineUncoordinatedCostEstimation

Variable	Units	Type	Description
<i>cost_ra</i>	N/A	real	Vector with online coordination-independent cost of each action; used to select RAs for single intruder
<i>cost_ta</i>	N/A	real	Vector with online coordination-independent cost of each action; used to determine TAs for single intruder
<i>s_c</i>	N/A	OnlineCostState(p. E-37)	State variables for estimating online costs

Algorithm 167 OnlineUncoordinatedCostEstimation

```

1 function OnlineUncoordinatedCostEstimation( mode_int::Z, h_own::R,
2           z_own_ave::R, dz_own_ave::R, z_int_ave::R, dz_int_ave::R,
3           b_tau_int::Vector{TauBelief(p.E-38)},
4           a_global_prev::GlobalAdvisory(p.E-35), a_indiv_prev::IndividualAdvisory(p.E-36),
5           equip_int::Bool, s_c::OnlineCostState(p.E-37) )
6   const N_actions::Z = params().actions.num_actions
7   (dz_min_prev::R, dz_max_prev::R) =
8     GetModifiedGlobalRates(p.192)( a_global_prev.dz_min, a_global_prev.dz_max,
9                                   a_indiv_prev.dz_min, a_indiv_prev.dz_max )
10  tau_expected::R          = ExpectedTau(p.327)( b_tau_int, false )
11  tau_expected_no_horizon::R = ExpectedTau(p.327)( b_tau_int, true )
12  cost_ra::Vector{R} = zeros( R, N_actions )
13  cost_ta::Vector{R} = zeros( R, N_actions )
14  update::Bool          = true
15  for act::Z in 1:N_actions
16    (dz_min::R, dz_max::R) = ActionToRates(p.325)( act, dz_own_ave, a_global_prev.action,
17                                                 a_global_prev.dz_min, a_global_prev.dz_max,
18                                                 a_global_prev.ddz )
19    (cost_ra[act], cost_ta[act]) =
20      OnlineUncoordinatedActionCostEstimation(p.193)( mode_int, dz_min, dz_max, h_own,
21                                                 a_global_prev, a_indiv_prev, dz_min_prev, dz_max_prev,
22                                                 z_own_ave, dz_own_ave, z_int_ave, dz_int_ave,
23                                                 tau_expected, tau_expected_no_horizon, equip_int, update, s_c )
24    update = false
25  end
26  return (cost_ra::Vector{R}, cost_ta::Vector{R})
27 end

```

Referenced In: StateAndCostEstimation(p. 161)

GETMODIFIEDGLOBALRATES (Algorithm 168) determines the values of the minimum and maximum vertical rates representing the advisory for the previous cycle. These vertical rate limits are used for computing online costs for a single intruder.

The vertical rate limits are adjusted in multi-intruder scenarios to properly account for differences between the resolution advisory (RA) for the individual intruder (or individual RA) and the global RA. When the individual RA is clear of conflict (COC) or the global RA is Multi-Threat Level-Off (MTLO), the vertical rate limits associated with the individual RA are used.

When there is a single intruder, the global RA and the individual RA will be the same, and so will their minimum and maximum vertical rates.

The individual RA is represented as the minimum and maximum vertical rates (*dz_min_int_prev* and *dz_max_int_prev*). The global RA is represented as the minimum and maximum vertical rates (*dz_min_prev* and *dz_max_prev*). Both RAs were produced on the previous TRM cycle.

GETMODIFIEDGLOBALRATES sets the vertical rate limits (*dz_min* and *dz_max*) as follows:

- When there was no RA on the previous cycle for the individual intruder, the vertical rate limits for clear of conflict (COC) are used.
- When the previous global RA was a Multi-Threat Level-Off (MTLO), the sense of the individual RA (*sense_indiv*) is used to determine the vertical rate limits. In this case, the vertical rate limits are set to 'Do Not Descend' if the sense is up and to 'Do Not Climb' if the sense is down. If the sense is neither up nor down, the vertical rate limits associated with the individual RA are used.

- Otherwise, the vertical rate limits for the global RA are used.

This algorithm takes as input dz_min_prev , dz_max_prev , $dz_min_int_prev$, and $dz_max_int_prev$. This algorithm returns dz_min and dz_max .

Algorithm 168 GetModifiedGlobalRates

```

1 function GetModifiedGlobalRates( dz_min_prev::R, dz_max_prev::R, dz_min_int_prev::R, dz_max_int_prev::R )
2   dz_min::R
3   dz_max::R
4   if IsCOC(p. 332)( dz_min_int_prev, dz_max_int_prev )
5     dz_min = -Inf
6     dz_max = Inf
7   elseif IsMTLO(p. 336)( dz_min_prev, dz_max_prev )
8     sense_indiv::Symbol = RatesToSense(p. 342)( dz_min_int_prev, dz_max_int_prev )
9     if (sense_indiv == :Up)
10       dz_min = 0
11       dz_max = Inf
12     elseif (sense_indiv == :Down)
13       dz_min = -Inf
14       dz_max = 0
15     else
16       dz_min = dz_min_int_prev
17       dz_max = dz_max_int_prev
18     end
19   else
20     dz_min = dz_min_prev
21     dz_max = dz_max_prev
22   end
23   return (dz_min::R, dz_max::R)
24 end
```

Referenced In: OnlineUncoordinatedCostEstimation(<i>p. 191</i>), OnlineCostEstimation(<i>p. 214</i>)

ONLINEUNCOORDINATEDACTIONCOSTESTIMATION (Algorithm 169) returns two online cost sums for the given action and the individual intruder. The online costs determined in this algorithm are independent of the coordination state of own aircraft and the intruding aircraft. The associated algorithm, ONLINECOORDINATEDACTIONCOSTESTIMATION, determines online costs that are dependent on the resolution advisory coordination state of own aircraft and the intruding aircraft.

The value *cost_ra* contains the sum of all the individual online costs for the given action. These costs are used when determining whether a resolution advisory should be issued, and if so, the type of resolution advisory to issue.

The value *cost_ta* contains the sum of the subset of individual online costs for the given action to be used when determining whether or not to issue a traffic advisory. Presently, the only coordination-independent online cost algorithm contributing to *cost_ta* is ALTITUDEDEPENDENTCOCCOST. The ALTITUDEDEPENDENTCOCCOST online cost is used to adjust the offline cost for altitude so traffic advisories will be useful precursors of resolution advisories at all altitudes.

Each online cost algorithm called in ONLINEUNCOORDINATEDACTIONCOSTESTIMATION corresponds to a different event to be penalized, such as reversing or restarting the resolution advisory. After initialization, the algorithms update their own state variables only once during each call to ONLINEUNCOORDINATEDACTIONCOSTESTIMATION, even though ONLINEUNCOORDINATEDACTIONCOSTESTIMATION, and consequently each online cost algorithm, is called once for each action. The *update* value is used to indicate when the individual online cost algorithms should

update their state variables, contained in the `ONLINECOSTSTATE` data structure. The value is set to `TRUE` when computing the online cost for the first action and `FALSE` subsequently.

In the individual online cost algorithms, an equipped intruder is an intruding aircraft that is able to send resolution advisory coordination messages. The equipped intruder is indicated by a value of TRUE for *equip_int*.

Common inputs to the individual online cost algorithms include the index to be used for acquiring the appropriate values of the configuration parameters (*mode_int*), the minimum vertical rate associated with the action (*dz_min*), the maximum vertical rate associated with the action (*dz_max*), *update*, and the relevant set of online cost state variables (*s_c*). The variable *mode_int* is the protection mode index set by GETPROTECTIONMODEINDEX (Algorithm 278) in INTRUDERPREP.

This algorithm takes as input `mode_int`, `dz_min`, `dz_max`, `h_own`, `a_global_prev`, `a_indiv_prev`, `dz_min_prev`, `dz_max_prev`, `z_own_ave`, `dz_own_ave`, `z_int_ave`, `dz_int_ave`, `tau_expected`, `tau_expected_no_horizon`, `equip_int`, `update` and `s_c`. This algorithm returns `cost_ra` and `cost_ta`.

Algorithm 169 OnlineUncoordinatedActionCostEstimation

```

1 function OnlineUncoordinatedActionCostEstimation( mode_int::Z, dz_min::R, dz_max::R, h_own::R,
2                                     a_global_prev::GlobalAdvisory(p.E-35), a_indiv_prev::IndividualAdvisory(p.E-36),
3                                     dz_min_prev::R, dz_max_prev::R,
4                                     z_own_ave::R, dz_own_ave::R, z_int_ave::R, dz_int_ave::R,
5                                     tau_expected::R, tau_expected_no_horizon::R, equip_int::Bool,
6                                     update::Bool, s_c::OnlineCostState(p.E-37) )
7
8     inc_cost::R = 0.0
9     cost_ra::R = inc_cost
10    cost_ta::R = 0.0
11    inc_cost = AltitudeInhibitCost(p.197)( mode_int, dz_min, dz_max, h_own, dz_min_prev, dz_max_prev,
12                                         s_c.alt_inhibit )
13    cost_ra = inc_cost
14    inc_cost = AdvisoryRestartCost(p.194)( mode_int, dz_min, dz_max, dz_min_prev, dz_max_prev,
15                                         update, s_c.advisory_restart )
16    cost_ra = cost_ra + inc_cost
17    inc_cost = InitializationCost(p.205)( mode_int, dz_min, dz_max, update, s_c.initialization )
18    cost_ra = cost_ra + inc_cost
19    inc_cost = BadTransitionCost(p.198)( mode_int, dz_min, dz_max, dz_own_ave, a_global_prev.dz_min,
20                                         a_global_prev.dz_max, a_indiv_prev.dz_min, a_indiv_prev.dz_max,
21                                         equip_int, update, s_c.bad_transition )
22    cost_ra = cost_ra + inc_cost
23    inc_cost = PreventEarlyCOCCost(p.206)( mode_int, dz_min, dz_max, dz_min_prev, dz_max_prev,
24                                         z_own_ave, dz_own_ave, z_int_ave, dz_int_ave,
25                                         tau_expected, update, s_c.prevent_early_coc )
26    cost_ra = cost_ra + inc_cost
27    inc_cost = AltitudeDependentCOCCost(p.196)( mode_int, dz_min, dz_max, dz_min_prev, dz_max_prev, dz_own_ave,
28                                         s_c.altitude_dependent_coc.scaled_cost_coc_ra )
29    cost_ra = cost_ra + inc_cost
30    inc_cost = AltitudeDependentCOCCost(p.196)( mode_int, dz_min, dz_max, dz_min_prev, dz_max_prev, dz_own_ave,
31                                         s_c.altitude_dependent_coc.scaled_cost_coc_ta )
32    cost_ta = cost_ta + inc_cost
33    inc_cost = CriticalIntervalProtectionCost(p.201)( mode_int, dz_min, dz_max, dz_min_prev, dz_max_prev,
34                                         z_own_ave, z_int_ave, dz_own_ave, dz_int_ave,
35                                         update, s_c.critical_interval_protection )
36    cost_ra = cost_ra + inc_cost
37    inc_cost = SafeCrossingRADeferralCost(p.208)( mode_int, dz_min, dz_max, equip_int, z_own_ave, z_int_ave,
38                                         dz_own_ave, dz_int_ave, dz_min_prev, dz_max_prev,
39                                         tau_expected_no_horizon, update, s_c.safe_crossing_ra_deferral )
40    cost_ra = cost_ra + inc_cost
41    return (cost_ra::R, cost_ta::R)
42 end

```

Referenced In: OnlineUncoordinatedCostEstimation(p. 191)

3.2.4.1 Advisory Restart Cost

ADVISORYRESTARTCOST (Algorithm 170) is used to penalize the issuance of a new advisory if a previous advisory was issued recently. The ADVISORYRESTARTCOST algorithm penalizes resolution advisories by $C_{advisory}$ whenever a prior resolution advisory has been terminated for fewer than T_{limit} processing cycles. To do this, ADVISORYRESTARTCOST maintains memory of whether an advisory has been issued ($s_c.alerted$), whether an advisory has terminated ($s_c.term$), and the amount of time for which an advisory has terminated ($s_c.t_term$).

Once per cycle, the state variables in s_c are set using the helper algorithm UPDATEADVISORYRESTARTCSTATE.

This algorithm takes as input $mode_int$, dz_min , dz_max , dz_min_prev , dz_max_prev , $update$ and s_c . This algorithm returns $cost$.

Algorithm 170 AdvisoryRestartCost

```

1 function AdvisoryRestartCost( mode_int::Z, dz_min::R, dz_max::R, dz_min_prev::R, dz_max_prev::R,
2           update::Bool, s_c::AdvisoryRestartCState(p. E-42) )
3   const T_limit::Z = params().modes[mode_int].cost_estimation.online.advisory_restart.T_limit
4   const C_advisory::R = params().modes[mode_int].cost_estimation.online.advisory_restart.C_advisory
5   if update
6     UpdateAdvisoryRestartCState(p. 195)( dz_min_prev, dz_max_prev, s_c )
7   end
8   cost::R = 0.0
9   if s_c.term && (s_c.t_term < T_limit) && !IsCOC(p. 332)( dz_min, dz_max )
10    cost = C_advisory
11  end
12  return cost::R
13 end
```

Referenced In: OnlineUncoordinatedActionCostEstimation(p. 193)

UPDATEADVISORYRESTARTCSTATE (Algorithm 171) is used to update the state variables of ADVISORYRESTARTCOST once per cycle. First, the algorithm checks to see if there was no alert on the previous cycle and either there was an alert two cycles prior or that an alert has been terminated any time previously. If true, the advisory termination flag ($s_c.term$) is set to TRUE and the alert termination counter ($s_c.t_term$) is incremented. If not, the advisory termination flag is set to FALSE and the alert termination counter is reset. Finally, the previously alerted flag ($s_c.alerted$) is updated using the vertical rate limits of the previous cycle for use in the next cycle.

This algorithm takes as input dz_min_prev , dz_max_prev , and s_c . The algorithm outputs updates to $s_c::AdvisoryRestartCState$.

Algorithm 171 UpdateAdvisoryRestartCState

```

1 function UpdateAdvisoryRestartCState( dz_min_prev::R, dz_max_prev::R, s_c::AdvisoryRestartCState(p_E-42) )
2   if (s_c.alerted || s_c.term) && IsCOC(p_332)(dz_min_prev, dz_max_prev)
3     s_c.term = true
4     s_c.t_term = s_c.t_term + 1
5   else
6     s_c.term = false
7     s_c.t_term = 0
8   end
9   s_c.alerted = !IsCOC(p_332)(dz_min_prev, dz_max_prev)
10 end

```

Referenced In: AdvisoryRestartCost(<i>p_194</i>)

3.2.4.2 Altitude Dependent Clear of Conflict Cost

ALTITUDEDEPENDENTCOCCOST (Algorithm 172) is used to penalize transitions to clear of conflict as a function of altitude. The effect is scaled so that at higher altitudes the transitions to clear of conflict or preventives are more heavily penalized. Transitions to preventives are also penalized as the own aircraft vertical rate approaches level as these advisories will not increase separation. This effect is controlled through the call to CALCULATETHRESHOLDRAMPDOWNFACTOR (Algorithm 274).

The altitude scaling and other geometric scaling takes place before this function when the *scaled_cost_coc* value is computed using DETERMINEALTITUDEDEPENDENTCOCFATOR (Algorithm 158). The *scaled_cost_coc* input is calculated once per intruder per cycle.

The ALTITUDEDEPENDENTCOCCOST algorithm is used twice, once for resolution advisories and once for traffic advisories. The two instances differ only in the value of *scaled_cost_coc*, which is provided as an input argument. The values of *scaled_cost_coc* for resolution advisories and traffic advisories are stored in ALTITUDEDEPENDENTCOCCSTATE.

This algorithm takes as input *mode_int*, *dz_min*, *dz_max*, *dz_min_prev*, *dz_max_prev*, *dz_own_ave*, and *scaled_cost_coc*. This algorithm returns *cost*.

Algorithm 172 AltitudeDependentCOCCost

```

1 function AltitudeDependentCOCCost( mode_int::Z, dz_min::R, dz_max::R, dz_min_prev::R, dz_max_prev::R,
2                                     dz_own_ave::R, scaled_cost_coc::R )
3     const R_threshold_lolo::R =
4         params().modes[mode_int].cost_estimation.online.altitude_dependent_coc.R_threshold_lolo
5     const R_rolloff_lolo::R =
6         params().modes[mode_int].cost_estimation.online.altitude_dependent_coc.R_rolloff_lolo
7     const X_maintain_factor::R =
8         params().modes[mode_int].cost_estimation.online.altitude_dependent_coc.X_maintain_factor
9     cost::R = 0.0
10    if IsCOC(p.332)( dz_min, dz_max )
11        cost = scaled_cost_coc;
12    elseif IsPreventive( dz_min, dz_max ) && !IsPreventive(p.336)( dz_min_prev, dz_max_prev )
13        if (dz_min < dz_own_ave < dz_max)
14            cost = scaled_cost_coc
15        else
16            cost = scaled_cost_coc * CalculateThresholdRampDownFactor(p.326)( abs( dz_own_ave ), 
17                           R_threshold_lolo, R_rolloff_lolo )
18    end
19    elseif IsMaintain(p.335)( dz_min, dz_max ) && IsCOC(p.332)( dz_min_prev, dz_max_prev )
20        cost = scaled_cost_coc * X_maintain_factor
21    end
22    return cost::R
22 end

```

Referenced In: OnlineUncoordinatedActionCostEstimation(p. 193)

3.2.4.3 Altitude Inhibit Cost

ALTITUDEINHIBITCOST (Algorithm 173) controls the alerting behavior against intruder aircraft as the ownship approaches the ground. This is done by penalizing sets of resolution advisories below specified altitudes with the $C_{inhibit}$ cost. The actions for which the cost applies are specified using R_{dz_lo} , R_{dz_hi} , $R_{dz_lo_prev}$, and $R_{dz_hi_prev}$. A negative value in $C_{inhibit}$ is used to limit the applicability of certain altitude inhibits that are too broadly applicable.

The inhibit costs are applied with a simple hysteresis check. To do this, the algorithm keeps knowledge of current altitude inhibited state ($s_{c.inhibited}$). The altitude inhibited state is updated by UPDATEALTITUDEINHIBITCSTATE prior to any threat processing. Whenever the aircraft is below H_{lo} , the cost is incurred for the associated actions. The cost continues to be applied until the aircraft climbs above H_{hi} , at which time the cost is removed. The cost can only become inhibited again if the aircraft descends below H_{lo} . Initialization of the $s_{c.inhibited}$ is defined by the parameter file. The cost is never applied if the action corresponds to clear of conflict (COC). The algorithm IsCOC (Algorithm 281) is used to indicate whether the action corresponds to a clear of conflict.

The parameter sets used in the ALTITUDEINHIBITCOST have the following effects on alerting:

- Increase Descend Inhibit - Inhibits strengthening of either Descend or Maintain (Down) to an Increase Descend RA. It does not alter an Increase Descend RA that is already issued. Set when below 1450' AGL. Reset when above 1650' AGL.
- Descend Inhibit - Inhibits initiation of Descend, Increase Descend or Maintain (Down RA), and if any of these are already issued (Descend RA, Increase Descend RA or Maintain (Down) RA) converts it to a Don't Climb RA. Set when below 1000' AGL. Reset when above 1200' AGL.
- All RA inhibit - Inhibits all RAs but does not result in issuing clear of conflict. Set when below

900' AGL. Reset when above 1100' AGL.

The variable *h_own* is the radar altitude above ground level (AGL). This is used for the low altitude inhibit rather than the barometric altitude as it provides a more precise indication of the height above the ground.

This algorithm takes as input *mode_int*, *dz_min*, *dz_max*, *h_own*, *dz_min_prev*, *dz_max_prev*, and *s_c*. This algorithm returns *cost*.

Algorithm 173 AltitudeInhibitCost

```

1 function AltitudeInhibitCost( mode_int::Z, dz_min::R, dz_max::R, h_own::R,
2                               dz_min_prev::R, dz_max_prev::R,
3                               s_c::AltitudeInhibitCState(p_E-42) )
4   const C_inhibit::Vector{R} =
5     params().modes[mode_int].cost_estimation.online.altitude_inhibit.C_inhibit
6   const R_dz_lo::Vector{R} =
7     params().modes[mode_int].cost_estimation.online.altitude_inhibit.R_dz_lo
8   const R_dz_hi::Vector{R} =
9     params().modes[mode_int].cost_estimation.online.altitude_inhibit.R_dz_hi
10  const R_dz_lo_prev::Vector{R} =
11    params().modes[mode_int].cost_estimation.online.altitude_inhibit.R_dz_lo_prev
12  const R_dz_hi_prev::Vector{R} =
13    params().modes[mode_int].cost_estimation.online.altitude_inhibit.R_dz_hi_prev
14  cost::R = 0.0
15  if !IsCOC(p_332)( dz_min, dz_max )
16    for i = 1:length( C_inhibit )
17      if s_c.inhibited[i] &&
18        ((R_dz_lo[i] <= dz_min) && (dz_max <= R_dz_hi[i])) &&
19        ((R_dz_lo_prev[i] <= dz_min_prev) && (dz_max_prev <= R_dz_hi_prev[i]))
20      cost = cost + C_inhibit[i]
21    end
22  end
23 end
24 return cost::R
25 end
```

Referenced In: OnlineUncoordinatedActionCostEstimation(p. 193)

3.2.4.4 Bad Transition Cost

BADTRANSITIONCOST (Algorithm 174) penalizes resolution advisories by *C_bad_transition* if the transition to that action from the current state is illegal. Penalties for some illegal transitions are encoded directly into the offline cost table. However, due to the limitations inherent to the offline cost table construction and represented states, some transitions must be dealt with online.

Illegal transitions are those matching one or more of the following conditions. Resolution advisories matching more than one of the conditions incur the penalty for each condition matched.

- Maintain Climb to Climb or Increase Descend
- Maintain Descend to Descend or Increase Climb
- Maintain Climb to Increase Climb (if not a strengthening)
- Maintain Descend to Increase Descend (if not a strengthening)
- Climb to Maintain Climb or Descend to Maintain Descend
- Increase Climb to Maintain Climb

- Increase Descend to Maintain Descend
- Reversals to Preventive RAs
- Transitions to Climbs, Descends, Increase Climbs, and Increase Descends if the aircraft has a vertical rate higher than that specified by the advisory
- Transitions to Maintain Climb or Maintain Descend if the aircraft has a vertical rate lower than 1500 fpm

The helper algorithm BADMAINTAINTRANSITIONCOST is used to calculate the costs associated with bad transitions to and from Maintain advisories.

Once per cycle, the state variables in s_c are set using the helper algorithm UPDATEBADTRANSITIONCSTATE.

This algorithm takes as input $mode_int$, dz_min , dz_max , dz_own_ave , $dz_min_global_prev$, $dz_max_global_prev$, $dz_min_indiv_prev$, $dz_max_indiv_prev$, $equip_int$, $update$ and s_c . This algorithm returns $cost$.

Algorithm 174 BadTransitionCost

```

1 function BadTransitionCost( mode_int::Z, dz_min::R, dz_max::R, dz_own_ave::R,
2                               dz_min_global_prev::R, dz_max_global_prev::R,
3                               dz_min_indiv_prev::R, dz_max_indiv_prev::R,
4                               equip_int::Bool, update::Bool, s_c::BadTransitionCState(p.E-42) )
5   const C_bad_transition::R =
6     params().modes[mode_int].cost_estimation.online.bad_transition.C_bad_transition
7   const R_corrective::R      = params().actions.corrective_rate
8   const R_strengthen::R    = params().actions.strengthen_rate
9   if update
10    UpdateBadTransitionCState(p.199)( dz_min_global_prev, dz_max_global_prev,
11                                      dz_min_indiv_prev, dz_max_indiv_prev, equip_int, s_c )
12  end
13  cost::R = 0.0
14  sense_own::Symbol      = RatesToSense(p.342)( dz_min, dz_max )
15  ra_is_maintain::Bool   = IsMaintain(p.335)( dz_min, dz_max )
16  ra_is_strengthening::Bool = IsStrengthening(p.338)( s_c.dz_min_prev, s_c.dz_max_prev, dz_min, dz_max )
17  cost = BadMaintainTransitionCost(p.200)( mode_int, dz_min, dz_max, dz_own_ave, C_bad_transition,
18                                            sense_own, ra_is_maintain, ra_is_strengthening, s_c )
19  if IsPreventive(p.336)( dz_min, dz_max ) && IsReversal(p.337)( s_c.sense_own_prev, sense_own )
20    cost = cost + C_bad_transition
21  end
22  if (R_corrective < dz_own_ave) &&
23    (dz_min == R_corrective) && (dz_max == Inf) &&
24    !(s_c.dz_min_prev == R_corrective) && (s_c.dz_max_prev == Inf))
25    cost = cost + C_bad_transition
26  elseif (R_strengthen < dz_own_ave) &&
27    (dz_min == R_strengthen) && (dz_max == Inf) &&
28    !(s_c.dz_min_prev == R_strengthen) && (s_c.dz_max_prev == Inf))
29    cost = cost + C_bad_transition
30  elseif (dz_own_ave < -R_corrective) &&
31    (dz_min == -Inf) && (dz_max == -R_corrective) &&
32    !(s_c.dz_min_prev == -Inf) && (s_c.dz_max_prev == -R_corrective))
33    cost = cost + C_bad_transition
34  elseif (dz_own_ave < -R_strengthen) &&
35    (dz_min == -Inf) && (dz_max == -R_strengthen) &&
36    !(s_c.dz_min_prev == -Inf) && (s_c.dz_max_prev == -R_strengthen))
37    cost = cost + C_bad_transition
38  end
39  return cost::R
40 end
```

Referenced In: OnlineUncoordinatedActionCostEstimation(p.193)
--

UPDATEBADTRANSITIONCSTATE (Algorithm 175) is used to update the state variables of BADTRANSITIONCOST once per cycle. Due to pairwise coordination between equipped aircraft, bad transitions for equipped intruders are calculated using the previous individual RA while bad transitions for unequipped intruders use the previous global RA. If either the maximum or minimum vertical rate limit changed on the previous cycle, additional processing is required. If the transition on the previous cycle was to clear of conflict, the previous own aircraft RA sense (*s_c.sense_own_prev*) and the previous RA was a Maintain (*s_c.ra_is_maintain_prev*) variables are reset. Otherwise, the previous RA limits are used to determine the sense and whether or not it was a Maintain RA. Finally, the state variables for the previous RA vertical rate limits (*s_c.dz_min_prev* and *s_c.dz_max_prev*) are updated.

This algorithm takes as input *dz_min_global_prev*, *dz_max_global_prev*, *dz_min_indiv_prev*, *dz_max_indiv_prev*, *equip_int*, *update* and *s_c*. This algorithm updates *s_c::BadTransitionCState*.

Algorithm 175 UpdateBadTransitionCState

```

1 function UpdateBadTransitionCState( dz_min_global_prev::R, dz_max_global_prev::R,
2                               dz_min_indiv_prev::R, dz_max_indiv_prev::R,
3                               equip_int::Bool, s_c::BadTransitionCState(p. E-42) )
4   if equip_int
5     dz_min_prev::R = dz_min_indiv_prev
6     dz_max_prev::R = dz_max_indiv_prev
7   else
8     dz_min_prev = dz_min_global_prev
9     dz_max_prev = dz_max_global_prev
10  end
11  if (s_c.dz_min_prev != dz_min_prev) || (s_c.dz_max_prev != dz_max_prev)
12    if IsCOC(p.332)( dz_min_prev, dz_max_prev )
13      s_c.ra_is_maintain_prev = false
14      s_c.sense_own_prev = :None
15    else
16      sense_own_prev = RatesToSense(p.342)( dz_min_prev, dz_max_prev )
17      s_c.ra_is_maintain_prev = IsMaintain(p.335)( dz_min_prev, dz_max_prev )
18      s_c.sense_own_prev = sense_own_prev
19    end
20    s_c.dz_min_prev = dz_min_prev
21    s_c.dz_max_prev = dz_max_prev
22  end
23 end
```

Referenced In: BadTransitionCost(p. 198)

BADMINTAINTRANSITIONCOST (Algorithm 176) is used to calculate the costs associated with bad transitions to and from Maintain advisories. First, the algorithm determines if a transition is being attempted from a Maintain RA to a non-Maintain RA. If so the transition is from Maintain Climb to either Climb or Increase Descend the cost is applied. Additionally, transitions from Maintain Descend to either Descend or Increase Climb are penalized. Finally, any transition from Maintain Climb to Increase Climb and from Maintain Descend to Increase Descend that is not also a strengthening also has the cost applied.

Similarly, the algorithm checks to see if the transition being attempted is from a non-Maintain RA to a Maintain RA. Transitions from Climb or Increase Climb to Maintain Climb and transitions from Descend or Increase Descend to Maintain Descend are penalized. Moreover, the initiation of a Maintain RA while the own aircraft vertical rate is below 1500 fpm is likewise penalized.

This algorithm takes as input *mode_int*, *dz_min*, *dz_max*, *dz_own_ave*, *C_bad_transition*, *sense_own*, *ra_is_maintain*, *ra_is_strengthening*, and *s_c*. This algorithm returns *cost*.

Algorithm 176 BadMaintainTransitionCost

```

1 function BadMaintainTransitionCost( mode_int::Z, dz_min::R, dz_max::R, dz_own_ave::R, C_bad_transition::R,
2                               sense_own::Symbol, ra_is_maintain::Bool, ra_is_strengthening::Bool,
3                               s_c::BadTransitionCState(p.E-42) )
4   const R_corrective::R = params().actions.corrective_rate
5   const R_strengthen::R = params().actions.strengthen_rate
6   const C_bad_maintain_initiation::R =
7     params().modes[mode_int].cost_estimation.online.bad_transition.C_bad_maintain_initiation
8   cost::R = 0.0
9   if s_c.ra_is_maintain_prev && !ra_is_maintain
10    if (s_c.sense_own_prev == :Up) &&
11      ( ((dz_min == R_corrective) && (dz_max == Inf)) ||
12        ((dz_min == -Inf) && (dz_max == -R_strengthen)) )
13      cost = C_bad_transition
14    elseif (s_c.sense_own_prev == :Down) &&
15      ( ((dz_min == -Inf) && (dz_max == -R_corrective)) ||
16        ((dz_min == R_strengthen) && (dz_max == Inf)) )
17      cost = C_bad_transition
18    elseif (s_c.sense_own_prev == :Up) && !ra_is_strengthening &&
19      ((dz_min == R_strengthen) && (dz_max == Inf))
20      cost = C_bad_transition
21    elseif (s_c.sense_own_prev == :Down) && !ra_is_strengthening &&
22      ((dz_min == -Inf) && (dz_max == -R_strengthen))
23      cost = C_bad_transition
24    end
25  elseif !s_c.ra_is_maintain_prev && ra_is_maintain
26    if (sense_own == :Up) &&
27      ( ((s_c.dz_min_prev == R_corrective) && (s_c.dz_max_prev == Inf)) ||
28        ((s_c.dz_min_prev == R_strengthen) && (s_c.dz_max_prev == Inf)) )
29      cost = C_bad_transition
30    elseif (sense_own == :Down) &&
31      ( ((s_c.dz_min_prev == -Inf) && (s_c.dz_max_prev == -R_corrective)) ||
32        ((s_c.dz_min_prev == -Inf) && (s_c.dz_max_prev == -R_strengthen)) )
33      cost = C_bad_transition
34    end
35    if (abs( dz_own_ave ) < R_corrective)
36      cost = cost + C_bad_maintain_initiation
37    end
38  end
39  return cost::R
40 end

```

Referenced In: BadTransitionCost(p.198)
--

3.2.4.5 Critical Interval Protection Cost

CRITICAL INTERVAL PROTECTION COST (Algorithm 177) is used to force alerts and eventually corrective actions in the particularly dangerous situation of extremely slow closure encounters. Slow closures are notoriously difficult as the estimation of tau becomes highly sensitive to the estimated closure rate. Additionally, they suffer from the unique characteristic of potentially being in horizontal conflict for extended periods of time. In most cases, other methods in the logic can provide sufficient robustness, but a few unique instances are covered by this method. To ensure safety, this cost uses the raw horizontal belief states, as opposed to tau, to identify cases where the horizontal separation and horizontal closure rate are sufficiently low to require action be taken.

Using the boolean flags in the cost state (*s_c*), alerts are forced at the prescribed levels. If *force_-*

alert is TRUE, transitions to Clear of Conflict are penalized by *C_force_alert*. If *force_climbdescend* is TRUE, transitions to RAs with a target rate below 1500 fpm are penalized by *C_force_alert*. If *force_inc_climbdescend* is TRUE, transitions to RAs with a target rate below 2500 fpm are penalized by *C_force_alert*. Furthermore, if either *force_climbdescend* or *force_inc_climbdescend* is TRUE, all convergent RAs will be penalized so long as the relative altitude between intruder and ownship is greater than *CROSSING_ALTITUDE_BUFFER*.

Once per cycle, the state variables in *s_c* are set using the helper algorithm **UPDATECRITICALINTERVALPROTECTIONCSTATE**.

This algorithm takes as input *mode_int*, *dz_min*, *dz_max*, *dz_min_prev*, *dz_max_prev*, *z_own_ave*, *dz_own_ave*, *z_int_ave*, *dz_int_ave*, *update* and *s_c*. This algorithm returns *cost*.

Algorithm 177 CriticalIntervalProtectionCost

```

1 function CriticalIntervalProtectionCost( mode_int::Z, dz_min::R, dz_max::R,
2                                     dz_min_prev::R, dz_max_prev::R,
3                                     z_own_ave::R, z_int_ave::R,
4                                     dz_own_ave::R, dz_int_ave::R,
5                                     update::Bool, s_c::CriticalIntervalProtectionCState(p. E-43) )
6   global CROSSING_ALTITUDE_BUFFER
7   const R_corrective::R = params().actions.corrective_rate
8   const R_strengthen::R = params().actions.strengthen_rate
9   const C_force_alert::R =
10     params().modes[mode_int].cost_estimation.online.critical_interval_protection.C_force_alert
11   if update
12     UpdateCriticalIntervalProtectionCState(p. 203)( mode_int, dz_min_prev, dz_max_prev,
13                                         z_own_ave, z_int_ave, dz_own_ave, dz_int_ave,
14                                         s_c )
15   end
16   cost::R = 0.0
17   z_rel::R = z_int_ave - z_own_ave
18   if s_c.force_inc_climbdescend
19     if (CROSSING_ALTITUDE_BUFFER < z_rel) && (-R_strengthen < dz_max)
20       cost = C_force_alert
21     elseif (z_rel < -CROSSING_ALTITUDE_BUFFER) && (dz_min < R_strengthen)
22       cost = C_force_alert
23     end
24   elseif s_c.force_climbdescend
25     if (CROSSING_ALTITUDE_BUFFER < z_rel) && (-R_corrective < dz_max)
26       cost = C_force_alert
27     elseif (z_rel < -CROSSING_ALTITUDE_BUFFER) && (dz_min < R_corrective)
28       cost = C_force_alert
29     end
30   elseif s_c.force_alert && IsCOC(p. 332)( dz_min, dz_max )
31     cost = C_force_alert
32   end
33   return cost::R
34 end
```

Referenced In: OnlineUncoordinatedActionCostEstimation(p. 193)

UPDATECRITICALINTERVALPROTECTIONCSTATE (Algorithm 178) is used to update the state variables of **CRITICALINTERVALPROTECTIONCOST** once per cycle.

The three boolean flags, *force_alert*, *force_climbdescend*, and *force_inc_climbdescend*, are set to TRUE when an alert, a climb or descend alert, or an increase rate alert is needed, respectively. These flags are initialized to false, meaning no alerts will be forced. These conditions limit the effect of this cost only to those cases where it is needed to provide safety.

Checks are performed to determine the strength of alert that is required. These involve projecting the vertical states of the own aircraft and intruder forward in time to determine the expected separation. A different time horizon is used for each alert strength. General alerts have a time horizon of $T_{proximate}$, climb and descend alerts have a time horizon of $T_{proximate_climbdescend}$, and increase rate alerts have a time horizon of $T_{proximate_inc_climbdescend}$. If the expected separation of a given alert strength falls below the required level, that type of alert will be forced. For the $force_climbdescend$ and $force_inc_climbdescend$ the RAs are required to be non-crossing. An exception to this general procedure is the case where the current vertical separation is below the crossing altitude buffer. In this case, a crossing alert is undefined so only $force_alert$ is set to TRUE while $force_climbdescend$ and $force_inc_climbdescend$ are set to FALSE.

A final check using CRITICALINTERVALREQUIRESVERTICALDIVERGENCE ensures only divergent RAs are issued in specific circumstances.

This algorithm takes as input $mode_int$, dz_min_prev , dz_max_prev , z_own_ave , dz_own_ave , z_int_ave , dz_int_ave , and s_c . This algorithm updates $s_c::CriticalIntervalProtectionCState$.

Algorithm 178 UpdateCriticalIntervalProtectionCState

```

1 function UpdateCriticalIntervalProtectionCState( mode_int::Z, dz_min_prev::R, dz_max_prev::R,
2                                     z_own_ave::R, z_int_ave::R,
3                                     dz_own_ave::R, dz_int_ave::R,
4                                     s_c::CriticalIntervalProtectionCState(p. E-43) )
5   global CROSSING_ALTITUDE_BUFFER
6   const R_corrective::R = params().actions.corrective_rate
7   const R_strengthen::R = params().actions.strengthen_rate
8   const T_proximate::R =
9     params().modes[mode_int].cost_estimation.online.critical_interval_protection.T_proximate
10  const T_proximate_climbdescend::R =
11    params().modes[mode_int].cost_estimation.online.critical_interval_protection.T_proximate_climbdescend
12  const T_proximate_inc_climbdescend::R =
13    params().modes[mode_int].cost_estimation.online.critical_interval_protection.T_proximate_inc-
14      climbdescend
14  const H_rel_required::R =
15    params().modes[mode_int].cost_estimation.online.critical_interval_protection.H_rel_required
16  s_c.force_inc_climbdescend = false
17  s_c.force_climbdescend = false
18  s_c.force_alert = false
19  z_rel::R = z_int_ave - z_own_ave
20  coc_prev::Bool = IsCOC(p.332)(dz_min_prev, dz_max_prev);
21  sense_own_prev::Symbol = RatesToSense(p.342)(dz_min_prev, dz_max_prev)
22  if IsOutsideCriticalInterval(p.204)(mode_int, coc_prev, dz_own_ave, dz_int_ave, s_c)
23    s_c.force_alert_prev = false
24  else
25    dz_rel::R = dz_int_ave - dz_own_ave
26    z_rel_proximate::R = z_rel + (dz_rel * T_proximate)
27    z_rel_proximate_climbdescend::R = z_rel + (dz_rel * T_proximate_climbdescend)
28    z_rel_proximate_inc_climbdescend::R = z_rel + (dz_rel * T_proximate_inc_climbdescend)
29    if s_c.force_alert_prev ||
30      (abs(z_rel) < H_rel_required) ||
31      (abs(z_rel_proximate) < H_rel_required) ||
32      IsProjectedCrossing(p.337)(z_rel, z_rel_proximate)
33      s_c.force_alert = true
34    end
35    s_c.force_alert_prev = s_c.force_alert
36    if s_c.force_alert && (R_corrective < abs(dz_int_ave)) &&
37      ((abs(z_rel_proximate_climbdescend) < H_rel_required) ||
38      IsProjectedCrossing(p.337)(z_rel, z_rel_proximate_climbdescend))
39      s_c.force_climbdescend = true
40    end
41    if s_c.force_alert && (R_strengthen < abs(dz_int_ave)) &&
42      ((abs(z_rel_proximate_inc_climbdescend) < H_rel_required) ||
43      IsProjectedCrossing(p.337)(z_rel, z_rel_proximate_inc_climbdescend)) &&
44      ((dz_min_prev < R_strengthen) && (R_corrective <= dz_min_prev) &&
45      (z_rel < -CROSSING_ALTITUDE_BUFFER)) ||
46      ((-R_strengthen < dz_max_prev) && (dz_max_prev <= -R_corrective) &&
47      (CROSSING_ALTITUDE_BUFFER < z_rel))
48      s_c.force_inc_climbdescend = true
49    end
50  end
51  if CriticalIntervalRequiresVerticalDivergence(p.205)(mode_int, coc_prev, sense_own_prev, z_rel, s_c)
52    s_c.force_alert_prev = true
53    s_c.force_alert = true
54    s_c.force_climbdescend = true
55  end
56 end
```

Referenced In: CriticalIntervalProtectionCost(p.201)

ISOUTSIDECRITICALINTERVAL (Algorithm 179) uses geometric and RA information to determine if the own aircraft is outside the critical interval. The own aircraft is outside the critical interval when any of the following conditions are true:

- The horizontal closure speed (magnitude) is above a threshold ($R_speed_ground_threshold$)
- The horizontal range is above a larger threshold ($D_range_ground_expanded$)
- The horizontal range is above a smaller threshold ($D_range_ground_required$) and the aircraft are diverging horizontally
- The intruder vertical rate is below a threshold ($R_int_vert_required$)
- The ownship vertical rate is below a threshold ($R_own_vert_required$)
- The own advisory is currently clear of conflict and the horizontal range is above the smaller threshold ($D_range_ground_required$)

If none of these conditions are true, the own aircraft is considered to be inside the critical interval.

This algorithm takes as input $mode_int$, coc_prev , dz_own_ave , dz_int_ave , and s_c . This algorithm returns $is_outside_ci$.

Algorithm 179 IsOutsideCriticalInterval

```

1 function IsOutsideCriticalInterval( mode_int::Z, coc_prev::Bool, dz_own_ave::R, dz_int_ave::R,
2                                         s_c::CriticalIntervalProtectionCState(p_E-43) )
3     const D_range_ground_required::R =
4         params().modes[mode_int].cost_estimation.online.critical_interval_protection.D_range_ground_required
5     const D_range_ground_expanded::R =
6         params().modes[mode_int].cost_estimation.online.critical_interval_protection.D_range_ground_expanded
7     const R_speed_ground_threshold::R =
8         params().modes[mode_int].cost_estimation.online.critical_interval_protection.R_speed_ground_threshold
9     const R_int_vert_threshold::R =
10        params().modes[mode_int].cost_estimation.online.critical_interval_protection.R_int_vert_threshold
11    const R_own_vert_threshold::R =
12        params().modes[mode_int].cost_estimation.online.critical_interval_protection.R_own_vert_threshold
13    is_outside_ci::Bool = false
14    if (R_speed_ground_threshold < s_c.speed) ||
15        (D_range_ground_expanded <= s_c.range) ||
16        ((D_range_ground_required <= s_c.range) && (s_c.angle < pi/2)) ||
17        (coc_prev && (D_range_ground_required <= s_c.range)) ||
18        (abs( dz_int_ave ) < R_int_vert_threshold) ||
19        (abs( dz_own_ave ) < R_own_vert_threshold)
20        is_outside_ci = true
21    end
22    return is_outside_ci::Bool
23 end
```

Referenced In: UpdateCriticalIntervalProtectionCState(p_203)

 CRITICALINTERVALREQUIRESVERTICALDIVERGENCE (Algorithm 180) uses geometric and RA information to determine if vertical divergence is required in the critical interval. The own aircraft is required to issue divergent alerts if all of the following conditions are true:

- The own aircraft has previously alerted or will be forced to alert on this cycle
- The horizontal range is below or equal to a threshold ($D_range_ground_required$)
- The horizontal closure speed is below a threshold ($R_speed_ground_force_diverge_threshold$)
- Either
 - The intruder is above the own aircraft and the previous RA sense is up
 - The intruder is below the own aircraft and the previous RA sense is down

If these conditions are met, the own aircraft is required to issue a divergent alert.

This algorithm takes as input *mode_int*, *coc_prev*, *sense_own_prev*, *z_rel*, and *s_c*. This algorithm returns *requires_diverge*.

Algorithm 180 CriticalIntervalRequiresVerticalDivergence

```

1 function CriticalIntervalRequiresVerticalDivergence( mode_int::Z, coc_prev::Bool, sense_own_prev::Symbol,
2           z_rel::R, s_c::CriticalIntervalProtectionCState(p.E-43) )
3   const D_range_ground_required::R =
4     params().modes[mode_int].cost_estimation.online.critical_interval_protection.D_range_ground_required
5   const R_speed_ground_force_diverge_threshold::R =
6     params().modes[mode_int].cost_estimation.online.critical_interval_protection.R_speed_ground_force_-
      diverge_threshold
7   requires_diverge::Bool = false
8   if (!coc_prev || s_c.force_alert) &&
9     (s_c.range <= D_range_ground_required) &&
10    (s_c.speed < R_speed_ground_force_diverge_threshold) &&
11    (((0 < z_rel) && (sense_own_prev == :Up)) ||
12     ((z_rel < 0) && (sense_own_prev == :Down)))
13   requires_diverge = true
14 end
15 return requires_diverge::Bool
16 end
```

Referenced In: UpdateCriticalIntervalProtectionCState(p.203)

3.2.4.6 Initialization Cost

INITIALIZATIONCOST (Algorithm 181) prohibits resolution advisories from being issued for the first *T_init* processing cycles. This cost allows the tracker estimates to stabilize. Initialized to 0, the variable *s_c.t_count* counts up until reaching *T_init*, after which resolution advisories are no longer penalized.

This algorithm takes as input *mode_int*, *dz_min*, *dz_max*, *update* and *s_c*. This algorithm returns *cost*.

Algorithm 181 InitializationCost

```

1 function InitializationCost( mode_int::Z, dz_min::R, dz_max::R, update::Bool, s_c::InitializationCState(p.E-44) )
2   const T_init::Z = params().modes[mode_int].cost_estimation.online.initialization.T_init
3   if update
4     s_c.t_count = min( T_init, s_c.t_count + 1 )
5   end
6   cost::R = 0.0
7   if (s_c.t_count < T_init) && !IsCOC(p.332)( dz_min, dz_max )
8     cost = Inf
9   end
10  return cost::R
11 end
```

Referenced In: OnlineUncoordinatedActionCostEstimation(p.193)
--

3.2.4.7 Prevent Early COC Cost

PREVENTEARLYCOCCOST (Algorithm 182) penalizes issuance of a clear of conflict by *C_coc* when there is an active resolution advisory and the geometry of the encounter necessitates the con-

tinuation of the alert. If the early clear of conflict flag is TRUE ($s_c.is_early_coc$) and the proposed action is clear of conflict, the cost is applied.

Once per cycle, the state variables in s_c are set using the helper algorithm UPDATEPREVENTEARLYCOCCSTATE.

This algorithm takes as input $mode_int$, dz_min , dz_max , dz_min_prev , dz_max_prev , z_own_ave , dz_own_ave , z_int_ave , dz_int_ave , $tau_expected$, $update$ and s_c . This algorithm returns $cost$.

Algorithm 182 PreventEarlyCOCCCost

```

1 function PreventEarlyCOCCCost( mode_int::Z, dz_min::R, dz_max::R,
2                               dz_min_prev::R, dz_max_prev::R, z_own_ave::R, dz_own_ave::R,
3                               z_int_ave::R, dz_int_ave::R, tau_expected::R, update::Bool,
4                               s_c::PreventEarlyCOCCState(p.E-45) )
5   const C_coc::R = params().modes[mode_int].cost_estimation.online.prevent_early_coc.C_coc
6   if update
7     UpdatePreventEarlyCOCCState(p.207)( mode_int, dz_min_prev, dz_max_prev, z_own_ave, dz_own_ave,
8                                         z_int_ave, dz_int_ave, tau_expected, s_c )
9   end
10  cost::R = 0.0
11  if s_c.is_early_coc && IsCOC(p.332)( dz_min, dz_max )
12    cost = C_coc
13  end
14  return cost::R
15 end
```

Referenced In: OnlineUncoordinatedActionCostEstimation(p. 193)

UPDATEPREVENTEARLYCOCCSTATE (Algorithm 183) is used to update the state variables of PREVENTEARLYCOCCCost once per cycle. The ultimate purpose of this algorithm is to set the early clear of conflict flag ($s_c.is_early_coc$) based on geometric considerations. This flag is initialized to FALSE.

If the action selected on the previous cycle was clear of conflict, the state variables are reset. If the range on the previous time step is undefined or greater than the current range there is no horizontal divergence and the early clear of conflict flag is TRUE. Otherwise, the range divergence time count ($s_c.t_consec_range_divergence$) is incremented, and can be checked against a threshold ($T_min_horizontal_divergence$). If the aircraft have diverged in horizontal range for less than the threshold, the early clear of conflict flag is TRUE. Likewise, if the horizontal range is not greater than a threshold (D_range_min), the early clear of conflict flag is TRUE.

While sufficient for most cases, the horizontal checks can lead to continued RAs even if the vertical separation has become excessive so additional checks are required. If the horizontal checks determine the early clear of conflict flag should be TRUE, a series of vertical tests are used to determine if clear of conflict can be issued to prevent unnecessary long duration RAs. This creates three layers that allow clear of conflict to be issued due to vertical geometry considerations even if the standard horizontal checks are not passed. These vertical tests rely on relative vertical separation (z_rel) and relative vertical convergence/divergence ($vertical_tau$). The resulting effect is that either greater vertical separation or greater vertical divergence is required if the horizontal checks are not passed. Eventually if sufficient vertical separation is achieved and the aircraft are not converging vertically, RAs will be allowed to clear regardless of horizontal geometry.

This algorithm takes as input $mode_int$, dz_min_prev , dz_max_prev , z_own_ave , dz_own_ave , z_int_ave ,

int_ave, dz_int_ave, tau_expected, and s_c. This algorithm updates *s_c::PreventEarlyCOCCState*.

Algorithm 183 UpdatePreventEarlyCOCCState

```

1 function UpdatePreventEarlyCOCCState( mode_int::Z, dz_min_prev::R, dz_max_prev::R,
2                                     z_own_ave::R, dz_own_ave::R, z_int_ave::R, dz_int_ave::R,
3                                     tau_expected::R, s_c::PreventEarlyCOCCState(p_E-45) )
4   const T_threshold::R =
5     params().modes[mode_int].cost_estimation.online.prevent_early_coc.T_threshold
6   const T_min_horizontal_divergence::Z =
7     params().modes[mode_int].cost_estimation.online.prevent_early_coc.T_min_horizontal_divergence
8   const D_range_min::R =
9     params().modes[mode_int].cost_estimation.online.prevent_early_coc.D_range_min
10  const R_rel_vert_threshold::R =
11    params().modes[mode_int].cost_estimation.online.prevent_early_coc.R_rel_vert_threshold
12  const H_rel_hi_threshold::R =
13    params().modes[mode_int].cost_estimation.online.prevent_early_coc.H_rel_hi_threshold
14  const H_rel_mid_threshold::R =
15    params().modes[mode_int].cost_estimation.online.prevent_early_coc.H_rel_mid_threshold
16  const H_rel_lo_threshold::R =
17    params().modes[mode_int].cost_estimation.online.prevent_early_coc.H_rel_lo_threshold
18  s_c.is_early_coc = false
19  if IsCOC(p_332)( dz_min_prev, dz_max_prev )
20    s_c.range_prev = Nan;
21    s_c.t_consec_range_divergence = 0;
22  else
23    if isnan(s_c.range_prev) || (s_c.range <= s_c.range_prev)
24      s_c.t_consec_range_divergence = 0
25      s_c.is_early_coc = true
26    else
27      s_c.t_consec_range_divergence = s_c.t_consec_range_divergence + 1
28      if (s_c.t_consec_range_divergence <= T_min_horizontal_divergence)
29        s_c.is_early_coc = true
30      elseif (s_c.range <= D_range_min)
31        s_c.is_early_coc = true
32      end
33    end
34    s_c.range_prev = s_c.range
35  end
36  if (s_c.is_early_coc == true)
37    z_rel::R = z_int_ave - z_own_ave
38    dz_rel::R = dz_int_ave - dz_own_ave
39    vertical_tau::R = 0.0
40    if (R_rel_vert_threshold < abs(dz_rel))
41      vertical_tau = z_rel / dz_rel
42    end
43    is_horizontal_sufficient::Bool = (T_threshold <= tau_expected) ||
44                                         (T_min_horizontal_divergence < s_c.t_consec_range_divergence)
45    if ( (H_rel_hi_threshold <= abs(z_rel)) && (0.0 <= vertical_tau) ) ||
46      ( (H_rel_mid_threshold <= abs(z_rel)) && (0.0 <= vertical_tau) && is_horizontal_sufficient ) ||
47      ( (H_rel_lo_threshold <= abs(z_rel)) && (0.0 < vertical_tau) && is_horizontal_sufficient &&
48        (D_range_min < s_c.range) )
49      s_c.is_early_coc = false
50  end
51 end
52 end
```

Referenced In: PreventEarlyCOCCCost(p_206)
--

3.2.4.8 Safe Crossing Resolution Advisory Deferral Cost

SAFECROSSINGRADEFERRALCOST (Algorithm 184) penalizes the issuance of any resolution advisory when an altitude crossing is expected to be safe until the crossing advisory can be issued.

The cost applied to issuing advisories is stored as a state variable ($s_c.c_deferral$). If the current action is clear of conflict and the proposed action is not clear of conflict, the cost is applied.

Once per cycle, the state variables in s_c are set using the helper algorithm UPDATESAFECROSSINGRADEFERRALCSTATE.

This algorithm takes as input $mode_int$, dz_min , dz_max , $equip_int$, z_own_ave , z_int_ave , dz_own_ave , dz_int_ave , dz_min_prev , dz_max_prev , $tau_expected_no_horizon$, $update$ and s_c . This algorithm returns $cost$.

Algorithm 184 SafeCrossingRADeferralCost

```

1 function SafeCrossingRADeferralCost( mode_int::Z, dz_min::R, dz_max::R, equip_int::Bool, z_own_ave::R,
2                                     z_int_ave::R, dz_own_ave::R, dz_int_ave::R, dz_min_prev::R,
3                                     dz_max_prev::R, tau_expected_no_horizon::R, update::Bool,
4                                     s_c::SafeCrossingRADeferralCState(p. E-46) )
5   if update
6     UpdateSafeCrossingRADeferralCState(p. 209)( mode_int, equip_int, z_own_ave, z_int_ave,
7                                               dz_own_ave, dz_int_ave, tau_expected_no_horizon, s_c )
8   end
9   cost::R = 0.0
10  if !IsCOC(dz_min, dz_max) && IsCOC(p. 332)(dz_min_prev, dz_max_prev)
11    cost = s_c.c_deferral
12  end
13  return (cost::R)
14 end
```

Referenced In: OnlineUncoordinatedActionCostEstimation(p. 193)

UPDATESAFECROSSINGRADEFERRALCSTATE (Algorithm 185) is used to update the state variables of SAFECROSSINGRADEFERRALCOST once per cycle. The ultimate purpose of this algorithm is to set deferral cost state variable ($s_c.c_deferral$) based on geometric and equipage considerations.

If the intruder is equipped no deferral cost is applied.

A linear projection of the vertical states of own aircraft and intruder is performed to determine the vertical separation expected at the closest point of approach ($z_cpa_expected$). Additionally, thresholds for the current vertical separation ($s_deferral_threshold$) and the separation at the closest point of approach ($cpa_separation$) are calculated using the ownship vertical rate. As the own aircraft vertical rate increases, the separation thresholds increase. Using these calculated values the following conditions are checked:

- If the vertical separation indicates an altitude crossing will occur
- If the vertical separation at closest point of approach is projected to be larger than the $cpa_separation$ threshold
- If the current vertical separation is currently larger than the $s_deferral_threshold$ value
- If the vertical rate of the intruder aircraft is above the $R_int_vertical_threshold$ value

If all of these conditions are true, the deferral cost state variable is set to $C_crossing_deferral$ for use in SAFECROSSINGRADEFERRALCOST.

This algorithm takes as input $mode_int$, $equip_int$, z_own_ave , z_int_ave , dz_own_ave , dz_int_ave , $tau_expected_no_horizon$, and s_c . This algorithm updates $s_c::SafeCrossingRADeferralCState$.

Algorithm 185 UpdateSafeCrossingRADeferralCState

```

1 function UpdateSafeCrossingRADeferralCState( mode_int::Z, equip_int::Bool, z_own_ave::R, z_int_ave::R,
2           dz_own_ave::R, tau_expected_no_horizon::R, s_c::SafeCrossingRADeferralCState(p. E-46) )
3   const C_crossing_deferral::R =
4     params().modes[mode_int].cost_estimation.online.safe_crossing_ra_deferral.C_crossing_deferral
5   const H_min_threshold::R =
6     params().modes[mode_int].cost_estimation.online.safe_crossing_ra_deferral.H_min_threshold
7   const H_cpa_separation_threshold::R =
8     params().modes[mode_int].cost_estimation.online.safe_crossing_ra_deferral.H_cpa_separation_threshold
9   const H_own_dz_vertical_threshold::R =
10    params().modes[mode_int].cost_estimation.online.safe_crossing_ra_deferral.H_own_dz_vertical-
11      threshold
11  const H_own_dz_cpa_vertical_threshold::R =
12    params().modes[mode_int].cost_estimation.online.safe_crossing_ra_deferral.H_own_dz_cpa_vertical-
13      threshold
13  const R_int_vertical_threshold::R =
14    params().modes[mode_int].cost_estimation.online.safe_crossing_ra_deferral.R_int_vertical_threshold
15  const R_own_threshold::R =
16    params().modes[mode_int].cost_estimation.online.safe_crossing_ra_deferral.R_own_threshold
17  const R_own_rolloff::R =
18    params().modes[mode_int].cost_estimation.online.safe_crossing_ra_deferral.R_own_rolloff
19  if equip_int
20    s_c.c_deferral = 0.0
21  else
22    z_rel::R           = z_int_ave - z_own_ave
23    dz_rel::R          = dz_int_ave - dz_own_ave
24    z_cpa_expected::R = z_rel + (dz_rel * tau_expected_no_horizon)
25    diff_sign_rates::Bool = ((dz_own_ave * dz_int_ave) < 0)
26    s_deferral_threshold::R = H_min_threshold
27    cpa_separation::R = H_cpa_separation_threshold
28    if diff_sign_rates
29      dz_own_scale_factor::R = CalculateThresholdRampUpFactor(p. 326)( abs( dz_own_ave ), R_own_threshold,
30                           R_own_rolloff )
30      s_deferral_threshold = s_deferral_threshold + (H_own_dz_vertical_threshold *
31                           dz_own_scale_factor)
31      cpa_separation       = cpa_separation + (H_own_dz_cpa_vertical_threshold * dz_own_scale_factor
32                           )
32    end
33    if IsProjectedCrossing(p. 337)( z_rel, z_cpa_expected ) &&
34      (s_deferral_threshold < abs( z_rel )) &&
35      (cpa_separation < abs( z_cpa_expected )) &&
36      (R_int_vertical_threshold < abs( dz_int_ave ))
37      s_c.c_deferral = C_crossing_deferral
38    else
39      s_c.c_deferral = 0.0
40    end
41  end
42 end

```

Referenced In: SafeCrossingRADeferralCost(p. 208)

3.2.5 Update Intruder Inputs Mid-Cycle

UPDATEINTRUDERINPUTS (Algorithm 186) is used to retrieve updated information from the STM in the middle of a processing cycle. It retrieves the most recent VRC values from the STM only for those intruders already being processed by the TRM. This retrieval happens at the latest time that VRC inputs can be ingested into the TRM for processing. Subsequent threat processing requires information from received resolution advisory coordination messages.

The call back to the STM, UPDATEINTRUDERVRC, updates the *vrc* field for each intruder repre-

sented in the *input_int* vector. UPDATEINTRUDERVRC also returns the *received_vrcs* vector, which is output in the TRMGROUNDMSGDATA and TRMRABROADCASTDATA as the Resolution Advisory Complements (RAC).

Once the TRM receives updated VRC information from the STM, UPDATEINTRUDERINPUTS loops over all intruders to set variables for later use.

In the loop, the algorithm first updates the *vrc_int* value when both the intruder and ownship are capable of producing resolution advisories, as indicated by *equip_int* having a value of TRUE. Otherwise, the value of *vrc_int* is always 0.

Next, it sets *vrcs_conflict*. If the intruder aircraft has sent a VRC, then the two aircraft are coordinating their advisories. In that case, the VRC from the intruder (*intruder_input.vrc*) is examined to determine if it conflicts with VRCs received from other intruders. The *vrcs_conflict* value is set to indicate the result. Note that *received_vrcs* is used for RAC output only and is not used to determine whether intruder VRCs conflict. That is because *received_vrcs* may reflect VRCs received from intruders that became known to the STM after the start of the TRM cycle; such intruders aren't reflected in the TRM input. *vrcs_conflict* is used in ONLINECOSTESTIMATION for COMPATIBILITYCOST.

Finally, the variable *st_int.is_dna_coordination* is set to indicate whether own aircraft should send VRCs to the intruder designated to Designated No Alerts (DNA) Xo mode (*st_int.no_alerts*). To be able to send VRCs to an intruder designated to DNA Xo mode, own aircraft must have received a VRC from the intruder (*sense_vrc* is not *:None*), the intruder must be equipped with TCAS II (*active_cas_version* is *ACTIVE_CAS_TCAS*), the intruder must be a slave with respect to coordination (*not master_int*), and own aircraft must be able to produce RAs on that intruder (*not exclude_int*). When all those conditions are met, *st_int.is_dna_coordination* is set to TRUE. The STM will identify an intruder with an active CAS as being TCAS II unless the intruder communicates information to explicitly indicate otherwise. Allowing coordination in the case of a TCAS-equipped slave intruder, mitigates the hazard of a TCAS slave being unable to perform a geometric reversal. An ACAS X slave is able to perform geometric reversals and, thus, mitigation is not required for an ACAS X slave. There is no hazard if the mitigation is applied to the ACAS X slave, but it will have more flexibility selecting resolution advisories if the mitigation is not applied.

The values of *equip_int* and *master_int* are unchanged by this algorithm, they retain the values set in INTRUDERPREP.

This algorithm takes as input *input_int*, *equip_int*, *master_int*, *exclude_int*, *st_int*, and *vrc_int*. This algorithm returns *received_vrcs* and *vrcs_conflict*.

Algorithm 186 UpdateIntruderInputs

```

1 function UpdateIntruderInputs( input_int::Vector{TRMIntruderInput(p. E-20)}, equip_int::Vector{Bool},
2                         master_int::Vector{Bool}, exclude_int::Vector{Bool},
3                         st_int::Vector{TRMIntruderState(p. E-33)}, vrc_int::Vector{UInt32} )
4     received_vrcs::Vector{Bool} = UpdateIntruderVRC(p. 89)( input_int )
5     vrcs_conflict::Bool = false
6     sense::Symbol = :None
7     for j in 1:length(input_int)
8         st_int[j].is_dna_coordination = false
9         if equip_int[j]
10            vrc_int[j] = input_int[j].vrc
11            sense_vrc::Symbol = VRCToSense(p. 344)( input_int[j].vrc )
12            if (:None != sense_vrc)
13                if (:None == sense)
14                    sense = sense_vrc
15                elseif (sense_vrc != sense)
16                    vrcs_conflict = true
17                end
18                if (ACTIVE_CAS_TCAS == input_int[j].active_cas_version) &&
19                   st_int[j].no_alerts && !exclude_int[j] && !master_int[j]
20                   st_int[j].is_dna_coordination = true
21                end
22            end
23        else
24            if (EQUIPAGE_CASRA != input_int[j].equipage)
25                vrc_int[j] = EncodeVRC(p. 272)( :None, EQUIPAGE_ATCRBS )
26            end
27        end
28    end
29    return (received_vrcs::Vector{Bool}, vrcs_conflict::Bool)
30 end

```

Referenced In: StateAndCostEstimation(p. 161)

3.2.6 Online Cost Estimation Dependent on Coordination

An overview of the ONLINECOSTESTIMATION component in STATEANDCOSTESTIMATION is shown in Figure 3-5.

ONLINECOSTESTIMATION (Algorithm 187) determines the values of the online costs that are dependent on coordination with intruding aircraft. Online costs capture characteristics of the current state, such as ACAS X coordination, that cannot be suitably reflected in the offline cost tables. The online costs determined in this algorithm are dependent on the resolution advisory coordination state of own aircraft and the intruder aircraft.

ONLINECOSTESTIMATION always performs its evaluation with regard to a single intruder, independent of any other intruders.

It first computes values needed by the various online cost algorithms.

GETMODIFIEDGLOBALRATES (Algorithm 168) determines the vertical rate limits for the previous cycle (*dz_min_prev* and *dz_max_prev*) to be used for computing online costs.

OWNRESPONSEESTIMATION (Algorithm 190) estimates the responsiveness of own aircraft to its own resolution advisories (*resp_own*).

INTRUDERRESPONSEESTIMATION (Algorithm 189) estimates the responsiveness of the intruder to its own advisories, as communicated to own aircraft in resolution advisory coordination messages (*resp_int*). If the intruder is not equipped to send coordination messages, it is treated as being

responsive.

EXPECTEDTAU (Algorithm 276) determines the weighted average value for tau to be used for computing online costs. It is called twice with a slight difference in input arguments. The first time it is called to compute the weighted average value for tau including the horizon (*tau_expected*). This variation is used to determine the applicability of a given functionality in an online cost. The second time it is called to compute the weighted average value for tau excluding the horizon (*tau_expected_no_horizon*). This variation is used when a projection to loss of horizontal separation is required.

See section 3.2.2.1 for a discussion of tau.

The cost is initialized to 0.0 for each action in the resolution advisory cost vectors (*cost_ra*, *cost_ra_subset*, *cost_ra_coord*, and *cost_ra_coord_subset*). These costs are used when determining whether a resolution advisory should be issued, and if so, the type of resolution advisory to issue.

The cost is also initialized to 0.0 for each action in the traffic advisory cost vectors (*cost_ta* and *cost_ta_coord*). These costs are used when determining whether or not to issue a traffic advisory.

The *update* variable is initialized to TRUE. The *update* variable is used to flag the first iteration so each online cost algorithm will know to update the state variables in *s_c*. That set of updates will occur only once for each intruder on each cycle.

ONLINECOSTESTIMATION then loops over all the possible actions to compute the online costs for each action. Each action (*act*) is represented as a minimum and maximum vertical rate (*dz_min* and *dz_max*).

ACTIONTORATES (Algorithm 273) is used to obtain the minimum and maximum vertical rates for the action under evaluation. **ACTIONTORATES** is a frequently used helper algorithm that is intentionally omitted from the flow diagrams.

For each action, **ONLINECOORDINATEDACTIONCOSTESTIMATION** (Algorithm 188) computes a sum of the online costs that are dependent on resolution advisory coordination between own aircraft and the intruding aircraft for a specific action (*cost_ra_coord[act]*, *cost_ra_coord_subset[act]*, and *cost_ta_coord[act]*). *cost_ra_coord[act]* and *cost_ra_coord_subset[act]* are each then summed with *cost_ra_uncoord[act]*, computed by **ONLINEUNCOORDINATEDACTIONCOSTESTIMATION**, to fill the *cost_ra* and *cost_ra_subset* vectors. Similarly, *cost_ta_coord[act]* is summed with *cost_ta_uncoord[act]*, computed by **ONLINEUNCOORDINATEDACTIONCOSTESTIMATION**, to fill the *cost_ta* vector. The *cost_ra*, *cost_ra_subset*, and *cost_ta* vectors are inputs to **INDIVIDUALCOSTESTIMATION**

The algorithm requires knowledge of numerous variables to calculate the various online costs. The required inputs include the previous global resolution advisory issued by the system (*a_global_prev*), the previous resolution advisory for the intruder considered in isolation (*a_indiv_prev*), and the intruder inputs from the STM (*input_int*). The weighted average of the own aircraft and intruder's vertical states (*z_own_ave*, *dz_own_ave*, *z_int_ave*, and *dz_int_ave*) are also needed along with the samples and corresponding weights for tau (*b_tau_int*) and whether the intruder is able to send resolution advisory coordination messages (*equip_int*). Input variables related to exchange of resolution advisory coordination messages are whether the intruder is master (*master_int*), the VRC received from the intruder (*vrc_int*), and whether conflicting VRCs were received from different intruders (*vrcs_conflict*).

The value of *st_int.is_dna_coordination* indicates whether an intruder designation to the Design-

nated No Alerts (DNA) Xo mode is being taken into account for purposes of resolution advisory coordination. The value of this variable is an input to INTRUDERRESPONSEESTIMATION and ON-LINECOORDINATEDACTIONCOSTESTIMATION.

ONLINECOSTESTIMATION takes as input *mode_int*, *h_own*, *z_own_ave*, *dz_own_ave*, *z_int_ave*, *dz_int_ave*, *b_tau_int*, *st_own*, *st_int*, *master_int*, *vrc_int*, *equip_int*, *vrcs_conflict*, *cost_ra_uncoord*, and *cost_ta_uncoord*. Descriptions of the ONLINECOSTESTIMATION output variables are found in Table 3-5.

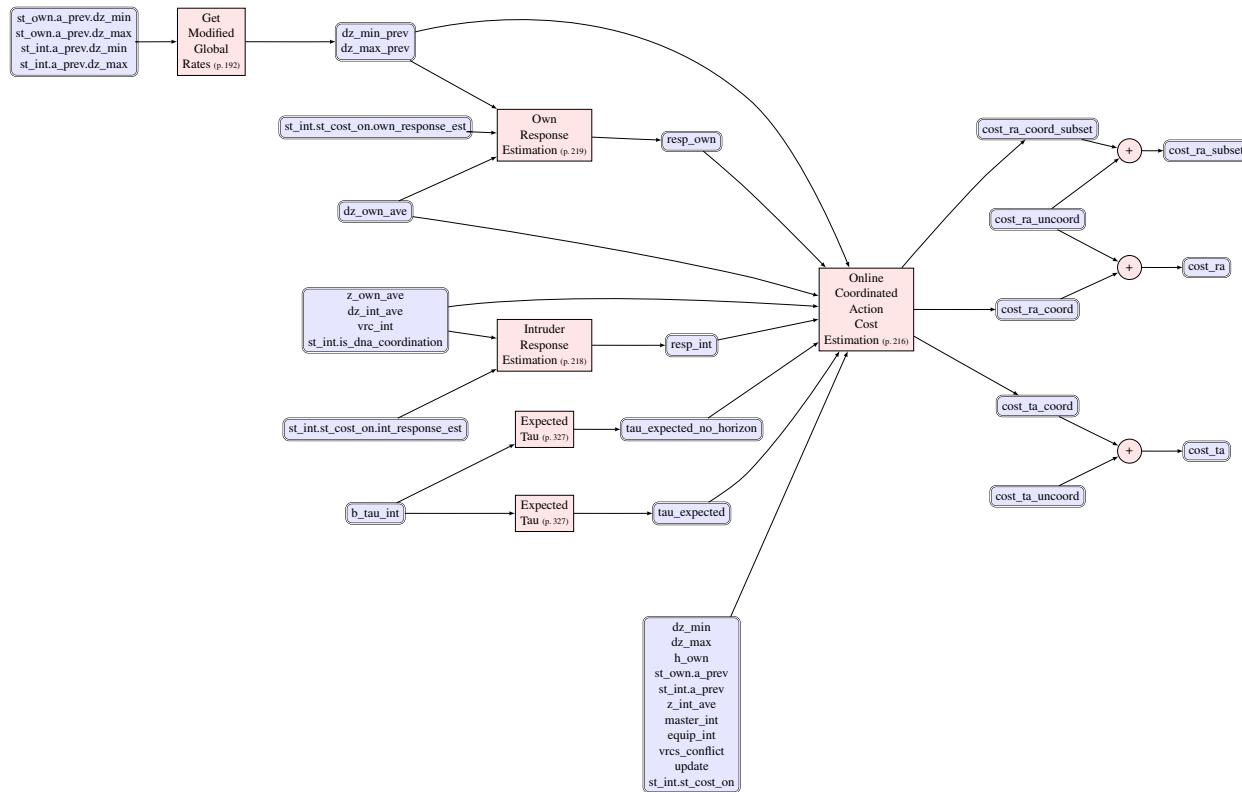


Figure 3-5. OnlineCostEstimation overview.

Table 3-5. TRM Algorithm Output Variables - OnlineCostEstimation

Variable	Units	Type	Description
<i>cost_ra</i>	N/A	real	Vector of combined offline and online cost of each action for single intruder; used for RAs
<i>cost_ra_subset</i>	N/A	real	Vector of combined offline cost and online cost subset of each action for single intruder; used for RAs
<i>cost_ta</i>	N/A	real	Vector of combined offline cost and online cost subset of each action for single intruder; used for TAs
<i>s_c</i>	N/A	OnlineCostState(p. E-37)	State variables for estimating online costs

Algorithm 187 OnlineCostEstimation

```

1 function OnlineCostEstimation( mode_int::Z, h_own::R,
2                               z_own_ave::R, dz_own_ave::R, z_int_ave::R, dz_int_ave::R,
3                               b_tau_int::Vector{TauBelief(p.E-38)},
4                               st_own::TRMOwnState(p.E-32), st_int::TRMIntruderState(p.E-33),
5                               master_int::Bool, vrc_int::UInt32, equip_int::Bool, vrcs_conflict::Bool,
6                               cost_ra_uncoord::Vector{R}, cost_ta_uncoord::Vector{R} )
7   const N_actions::Z = params().actions.num_actions
8   (dz_min_prev::R, dz_max_prev::R) =
9     GetModifiedGlobalRates(p.192)( st_own.a_prev.dz_min, st_own.a_prev.dz_max,
10                                st_int.a_prev.dz_min, st_int.a_prev.dz_max )
11  resp_own::Bool = OwnResponseEstimation(p.219)( mode_int, dz_min_prev, dz_max_prev, dz_own_ave,
12                                                st_int.st_cost_on.own_response_est )
13  resp_int::Bool = IntruderResponseEstimation(p.218)( mode_int, z_own_ave, dz_int_ave, vrc_int,
14                                                    st_int.is_dna_coordination,
15                                                    st_int.st_cost_on.int_response_est )
16  tau_expected::R = ExpectedTau(p.327)( b_tau_int, false )
17  tau_expected_no_horizon::R = ExpectedTau(p.327)( b_tau_int, true )
18  cost_ra::Vector{R} = zeros( R, N_actions )
19  cost_ra_subset::Vector{R} = zeros( R, N_actions )
20  cost_ta::Vector{R} = zeros( R, N_actions )
21  cost_ra_coord::Vector{R} = zeros( R, N_actions )
22  cost_ra_coord_subset::Vector{R} = zeros( R, N_actions )
23  cost_ta_coord::Vector{R} = zeros( R, N_actions )
24  update::Bool = true
25  for act::Z in 1:N_actions
26    (dz_min::R, dz_max::R) =
27      ActionToRates(p.325)( act, dz_own_ave, st_own.a_prev.action, st_own.a_prev.dz_min,
28                            st_own.a_prev.dz_max, st_own.a_prev.ddz )
29    (cost_ra_coord[act], cost_ra_coord_subset[act], cost_ta_coord[act]) =
30      OnlineCoordinatedActionCostEstimation(p.216)( mode_int, dz_min, dz_max, h_own,
31                                                    st_own.a_prev, st_int.a_prev, dz_min_prev, dz_max_prev,
32                                                    z_own_ave, dz_own_ave, z_int_ave, dz_int_ave,
33                                                    resp_own, resp_int, tau_expected, tau_expected_no_horizon,
34                                                    master_int, vrc_int, equip_int, vrcs_conflict,
35                                                    st_int.is_dna_coordination, update, st_int.st_cost_on )
36    update = false
37    cost_ra[act] = cost_ra_uncoord[act] + cost_ra_coord[act]
38    cost_ra_subset[act] = cost_ra_uncoord[act] + cost_ra_coord_subset[act]
39    cost_ta[act] = cost_ta_uncoord[act] + cost_ta_coord[act]
40  end
41  return (cost_ra::Vector{R}, cost_ra_subset::Vector{R}, cost_ta::Vector{R})
42 end

```

Referenced In: StateAndCostEstimation(p.161)

ONLINECOORDINATEDACTIONCOSTESTIMATION (Algorithm 188) returns three online cost sums for the given action and the individual intruder. The online costs determined in this algorithm are dependent on the coordination state of own aircraft and the intruding aircraft. The associated algorithm, ONLINEUNCOORDINATEDACTIONCOSTESTIMATION, determines online costs that are independent of the resolution advisory coordination state of own aircraft and the intruding aircraft.

Two of the cost sums, *cost_ra* and *cost_ra_subset*, are used when determining whether or not to issue a resolution advisory. The value *cost_ra* contains the sum of all the individual coordination-dependent online costs for the given action. The value *cost_ra_subset* contains the sum of the subset of individual coordination-dependent online costs for the given action. An online cost is excluded from *cost_ra_subset* when that cost is designed to force a specific resolution advisory. The costs excluded from the subset are the SA01HEURISTIC (Algorithm 210), the COMPATIBILITYCOST (Algorithm 192), and the CROSSINGNOALERTCOST (Algorithm 198). Each of these costs will force specific advisories to be given and are not included in the *cost_ra_subset*.

The third cost sum, *cost_ta*, contains the sum of the subset of individual online costs for the given action to be used when determining whether or not to issue a traffic advisory. Presently, the only coordination-dependent online cost algorithm contributing to *cost_ta* is LOWALTITUDEPARALLELRADEFERRALCOST. The LOWALTITUDEPARALLELRADEFERRALCOST online cost is used to adjust the offline cost for low altitudes so traffic advisories will be useful precursors of resolution advisories at all altitudes.

Each online cost algorithm called in ONLINECOORDINATEDACTIONCOSTESTIMATION corresponds to a different event to be penalized, such as reversing the previous resolution advisory. After initialization, the algorithms update their own state variables only once during each call to ONLINECOORDINATEDACTIONCOSTESTIMATION, even though ONLINECOORDINATEDACTIONCOSTESTIMATION, and consequently each online cost algorithm, is called once for each action. The *update* value is used to indicate when the individual online cost algorithms should update their state variables, contained in the ONLINECOSTSTATE data structure. The value is set to TRUE when computing the online cost for the first action and FALSE subsequently.

In the individual online cost algorithms, an equipped intruder is an intruding aircraft that is able to send resolution advisory coordination messages. The equipped intruder is indicated by a value of TRUE for *equip_int*.

Common inputs to the individual online cost algorithms include the index to be used for acquiring the appropriate values of the configuration parameters (*mode_int*), whether the intruder aircraft is a master or not (*master_int*), the minimum vertical rate associated with the action (*dz_min*), the maximum vertical rate associated with the action (*dz_max*), *update*, and the relevant set of online cost state variables (*s_c*). *mode_int* is the protection mode index set by GETPROTECTIONMODEINDEX (Algorithm 278) in INTRUDERPREP. ISINTRUDERMASTER (Algorithm 287) is used in INTRUDERPREP to set *master_int*.

The value of *is_dna_int* indicates whether an intruder designation to the Designated No Alerts (DNA) Xo mode is being taken into account for purposes of resolution advisory coordination. The use of this variable is described in COMPATIBILITYCOST, the only one of these algorithms that uses it.

This algorithm takes as input *mode_int*, *dz_min*, *dz_max*, *h_own*, *a_global_prev*, *a_indiv_prev*, *dz_min_prev*, *dz_max_prev*, *z_own_ave*, *dz_own_ave*, *z_int_ave*, *dz_int_ave*, *resp_own*, *resp_int*, *tau_expected*, *tau_expected_no_horizon*, *master_int*, *vrc_int*, *equip_int*, *vrcs_conflict*, *is_dna_int*, *update* and *s_c*. This algorithm returns *cost_ra*, *cost_ra_subset* and *cost_ta*.

Algorithm 188 OnlineCoordinatedActionCostEstimation

```

1 function OnlineCoordinatedActionCostEstimation( mode_int::Z, dz_min::R, dz_max::R, h_own::R,
2                                     a_global_prev::GlobalAdvisory(p.E-35), a_indiv_prev::IndividualAdvisory(p.E-36),
3                                     dz_min_prev::R, dz_max_prev::R,
4                                     z_own_ave::R, dz_own_ave::R, z_int_ave::R, dz_int_ave::R,
5                                     resp_own::Bool, resp_int::Bool,
6                                     tau_expected::R, tau_expected_no_horizon::R,
7                                     master_int::Bool, vrc_int::UInt32, equip_int::Bool, vrcs_conflict::Bool,
8                                     is_dna_int::Bool, update::Bool, s_c::OnlineCostState(p.E-37) )
9   inc_cost::R = 0.0
10  cost_ra::R = 0.0
11  cost_ta::R = 0.0
12  inc_cost = CoordinationDelayCost(p.227)( mode_int, dz_min, dz_max, equip_int, vrc_int,
13                                         update, s_c.coord_delay )
14  cost_ra = cost_ra + inc_cost
15  inc_cost = RestrictCOCDueToReversal(p.238)( mode_int, dz_min, dz_max, vrc_int, master_int,
16                                         dz_min_prev, dz_max_prev, update, s_c.restrict_coc )
17  cost_ra = cost_ra + inc_cost
18  inc_cost = MaxReversalCost(p.235)( mode_int, dz_min, dz_max, vrc_int, master_int,
19                                         a_indiv_prev.dz_min, a_indiv_prev.dz_max, equip_int,
20                                         a_global_prev.multithreat, update, s_c.max_reversal )
21  cost_ra = cost_ra + inc_cost
22  inc_cost = PreventEarlyWeakeningCost(p.237)( mode_int, dz_min, dz_max, dz_min_prev, dz_max_prev, z_own_ave,
23                                         dz_own_ave, z_int_ave, dz_int_ave, vrc_int, tau_expected )
24  cost_ra = cost_ra + inc_cost
25  inc_cost = CoordinatedRADeferralCost(p.225)( mode_int, dz_min, dz_max, dz_min_prev, dz_max_prev,
26                                         z_own_ave, dz_own_ave, z_int_ave, dz_int_ave,
27                                         equip_int, vrc_int,
28                                         update, s_c.coord_ra_deferral )
29  cost_ra = cost_ra + inc_cost
30  inc_cost = LowAltitudeParallelRADeferralCost(p.229)( mode_int, dz_min, dz_max, dz_min_prev, dz_max_prev,
31                                         z_own_ave, h_own, equip_int, tau_expected,
32                                         update, s_c.low_alt_parallel_ra_deferral )
33  cost_ra = cost_ra + inc_cost
34  inc_cost = LowAltitudeParallelRADeferralCost(p.229)( mode_int, dz_min, dz_max, dz_min_prev, dz_max_prev,
35                                         z_own_ave, h_own, equip_int, tau_expected,
36                                         update, s_c.ta_low_alt_parallel_ra_deferral )
37  cost_ta = cost_ta + inc_cost
38  inc_cost = TimeBasedNonComplianceCost(p.242)( mode_int, dz_min, dz_max, dz_min_prev, dz_max_prev,
39                                         z_own_ave, dz_own_ave, z_int_ave, dz_int_ave,
40                                         tau_expected_no_horizon, equip_int, resp_int,
41                                         update, s_c.time_based_non_compliance )
42  cost_ra = cost_ra + inc_cost
43  cost_ra_subset::R = cost_ra
44  inc_cost = SA01Heuristic(p.239)( mode_int, dz_min, dz_max, vrc_int, equip_int, resp_own, master_int,
45                                         (z_int_ave - z_own_ave), dz_own_ave, dz_int_ave, dz_min_prev, dz_max_prev,
46                                         tau_expected_no_horizon, update, s_c.sa01_heuristic )
47  cost_ra = cost_ra + inc_cost
48  inc_cost = CompatibilityCost(p.223)( mode_int, dz_min, dz_max, vrc_int, resp_own, master_int,
49                                         z_own_ave, z_int_ave, a_indiv_prev.dz_min, a_indiv_prev.dz_max,
50                                         a_global_prev.multithreat, resp_int, vrcs_conflict, is_dna_int,
51                                         update, s_c.compatibility_cost )
52  cost_ra = cost_ra + inc_cost
53  inc_cost = CrossingNoAlertCost(p.228)( mode_int, dz_min, dz_max, z_own_ave, z_int_ave, vrc_int,
54                                         tau_expected, update, s_c.crossing_no_alert )
55  cost_ra = cost_ra + inc_cost
56  return (cost_ra::R, cost_ra_subset::R, cost_ta::R)
57 end

```

Referenced In: OnlineCostEstimation(p.214)

3.2.6.1 Response Estimation

Determination of responsiveness to active resolution advisories is implemented by separate algo-

rithms for the intruder aircraft and for own aircraft. Both algorithms output a value (*is_responding*) indicating whether it appears the aircraft is responding to the active resolution advisory or not. The two algorithms differ only in the inputs to DETERMINERESPONSE. After setting the state variables in the response estimation cost state data structure (*s_c*), each algorithm calls DETERMINERESPONSE to get the value of *is_responding*.

INTRUDERRESPONSEESTIMATION (Algorithm 189) is used to estimate the responsiveness of the intruder aircraft to resolution advisories. The received VRC (*vrc_int*) is used to establish the sense of the active resolution advisory. The sense is set to *:None* for an unequipped intruder. The value of *is_dna_int* indicates whether an intruder designation to the Designated No Alerts (DNA) Xo mode is being taken into account for purposes of resolution advisory coordination. When own aircraft is coordinating with an intruder designated to DNA Xo mode, the intruder maximum response time (*t_wait_int*) is set to a very large value. As a result, the intruder is treated as always responsive to its active resolution advisory. Similarly, if the own aircraft is below a given altitude threshold (*h_own_wait_int_threshold*) the time delay in determining if the intruder is responding to advisories is shortened. At lower altitudes the own aircraft advisories can be issued much later due to altitude sensitivity and the determination of intruder response must be performed in a more timely manner. The vertical rate sensitivity for the intruder aircraft is given by *R_buffer_int*.

This algorithm takes as input *mode_int*, *z_own_ave*, *dz_int_ave*, *vrc_int*, *is_dna_int*, and *s_c*. This algorithm returns *is_responding*. This algorithm updates *s_c::ResponseEstimationCState*.

Algorithm 189 IntruderResponseEstimation

```

1 function IntruderResponseEstimation( mode_int::Z, z_own_ave::R, dz_int_ave::R, vrc_int::UInt32,
2                                     is_dna_int::Bool, s_c::ResponseEstimationCState(p.E-45) )
3   const R_buffer_int::R =
4     params().modes[mode_int].cost_estimation.online.response_estimation.R_buffer_int
5   const H_own_wait_int_threshold::R =
6     params().modes[mode_int].cost_estimation.online.response_estimation.H_own_wait_int_threshold
7   t_wait_int::Z = 0
8   if is_dna_int
9     t_wait_int = params().modes[mode_int].cost_estimation.online.response_estimation.resp_dna.T_wait_int
10  elseif (z_own_ave <= H_own_wait_int_threshold)
11    t_wait_int = params().modes[mode_int].cost_estimation.online.response_estimation.T_wait_int_low_alt
12  else
13    t_wait_int = params().modes[mode_int].cost_estimation.online.response_estimation.T_wait_int
14  end
15  sense_int::Symbol = VRCToSense(p.344)( vrc_int )
16  if (sense_int == :None)
17    s_c.t_same_sense      = 0
18    s_c.sense_prev        = :None
19    s_c.dz_after_wait     = NaN
20    s_c.is_responding_prev = true
21    s_c.dz_response_initial = dz_int_ave
22    s_c.dz_response_max   = dz_int_ave
23    s_c.t_dz_response_max = 0
24  elseif (s_c.sense_prev == sense_int)
25    s_c.t_same_sense = s_c.t_same_sense + 1;
26    if (abs( s_c.dz_response_max ) < abs( dz_int_ave ))
27      s_c.dz_response_max = dz_int_ave
28      s_c.t_dz_response_max = s_c.t_same_sense
29  end
30  else
31    s_c.t_same_sense      = 0
32    s_c.dz_response_initial = dz_int_ave
33    s_c.dz_response_max   = dz_int_ave
34    s_c.t_dz_response_max = 0
35  end
36  if (s_c.t_same_sense < t_wait_int)
37    s_c.dz_after_wait = NaN
38  elseif (s_c.t_same_sense == t_wait_int)
39    s_c.dz_after_wait = dz_int_ave
40  end
41  is_responding::Bool = DetermineResponse(p.220)( mode_int, sense_int, dz_int_ave,
42                                               R_buffer_int, t_wait_int, s_c )
43  s_c.is_responding_prev = is_responding
44  s_c.sense_prev         = sense_int
45  return is_responding::Bool
46 end

```

Referenced In: OnlineCostEstimation(p.214)

OWNRESPONSEESTIMATION (Algorithm 190) is used to estimate the responsiveness of own aircraft to resolution advisories. The previous global vertical rate limits, as provided by GETMODIFIEDGLOBALRATES, are used to establish the sense of the active resolution advisory. The own aircraft maximum response time and vertical rate sensitivity are given by T_{wait_own} and R_buffer_own .

This algorithm takes as input $mode_int$, dz_min_prev , dz_max_prev , dz_own_ave , and s_c . This algorithm returns $is_responding$. This algorithm updates $s_c::ResponseEstimationCState$.

Algorithm 190 OwnResponseEstimation

```

1 function OwnResponseEstimation( mode_int::Z, dz_min_prev::R, dz_max_prev::R, dz_own_ave::R,
2                               s_c::ResponseEstimationCState(p_E-45) )
3   const T_wait_own::Z = params().modes[mode_int].cost_estimation.online.response_estimation.T_wait_own
4   const R_buffer_own::R = params().modes[mode_int].cost_estimation.online.response_estimation.R_buffer_own
5   sense_own::Symbol = RatesToSense(p_342)( dz_min_prev, dz_max_prev )
6   if (sense_own == :None)
7     s_c.t_same_sense      = 0
8     s_c.sense_prev        = :None
9     s_c.dz_after_wait    = NaN
10    s_c.is_responding_prev = true
11    s_c.dz_response_initial = dz_own_ave
12    s_c.dz_response_max    = dz_own_ave
13    s_c.t_dz_response_max = 0
14  elseif (s_c.sense_prev == sense_own)
15    s_c.t_same_sense = s_c.t_same_sense + 1
16    if (abs( s_c.dz_response_max ) < abs( dz_own_ave ))
17      s_c.dz_response_max = dz_own_ave
18      s_c.t_dz_response_max = s_c.t_same_sense
19    end
20  else
21    s_c.t_same_sense      = 0
22    s_c.dz_response_initial = dz_own_ave
23    s_c.dz_response_max    = dz_own_ave
24    s_c.t_dz_response_max = 0
25  end
26  if (s_c.t_same_sense < T_wait_own)
27    s_c.dz_after_wait = NaN
28  elseif (s_c.t_same_sense == T_wait_own)
29    s_c.dz_after_wait = dz_own_ave
30  end
31  is_responding::Bool = DetermineResponse(p_220)( mode_int, sense_own, dz_own_ave,
32                                              R_buffer_own, T_wait_own, s_c )
33  s_c.is_responding_prev = is_responding
34  s_c.sense_prev        = sense_own
35  return is_responding::Bool
36 end

```

Referenced In: OnlineCostEstimation(p.214)

DETERMINE_RESPONSE (Algorithm 191) determines whether the aircraft appears to be responding to its active resolution advisory based on the sense of the active resolution advisory, the vertical rate of the aircraft, the vertical rate sensitivity, the maximum response time, and the response estimation state. It always outputs the aircraft is responding when there is no active resolution advisory, as indicated by a sense of *:None*. When the active resolution advisory has an up sense or down sense, responsiveness is determined by estimating the vertical acceleration. The aircraft is considered to be non-responsive if the acceleration is not compliant with the active resolution advisory or the aircraft appears to be leveling off in a direction opposite to the sense of the active resolution advisory.

This algorithm takes as input *mode_int*, *sense*, *dz_curr*, *dz_buffer*, *t_wait*, and *s_c*. This algorithm returns *is_responding*.

Algorithm 191 DetermineResponse

```

1 function DetermineResponse( mode_int::Z, sense::Symbol, dz_curr::R,
2                               dz_buffer::R, t_wait::Z, s_c::ResponseEstimationCState(p.E-45) )
3   const R_acceleration::R =
4     params().modes[mode_int].cost_estimation.online.response_estimation.R_acceleration
5   const R_threshold_acceleration_lo::R =
6     params().modes[mode_int].cost_estimation.online.response_estimation.R_threshold_acceleration_lo
7   const R_threshold_acceleration_hi::R =
8     params().modes[mode_int].cost_estimation.online.response_estimation.R_threshold_acceleration_hi
9   is_responding::Bool = true
10  sense_sign::Z = 1
11  if (sense == :Down)
12    sense_sign = -1
13  end
14  if (sense == :None) || isnan( s_c.dz_after_wait ) || (0 == s_c.t_same_sense)
15    is_responding = true
16  else
17    ddz_response::R      = (s_c.dz_response_initial - dz_curr) / s_c.t_same_sense
18    ddz_response_max::R = 0.0
19    if (s_c.t_same_sense != s_c.t_dz_response_max)
20      ddz_response_max = (s_c.dz_response_max - dz_curr) / (s_c.t_same_sense - s_c.t_dz_response_max)
21    end
22    if (R_threshold_acceleration_lo < abs( ddz_response )) && (0 < (ddz_response * sense_sign)) &&
23      ((abs( ddz_response_max ) < R_threshold_acceleration_hi) ||
24      (0 < (ddz_response_max * sense_sign)))
25      is_responding = false
26    else
27      dz_target::R = s_c.dz_after_wait - (sense_sign * dz_buffer) +
28                    (sense_sign * (s_c.t_same_sense - t_wait) * R_acceleration)
29      if (0 < ((dz_target + (sense_sign * dz_buffer)) * sense_sign))
30        dz_target = -1 * sense_sign * dz_buffer
31      end
32      dz_target_wob::R = s_c.dz_after_wait +
33                        (sense_sign * (s_c.t_same_sense - t_wait) * R_acceleration)
34      if (0 < (dz_target_wob * sense_sign))
35        dz_target_wob = -1 * sense_sign
36      end
37      if (0 < (sense_sign * (dz_curr + (sense_sign * dz_buffer))))
38        is_responding = true
39      elseif s_c.is_responding_prev
40        is_responding = (0 < (sense_sign * (dz_curr - dz_target)))
41      else
42        is_responding = (0 < (sense_sign * (dz_curr - dz_target_wob)))
43      end
44    end
45  end
46  return is_responding::Bool
47 end

```

Referenced In: OwnResponseEstimation(p. 219), IntruderResponseEstimation(p. 218)

3.2.6.2 Compatibility Cost

COMPATIBILITY COST (Algorithm 192) adds a cost each time the resolution advisory has a sense that conflicts with the VRC message of an intruder. It is used, in combination with other online costs, to enforce coordination rules. Small costs allow for flexibility in selecting a resolution advisory but still reduce reversal frequency in equipped encounters. Large costs enforce compatibility. The sense of the resolution advisory under consideration for own aircraft is *sense_own*. The sense of the intruder, as inferred from the VRC message, is *sense_int*. These two senses conflict if they are the same, that is, if own aircraft and the intruder are both climbing or both are descending. If the ownership is a master, several conditions are checked to determine the value of the cost to apply. Ad-

ditionally, these costs only apply to the master if the intruder aircraft is determined to be responding, based on *resp_int*, which is calculated in INTRUDERRESPONSEESTIMATION (Algorithm 189). This check is necessary to ensure the master is unconstrained in the event of a non-responsive intruder and can maintain an appropriate level of safety. Similar to the SA01HEURISTIC (Algorithm 210), which protects the intruder if the ownship pilot chooses not to respond, this check has a significant impact on the safety of vertical chase scenarios where one pilot does not respond. Furthermore if the master ownship is determined to be non responsive when the slave intruder is responsive the cost is also removed to allow unconstrained reversals. If the intruder is a master, the ownship is required to pick a compliant action. Thus, the cost for a conflicting action in this case, *C_master* is large.

There are seven different cost values:

- *C_master* is a large cost, used to prevent a reversal when the intruder is master;
- *C_slave_init* is a large cost, used to enforce coordination rules that prevent the master from reversing the slave intruder when the previous action of the own aircraft was clear of conflict (COC) and there is no potential crossing;
- *C_slave_init_crossing* is a very small cost, used to allow the master to reverse the slave intruder when there is no previous resolution advisory, there is a potential crossing, and the slave VRC was received within 3 seconds, under the coordination rules;
- *C_slave_sub* is a moderate cost, used to constrain the master reversing the slave intruder when the previous action of the own aircraft was neither COC nor a crossing advisory and there was no prior multithreat advisory;
- *C_slave_sub_noncrossing* is a small cost, used to constrain the master reversing the slave intruder when the previous action of the own aircraft was a crossing advisory and there was no prior multithreat advisory;
- *C_slave_sub_no_response* is a null cost, used to remove the constraints on the master reversing the slave intruder when the master is not responding;
- *C_slave_sub_multithreat* is a small cost, used to constrain the master reversing the slave intruder when the previous action was not COC and there was a previous multithreat situation.

Five of these cost values are modified when the input variable *is_dna_int* is TRUE. The value of *is_dna_int* indicates whether an intruder designation to the Designated No Alerts (DNA) Xo mode is being taken into account for purposes of resolution advisory coordination. When own aircraft is coordinating with an intruder designated to DNA Xo mode and that intruder is a slave to own aircraft, compatibility with the slave is encouraged by setting *C_slave_init*, *C_slave_init_crossing*, *C_slave_sub*, *C_slave_sub_noncrossing*, and *C_slave_sub_no_response* to moderately large values. Adjusting the cost values in this way discourages, but doesn't prevent, own aircraft from reversing the slave.

When own aircraft is master and there are multiple intruders from which own aircraft is receiving conflicting VRCs, no cost is applied. Eliminating the cost in that situation allows own aircraft more flexibility in choosing the best resolution advisory.

The *master_int* variable is a boolean representing if the intruder aircraft is the master. It is set via ISINTRUDERMASTER. The *vrcs_conflict* variable is a boolean indicating whether multiple intruders are sending conflicting VRCs to own aircraft. It is set in UPDATEINTRUDERINPUTS.

The state variable *s_c.t_since_first_vrc* is used with *T_reversal_thres* when there is a potential crossing. It is used to determine when to enforce a coordination rule for a slave reversal and apply *C_slave_init*, a large cost, instead of *C_slave_init_crossing*, a small cost. In a similar manner, the

$H_{noncrossing_thres}$ and $T_{noncrossing_thres}$ are used to determine if a subsequent advisory will force a reversal to a non-crossing RA early in the encounter. If this is true, the smaller $C_{slave_sub_noncrossing}$ will be applied as opposed to the larger C_{slave_sub} .

Once per cycle, the state variables in s_c are set using the helper algorithm UPDATECOMPATIBILITYCSTATE.

The COMPATIBILITYCOST is excluded from $cost_ra_subset$, which is used in INDIVIDUALCOST-ESTIMATION to determine whether to penalize an action to force a resolution advisory.

This algorithm takes as input $mode_int$, dz_min , dz_max , vrc_int , $resp_own$, $master_int$, z_own_ave , z_int_ave , dz_min_prev , dz_max_prev , $multithreat_prev$, $resp_int$, $vrcs_conflict$, is_dna_int , $update$ and s_c . This algorithm returns $cost$.

.....

Algorithm 192 CompatibilityCost

```

1 function CompatibilityCost( mode_int::Z, dz_min::R, dz_max::R, vrc_int::UInt32,
2             resp_own::Bool, master_int::Bool, z_own_ave::R,
3             z_int_ave::R, dz_min_prev::R, dz_max_prev::R,
4             multithreat_prev::Bool, resp_int::Bool, vrscs_conflict::Bool,
5             is_dna_int::Bool, update::Bool, s_c::CompatibilityCState(p.E-43) )
6     const C_master::R =
7         params().modes[mode_int].cost_estimation.online.compatibility.C_master
8     const C_slave_sub_multithreat::R =
9         params().modes[mode_int].cost_estimation.online.compatibility.C_slave_sub_multithreat
10    const T_reversal_thres::Z =
11        params().modes[mode_int].cost_estimation.online.compatibility.T_reversal_thres
12    const H_noncrossing_thres::R =
13        params().modes[mode_int].cost_estimation.online.compatibility.H_noncrossing_thres;
14    const T_noncrossing_thres::Z =
15        params().modes[mode_int].cost_estimation.online.compatibility.T_noncrossing_thres;
16    if update
17        UpdateCompatibilityCState(p.224)( mode_int, vrc_int, is_dna_int, s_c )
18    end
19    cost::R = 0.0
20    sense_int::Symbol      = VRCToSense(p.344)( vrc_int )
21    sense_own::Symbol      = RatesToSense(p.342)( dz_min, dz_max )
22    sense_own_prev::Symbol = RatesToSense(p.342)( dz_min_prev, dz_max_prev )
23    if (:None != sense_own) && (sense_own == sense_int)
24        if master_int
25            cost = C_master
26        elseif IsCOC(p.332)( dz_min_prev, dz_max_prev )
27            if vrscs_conflict
28                cost = 0.0
29            elseif (s_c.t_since_first_vrc <= T_reversal_thres) &&
30                IsCrossing(p.333)( z_own_ave, z_int_ave, sense_int )
31                cost = s_c.c_slave_init_crossing
32            else
33                cost = s_c.c_slave_init
34            end
35        elseif resp_int && (sense_own != sense_own_prev)
36            if multithreat_prev
37                cost = C_slave_sub_multithreat
38            elseif !resp_own
39                cost = s_c.c_slave_sub_no_response
40            elseif IsCrossing(p.333)( z_int_ave, z_own_ave, sense_own_prev ) &&
41                !IsCrossing(p.333)( z_int_ave, z_own_ave, sense_own ) &&
42                (H_noncrossing_thres < abs( z_own_ave - z_int_ave )) &&
43                (s_c.t_since_first_vrc < T_noncrossing_thres)
44                cost = s_c.c_slave_sub_noncrossing
45            else
46                cost = s_c.c_slave_sub
47            end
48        end
49    end
50    return cost::R
51 end

```

Referenced In: OnlineCoordinatedActionCostEstimation(p.216)

UPDATECOMPATIBILITYCSTATE (Algorithm 193) is used to update the state variables of COMPATIBILITYCOST once per cycle. This algorithm has two primary goals. The first is to keep a record of how long the intruder has been sending a VRC (*s_c.t_since_first_vrc*). The second is to set the correct parameters in the state variable based on whether the intruder is subject to DNA processing (*is_dna_int*).

This algorithm takes as input *mode_int*, *vrc_int*, *is_dna_int*, and *s_c*. The algorithm outputs updates

to $s_c::CompatibilityCState$.

Algorithm 193 UpdateCompatibilityCState

```

1 function UpdateCompatibilityCState( mode_int::Z, vrc_int::UInt32, is_dna_int::Bool, s_c::  

  CompatibilityCState(p. E-43) )  

2   sense_int::Symbol = VRCToSense(p. 344)( vrc_int )  

3   if (:None == sense_int)  

4     s_c.t_since_first_vrc = 0  

5   else  

6     s_c.t_since_first_vrc = s_c.t_since_first_vrc + 1  

7   end  

8   if is_dna_int  

9     s_c.c_slave_init =  

10    params().modes[mode_int].cost_estimation.online.compatibility.compat_dna.C_slave_init  

11    s_c.c_slave_init_crossing =  

12    params().modes[mode_int].cost_estimation.online.compatibility.compat_dna.C_slave_init_crossing  

13    s_c.c_slave_sub =  

14    params().modes[mode_int].cost_estimation.online.compatibility.compat_dna.C_slave_sub  

15    s_c.c_slave_sub_noncrossing =  

16    params().modes[mode_int].cost_estimation.online.compatibility.compat_dna.C_slave_sub_noncrossing  

17    s_c.c_slave_sub_no_response =  

18    params().modes[mode_int].cost_estimation.online.compatibility.compat_dna.C_slave_sub_no_response  

19   else  

20     s_c.c_slave_init =  

21     params().modes[mode_int].cost_estimation.online.compatibility.C_slave_init  

22     s_c.c_slave_init_crossing =  

23     params().modes[mode_int].cost_estimation.online.compatibility.C_slave_init_crossing  

24     s_c.c_slave_sub =  

25     params().modes[mode_int].cost_estimation.online.compatibility.C_slave_sub  

26     s_c.c_slave_sub_noncrossing =  

27     params().modes[mode_int].cost_estimation.online.compatibility.C_slave_sub_noncrossing  

28     s_c.c_slave_sub_no_response =  

29     params().modes[mode_int].cost_estimation.online.compatibility.C_slave_sub_no_response  

30   end  

31 end
```

Referenced In: CompatibilityCost(p. 223)

3.2.6.3 Coordinated Resolution Advisory Deferral Cost

COORDINATEDRADEFERRALCOST (Algorithm 194) is used to reduce the frequency of both aircraft issuing corrective alerts in a coordinated encounter. As many geometries can be resolved with just one aircraft issuing a corrective alert, this cost suppresses non-preventive alerts on the own aircraft if the intruder has already alerted. If the deferral cost state variable ($s_c.deferral_cost$) is set, the current action is clear of conflict, and the proposed action is not clear of conflict or a preventive advisory, the cost is applied.

Once per cycle, the state variables in s_c are set using the helper algorithm UPDATECOORDINATE-DRADFERRALCSTATE.

This algorithm takes as input $mode_int$, dz_min , dz_max , dz_min_prev , dz_max_prev , z_own_ave , dz_own_ave , z_int_ave , dz_int_ave , $equip_int$, vrc_int , $update$ and s_c . This algorithm returns $cost$.

Algorithm 194 CoordinatedRADeferralCost

```

1 function CoordinatedRADeferralCost( mode_int::Z, dz_min::R, dz_max::R,
2                               dz_min_prev::R, dz_max_prev::R,
3                               z_own_ave::R, dz_own_ave::R,
4                               z_int_ave::R, dz_int_ave::R,
5                               equip_int::Bool, vrc_int::UInt32, update::Bool,
6                               s_c::CoordinatedRADeferralCState(p. E-43) )
7   if update
8     UpdateCoordinatedRADeferralCState(p. 226)( mode_int, z_own_ave, dz_own_ave, z_int_ave, dz_int_ave,
9                                               equip_int, vrc_int, s_c )
10  end
11  cost::R = 0.0
12  if IsCOC(p. 332)( dz_min_prev, dz_max_prev ) &&
13    !IsCOC(p. 332)( dz_min, dz_max ) &&
14    !IsPreventive(p. 336)( dz_min, dz_max )
15    cost = s_c.deferral_cost
16  end
17  return cost::R
18 end

```

Referenced In: OnlineCoordinatedActionCostEstimation(p. 216)

UPDATECOORDINATEDRADEFERRALCSTATE (Algorithm 195) is used to update the state variables of COORDINATEDRADEFERRALCOST once per cycle.

The RA deferral is subject to several requirements to ensure safety can be maintained by the aircraft during the encounter. The first requirement is that the intruder aircraft must be equipped and have an active RA against own aircraft. Furthermore, the RA on the intruder aircraft must be a non-crossing RA, as most crossings should be secured by advisories on both aircraft. If these conditions are met, the vertical states of own aircraft and intruder are projected forward in time $T_{proximate}$ seconds to determine the expected level of separation in the near future. If sufficient separation is expected, then corrective RAs will be suppressed by a moderate cost. If loss of separation or a crossing is expected, then the cost applied to corrective RAs will be reduced to zero, with interpolation when the separation is between.

This algorithm takes as input $mode_int$, z_own_ave , dz_own_ave , z_int_ave , dz_int_ave , $equip_int$, vrc_int , and s_c . The algorithm outputs updates to $s_c::CoordinatedRADeferralCState$.

.....

Algorithm 195 UpdateCoordinatedRADeferralCState

```

1 function UpdateCoordinatedRADeferralCState( mode_int::Z, z_own_ave::R, dz_own_ave::R,
2                                     z_int_ave::R, dz_int_ave::R,
3                                     equip_int::Bool, vrc_int::UInt32,
4                                     s_c::CoordinatedRADeferralCState(p. E-43) )
5   const T_proximate::R =
6     params().modes[mode_int].cost_estimation.online.coord_ra_deferral.T_proximate
7   const H_threshold_proximate::R =
8     params().modes[mode_int].cost_estimation.online.coord_ra_deferral.H_threshold_proximate
9   const H_rolloff_proximate::R =
10    params().modes[mode_int].cost_estimation.online.coord_ra_deferral.H_rolloff_proximate
11   const C_deferral::R =
12     params().modes[mode_int].cost_estimation.online.coord_ra_deferral.C_deferral
13   s_c.deferral_cost = 0.0
14   sense_int::Symbol = VRCToSense(p. 344)( vrc_int )
15   if equip_int && (:None != sense_int)
16     z_rel::R = z_int_ave - z_own_ave
17     z_rel_proximate::R = z_rel + ((dz_int_ave - dz_own_ave) * T_proximate)
18     if IsDiverging(p. 333)( z_own_ave, z_int_ave, sense_int ) &&
19       !IsProjectedCrossing(p. 337)( z_rel, z_rel_proximate )
20       z_rel_proximate = min( abs( z_rel ), abs( z_rel_proximate ) )
21       scale_factor::R = CalculateThresholdRampUpFactor(p. 326)( abs( z_rel_proximate ),
22                                         H_threshold_proximate, H_rolloff_proximate )
22       s_c.deferral_cost = C_deferral * scale_factor
23   end
24 end
25 end

```

Referenced In: CoordinatedRADeferralCost(p. 225)

3.2.6.4 Coordination Delay Cost

COORDINATIONDELAYCOST (Algorithm 196) prohibits resolution advisories (RAs) from being issued immediately for a CAS-equipped intruder that can issue RAs. The delay associated with this cost ($s_c.t_count$) allows time for a coordination message to be received from a CAS-equipped intruder following transition from TA-Only operational mode to TA/RA operational mode on ownership. $s_c.t_count$ is initialized to 0 when the intruder becomes eligible for RA processing. That occurs when the intruder is first received for TRM processing, when it no longer has degraded surveillance, or when ownership transitions from TA-Only operational mode to TA/RA operational mode. If ownership is in TA-Only operational mode, the intruder is unequipped with a CAS, or the intruder is equipped with a CAS but cannot issue RAs, $s_c.t_count$ is set to T_{init} . Otherwise, $s_c.t_count$ is updated based on receipt of a VRC from the intruder. Once $s_c.t_count$ is greater than or equal to T_{init} , resolution advisories are not penalized.

Once per cycle, the state variables in s_c are set using the helper algorithm UPDATECOORDINATIONDELAYCSTATE.

This algorithm takes as input $mode_int$, dz_min , dz_max , $equip_int$, vrc_int , $update$ and s_c . This algorithm returns $cost$.

Algorithm 196 CoordinationDelayCost

```

1 function CoordinationDelayCost( mode_int::Z, dz_min::R, dz_max::R, equip_int::Bool, vrc_int::UInt32,
2                               update::Bool, s_c::CoordinationDelayCState(p.E-43) )
3   const T_init::Z = params().modes[mode_int].cost_estimation.online.coord_delay.T_init
4   if update
5     UpdateCoordinationDelayCState(p.227)( mode_int, equip_int, vrc_int, s_c )
6   end
7   cost::R = 0.0
8   if (s_c.t_count < T_init) && !IsCOC(p.332)( dz_min, dz_max )
9     cost = Inf
10  end
11  return cost::R
12 end

```

Referenced In: OnlineCoordinatedActionCostEstimation(p.216)

UPDATECOORDINATIONDELAYCSTATE (Algorithm 197) is used to update the state variables of COORDINATIONDELAYCOST once per cycle.

Variable *s_c.t_count* is initialized to 0.

When ownership and the intruder can coordinate resolution advisories as indicated by *equip_int*, and *s_c.is_count_enabled* is TRUE, *s_c.t_count* counts up until a VRC is received from the intruder or *T_init* is reached. When a VRC is received from the intruder, the sense of the intruder (*sense_int*) has a value other than :NONE and *s_c.t_count* is set to *T_init*.

If either ownership or the intruder cannot issue resolution advisories, or *s_c.is_count_enabled* is FALSE, *s_c.t_count* is set to *T_init*.

This algorithm takes as input *mode_int*, *equip_int*, *vrc_int*, and *s_c*. The algorithm outputs updates to *s_c::CoordinationDelayCState*.

Algorithm 197 UpdateCoordinationDelayCState

```

1 function UpdateCoordinationDelayCState( mode_int::Z, equip_int::Bool, vrc_int::UInt32,
2                                         s_c::CoordinationDelayCState(p.E-43) )
3   const T_init::Z = params().modes[mode_int].cost_estimation.online.coord_delay.T_init
4   if equip_int && s_c.is_count_enabled
5     sense_int::Symbol = VRCToSense(p.344)( vrc_int )
6     if (:None == sense_int)
7       s_c.t_count = min( T_init, s_c.t_count + 1 )
8     else
9       s_c.t_count = T_init
10    end
11  else
12    s_c.t_count = T_init
13  end
14 end

```

Referenced In: CoordinationDelayCost(p.227)

3.2.6.5 Crossing No Alert Cost

CROSSINGNOALERTCOST (Algorithm 198) forces the aircraft to issue a resolution advisory (RA) when the intruder intends to cross altitudes. This algorithm forces the aircraft to issue an RA by penalizing the clear of conflict (COC) action by *C_coc*. This ensures that when one intruder

.....

initiates a crossing, the ownship will issue an RA, possibly a reversal, subject to coordination rules in COMPATIBILITYCOST. If the crossing flag state variable (*s_c.is_crossing*) is set, the proposed action is clear of conflict, the geometric crossing flag (*s_c.is_crossing_caused_by_geometry*) is not set, and tau is less than a threshold (*T_threshold*), the cost is applied. The value of tau (*tau_expected*) must be within the threshold to prevent inappropriate application of the cost during slow closure encounters. Tau is the time until the intruder comes within a lateral conflict distance of own aircraft. EXPECTEDTAU (Algorithm 276) is used to estimate a value for tau from the distribution. See section 3.2.2.1 for a discussion of tau.

Once per cycle, the state variables in *s_c* are set using the helper algorithm UPDATECROSSING-NOALERTCSTATE.

The CROSSINGNOALERTCOST is excluded from *cost_ra_subset*, which is used in INDIVIDUAL-COSTESTIMATION to determine whether to penalize an action to force a resolution advisory.

This algorithm takes as input *mode_int*, *dz_min*, *dz_max*, *z_own_ave*, *z_int_ave*, *vrc_int*, *tau_expected*, *update* and *s_c*. This algorithm returns *cost*.

Algorithm 198 CrossingNoAlertCost

```

1 function CrossingNoAlertCost( mode_int::Z, dz_min::R, dz_max::R,
2                               z_own_ave::R, z_int_ave::R, vrc_int::UInt32, tau_expected::R,
3                               update::Bool, s_c::CrossingNoAlertCState(p.E-44) )
4   const C_coc::R =
5     params().modes[mode_int].cost_estimation.online.crossing_no_alert.C_coc
6   const T_threshold::R =
7     params().modes[mode_int].cost_estimation.online.crossing_no_alert.T_threshold
8   if update
9     UpdateCrossingNoAlertCState(p.229)( z_own_ave, z_int_ave, vrc_int, s_c )
10  end
11  cost::R = 0.0
12  if s_c.is_crossing &&
13    IsCOC(p.332)( dz_min, dz_max ) &&
14    !s_c.is_crossing_caused_by_geometry &&
15    (tau_expected < T_threshold)
16    cost = C_coc
17  end
18  return cost::R
19 end
```

Referenced In: OnlineCoordinatedActionCostEstimation(p.216)

UPDATECROSSINGNOALERTCSTATE (Algorithm 199) is used to update the state variables of CROSSINGNOALERTCOST once per cycle. The intruder's intent is inferred from the received VRC message and the altitude of each aircraft using ISCROSSING. If the intruder's intent indicates crossing on this cycle, but did not indicate a crossing on the previous cycle, and the VRC did not change, the crossing indication is due to aircraft geometry changing. For example, an intruder above the ownship may issue a climb resolution advisory while descending and, if the response is too slow, it may cross altitudes with the ownship. In this case, the cost is not applied to prevent it from forcing an alert that could potentially reverse the intruder.

This algorithm takes as input *z_own_ave*, *z_int_ave*, *vrc_int*, and *s_c*. The algorithm outputs updates to *s_c::CrossingNoAlertCState*.

Algorithm 199 UpdateCrossingNoAlertCState

```

1 function UpdateCrossingNoAlertCState( z_own_ave::R, z_int_ave::R, vrc_int::UInt32,
2                                     s_c::CrossingNoAlertCState(p.E-44) )
3   sense_int::Symbol = VRCToSense(p.344)( vrc_int )
4   s_c.is_crossing = IsCrossing(p.333)( z_own_ave, z_int_ave, sense_int )
5   if s_c.is_crossing && !s_c.is_crossing_prev && (sense_int != :None) && (vrc_int == s_c.vrc_int_prev)
6     s_c.is_crossing_caused_by_geometry = true
7   else
8     s_c.is_crossing_caused_by_geometry = false
9     s_c.is_crossing_prev = s_c.is_crossing
10  end
11  s_c.vrc_int_prev = vrc_int
12 end
```

Referenced In: CrossingNoAlertCost(p.228)

3.2.6.6 Low Altitude Parallel Resolution Advisory Deferral Cost

LOWALTITUDEPARALLELRADEFERRALCOST (Algorithm 200) is used to penalize an initial RA based on the height of the own aircraft and the raw horizontal states of the two aircraft. This method attempts to identify acceptable parallel geometries that occur near the ground and defer issuing an RA as long as possible. If the own aircraft has not yet alerted, and the proposed action is not-clear of conflict, the costs stored in the state variables will be applied. Preventive and Maintain advisories are penalized by different amounts than corrective advisories.

This algorithm is used twice, once for resolution advisories and once for traffic advisories. The two instances differ in the values in the cost state data structure (*s_c*) and the parameters used for the range, tau, and altitude thresholds.

Once per cycle, the state variables in *s_c* are set for each instance using the helper algorithm UPDATELOWALTITUDEPARALLELRADEFERRALCSTATE.

This algorithm takes as input *mode_int*, *dz_min*, *dz_max*, *dz_min_prev*, *dz_max_prev*, *z_own_ave*, *h_own*, *tau_expected*, *equip_int*, *update* and *s_c*. This algorithm returns *cost*.

Algorithm 200 LowAltitudeParallelRADeferralCost

```

1 function LowAltitudeParallelRADeferralCost( mode_int::Z, dz_min::R, dz_max::R,
2                                              dz_min_prev::R, dz_max_prev::R, z_own_ave::R, h_own::R,
3                                              equip_int::Bool, tau_expected::R, update::Bool,
4                                              s_c::LowAltitudeParallelRADeferralCState(p.E-44) )
5   if update
6     UpdateLowAltitudeParallelRADeferralCState(p.231)( mode_int, z_own_ave, h_own, equip_int, tau_expected, s_c
      )
7   end
8   cost::R = 0.0
9   if !IsCOC( dz_min, dz_max ) && IsCOC(p.332)( dz_min_prev, dz_max_prev )
10    if IsPreventive(p.336)( dz_min, dz_max ) || IsMaintain(p.335)( dz_min, dz_max )
11      cost = s_c.c_non_climb_descend
12    else
13      cost = s_c.c_alert_deferral
14    end
15  end
16  return cost::R
17 end
```

Referenced In: OnlineCoordinatedActionCostEstimation(p.216)

UPDATEROWALTITUDEPARALLELRADEFERRALCSTATE (Algorithm 201) is used to update the state variables of **LOWALTITUDEPARALLELRADEFERRALCOST** once per cycle. This algorithm is used twice, once for resolution advisories and once for traffic advisories. The two instances differ in the values in the cost state data structure (s_c) and the parameters used for the range, tau, and altitude thresholds.

Three mechanisms are used to delay alerts as the aircraft altitude approaches the ground:

- A parallel discriminator to penalize alerts when in parallel geometries;
- A general penalty on non-climb\descend RAs below a given altitude threshold
- General penalties on all alerts below given altitude thresholds

The parallel discriminator identifies cases where the aircraft are closing sufficiently slowly and have sufficient horizontal separation for advisories to be deferred. The acceptable level of horizontal separation is dependent on the closure rate. The own aircraft height above ground (or barometric altitude, if radar altimeter measurements are unavailable) is also used to limit the applicability of the cost. The value must be below the threshold parameters for the cost to be applied. Furthermore, the aircraft must be closing sufficiently slowly and have sufficient horizontal separation for advisories to be deferred. So long as the aircraft are separated horizontally and the estimated closure rate is slow, advisories will be deferred indefinitely.

The general penalty on non-climb\descend RAs applies to preventive and maintain RAs so long as tau is below $T_{threshold_dnx}$. The cost is scaled based on altitude so that as altitude decreases the cost increases. Additionally, the penalty on non-climb\descend RAs is always required to be no less than the parallel discriminator cost.

The general penalties on all alerts below a given altitude threshold are computed using **DETERMINELOWALTITUDECOST**. These costs are added to the penalties on parallel alerts and non-climb\descend RAs.

This algorithm takes as input *mode_int*, *h_own*, *z_own_ave*, *tau_expected*, *equip_int*, and *s_c*. The algorithm outputs updates to *s_c*::*LowAltitudeParallelRADeferralCState*.

Algorithm 201 UpdateLowAltitudeParallelRADeferralCState

```

1 function UpdateLowAltitudeParallelRADeferralCState( mode_int::Z, z_own_ave::R, h_own::R, equip_int::Bool,
2                                     tau_expected::R, s_c::LowAltitudeParallelRADeferralCState(p.E-44) )
3   const H_threshold::R =
4     params().modes[mode_int].cost_estimation.online.low_alt_parallel_ra_deferral.H_threshold
5   const H_rolloff::R =
6     params().modes[mode_int].cost_estimation.online.low_alt_parallel_ra_deferral.H_rolloff
7   const H_threshold_dnx::R =
8     params().modes[mode_int].cost_estimation.online.low_alt_parallel_ra_deferral.H_threshold_dnx
9   const H_rolloff_dnx::R =
10    params().modes[mode_int].cost_estimation.online.low_alt_parallel_ra_deferral.H_rolloff_dnx
11   const T_threshold_dnx::R =
12     params().modes[mode_int].cost_estimation.online.low_alt_parallel_ra_deferral.T_threshold_dnx
13   const D_range_ground_threshold::R =
14     params().modes[mode_int].cost_estimation.online.low_alt_parallel_ra_deferral.D_range_ground_threshold
15   const D_range_ground_rolloff::R =
16     params().modes[mode_int].cost_estimation.online.low_alt_parallel_ra_deferral.D_range_ground_rolloff
17   const R_speed_ground_threshold::R =
18     params().modes[mode_int].cost_estimation.online.low_alt_parallel_ra_deferral.R_speed_ground_threshold
19   const R_speed_ground_rolloff::R =
20     params().modes[mode_int].cost_estimation.online.low_alt_parallel_ra_deferral.R_speed_ground_rolloff
21   const D_range_expansion::R =
22     params().modes[mode_int].cost_estimation.online.low_alt_parallel_ra_deferral.D_range_expansion
23   const C_deferral::R =
24     params().modes[mode_int].cost_estimation.online.low_alt_parallel_ra_deferral.C_deferral
25   const C_deferral_dnx::R =
26     params().modes[mode_int].cost_estimation.online.low_alt_parallel_ra_deferral.C_deferral_dnx
27   s_c.c_alert_deferral = 0.0
28   s_c.c_non_climb_descend = 0.0
29   height_own::R = GetOwnHeight(p.327)( h_own, z_own_ave )
30   z_factor::R = CalculateThresholdRampDownFactor(p.326)( height_own, H_threshold, H_rolloff )
31   s_ground_factor::R = CalculateThresholdRampDownFactor(p.326)( s_c.speed_rel, R_speed_ground_threshold,
32                 R_speed_ground_rolloff )
33   range_expansion::R = (1.0 - s_ground_factor) * D_range_expansion
34   r_ground_factor::R = CalculateThresholdRampUpFactor(p.326)( s_c.range, (D_range_ground_threshold +
35                 range_expansion), D_range_ground_rolloff )
36   s_c.c_alert_deferral = z_factor * min( r_ground_factor, s_ground_factor ) * C_deferral
37   if (T_threshold_dnx <= tau_expected)
38     s_c.c_non_climb_descend = C_deferral_dnx * CalculateThresholdRampDownFactor(p.326)( height_own,
39                               H_threshold_dnx, H_rolloff_dnx )
40   end
41   if (s_c.c_non_climb_descend < s_c.c_alert_deferral)
42     s_c.c_non_climb_descend = s_c.c_alert_deferral
43   end
44   c_altitude::R = DetermineLowAltitudeCost(p.232)( mode_int, height_own, h_own, equip_int, s_c )
45   s_c.c_alert_deferral = s_c.c_alert_deferral + c_altitude
46   s_c.c_non_climb_descend = s_c.c_non_climb_descend + c_altitude
47 end

```

Referenced In: LowAltitudeParallelRADeferralCost(p.229)

DETERMINELOWALTITUDECOST (Algorithm 202) penalizes all alerts at low altitudes based on a series of tau, range, and altitude thresholds. Due to the small horizontal ranges and low closing speeds experienced at low altitudes, this algorithm uses subsequent range belief states to estimate the time until closest point of approach (τ_{range}). Two distinct altitude thresholds are used to create a layered reduction in alerting. The lower layer uses only the radio altimeter to reduce alerting, while the upper layer uses barometric altitude. So long as the range and tau are sufficient and the altitude is below the threshold, alerts on equipped intruders will be deferred.

This algorithm is used twice, once for resolution advisories and once for traffic advisories. The two instances differ in the values in the cost state data structure (s_c) and the parameters used

for the range, tau, and altitude thresholds. The value of *s_c.for_ta* indicates which instance is being used. Specific parameters for the thresholds used for resolution advisories are returned by GETLOWALTITUDECOSTRAPARAMETERS. Specific parameters for the thresholds used for traffic advisories are returned by GETLOWALTITUDECOSTTAPARAMETERS.

This algorithm takes as input *mode_int*, *height_own*, *h_own*, *equip_int*, and *s_c*. This algorithm returns *c_altitude*.

Algorithm 202 DetermineLowAltitudeCost

```

1 function DetermineLowAltitudeCost( mode_int::Z, height_own::R, h_own::R, equip_int::Bool,
2                               s_c::LowAltitudeParallelRADeferralCState(p.E-44) )
3   const H_very_low_alt::R =
4     params().modes[mode_int].cost_estimation.online.low_alt_parallel_ra_deferral.H_very_low_alt
5   const H_low_alt::R =
6     params().modes[mode_int].cost_estimation.online.low_alt_parallel_ra_deferral.H_low_alt
7   const D_range_delta_min::R =
8     params().modes[mode_int].cost_estimation.online.low_alt_parallel_ra_deferral.D_range_delta_min
9   const C_deferral_very_low_alt::R =
10    params().modes[mode_int].cost_estimation.online.low_alt_parallel_ra_deferral.C_deferral_very_low_alt
11   const C_deferral_low_alt::R =
12     params().modes[mode_int].cost_estimation.online.low_alt_parallel_ra_deferral.C_deferral_low_alt
13   t_very_low_alt::R      = 0.0
14   d_range_very_low_alt::R = 0.0
15   t_low_alt::R          = 0.0
16   d_range_low_alt::R    = 0.0
17   c_altitude::R         = 0.0
18   tau_range::R          = Inf
19   range_delta::R = s_c.range_prev - s_c.range
20   if !isnan(s_c.range_prev) && (D_range_delta_min < range_delta)
21     tau_range = s_c.range / range_delta
22   end
23   s_c.range_prev = s_c.range
24   if s_c.for_ta
25     (t_very_low_alt, d_range_very_low_alt, t_low_alt, d_range_low_alt) =
26       GetLowAltitudeCostTAParameters(p.233)( mode_int, equip_int )
27   else
28     (t_very_low_alt, d_range_very_low_alt, t_low_alt, d_range_low_alt) =
29       GetLowAltitudeCostRAParameters(p.234)( mode_int, equip_int )
30   end
31   if !isnan(h_own)           &&
32     (h_own < H_very_low_alt)    &&
33     (t_very_low_alt <= tau_range) &&
34     (d_range_very_low_alt <= s_c.range)
35     c_altitude = C_deferral_very_low_alt
36   elseif (height_own < H_low_alt)  &&
37     (t_low_alt <= tau_range)    &&
38     (d_range_low_alt <= s_c.range)
39     c_altitude = C_deferral_low_alt
40   end
41   return c_altitude::R
42 end
```

Referenced In: UpdateLowAltitudeParallelRADeferralCState(p.231)
--

GETLOWALTITUDECOSTTAPARAMETERS (Algorithm 203) returns the traffic advisory threshold parameters for DETERMINELOWALTITUDECOST based on intruder equipage. Equipped and un-equipped aircraft have different parameters due to the unique types of operations that are performed by each.

This algorithm takes as input *mode_int* and *equip_int*. This algorithm returns *t_very_low_alt*, *d* -

range_low_alt, *t_low_alt*, and *d_range_low_alt*.

Algorithm 203 GetLowAltitudeCostTAParameters

```

1 function GetLowAltitudeCostTAParameters( mode_int::Z, equip_int::Bool )
2   t_very_low_alt::R      = 0.0
3   d_range_very_low_alt::R = 0.0
4   t_low_alt::R           = 0.0
5   d_range_low_alt::R     = 0.0
6   if equip_int
7     t_very_low_alt = params().modes[mode_int].track_threat.ta_low_alt_parallel_ra_deferral.ta_lowalt-
8       equip.T_very_low_alt
9     t_low_alt      = params().modes[mode_int].track_threat.ta_low_alt_parallel_ra_deferral.ta_lowalt-
10      equip.T_low_alt
11    d_range_very_low_alt = params().modes[mode_int].track_threat.ta_low_alt_parallel_ra_deferral.ta-
12      lowalt_equip.D_range_very_low_alt
13    d_range_low_alt     = params().modes[mode_int].track_threat.ta_low_alt_parallel_ra_deferral.ta-
14      lowalt_equip.D_range_low_alt
15  else
16    t_very_low_alt = params().modes[mode_int].track_threat.ta_low_alt_parallel_ra_deferral.ta_lowalt-
17      unequip.T_very_low_alt
18    t_low_alt      = params().modes[mode_int].track_threat.ta_low_alt_parallel_ra_deferral.ta_lowalt-
19      unequip.T_low_alt
20    d_range_very_low_alt = params().modes[mode_int].track_threat.ta_low_alt_parallel_ra_deferral.ta-
21      lowalt_unequip.D_range_very_low_alt
22    d_range_low_alt     = params().modes[mode_int].track_threat.ta_low_alt_parallel_ra_deferral.ta-
23      lowalt_unequip.D_range_low_alt
24  end
25  return (t_very_low_alt::R, d_range_very_low_alt::R, t_low_alt::R, d_range_low_alt::R)
26 end

```

Referenced In: DetermineLowAltitudeCost(p.232)

GETLOWALTITUDECOSTRAPARAMETERS (Algorithm 204) returns the resolution advisory threshold parameters for DETERMINELOWALTITUDECOST based on intruder equipage. Equipped and unequipped aircraft have different parameters due to the unique types of operations that are performed by each.

This algorithm takes as input *mode_int* and *equip_int*. This algorithm returns *t_very_low_alt*, *d_range_low_alt*, *t_low_alt*, and *d_range_low_alt*.

Algorithm 204 GetLowAltitudeCostRAParameters

```

1 function GetLowAltitudeCostRAParameters( mode_int::Z, equip_int::Bool )
2   t_very_low_alt::R      = 0.0
3   d_range_very_low_alt::R = 0.0
4   t_low_alt::R           = 0.0
5   d_range_low_alt::R     = 0.0
6   if equip_int
7     t_very_low_alt = params().modes[mode_int].cost_estimation.online.low_alt_parallel_ra-
      deferral.lowalt_equip.T_very_low_alt
8     t_low_alt      = params().modes[mode_int].cost_estimation.online.low_alt_parallel_ra-
      deferral.lowalt_equip.T_low_alt
9     d_range_very_low_alt = params().modes[mode_int].cost_estimation.online.low_alt_parallel_ra-
      deferral.lowalt_equip.D.range_very_low_alt
10    d_range_low_alt     = params().modes[mode_int].cost_estimation.online.low_alt_parallel_ra-
      deferral.lowalt_equip.D.range_low_alt
11  else
12    t_very_low_alt = params().modes[mode_int].cost_estimation.online.low_alt_parallel_ra-
      deferral.lowalt_unequip.T_very_low_alt
13    t_low_alt      = params().modes[mode_int].cost_estimation.online.low_alt_parallel_ra-
      deferral.lowalt_unequip.T_low_alt
14    d_range_very_low_alt = params().modes[mode_int].cost_estimation.online.low_alt_parallel_ra-
      deferral.lowalt_unequip.D.range_very_low_alt
15    d_range_low_alt     = params().modes[mode_int].cost_estimation.online.low_alt_parallel_ra-
      deferral.lowalt_unequip.D.range_low_alt
16  end
17  return (t_very_low_alt::R, d_range_very_low_alt::R, t_low_alt::R, d_range_low_alt::R)
18 end

```

Referenced In: DetermineLowAltitudeCost(p.232)

3.2.6.7 Max Reversal Cost

MAXREVERSALCOST (Algorithm 205) penalizes certain actions if the number of reversals exceeds a threshold (N_{limit}). To prevent premature advisory termination, clear of conflict is restricted by the cost C_{coc} for $T_{coc_threshold}$ TRM cycles after the reversal limit is reached. Reversals are restricted by the cost $C_{reversal}$ if the reversal limit has been reached, unless the master intruder aircraft is forcing the reversal, as determined by ISMASTERFORCINGREVERSAL. The reversal count and time since reaching the reversal limit are contained within the internal state information (s_c).

Once per cycle, the state variables in s_c are set using the helper algorithm UPDATEREVERSALSTATE.

This algorithm takes as input $mode_int$, dz_min , dz_max , vrc_int , $master_int$, dz_min_prev , dz_max_prev , $multithreat_prev$, $equip_int$, $update$ and s_c . This algorithm returns $cost$.

Algorithm 205 MaxReversalCost

```

1 function MaxReversalCost( mode_int::Z, dz_min::R, dz_max::R, vrc_int::UInt32, master_int::Bool,
2           dz_min_prev::R, dz_max_prev::R, equip_int::Bool,
3           multithreat_prev::Bool, update::Bool, s_c::MaxReversalCState(p.E-44) )
4   const T_coc_threshold::Z = params().modes[mode_int].cost_estimation.online.max_reversal.T_coc_threshold
5   const N_limit::Z       = params().modes[mode_int].cost_estimation.online.max_reversal.N_limit
6   const C_reversal::R    = params().modes[mode_int].cost_estimation.online.max_reversal.C_reversal
7   const C_coc::R        = params().modes[mode_int].cost_estimation.online.max_reversal.C_coc
8   if update
9     UpdateMaxReversalCState(p.236)( mode_int, vrc_int, master_int, dz_min_prev, dz_max_prev,
10                                equip_int, multithreat_prev, s_c )
11   end
12   cost::R = 0.0
13   if IsCOC(p.332)( dz_min, dz_max )
14     if (s_c.crossed_thres_time < T_coc_threshold) && (N_limit <= s_c.num_reversals)
15       cost = C_coc
16     end
17   elseif (N_limit <= s_c.num_reversals) &&
18     !IsMasterForcingReversal(p.335)( master_int, s_c.sense_own_prev, s_c.sense_int )
19     sense_own = RatesToSense(p.342)( dz_min, dz_max )
20     if IsReversal(p.337)( s_c.sense_own_prev, sense_own )
21       cost = C_reversal
22     end
23   end
24   return cost::R
25 end
```

Referenced In: OnlineCoordinatedActionCostEstimation(p.216)

UPDATEMAXREVERSALCSTATE (Algorithm 206) is used to update the state variables of MAXREVERSALCOST once per cycle. The reversal count is incremented if there was a reversal in the previous TRM cycle, with two additional constraints. The first constraint is that the reversal was not forced by an intruder master aircraft, as determined by ISMASTERFORCINGREVERSAL. The second constraint is that ownership was not in a multithreat encounter (*multithreat_prev*) with an unequipped intruder (*equip_int*). Reversals for unequipped intruders in multithreat encounters are not counted in order to maintain performance with equipped intruders. When the previous action was clear of conflict, the reversal counter and time since last reversal are reset.

This algorithm takes as input *mode_int*, *vrc_int*, *master_int*, *dz_min_prev*, *dz_max_prev*, *equip_int*, *multithreat_prev*, and *s_c*. The algorithm outputs updates to *s_c::MaxReversalCState*.

Algorithm 206 UpdateMaxReversalCState

```

1 function UpdateMaxReversalCState( mode_int::Z, vrc_int::UInt32, master_int::Bool,
2           dz_min_prev::R, dz_max_prev::R, equip_int::Bool,
3           multithreat_prev::Bool, s_c::MaxReversalCState(p_E-44) )
4   const N_limit::Z = params().modes[mode_int].cost_estimation.online.max_reversal.N_limit
5   sense_own_prev::Symbol = RatesToSense(p_342)( dz_min_prev, dz_max_prev )
6   if IsReversal(p_337)( s_c.sense_own_prev, sense_own_prev )
7     if !IsMasterForcingReversal(p_335)( master_int, s_c.sense_own_prev, s_c.sense_int ) &&
8       (!multithreat_prev || (multithreat_prev & equip_int))
9     s_c.num_reversals = s_c.num_reversals + 1
10    end
11  elseif IsCOC(p_332)( dz_min_prev, dz_max_prev )
12    s_c.num_reversals = 0
13  end
14  if (N_limit <= s_c.num_reversals)
15    s_c.crossed_thres_time = s_c.crossed_thres_time + 1
16  else
17    s_c.crossed_thres_time = 0
18  end
19  s_c.sense_int = VRCToSense(p_344)( vrc_int )
20  s_c.sense_own_prev = sense_own_prev
21 end

```

Referenced In: MaxReversalCost(p.235)

3.2.6.8 Prevent Early Weakening Cost

PREVENTEARLYWEAKENINGCOST (Algorithm 207) penalizes issuance of a weakening resolution advisory by $C_{weakening}$ in equipped-equipped encounters to ensure the coordinating aircraft continue to issue corrective resolution advisories until they are projected to diverge. Divergence is determined by verifying that the expected vertical separation at minimum tau is positive if intruder VRC sense is positive or that the expected vertical separation at minimum tau is negative if intruder VRC sense is negative. Tau is the time until the intruder comes within a lateral conflict distance of own aircraft and is provided as an input ($\tau_{expected}$).

This algorithm takes as input $mode_int$, dz_min , dz_max , dz_min_prev , dz_max_prev , z_own_ave , dz_own_ave , z_int_ave , dz_int_ave , vrc_int , and $\tau_{expected}$. This algorithm returns $cost$.

Algorithm 207 PreventEarlyWeakeningCost

```

1 function PreventEarlyWeakeningCost( mode_int::Z, dz_min::R, dz_max::R,
2                               dz_min_prev::R, dz_max_prev::R,
3                               z_own_ave::R, dz_own_ave::R,
4                               z_int_ave::R, dz_int_ave::R,
5                               vrc_int::UInt32, tau_expected::R )
6   const C_weakening::R =
7     params().modes[mode_int].cost_estimation.online.prevent_early_weakening
8   cost::R = 0.0
9   sense_int::Symbol = VRCToSense(p.344)( vrc_int )
10  if (:None != sense_int) && (IsDND(p.334)( dz_min, dz_max ) || IsDNC(p.334)( dz_min, dz_max ))
11    z_rel_projected::R = (z_own_ave - z_int_ave) + ((dz_own_ave - dz_int_ave) * tau_expected)
12    if IsCorrective(p.332)( dz_min_prev, dz_max_prev ) &&
13      (((0 <= z_rel_projected) && (:Up == sense_int)) ||
14       ((z_rel_projected < 0) && (:Down == sense_int)))
15      cost = C_weakening
16    end
17  end
18  return cost::R
19 end

```

Referenced In: OnlineCoordinatedActionCostEstimation(p.216)

3.2.6.9 Restrict Clear of Conflict Due To Reversal

RESTRICTCOCDUETOREVERSAL (Algorithm 208) penalizes the transition to clear of conflict (COC) by $C_{restrict_coc}$ when own aircraft is a slave and is forced to reverse due to coordination. The $master_int$ variable is a boolean representing if the intruder aircraft is the master. ISMASTERFORCINGREVERSAL (Algorithm 289) ensures that the restriction of COC only occurs for the one time step that a VRC message is received from an intruder aircraft that is the master. It forces an opposite sense advisory than was already issued. Whether an intruder is the master is determined in VERTICALTRMUPDATE using ISINTRUDERMASTER.

Once per cycle, the state variables in s_c are set using the helper algorithm UPDATERESTRICTCOCDUETOREVERSALCSTATE.

This algorithm takes as input $mode_int$, dz_min , dz_max , vrc_int , $master_int$, dz_min_prev , dz_max_prev , $update$ and s_c . This algorithm returns $cost$.

Algorithm 208 RestrictCOCDueToReversal

```

1 function RestrictCOCDueToReversal( mode_int::Z, dz_min::R, dz_max::R, vrc_int::UInt32,
2                               master_int::Bool, dz_min_prev::R, dz_max_prev::R,
3                               update::Bool, s_c::RestrictCOCCState(p. E-45) )
4   const C_restrict_coc::R =
5     params().modes[mode_int].cost_estimation.online.restrict_coc_due_to_reversal.C_restrict_coc
6   if update
7     UpdateRestrictCOCDueToReversalCState(p.238)( dz_min_prev, dz_max_prev, s_c )
8   end
9   cost::R = 0.0
10  if IsCOC(p.332)( dz_min, dz_max )
11    sense_int::Symbol = VRCToSense(p.344)( vrc_int )
12    if IsMasterForcingReversal(p.335)( master_int, s_c.sense_own_prev, sense_int ) && !s_c.coc_prev
13      cost = C_restrict_coc
14    end
15  end
16  return cost::R
17 end

```

Referenced In: OnlineCoordinatedActionCostEstimation(p.216)
--

UPDATERESTRICTCOCDUETOREVERSALCSTATE (Algorithm 209) is used to update the state variables of RESTRICTCOCDUETOREVERSAL once per cycle. A flag indicating clear of conflict on the previous cycle and memory of the own aircraft sense on the previous cycle are updated.

This algorithm takes as input *dz_min*, *dz_max*, and *s_c*. The algorithm outputs updates to *s_c::RestrictCOCCState*.

Algorithm 209 UpdateRestrictCOCDueToReversalCState

```

1 function UpdateRestrictCOCDueToReversalCState( dz_min_prev::R, dz_max_prev::R, s_c::RestrictCOCCState(p. E-45) )
2   s_c.coc_prev = IsCOC(p.332)( dz_min_prev, dz_max_prev )
3   s_c.sense_own_prev = RatesToSense(p.342)( dz_min_prev, dz_max_prev )
4 end

```

Referenced In: RestrictCOCDueToReversal(p.238)

3.2.6.10 SA01 Heuristic

SA01HEURISTIC (Algorithm 210) penalizes resolution advisories to force a reversal. If the force reversal flag (*s_c.force_reversal*) is set, clear of conflict and all non-reverse advisories are penalized. This cost uses the sense of the own aircraft advisory from the previous cycle to determine which proposed actions are not reversals.

Once per cycle, the state variables in *s_c* are set using UPDATESA01HEURISTICCSTATE (Algorithm 211), a helper algorithm.

The SA01HEURISTIC is excluded from *cost_ra_subset*, which is used in INDIVIDUALCOSTESTIMATION to determine whether to penalize an action to force a resolution advisory.

This algorithm takes as input *mode_int*, *dz_min*, *dz_max*, *vrc_int*, *equip_int*, *resp_own*, *master_int*, *z_rel*, *dz_own_ave*, *dz_int_ave*, *dz_min_prev*, *dz_max_prev*, *tau_expected_no_horizon*, *update* and *s_c*. This algorithm returns *cost*.

Algorithm 210 SA01Heuristic

```

1 function SA01Heuristic( mode_int::Z, dz_min::R, dz_max::R, vrc_int::UInt32, equip_int::Bool,
2             resp_own::Bool, master_int::Bool, z_rel::R, dz_own_ave::R, dz_int_ave::R,
3             dz_min_prev::R, dz_max_prev::R, tau_expected_no_horizon::R,
4             update::Bool, s_c::SA01HeuristicCState(p. E-46) )
5     const C_non_reversal::R = params().modes[mode_int].cost_estimation.online.SA01.C_non_reversal
6     const C_coc::R          = params().modes[mode_int].cost_estimation.online.SA01.C_coc
7     if update
8         UpdateSA01HeuristicCState(p. 240)( mode_int, vrc_int, equip_int, resp_own, master_int, z_rel,
9                                         dz_own_ave, dz_int_ave, dz_min_prev, dz_max_prev,
10                                        tau_expected_no_horizon, s_c )
11    end
12    sense_own_prev::Symbol = RatesToSense(p. 342)( dz_min_prev, dz_max_prev )
13    sense_own::Symbol      = RatesToSense(p. 342)( dz_min, dz_max )
14    cost::R = 0.0
15    if s_c.force_reversal
16        if IsCOC(p. 332)( dz_min, dz_max )
17            cost = C_coc
18        elseif (sense_own_prev == sense_own) && (sense_own_prev != :None)
19            cost = C_non_reversal
20        end
21    end
22    return cost::R
23 end

```

Referenced In: OnlineCoordinatedActionCostEstimation(p. 216)

UPDATESA01HEURISTICCSTATE (Algorithm 211) is used once per cycle to update the state variables of SA01HEURISTIC. If the own aircraft is not responding to advisories (*resp_own*), the own aircraft and intruder have vertical rates in the same direction and of sufficient magnitude (*R_min*), and the intruder is equipped but not master, the SHOULDREVERSE algorithm is invoked. These criteria help identify an SA01 (vertical chase) encounter when the master is not responding and there is an equipped intruder. During these situations, a reversal is often the only way to resolve an encounter. The state variable *s_c.force_reversal* stores the result of SHOULDREVERSE, which is used to determine if a reversal should be issued.

This algorithm takes as input *mode_int*, *vrc_int*, *equip_int*, *resp_own*, *master_int*, *z_rel*, *dz_own_ave*, *dz_int_ave*, *dz_min_prev*, *dz_max_prev*, *tau_expected_no_horizon*, and *s_c*. The algorithm outputs updates to *s_c::SA01HeuristicCState*.

Algorithm 211 UpdateSA01HeuristicCState

```

1 function UpdateSA01HeuristicCState( mode_int::Z, vrc_int::UInt32, equip_int::Bool,
2                               resp_own::Bool, master_int::Bool,
3                               z_rel::R, dz_own_ave::R, dz_int_ave::R,
4                               dz_min_prev::R, dz_max_prev::R,
5                               tau_expected_no_horizon::R, s_c::SA01HeuristicCState(p.E-46) )
6   const R_min::R = params().modes[mode_int].cost_estimation.online.SA01.R_min
7   sense_own_prev::Symbol = RatesToSense(p.342)( dz_min_prev, dz_max_prev )
8   same_sign_rates::Bool = (0 < (dz_own_ave * dz_int_ave)) &&
9     (R_min < abs( dz_own_ave )) &&
10    (R_min < abs( dz_int_ave ))
11   s_c.force_reversal = false
12   if !resp_own && same_sign_rates && equip_int && !master_int
13     s_c.force_reversal = ShouldReverse(p.241)( mode_int, z_rel, dz_own_ave, dz_int_ave,
14                                               tau_expected_no_horizon, vrc_int, sense_own_prev,
15                                               s_c.range )
16   end
17 end

```

Referenced In: SA01Heuristic(p.239)

SHOULDREVERSE (Algorithm 212) uses a linear approximation, with constant acceleration to the vertical rate for the current resolution advisory, to estimate the vertical separation with the current resolution advisory (z_{rel_proj}), and after issuing a reversal ($z_{rel_proj_rev}$). The algorithm then uses the estimates to determine whether a reversal should be issued. If more separation is expected with the reversal or an altitude crossing is expected without a reversal, a reversal will be forced. SHOULDREVERSE returns FALSE if the estimated value of tau is too short to allow time for the intruder to reverse (T_{min_tau}) or if the estimated value of tau is long enough that a reversal may not be necessary (T_{max_tau}). It also returns FALSE if a reversal would cause an altitude crossing and the vertical separation between the two aircraft is greater than $H_{rel_threshold}$ or the horizontal separation is less than $D_{range_threshold}$.

This algorithm takes as input $mode_int$, z_{rel} , dz_{own} , dz_{int} , tau , vrc_int , $sense_own_prev$, and $range$. This algorithm returns $force_reversal$.

Algorithm 212 ShouldReverse

```

1 function ShouldReverse( mode_int::Z, z_rel::R, dz_own::R, dz_int::R, tau::R,
2           vrc_int::UInt32, sense_own_prev::Symbol, range::R )
3   const R_acceleration::R = params().modes[mode_int].cost_estimation.online.SA01.R_acceleration
4   const R_strengthen::R = params().modes[mode_int].cost_estimation.online.SA01.R_strengthen
5   const R_standard::R = params().modes[mode_int].cost_estimation.online.SA01.R_standard
6   const T_min::R = params().modes[mode_int].cost_estimation.online.SA01.T_min
7   const T_max::R = params().modes[mode_int].cost_estimation.online.SA01.T_max
8   const T_delay::R = params().modes[mode_int].cost_estimation.online.SA01.T_delay
9   const H_rel_threshold::R = params().modes[mode_int].cost_estimation.online.SA01.H_rel_threshold
10  const D_range_threshold::R = params().modes[mode_int].cost_estimation.online.SA01.D_range_threshold
11  sense_int::Symbol = VRCToSense(p.344)( vrc_int )
12  force_reversal::Bool = false
13  sense_own_prev_sign::Z = 1
14  if (sense_own_prev == :Down)
15    sense_own_prev_sign = -1
16  end
17  reversal_is_cross::Bool = 0 < z_rel*dz_own
18  if (sense_int != :None) && ((sense_own_prev_sign * dz_own) <= 0) &&
19    (T_min <= tau) && (tau <= T_max) &&
20    (!reversal_is_cross || ((abs(z_rel) <= H_rel_threshold) && (D_range_threshold <= range)))
21  t_max::Z = floor( tau - T_delay - 1 )
22  sense_int_sign::Z = 1
23  if (sense_int == :Down)
24    sense_int_sign = -1
25  end
26  z_rel_proj::R = z_rel + ((dz_int - dz_own) * T_delay)
27  dz_int_new::R = dz_int
28  for t = 1:t_max
29    if (0 < (dz_int * sense_int_sign)) && (R_strengthen <= abs( dz_int ))
30      dz_int_new = dz_int
31    elseif ((dz_int_new * sense_int_sign) < 0) || (abs( dz_int_new ) < R_strengthen)
32      dz_int_new = dz_int_new + (sense_int_sign * R_acceleration)
33    elseif (R_strengthen < abs( dz_int_new ))
34      dz_int_new = sense_int_sign * R_strengthen
35    end
36    z_rel_proj = z_rel_proj + (dz_int_new - dz_own)
37  end
38  z_rel_proj_rev = z_rel + ((dz_int - dz_own) * T_delay)
39  dz_int_new = dz_int
40  for t = 1:t_max
41    if ((dz_int_new * sense_own_prev_sign) < 0) || (abs( dz_int_new ) < R_standard)
42      dz_int_new = dz_int_new + (sense_own_prev_sign * R_acceleration)
43    elseif (R_standard < abs( dz_int_new ))
44      dz_int_new = sense_own_prev_sign * R_standard
45    end
46    z_rel_proj_rev = z_rel_proj_rev + (dz_int_new - dz_own)
47  end
48  if (abs( z_rel_proj ) < abs( z_rel_proj_rev ))
49    force_reversal = true
50  elseif (0 < sense_own_prev_sign) && (0 < z_rel_proj)
51    force_reversal = true
52  elseif (sense_own_prev_sign < 0) && (z_rel_proj < 0)
53    force_reversal = true
54  else
55    force_reversal = false
56  end
57  end
58  return force_reversal::Bool
59 end

```

Referenced In: UpdateSA01HeuristicCState(p.240)

3.2.6.11 Time-Based Non-Compliance Cost

TIMEBASEDNONCOMPLIANCECOST (Algorithm 213) penalizes persisting the current resolution advisory when either aircraft is not responding to RAs and the situation has continued to deteriorate. The necessity of these penalties is determined by the current RA cost state variable ($s_c.current_ra_cost$). A cost (C_coc) is also applied to clear of conflict whenever the current RA is penalized to prevent a transition to no alert.

Once per cycle, the state variables in s_c are set using the helper algorithm UPDATETIMEBASEDNONCOMPLIANCECSTATE.

This algorithm takes as input $mode_int$, dz_min , dz_max , dz_min_prev , dz_max_prev , z_own_ave , dz_own_ave , z_int_ave , dz_int_ave , $tau_expected_no_horizon$, $equip_int$, $resp_int$, $update$, and s_c . This algorithm returns $cost$.

Algorithm 213 TimeBasedNonComplianceCost

```

1 function TimeBasedNonComplianceCost( mode_int::Z, dz_min::R, dz_max::R, dz_min_prev::R, dz_max_prev::R,
2                                     z_own_ave::R, dz_own_ave::R, z_int_ave::R, dz_int_ave::R,
3                                     tau_expected_no_horizon::R, equip_int::Bool, resp_int::Bool,
4                                     update::Bool, s_c::TimeBasedNonComplianceCState(p. E-46) )
5   const C_coc::R =
6     params().modes[mode_int].cost_estimation.online.time_based_non_compliance.C_coc
7   const C_ra_epsilon::R =
8     params().modes[mode_int].cost_estimation.online.time_based_non_compliance.C_ra_epsilon
9   if update
10    UpdateTimeBasedNonComplianceCState(p. 244)( mode_int, dz_min_prev, dz_max_prev,
11                                              z_own_ave, dz_own_ave, z_int_ave, dz_int_ave,
12                                              tau_expected_no_horizon, equip_int, resp_int, s_c )
13  end
14  cost::R = 0.0
15  if (C_ra_epsilon < s_c.current_ra_cost)
16    if (dz_min_prev == dz_min) && (dz_max_prev == dz_max)
17      cost = s_c.current_ra_cost
18    elseif IsCOC(p. 332)( dz_min, dz_max )
19      cost = C_coc
20    end
21  end
22  return cost::R
23 end
```

Referenced In: OnlineCoordinatedActionCostEstimation(p. 216)

UPDATETIMEBASEDNONCOMPLIANCECSTATE (Algorithm 214) is used to update the state variables of TIMEBASEDNONCOMPLIANCECOST once per cycle. The algorithm first calculates a factor ($factor_separation$) based on the expected separation at closest point of approach (z_rel_actual) and a set of thresholds to determine cost applicability. If an altitude crossing is expected, larger thresholds are used. Three values are then calculated to determine factors that reduce or increase the value of the cost applied:

- $current_ra_max_tau$ - the largest tau value observed during the current RA
- $current_ra_min_tau_time$ - the number of cycles that the tau value has been below $T_threshold_min_tau$ during the current RA
- $current_ra_dz_initial_delta$ - the difference between the ownship vertical rate when the RA was issued and the target vertical rate of the current RA
 - this value is calculated using the helper algorithm FINDCOMPLIANTRATEDELTA

All of these values are reset each time the RA is clear of conflict or whenever the RA changes. Additionally the *current_ra_dz_initial_delta* value is bounded by the *R_vert_rel_compare_min* parameter to prevent the possibility of division by zero.

The own aircraft contribution to the non-compliance cost is calculated using DETERMINEOWN-NONCOMPLIANCEFACTOR. It is combined with the separation factor (*factor_separation*) to determine the total cost to apply. The separation factor is also used in conjunction with the standard intruder response model calculated with INTRUDERRESPONSEESTIMATION. If the intruder is believed to be non-responsive and the separation appears to be insufficient, the intruder factor (*factor_int*) will encourage transitions to stronger RAs, especially if reversals are prohibited by coordination.

The greater of the two aircraft factors (*factor_own* and *factor_int*) is applied to the nominal cost $C_{-current_ra}$. This ensures that the cost will be applied if either of the two aircraft is not responding and the separation appears to be insufficient. However, in the event that the current RA is clear of conflict or has a target rate of magnitude 2,500 fpm or higher the cost is removed. This prevents initial alerts and undesirable transitions from the strongest possible RAs due to this cost.

This algorithm takes as input *mode_int*, *dz_min_prev*, *dz_max_prev*, *z_own_ave*, *dz_own_ave*, *z_int_ave*, *dz_int_ave*, *tau_expected_no_horizon*, *equip_int*, *resp_int*, and *s_c*. The algorithm outputs updates to *s_c::TimeBasedNonComplianceCState*.

Algorithm 214 UpdateTimeBasedNonComplianceCState

```

1 function UpdateTimeBasedNonComplianceCState( mode_int::Z, dz_min_prev::R, dz_max_prev::R,
2                                     z_own_ave::R, dz_own_ave::R, z_int_ave::R, dz_int_ave::R,
3                                     tau_expected_no_horizon::R, equip_int::Bool,
4                                     resp_int::Bool, s_c::TimeBasedNonComplianceCState(p.E-46) )
5   const R_strengthen::R = params().actions.strengthen_rate
6   const T_threshold_min_tau::R =
7     params().modes[mode_int].cost_estimation.online.time_based_non_compliance.T_threshold_min_tau
8   const R_vert_rel_compare_min::R =
9     params().modes[mode_int].cost_estimation.online.time_based_non_compliance.R_vert_rel_compare_min
10  const H_rel_threshold::R =
11    params().modes[mode_int].cost_estimation.online.time_based_non_compliance.H_rel_threshold
12  const H_rel_rolloff::R =
13    params().modes[mode_int].cost_estimation.online.time_based_non_compliance.H_rel_rolloff
14  const H_rel_threshold_cross::R =
15    params().modes[mode_int].cost_estimation.online.time_based_non_compliance.H_rel_threshold_cross
16  const H_rel_rolloff_cross::R =
17    params().modes[mode_int].cost_estimation.online.time_based_non_compliance.H_rel_rolloff_cross
18  const C_current_ra::R =
19    params().modes[mode_int].cost_estimation.online.time_based_non_compliance.C_current_ra
20  dz_delta::R = FindCompliantRateDelta(p.245)( dz_own_ave, dz_min_prev, dz_max_prev )
21  z_rel_actual::R = (z_int_ave - z_own_ave) + ((dz_int_ave - dz_own_ave) * tau_expected_no_horizon)
22  factor_separation::R = 0.0
23  if IsProjectedCrossing(p.337)( (z_int_ave - z_own_ave), z_rel_actual )
24    factor_separation =
25      CalculateThresholdRampDownFactor(p.326)( abs(z_rel_actual), H_rel_threshold_cross,
26                                              H_rel_rolloff_cross )
27  else
28    factor_separation =
29      CalculateThresholdRampDownFactor(p.326)( abs(z_rel_actual), H_rel_threshold, H_rel_rolloff )
30  end
31  if IsCOC(p.332)( dz_min_prev, dz_max_prev ) ||
32    (dz_min_prev != s_c.dz_min_prev2) ||
33    (dz_max_prev != s_c.dz_max_prev2)
34    s_c.current_ra_max_tau = tau_expected_no_horizon
35    s_c.current_ra_min_tau_time = 0
36    s_c.current_ra_dz_initial_delta = dz_delta
37  else
38    if (s_c.current_ra_max_tau < tau_expected_no_horizon)
39      s_c.current_ra_max_tau = tau_expected_no_horizon
40    end
41    if (tau_expected_no_horizon < T_threshold_min_tau)
42      s_c.current_ra_min_tau_time = s_c.current_ra_min_tau_time + 1
43    end
44  end
45  if (s_c.current_ra_dz_initial_delta < R_vert_rel_compare_min)
46    s_c.current_ra_dz_initial_delta = R_vert_rel_compare_min
47  end
48  factor_own::R = factor_separation *
49  DetermineOwnNonComplianceFactor(p.246)( mode_int, tau_expected_no_horizon, dz_delta, s_c )
50  factor_int::R = 0.0
51  if equip_int && !resp_int
52    factor_int = factor_separation
53  end
54  s_c.current_ra_cost = C_current_ra * max( factor_own, factor_int )
55  if (R_strengthen <= dz_min_prev) || (dz_max_prev <= -R_strengthen) || IsCOC(p.332)( dz_min_prev,
56    dz_max_prev )
57    s_c.current_ra_cost = 0.0
58  end
59  s_c.dz_min_prev2 = dz_min_prev
60  s_c.dz_max_prev2 = dz_max_prev
60 end
```

Referenced In: TimeBasedNonComplianceCost(p.242)

FINDCOMPLIANTRATEDELTA (Algorithm 215) calculates the difference between the current own aircraft vertical rate and the vertical rate limits of the advisory. The advisory vertical rate limits are compared to the own aircraft vertical rate to determine which one should be used in the calculation. The calculated difference (*dz_delta*) is returned.

This algorithm takes as input *dz_own*, *dz_min*, and *dz_max*. This algorithm returns *dz_delta*.

Algorithm 215 FindCompliantRateDelta

```

1 function FindCompliantRateDelta( dz_own::R, dz_min::R, dz_max::R )
2   dz_delta::R = 0.0
3   if (dz_own < dz_min)
4     dz_delta = dz_min - dz_own
5   elseif (dz_max < dz_own)
6     dz_delta = dz_own - dz_max
7   end
8   return dz_delta::R
9 end
```

Referenced In: UpdateTimeBasedNonComplianceCState(p. 244)
--

DETERMINEOWNNONCOMPLIANCEFACTOR (Algorithm 216) uses tau and own aircraft vertical rate to determine whether to apply the non-compliance cost. Three factors are calculated to determine the overall factor for own aircraft behavior *factor_own*. The *factor_tau_apply* ensures that the cost is only applied if *tau_expected_no_horizon* is low enough, i.e., when the aircraft are close to CPA (closest point of approach). The *factor_tau_diff* ensures that the cost is only applied if tau has decreased sufficiently and\or been below the minimum for enough time. The *factor_rate* ensures that the cost is only applied if the own aircraft vertical rate is not compliant with the target rate of the current RA. All three factors are combined and returned to UPDATETIMEBASEDNONCOMPLIANCECSTATE.

This algorithm takes as input *mode_int*, *tau_expected_no_horizon*, *dz_delta*, and *s_c*. This algorithm returns *factor_own*.

Algorithm 216 DetermineOwnNonComplianceFactor

```

1 function DetermineOwnNonComplianceFactor( mode_int::Z, tau_expected_no_horizon::R, dz_delta::R,
2                                         s_c::TimeBasedNonComplianceCState(p.E-46) )
3     const T_threshold_apply::R =
4         params().modes[mode_int].cost_estimation.online.time_based_non_compliance.T_threshold_apply
5     const T_rolloff_apply::R =
6         params().modes[mode_int].cost_estimation.online.time_based_non_compliance.T_rolloff_apply
7     const T_threshold_tau_delta::R =
8         params().modes[mode_int].cost_estimation.online.time_based_non_compliance.T_threshold_tau_delta
9     const T_rolloff_tau_delta::R =
10        params().modes[mode_int].cost_estimation.online.time_based_non_compliance.T_rolloff_tau_delta
11    const X_vert_rel_factor_max::R =
12        params().modes[mode_int].cost_estimation.online.time_based_non_compliance.X_vert_rel_factor_max
13    factor_tau_apply::R =
14        CalculateThresholdRampDownFactor(p.326)( tau_expected_no_horizon, T_threshold_apply, T_rolloff_apply )
15    tau_delta = s_c.current_ra_max_tau - tau_expected_no_horizon + s_c.current_ra_min_tau_time
16    factor_tau_diff::R =
17        CalculateThresholdRampUpFactor(p.326)( tau_delta, T_threshold_tau_delta, T_rolloff_tau_delta )
18    factor_rate::R = dz_delta / s_c.current_ra_dz_initial_delta
19    if (X_vert_rel_factor_max < factor_rate)
20        factor_rate = X_vert_rel_factor_max
21    end
22    factor_own::R = factor_tau_apply * factor_tau_diff * factor_rate
23    return factor_own::R
24 end

```

Referenced In: UpdateTimeBasedNonComplianceCState(p. 244)

3.2.7 Individual Cost Estimation

INDIVIDUALCOSTESTIMATION (Algorithm 217) processes and combines the offline and online costs for an individual intruder aircraft. It takes as input the estimated offline and online costs, as well as additional inputs, such as own altitude above ground, shown collectively in Figure 3-6.

The estimated offline and online costs are combined using vector addition represented by “+” in Figure 3-6. The offline costs are combined with three distinct online cost vectors: *cost_online_ta*, *cost_online_ra*, and *cost_online_ra_subset*. The index of the best action in *cost_online_ra_subset* is determined through a call to MINCOSTINDEX (Algorithm 298) and stored in *act_subset*. It is used to access the value of the best action, which is stored in *act_subset_value*.

The subset of costs (*cost_ra_subset*) is used to determine whether to apply an additional penalty (*C_force_alert*) on Clear of Conflict and preventive advisories to force a corrective advisory. This penalty is only applied if the current height of ownship (*height_own*) is above a threshold (*H_threshold_lower*) to prevent adverse interactions with ALTITUDEINHIBITCOST and COMPATIBILITYCOST. Additionally, different thresholds are used in the processing of *cost_ra_subset* as the height of ownship increases.

If the current RA is clear of conflict and the offline cost for clear of conflict is above a threshold ($c_coc_threshold$), the *ra_imminent* flag is set to TRUE. This allows INDIVIDUALCOSTESTIMATION to bypass the remaining checks that involve *cost_ra_subset* and ensures both Clear of Conflict and preventive RAs are always penalized by *C_force_alert*.

Even if $ra_imminent$ is FALSE, the values from $cost_ra_subset$ can still result in the penalty being applied. If the lowest cost action from the subset, act_subset , is not Clear of Conflict, C_force_alert will be applied to Clear of Conflict. Furthermore, if the lowest cost action from the subset is not a preventive RA, both Clear of Conflict and all preventive RAs will be penalized by C_force_alert .

Finally, if the current RA is not Clear of Conflict and the value of the lowest cost from the subset (*act_subset_value*) is above *c_coc_threshold*, Clear of Conflict will be penalized by *C_force_alert*.

To prevent the multi-threat level-off (MTLO) action from being selected for an individual threat, *C_restrict* is always applied to the MTLO action.

The vector of combined online and offline costs *cost_ta*, is sent to the TRACKTHREATASSESSMENT component. The vector of combined online and offline costs, *cost_ra*, is sent to the ACTIONSELECTION component.

This algorithm takes as input *mode_int*, *height_own*, *dz_own_ave*, *st_own*, *cost_offline*, *cost_online_ra*, *cost_online_ra_subset*, *cost_online_ta*. Descriptions of the INDIVIDUALCOSTESTIMATION output variables are found in Table 3-6.

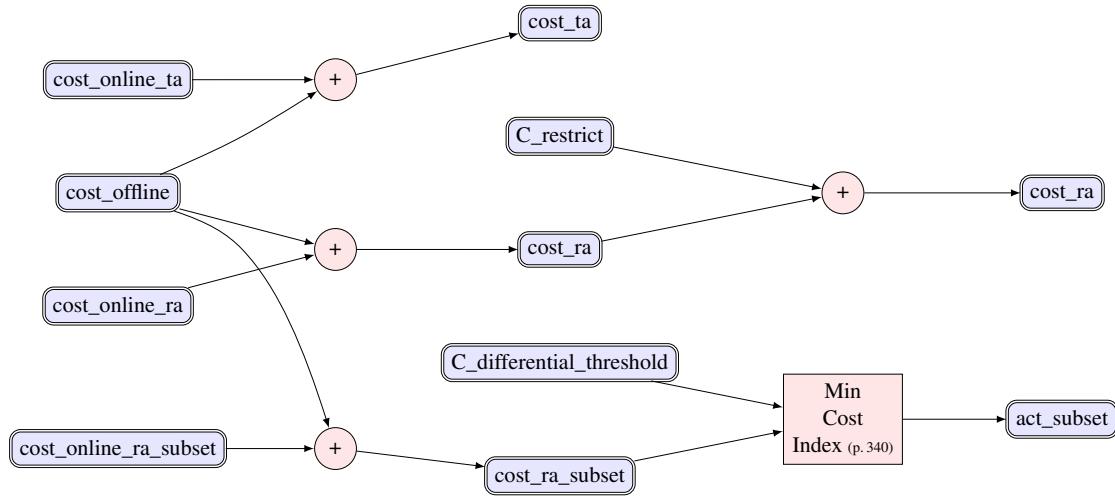


Figure 3-6. IndividualCostEstimation overview.

Table 3-6. TRM Algorithm Output Variables - IndividualCostEstimation

Variable	Units	Type	Description
<i>cost_ra</i>	N/A	real	Vector of total online cost of each action for single intruder; used to select RAs
<i>cost_ta</i>	N/A	real	Vector of total online cost of each action for single intruder; used to determine TAs

Algorithm 217 IndividualCostEstimation

```

1 function IndividualCostEstimation( mode_int::Z, height_own::R, dz_own_ave::R, st_own::TRMOwnState(p. E-32),
2                               cost_offline::Vector{R}, cost_online_ra::Vector{R},
3                               cost_online_ra_subset::Vector{R}, cost_online_ta::Vector{R} )
4   const N_actions::Z = params().actions.num_actions
5   const C_force_alert::R =
6     params().modes[mode_int].cost_estimation.online.force_alert.C_force_alert
7   const H_threshold_lower::R =
8     params().modes[mode_int].cost_estimation.online.force_alert.H_threshold_lower
9   const H_threshold_upper::R =
10    params().modes[mode_int].cost_estimation.online.force_alert.H_threshold_upper
11  const C_restrict::R           = params().threat_resolution.C_restrict
12  const C_differential_threshold::R = params().threat_resolution.C_differential_threshold
13  cost_ta::Vector{R}          = cost_offline + cost_online_ta
14  cost_ra::Vector{R}          = cost_offline + cost_online_ra
15  cost_ra_subset::Vector{R}   = cost_offline + cost_online_ra_subset
16  act_subset::Z              = MinCostIndex(p.340)( cost_ra_subset, C_differential_threshold )
17  act_subset_value::R        = cost_ra_subset[act_subset]
18  c_coc_threshold::R        = 0
19  if (H_threshold_lower < height_own)
20    if (H_threshold_upper < height_own)
21      c_coc_threshold = params().modes[mode_int].cost_estimation.online.force_alert.C_coc_threshold_
22      upper
23    else
24      c_coc_threshold = params().modes[mode_int].cost_estimation.online.force_alert.C_coc_threshold_
25      lower
26    end
27    ra_imminent::Bool = false
28    if (st_own.a_prev.action == COC) && (c_coc_threshold < cost_offline[COC])
29      ra_imminent = true
30    end
31    if (act_subset != COC) || ra_imminent
32      (dz_act_min::R, dz_act_max::R) =
33        ActionToRates(p.325)( act_subset, dz_own_ave, st_own.a_prev.action,
34                               st_own.a_prev.dz_min, st_own.a_prev.dz_max, st_own.a_prev.ddz )
35      if !IsPreventive(p.336)( dz_act_min, dz_act_max ) || ra_imminent
36        for act::Z in 1:N_actions
37          (dz_min::R, dz_max::R) =
38            ActionToRates(p.325)( act, dz_own_ave, st_own.a_prev.action,
39                               st_own.a_prev.dz_min, st_own.a_prev.dz_max, st_own.a_prev.ddz )
40          if IsPreventive(p.336)( dz_min, dz_max )
41            cost_ra[act] = cost_ra[act] + C_force_alert
42          end
43        end
44        cost_ra[COC] = cost_ra[COC] + C_force_alert
45      elseif (st_own.a_prev.action != COC) && (c_coc_threshold < act_subset_value)
46        cost_ra[COC] = cost_ra[COC] + C_force_alert
47      end
48      mtlo::Z = MTLOAction(p.340)()
49      cost_ra[mtlo] = cost_ra[mtlo] + C_restrict
50    return (cost_ra::Vector{R}, cost_ta::Vector{R})
51 end

```

Referenced In: StateAndCostEstimation(p. 161)
--

3.3 Action Selection

An overview of the ACTIONSELECTION component of the Threat Resolution Module is shown in Figure 3-7.

ACTIONSELECTION (Algorithm 218) determines the global action for own aircraft along with individual actions for each intruder considered in isolation. It accepts a set of action costs from each of the intruders as well as additional inputs, such as the intruder altitude samples and associated weights.

Action selection is handled differently for the cases of a single intruder and multiple intruders.

DETERMINEMINIMUMCOSTACTION (Algorithm 220) determines the global action and individual action when there is a single intruder.

DETERMINEMULTINTRUDERACTION (Algorithm 221) determines the global and individual actions when there are multiple intruders.

One or more intruders may have degraded surveillance that precludes them from being used to produce resolution advisories. In that case, the individual action for such an intruder is determined but is not considered when determining the global action. The *exclude_int* variable is used to indicate when an intruder should be excluded from consideration for the global action.

If no valid intruders are input into the TRM, **NOINTRUDERSACTION** is used to produce the appropriate action and vertical rate values in lieu of ACTIONSELECTION.

This algorithm takes as input *mode_int*, *cost_int*, *z_int_ave*, *dz_int_ave*, *z_own_ave*, *dz_own_ave*, *tau_int*, *equip_int*, *exclude_int*, *st_own*, and *st_int*. A description of the ACTIONSELECTION output variables appears in Table 3-7.

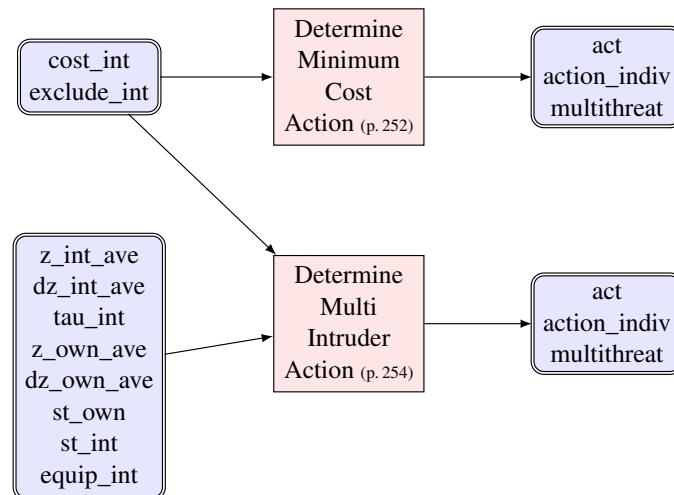


Figure 3-7. ActionSelection overview.

Table 3-7. TRM Algorithm Output Variables - ActionSelection

Variable	Units	Type	Description
act	N/A	integer	Single global resolution advisory for own aircraft
action_indiv	N/A	integer	Array of individual resolution advisories, one for each intruder
ddz	ft/s ²	real	Vertical acceleration compatible with global advisory
dz_max	ft/s	real	Maximum vertical rate compatible with global advisory
dz_min	ft/s	real	Minimum vertical rate compatible with global advisory
dz_own_ave_prev	ft/s	real	Estimated own vertical rate to be saved for next cycle
multithreat	N/A	bool	Whether or not there are multiple threats

Algorithm 218 ActionSelection

```

1 function ActionSelection( mode_int::Vector{Z}, cost_int::Matrix{R},
2                           z_int_ave::Vector{R}, dz_int_ave::Vector{R}, tau_int::Vector{R},
3                           z_own_ave::R, dz_own_ave::R, equip_int::Vector{Bool},
4                           exclude_int::Vector{Bool}, st_own::TRMOwnState(p.E-32),
5                           st_int::Vector{TRMIIntruderState(p.E-33)} )
6   const N_intruders::Z = length( z_int_ave )
7   dz_min::R = 0.0
8   dz_max::R = 0.0
9   ddz::R    = 0.0
10  act::Z    = COC
11  action_indiv::Vector{Z} = Z[]
12  multithreat::Bool = false
13  num_considered::Z = 0
14  for i = 1:N_intruders
15    if !exclude_int[i]
16      num_considered = num_considered + 1
17    end
18  end
19  if (num_considered <= 1)
20    (act, action_indiv, multithreat) =
21      DetermineMinimumCostAction(p.252)( cost_int, exclude_int )
22    st_own.st_multithreat_cost_balancing = MultithreatCostBalancingCState(p.E-45)()
23    st_own.st_arbitrate = ActionArbitrationGlobalCState(p.E-41)()
24  else
25    (act, action_indiv, multithreat) =
26      DetermineMultiIntruderAction(p.254)( mode_int, cost_int, z_int_ave, dz_int_ave, tau_int,
27                                              z_own_ave, dz_own_ave, st_own, st_int, equip_int,
28                                              exclude_int )
29  end
30  (dz_min, dz_max, ddz) = ActionToRates(p.325)( act, dz_own_ave, st_own.a_prev.action,
31                                              st_own.a_prev.dz_min, st_own.a_prev.dz_max, st_own.a_prev.ddz )
32  dz_own_ave_prev::R = dz_own_ave
33  return (act::Z, dz_min::R, dz_max::R, ddz::R, action_indiv::Vector{Z},
34          dz_own_ave_prev::R, multithreat::Bool)
35 end

```

Referenced In: VerticalTRMUpdate(p.152)

NOINTRUDERSACTION (Algorithm 219) is used when there are no valid intruders for TRM processing. It sets all necessary outputs and state variables to appropriate values for the condition when no intruders are valid.

This algorithm takes as input *dz_own_ave* and *st_own*. This algorithm returns *act*, *dz_min*, *dz_max*, *ddz*, and *multithreat*. The algorithm also updates *st_own::TRMOwnState*.

Algorithm 219 NoIntrudersAction

```

1 function NoIntrudersAction( dz_own_ave::R, st_own::TRMOwnState(p. E-32) )
2   dz_min::R = 0.0
3   dz_max::R = 0.0
4   ddz::R = 0.0
5   act::Z = COC
6   multithreat::Bool = false
7   (dz_min, dz_max, ddz) = ActionToRates(p. 325)( act, dz_own_ave, st_own.a_prev.action,
8                                         st_own.a_prev.dz_min, st_own.a_prev.dz_max, st_own.a_prev.ddz )
9   st_own.dz_ave_prev = dz_own_ave
10  st_own.action_prev = COC
11  st_own.word_prev = 0
12  st_own.crossing_prev = false
13  st_own.strength_prev = 0
14  st_own.ra_suppressed_prev = false
15  st_own.st_multithreat_cost_balancing = MultithreatCostBalancingCState(p. E-45)()
16  st_own.st_arbitrate = ActionArbitrationGlobalCState(p. E-41)()
17  return (act::Z, dz_min::R, dz_max::R, ddz::R, multithreat::Bool)
18 end

```

Referenced In: VerticalTRMUpdate(p. 152)

3.3.1 Single Intruder Rate Selection

For the single intruder case, determining the best action is performed by DETERMINEMINIMUM-COSTACTION (Algorithm 220), which determines the action with minimum cost for the single intruder that is valid as indicated by *exclude_int*. To ensure that the multi-threat level-off (MTLO) table entry is never used during single threat processing the MTLO action cost receives the penalty *C_restrict* prior to determining the minimum cost for each action. The minimum cost action is found through a call to MINCOSTINDEX (Algorithm 298). The global action corresponds to the individual action for the single valid intruder; the individual actions for invalid intruders are not used.

The *exclude_int* variable is used to indicate when an intruder, such as one with degraded surveillance, should be excluded as a threat and from consideration for the global action. There are no special logic considerations for designated intruders.

This algorithm takes as input *cost_int* and *exclude_int*. This algorithm returns *action*, *action_indiv*, and *multithreat*.

Algorithm 220 DetermineMinimumCostAction

```

1 function DetermineMinimumCostAction( cost_int::Matrix{R}, exclude_int::Vector{Bool} )
2   const C_restrict::R           = params().threat_resolution.C_restrict
3   const C_differential_threshold::R = params().threat_resolution.C_differential_threshold
4   const N_intruders::Z = length( exclude_int )
5   mtlo_action::Z = MTLOAction(p.340)()
6   action_indiv::Vector{Z} = zeros( Z, N_intruders )
7   action::Z = COC
8   for i = 1:N_intruders
9     cost_int[i, mtlo_action] = cost_int[i, mtlo_action] + C_restrict
10    action_indiv[i] = MinCostIndex(p.340)( ToVec(p.3-3)( cost_int[i,:] ), C_differential_threshold )
11    if !exclude_int[i]
12      action = action_indiv[i]
13    end
14  end
15  multithreat::Bool = false
16  return (action::Z, action_indiv::Vector{Z}, multithreat::Bool)
17 end

```

Referenced In: ActionSelection(p.250)

3.3.2 Multiple Intruder Rate Selection

When there are multiple valid intruders, DETERMINEMULTINTRUDERACTION (Algorithm 221) is used to determine the action to address all intruders. Because the effects of coordination between aircraft have a major role in multithreat encounters, action selection for multiple intruders treats intruders that are unequipped differently from those that are equipped. Since unequipped aircraft do not need to account for coordination effects, the unequipped intruders are treated together, independently from equipped intruders and the global action is then determined. In addition to selecting an action based on the individual actions, the DETERMINEMULTINTRUDERACTION logic applies some online costs that are specific to multithreat encounters and dependent on multiple aircraft.

In INDIVIDUALSELECTIONCOSTESTIMATION (Algorithm 222), the individual actions for each intruder are determined individually after applying additional online costs.

Once the individual intruder actions and the number of unequipped threat intruders have been determined, the global action is determined. First the best resolution advisory (RA) that addresses all unequipped threats is determined in UNEQUIPPEDCOSTFUSION (Algorithm 224). In MTLODETERMINATION (Algorithm 225), individual intruder actions are evaluated to determine if a 'Multi-Threat Level-Off' (MTLO) advisory is necessary due to conflicting up and down actions for different intruders. If an MTLO action is not selected, the global RA is determined by arbitration in ACTIONARBITRATION (Algorithm 230) to be the strongest up or down sense advisory. PERSISTMTLO (Algorithm 300) is then used to determine whether to persist an MTLO advisory rather than initiate a 'Do Not Climb' or 'Do Not Descend' advisory.

Finally, if a global MTLO advisory is continuing, the previous individual intruder advisories are persisted to maintain consistency. When the global RA is a continuing MTLO advisory and a previous individual intruder advisory is a maintain, the individual advisory is converted to a 'Do Not Climb' or 'Do Not Descend' advisory. That conversion properly preserves the individual advisory sense through a subsequent level-off. All other individual intruder advisories are persisted without change.

One or more intruders may have degraded surveillance that requires processing for traffic advisories only (TA-only). In that case, the individual action for each such intruder is determined but is not

considered when determining the global action. In a situation with multiple intruders, even if the individual action is something other than clear of conflict for the intruder undergoing degraded surveillance TA-only processing, that intruder will not be counted as a threat. In each of the algorithms mentioned above, the *exclude_int* variable is used to indicate when an intruder should be excluded as a threat and from consideration for the global action. There are no special logic considerations for intruders designated to an Xo mode.

This algorithm takes as input *mode_int*, *cost_min*, *z_int_ave*, *dz_int_ave*, *tau_int*, *z_own_ave*, *dz_own_ave*, *st_own*, *st_int*, *equip_int*, and *exclude_int*. This algorithm returns *action*, *action_indiv*, and *multithreat*.

Algorithm 221 DetermineMultiIntruderAction

```

1 function DetermineMultiIntruderAction( mode_int::Vector{Z}, cost_int::Matrix{R},
2                                         z_int_ave::Vector{R}, dz_int_ave::Vector{R}, tau_int::Vector{R},
3                                         z_own_ave::R, dz_own_ave::R,
4                                         st_own::TRMOwnState(p.E-32), st_int::Vector{TRMIntruderState(p.E-33)},
5                                         equip_int::Vector{Bool}, exclude_int::Vector{Bool} )
6   const R_initial_ddz::R = params().actions.initial_acceleration
7   const N_intruders::Z = length( st_int )
8   num_unequipped::Z = 0
9   for i = 1:N_intruders
10     if !equip_int[i] && !exclude_int[i]
11       num_unequipped = num_unequipped + 1
12     end
13   end
14   (act_indiv::Vector{Z}, cost_indiv::Vector{R}, num_threats::Z, num_unequipped_threats::Z) =
15     IndividualSelectionCostEstimation(p.255)( mode_int, cost_int, num_unequipped,
16                                               z_own_ave, dz_own_ave,
17                                               z_int_ave, dz_int_ave, tau_int,
18                                               equip_int, exclude_int )
19   multithreat::Bool = (1 < num_threats)
20   act_unequipped::Z = COC
21   if (0 < num_unequipped)
22     act_unequipped =
23       UnequippedCostFusion(p.258)( act_indiv, cost_int, num_unequipped_threats,
24                                     equip_int, exclude_int )
25   end
26   (issue_mtlo::Bool, arbitrate_conflicting_senses::Bool) =
27     MTLODetermination(p.259)( mode_int, act_indiv, cost_indiv, act_unequipped,
28                               multithreat, z_own_ave, dz_own_ave, st_own,
29                               z_int_ave, dz_int_ave, equip_int, exclude_int )
30   act::Z = COC
31   MTL0::Z = MTLOAction(p.340)()
32   if !issue_mtlo
33     act = ActionArbitration(p.264)( act_unequipped, act_indiv, cost_indiv,
34                                   dz_own_ave, arbitrate_conflicting_senses,
35                                   num_unequipped_threats, st_int, equip_int, exclude_int,
36                                   st_own.st_arbitrate )
37     if PersistMTLO(p.341)( act, multithreat, dz_own_ave, st_own.dz_ave_prev, st_own.a_prev )
38       act = MTL0
39       st_own.st_arbitrate = ActionArbitrationGlobalCState(p.E-41)()
40     end
41   else
42     act = MTL0
43     st_own.st_arbitrate = ActionArbitrationGlobalCState(p.E-41)()
44   end
45   if (MTL0 == act) && (MTL0 == st_own.a_prev.action)
46     maintain_action::Z = RatesToAction(p.342)( NaN, NaN, NaN )
47     for i = 1:N_intruders
48       if (maintain_action == st_int[i].a_prev.action)
49         if (0 <= dz_own_ave)
50           act_indiv[i] = RatesToAction(p.342)( 0.0, Inf, R_initial_ddz )
51         else
52           act_indiv[i] = RatesToAction(p.342)( -Inf, 0.0, R_initial_ddz )
53         end
54       else
55         act_indiv[i] = st_int[i].a_prev.action
56       end
57     end
58   end
59   return (act::Z, act_indiv::Vector{Z}, multithreat::Bool)
60 end

```

Referenced In: ActionSelection(p.250)

INDIVIDUALSELECTIONCOSTESTIMATION (Algorithm 222) determines the individual actions for each intruder individually after applying two additional online costs.

The first is a penalty to the MTLO action if a MTLO is not forced due to coordination.

The second is a penalty to prevent resolution advisories that may lead to an encounter where own aircraft is sandwiched between intruder aircraft. That is, one intruder aircraft is above own aircraft at the same time another intruder aircraft is below. At least one of the intruders must be unequipped for this penalty to be applied. SANDWICHPREVENTIONCOST (Algorithm 223) is used to determine to what actions the sandwich penalty should apply.

This algorithm takes as input *mode_int*, *cost_min*, *num_unequipped*, *z_own_ave*, *dz_own_ave*, *z_int_ave*, *dz_int_ave*, *tau_int*, *equip_int*, and *exclude_int*. This algorithm returns *action_indiv*, *cost_indiv*, *num_threats*, and *num_unequipped_threats*.

Algorithm 222 IndividualSelectionCostEstimation

```

1 function IndividualSelectionCostEstimation( mode_int::Vector{Z}, cost_int::Matrix{R}, num_unequipped::Z,
2                                     z_own_ave::R, dz_own_ave::R,
3                                     z_int_ave::Vector{R}, dz_int_ave::Vector{R}, tau_int::Vector{R},
4                                     equip_int::Vector{Bool}, exclude_int::Vector{Bool} )
5   const C_restrict::R = params().threat_resolution.C_restrict
6   const C_differential_threshold::R = params().threat_resolution.C_differential_threshold
7   const N_intruders::Z = length( z_int_ave )
8   act_indiv::Vector{Z} = zeros( Z, N_intruders )
9   cost_indiv::Vector{R} = zeros( R, N_intruders )
10  mtlo_action::Z = MTLOAction(p.340)()
11  num_threats::Z = 0
12  num_unequipped_threats::Z = 0
13  cost_coc::Vector{R} = ToVec(p.1-3)( cost_int[:,COC] )
14  for i = 1:N_intruders
15    cost_int[i, mtlo_action] = cost_int[i, mtlo_action] + C_restrict
16    has_unequipped_secondary::Bool = false
17    if !equip_int[i]
18      has_unequipped_secondary = (1 < num_unequipped)
19    else
20      has_unequipped_secondary = (0 < num_unequipped)
21    end
22    if has_unequipped_secondary
23      inc_cost::Vector{R} =
24        SandwichPreventionCost(p.257)( i, mode_int[i], cost_coc, z_own_ave, dz_own_ave,
25                                       z_int_ave, dz_int_ave, tau_int,
26                                       equip_int, exclude_int )
27      cost_int[i,:] = cost_int[i,:].+ reshape( inc_cost[:,], (1, length( inc_cost )) )
28    end
29    act_indiv[i] = MinCostIndex(p.340)( ToVec(p.1-3)( cost_int[i,:] ), C_differential_threshold )
30    cost_indiv[i] = cost_int[i,act_indiv[i]]
31    if (act_indiv[i] != COC) && !exclude_int[i]
32      num_threats = num_threats + 1
33      if !equip_int[i]
34        num_unequipped_threats = num_unequipped_threats + 1
35      end
36    end
37  end
38  return (act_indiv::Vector{Z}, cost_indiv::Vector{R}, num_threats::Z, num_unequipped_threats::Z)
39 end
```

Referenced In: DetermineMultiIntruderAction(p.254)

3.3.2.1 Sandwich Prevention Cost for Unequipped Intruders

SANDWICHPREVENTIONCOST (Algorithm 223) helps to discourage actions towards a primary intruder when a secondary intruder will be in the same direction at the closest point of approach (CPA). To prevent unnecessary application, the cost is only applied when the risk that the secondary intruder poses, indicated by the clear of conflict (COC) cost of the secondary intruder (*cost_coc*), is above a threshold (*C_sandwich_prevention_coc_threshold*). The projected altitude of the primary intruder at CPA (*z_int_idx_at_tau*) and the projected altitude of each secondary intruder at CPA (*z_int_j_at_tau*) are used over current altitudes to account for the possibility that the primary and secondary intruder cross altitudes before CPA. Separate checks are made depending on the current location of the primary intruder. If the primary intruder is below the ownship and the secondary intruder will be below the primary intruder at CPA, the *penalize_downsense* variable is set to TRUE. If the primary intruder is above the ownship and the secondary intruder will be above the primary intruder at CPA, the *penalize_upsense* variable is set to TRUE.

If only one of the penalize flags (*penalize_downsense*, *penalize_upsense*) is TRUE, the corresponding sense RAs will be penalized by *C_sandwich*. Equipped and unequipped aircraft are penalized by different amounts due to the effects of coordination costs and other equipage dependent costs. Both preventive and maintain advisories are excluded from the penalty. If both penalize flags are TRUE (or FALSE) no penalty will be applied. A vector containing the costs to be applied to the actions that should be penalized is updated and returned.

The *exclude_int* variable is used to omit intruders with degraded surveillance from the checks.

This algorithm takes as input *intruder_idx*, *mode_int*, *cost_coc*, *z_own_ave*, *dz_own_ave*, *z_int_ave*, *dz_int_ave*, *tau_int*, *equip_int*, and *exclude_int*. This algorithm returns *cost*.

Algorithm 223 SandwichPreventionCost

```

1 function SandwichPreventionCost( intruder_idx::Z, mode_int::Z, cost_coc::Vector{R},
2                               z_own_ave::R, dz_own_ave::R, z_int_ave::Vector{R}, dz_int_ave::Vector{R},
3                               tau_int::Vector{R}, equip_int::Vector{Bool}, exclude_int::Vector{Bool} )
4   const N_actions::Z      = params().actions.num_actions
5   const C_coc_threshold::R = params().modes[mode_int].cost_estimation.online.sandwich_prevention.C_coc_
6     threshold
7   c_sandwich::R = params().modes[mode_int].cost_estimation.online.sandwich_prevention.C_sandwich_unequip
8   if equip_int[intruder_idx]
9     c_sandwich = params().modes[mode_int].cost_estimation.online.sandwich_prevention.C_sandwich_equip
10   end
11   cost::Vector{R} = zeros( R, N_actions )
12   z_int_idx_at_tau::R = z_int_ave[intruder_idx] + (dz_int_ave[intruder_idx] * tau_int[intruder_idx])
13   penalize_upsense::Bool = false
14   penalize_downsense::Bool = false
15   for j = 1:length( cost_coc )
16     if (j != intruder_idx) && !equip_int[j] && (C_coc_threshold < cost_coc[j]) && !exclude_int[j]
17       z_int_j_at_tau::R = z_int_ave[j] + (dz_int_ave[j] * tau_int[intruder_idx])
18       if (z_int_j_at_tau < z_int_idx_at_tau) && (z_int_ave[j] <= z_own_ave)
19         penalize_downsense = true
20       elseif (z_int_idx_at_tau <= z_int_j_at_tau) && (z_own_ave < z_int_ave[j])
21         penalize_upsense = true
22       end
23     end
24   if (penalize_upsense && !penalize_downsense) || (!penalize_upsense && penalize_downsense)
25     for act = 1:N_actions
26       (dz_min::R, dz_max::R) = ActionToRates(p.325)( act, dz_own_ave, -1, NaN, NaN, NaN )
27       sense_own::Symbol = RatesToSense(p.342)( dz_min, dz_max )
28       if !IsPreventive(p.336)( dz_min, dz_max ) && !IsMaintain(p.335)( dz_min, dz_max ) &&
29         ((penalize_upsense && (:Up == sense_own)) || (penalize_downsense && (:Down == sense_own)))
30         cost[act] = c_sandwich
31     end
32   end
33 end
34 return cost::Vector{R}
35 end

```

Referenced In: IndividualSelectionCostEstimation(p. 255)

3.3.2.2 Cost Fusion for Unequipped Intruders

UNEQUIPPEDCOSTFUSION (Algorithm 224) determines the optimal action that attempts to address all unequipped intruders. The individual actions are first found by taking the lowest cost action for each unequipped intruder. An intruder is a threat if the optimal action toward that intruder in isolation is not clear of conflict (COC).

The costs from the individual unequipped intruders are fused by taking the worst cost (maximum) over all unequipped intruders for every action (q_{star}) to give each element of $fused_cost$. If there is more than one unequipped threat, Clear of Conflict is penalized by $C_restrict$ to ensure an alert is always issued. To prevent the selection of a multi-threat level-off (MTLO) advisory during cost fusion, a penalty ($C_restrict$) is applied to the MTLO for all intruders. Although this prevents the MTLO advisory from being selected during the fusion process, MTLODETERMINATION (Algorithm 225) can override this effect. The single action for all unequipped intruders is determined by finding the best action over the new $fused_cost$ vector through a call to MINCOSTINDEX (Algorithm 298).

The $exclude_int$ variable is used to omit intruders with degraded surveillance from the checks.

This algorithm takes as input *act_indiv*, *cost_int*, *num_unequipped_threats*, *equip_int*, and *exclude_int*. This algorithm returns *action*.

Algorithm 224 UnequippedCostFusion

```

1 function UnequippedCostFusion( act_indiv::Vector{Z}, cost_int::Matrix{R}, num_unequipped_threats::Z,
2                               equip_int::Vector{Bool}, exclude_int::Vector{Bool} )
3   const C_restrict::R           = params().threat_resolution.C_restrict
4   const C_differential_threshold::R = params().threat_resolution.C_differential_threshold
5   const N_actions::Z          = params().actions.num_actions
6   mtlo_action::Z = MTLOAction(p.340)
7   fused_cost::Vector{R} = zeros( R, N_actions )
8   for act = 1:N_actions
9     q_star::R = -Inf
10    for j = 1:length( act_indiv )
11      q::R = cost_int[j,act]
12      if !equip_int[j] && ((0 == num_unequipped_threats) || (act_indiv[j] != COC)) && !exclude_int[j]
13        q_star = max( q_star, q )
14      end
15    end
16    fused_cost[act] = q_star
17  end
18  if (1 < num_unequipped_threats)
19    fused_cost[COC] = fused_cost[COC] + C_restrict
20  end
21  fused_cost[mtlo_action] = fused_cost[mtlo_action] + C_restrict
22  action::Z = MinCostIndex(p.340)( fused_cost, C_differential_threshold )
23  return (action::Z)
24 end
```

Referenced In: DetermineMultiIntruderAction(p.254)

3.3.2.3 Multi Threat Level Off (MTLO) Determination

MTLODETERMINATION (Algorithm 225) determines if a multi-threat level-off (MTLO) can or should be issued under the current conditions. The issuance of the MTLO advisory is limited in a number of situations where the issuance can lead to undesirable behaviors.

If PERSISTMTLO (Algorithm 300) determines that a MTLO action should continue from the previous cycle, the *issue_mtlo* flag is set to TRUE.

DETERMINESENSES (Algorithm 226) provides the sense of the action the logic has selected for each intruder independently for use in subsequent algorithms. MULTITHREATCOSTBALANCING (Algorithm 229) sets *costs Allows Mtlo* to TRUE if it determines the individual costs for each intruder would support the selection of MTLO. While the conclusion reached in PERSISTMTLO can make the output of MULTITHREATCOSTBALANCING irrelevant, it must be called on each cycle to keep the values in *st_own.st_multithreat_cost_balancing* up to date.

Two other checks are performed to determine if MTLO should be issued. DETERMINECONFLICTINGSENSES (Algorithm 227) sets *conflicting_senses* to TRUE if the senses selected for individual intruders conflict in sense. ALLOWUNEQUIPPEDMTLO (Algorithm 228) sets *geometry Allows Mtlo* to TRUE if the geometry of unequipped intruders and the presence of equipped intruders would support the selection of MTLO. The three flags (*costs Allows Mtlo*, *conflicting_senses*, and *geometry Allows Mtlo*) must all be TRUE for *issue_mtlo* to be TRUE. Additionally, *arbitrate_conflicting_senses* is set as a direct copy of *conflicting_senses* for use in ACTIONARBITRATION (Algorithm 230).

The *exclude_int* variable is used to omit intruders with degraded surveillance from the checks.

This algorithm takes as input *mode_int*, *act_indiv*, *cost_indiv*, *act_unequipped*, *multithreat*, *z_own_ave*, *dz_own_ave*, *st_own*, *z_int_ave*, *dz_int_ave*, *equip_int*, and *exclude_int*. This algorithm returns *issue_mtlo* and *arbitrate_conflicting_senses*.

Algorithm 225 MTLODetermination

```

1 function MTLODetermination( mode_int::Vector{Z}, act_indiv::Vector{Z}, cost_indiv::Vector{R},
2                               act_unequipped::Z, multithreat::Bool,
3                               z_own_ave::R, dz_own_ave::R, st_own::TRMOwnState(p.E-32),
4                               z_int_ave::Vector{R}, dz_int_ave::Vector{R},
5                               equip_int::Vector{Bool}, exclude_int::Vector{Bool} )
6   issue_mtlo::Bool = false
7   arbitrate_conflicting_senses::Bool = false
8   costs_allow_mtlo::Bool = true
9   if PersistMTLO(p.341)( act_unequipped, multithreat, dz_own_ave, dz_own_ave, st_own.a_prev )
10    issue_mtlo = true
11  end
12  senses_own_indiv::Vector{Symbol} = DetermineSenses(p.260)( act_indiv, dz_own_ave, st_own.a_prev )
13  costs_allow_mtlo = MultithreatCostBalancing(p.263)( act_indiv, cost_indiv, senses_own_indiv, exclude_int,
14                                                    st_own.st_multithreat_cost_balancing )
15  if !issue_mtlo
16    conflicting_senses::Bool = DetermineConflictingSenses(p.260)( senses_own_indiv, exclude_int )
17    geometryAllows_mtlo::Bool = AllowUnequippedMTLO(p.261)( mode_int, z_own_ave, dz_own_ave,
18                                                               z_int_ave, dz_int_ave, senses_own_indiv,
19                                                               equip_int, exclude_int )
20    arbitrate_conflicting_senses = conflicting_senses
21    issue_mtlo = (conflicting_senses && geometryAllows_mtlo && costs_allow_mtlo)
22  end
23  return (issue_mtlo::Bool, arbitrate_conflicting_senses::Bool)
24 end
```

Referenced In: DetermineMultiIntruderAction(p.254)

DETERMINESENSES (Algorithm 226) provides the sense of each action in the input array. Each element of *action_indiv* contains an action index that can be used to look up the vertical rate limits (*dz_min*, *dz_max*) of each action. An explicit check for *NaN* is required to identify Maintain RAs. If the Maintain RA was also selected on the previous cycle, the vertical rate limits from the previous cycle are used. Otherwise, a call to MAINTAINRATES (Algorithm 297) sets the vertical rate limits appropriately.

The sense of each action is determined by invoking RATESTOSENSE (Algorithm 302) with the vertical rate limits (*dz_min* and *dz_max*).

This algorithm takes as input *act_indiv*, *dz_own_ave*, and *a_global*. This algorithm returns *senses_own_indiv*.

Algorithm 226 DetermineSenses

```

1 function DetermineSenses( action_intriv::Vector{Z}, dz_own_ave::R, a_global::GlobalAdvisory(p.E-35 ) )
2   const R_dz_min_adv::Vector{R} = params().actions.min_rates
3   const R_dz_max_adv::Vector{R} = params().actions.max_rates
4   const N_intruders::Z = length( action_intriv )
5   senses_own_intriv::Vector{Symbol} = Array( Symbol, N_intruders )
6   for i = 1:N_intruders
7     dz_min::R = 0.0
8     dz_max::R = 0.0
9     act::Z = action_intriv[i]
10    if isnan( R_dz_min_adv[act] ) || isnan( R_dz_max_adv[act] )
11      if (a_global.action == act)
12        dz_min = a_global.dz_min
13        dz_max = a_global.dz_max
14      else
15        (dz_min, dz_max) = MaintainRates(p.339)( dz_own_ave )
16      end
17    else
18      dz_min = R_dz_min_adv[act]
19      dz_max = R_dz_max_adv[act]
20    end
21    senses_own_intriv[i] = RatesToSense(p.342)( dz_min, dz_max )
22  end
23  return senses_own_intriv::Vector{Symbol}
24 end
```

Referenced In: MTLODetermination(*p.259*)

DETERMINECONFLICTINGSENSES (Algorithm 227) is used to determine if the actions selected for individual intruders conflict in sense. This is done by looping over the sense of the action selected for each intruder by ownership and setting the appropriate flag if the sense is up (*up_sense_issued*) or down (*down_sense_issued*). If both up and down sense RAs have been selected across the set of threats, *conflicting_senses* is set to TRUE.

The *exclude_int* variable is used to omit intruders with degraded surveillance from the checks.

This algorithm takes as input *senses_own_intriv* and *exclude_int*. This algorithm returns *conflicting_senses*.

Algorithm 227 DetermineConflictingSenses

```

1 function DetermineConflictingSenses( senses_own_intriv::Vector{Symbol}, exclude_int::Vector{Bool} )
2   up_sense_issued::Bool = false
3   down_sense_issued::Bool = false
4   for i = 1:length( senses_own_intriv )
5     if !exclude_int[i]
6       if (senses_own_intriv[i] == :Up)
7         up_sense_issued = true
8       elseif (senses_own_intriv[i] == :Down)
9         down_sense_issued = true
10      end
11    end
12  end
13  conflicting_senses::Bool = up_sense_issued && down_sense_issued
14  return conflicting_senses::Bool
15 end
```

Referenced In: MTLODetermination(*p.259*)

ALLOWUNEQUIPPEDMTLO (Algorithm 228) determines if the geometry of unequipped intruders or the presence of an equipped intruder would allow the selection of MTLO. The thresholds used in the geometric considerations ($H_{threshold}$ and $R_{threshold}$) are mode dependent and can be different for each intruder. Any intruder that did not have an action selected during single-threat processing is omitted from the checks. If any intruder is equipped, ALLOWUNEQUIPPEDMTLO will always consider MTLO to be an allowable action. The remaining intruders are subjected to the geometric tests. In order for MTLO to be permitted, an intruder being checked must have more than $H_{threshold}$ vertical separation or be diverging vertical at a rate greater than $R_{threshold}$. If any intruder does not meet these requirements, *allow_mtlo* is set to FALSE.

The *exclude_int* variable is used to omit intruders with degraded surveillance from the checks.

This algorithm takes as input *mode_int*, *z_own_ave*, *dz_own_ave*, *z_int_ave*, *dz_int_ave*, *senses_own_indiv*, *equip_int*, and *exclude_int*. This algorithm returns *allow_mtlo*.

Algorithm 228 AllowUnequippedMTLO

```

1 function AllowUnequippedMTLO( mode_int::Vector{Z}, z_own_ave::R, dz_own_ave::R,
2                               z_int_ave::Vector{R}, dz_int_ave::Vector{R},
3                               senses_own_indiv::Vector{Symbol}, equip_int::Vector{Bool},
4                               exclude_int::Vector{Bool} )
5   const N_intruders::Z = length( mode_int )
6   H_threshold::Vector{R} = zeros( R, N_intruders )
7   R_threshold::Vector{R} = zeros( R, N_intruders )
8   for i = 1:N_intruders
9     H_threshold[i] = params().modes[mode_int[i]].cost_estimation.online.unequipped_mtlo.H_threshold
10    R_threshold[i] = params().modes[mode_int[i]].cost_estimation.online.unequipped_mtlo.R_threshold
11  end
12  allow_mtlo::Bool = true
13  equipped_intruder::Bool = false
14  for i = 1:N_intruders
15    if (:None != senses_own_indiv[i]) && !exclude_int[i]
16      z_rel::R = z_own_ave - z_int_ave[i]
17      dz_rel::R = dz_own_ave - dz_int_ave[i]
18      if equip_int[i]
19        equipped_intruder = true
20      elseif (abs( z_rel ) < H_threshold[i])
21        if (z_rel < 0) && (-R_threshold[i] < dz_rel)
22          allow_mtlo = false
23        elseif (0 < z_rel) && (dz_rel < R_threshold[i])
24          allow_mtlo = false
25        end
26      end
27    end
28  end
29  if equipped_intruder
30    allow_mtlo = true
31  end
32  return allow_mtlo::Bool
33 end
```

Referenced In: MTLODetermination(p. 259)
--

3.3.2.4 Balancing Intruder Costs

MULTITHREATCOSTBALANCING (Algorithm 229) compares costs between intruders and allows multi-threat level-off (MTLO) as a global action if the threat level of each intruder is within a tolerance.

Because the cost values for each action can be interpreted to represent the relative threat of an intruder, comparing the costs for the actions selected for each individual intruder allows a rough determination of the relative threat for each intruder. When there are multiple actions selected that have large differences in costs and the selected actions have conflicting senses, it is an indicator that the MTLO that would be issued may not adequately reduce risk for one of the threats. In this undesirable situation, the MULTITHREATCOSTBALANCING algorithm compares costs between intruders and allows the prohibition of MTLO as a global action until the threat level of each intruder, as indicated by the individual action cost, is within a tolerance. The tolerance is initially given by $C_{differential_threshold_lo}$ but when the costs were not balanced previously, $C_{differential_threshold_hi}$ adds hysteresis to the tolerance. When the costs of both intruders are larger than $C_{absolute_threshold_hi}$, the up and down sense actions have become critical. If both up and down sense actions are becoming critical, MTLO is not allowed even if the costs are balanced. Once this condition arises, $C_{absolute_threshold_lo}$ is used for hysteresis.

The *exclude_int* variable is used to indicate when an intruder should be excluded as a threat and from consideration for the global action.

This algorithm takes as input *act_indiv*, *cost_indiv*, *senses_own_indiv*, *exclude_int*, and *s_c*. This algorithm returns *cost_allow_mtlo*. This algorithm updates *s_c*::*MultithreatCostBalancingCState*.

Algorithm 229 MultithreatCostBalancing

```

1 function MultithreatCostBalancing( act_indiv::Vector{Z}, cost_indiv::Vector{R},
2                                     senses_own_indiv::Vector{Symbol}, exclude_int::Vector{Bool},
3                                     s_c::MultithreatCostBalancingCState(p_E-45) )
4     const C_differential_threshold_hi::R =
5         params().threat_resolution.balance_costs.C_differential_threshold_hi
6     const C_differential_threshold_lo::R =
7         params().threat_resolution.balance_costs.C_differential_threshold_lo
8     const C_absolute_threshold_hi::R =
9         params().threat_resolution.balance_costs.C_absolute_threshold_hi
10    const C_absolute_threshold_lo::R =
11        params().threat_resolution.balance_costs.C_absolute_threshold_lo
12    const N_intruders::Z = length( act_indiv )
13    costs_allow_mtlo::Bool = true
14    diff_max::R = 0.0
15    diff_set::Bool = false
16    above_abs_threshold_hi::Bool = false
17    above_abs_threshold_lo::Bool = false
18    for i = 1:(N_intruders-1)
19        if (COC != act_indiv[i]) && !exclude_int[i]
20            for j = (i+1):N_intruders
21                if (COC != act_indiv[j]) && (senses_own_indiv[j] != senses_own_indiv[i]) && !exclude_int[j]
22                    diff_max = max( diff_max, abs( cost_indiv[i] - cost_indiv[j] ) )
23                    diff_set = true
24                    if (C_absolute_threshold_hi < abs( cost_indiv[i] )) &&
25                        (C_absolute_threshold_hi < abs( cost_indiv[j] ))
26                        above_abs_threshold_hi = true
27                    end
28                    if (C_absolute_threshold_lo < abs( cost_indiv[i] )) &&
29                        (C_absolute_threshold_lo < abs( cost_indiv[j] ))
30                        above_abs_threshold_lo = true
31                    end
32                end
33            end
34        end
35        if above_abs_threshold_hi && !s_c.above_abs_threshold_prev
36            costs_allow_mtlo = false
37            s_c.costs_allow_mtlo_prev = true
38            s_c.above_abs_threshold_prev = true
39        elseif above_abs_threshold_lo && s_c.above_abs_threshold_prev
40            costs_allow_mtlo = false
41            s_c.costs_allow_mtlo_prev = true
42            s_c.above_abs_threshold_prev = true
43        elseif diff_set
44            if s_c.costs_allow_mtlo_prev && (C_differential_threshold_hi <= diff_max)
45                costs_allow_mtlo = false
46            elseif !s_c.costs_allow_mtlo_prev && (C_differential_threshold_lo <= diff_max)
47                costs_allow_mtlo = false
48            else
49                costs_allow_mtlo = true
50            end
51            s_c.costs_allow_mtlo_prev = costs_allow_mtlo
52            s_c.above_abs_threshold_prev = false
53        else
54            costs_allow_mtlo = true
55            s_c.costs_allow_mtlo_prev = costs_allow_mtlo
56            s_c.above_abs_threshold_prev = false
57        end
58    end
59    return costs_allow_mtlo::Bool
60 end

```

Referenced In: MTLODetermination(p.259)
--

3.3.2.5 Action Arbitration

ACTIONARBITRATION (Algorithm 230) is utilized to select the global action when a multi-threat level-off (MTLO) advisory is not issued and there are multiple threats.

ARBITRATECONFLICTINGSENSES (Algorithm 231) is used for the special cases where MTLO was prohibited despite the presence of multiple individual actions with conflicting senses. The input flag (*arbitrate_conflicting_senses*) that enables this processing is determined by MTLODETERMINATION (Algorithm 225) through a call to DETERMINECONFLICTINGSENSES (Algorithm 227).

ARBITRATEMATCHINGSENSES (Algorithm 232) is used when there are multiple actions that have a common sense. In this case, the online cost state variables associated with ARBITRATECONFLICTINGSENSES in ACTIONARBITRATIONGLOBALCSTATE must be reset.

The *exclude_int* variable is used to indicate when an intruder should be excluded as a threat and from consideration for the global action.

This algorithm takes as input *act_unequipped*, *act_indiv*, *cost_indiv*, *dz_own_ave*, *arbitrate_conflicting_senses*, *num_unequipped_threats*, *st_int*, *equip_int*, *exclude_int*, and *s_c*. This algorithm returns *act*. This algorithm updates *s_c::ActionArbitrationGlobalCState*.

Algorithm 230 ActionArbitration

```

1 function ActionArbitration( act_unequipped::Z, act_indiv::Vector{Z}, cost_indiv::Vector{R}, dz_own_ave::R,
2           arbitrate_conflicting_senses::Bool, num_unequipped_threats::Z,
3           st_int::Vector{TRMIIntruderState(p.E-33)}, equip_int::Vector{Bool}, exclude_int::Vector{Bool}
4           ),
5           s_c::ActionArbitrationGlobalCState(p.E-41) )
6
7   act::Z = COC
8   if arbitrate_conflicting_senses
9     act = ArbitrateConflictingSenses(p.265)( act_indiv, cost_indiv, st_int, exclude_int, s_c )
10  else
11    s_c.mtlo_prohibited_prev = false
12    s_c.worst_case_cost_prev = -Inf
13    s_c.worst_case_action_prev = COC
14    act = ArbitrateMatchingSenses(p.267)( act_unequipped, act_indiv, dz_own_ave, num_unequipped_threats,
15                                         equip_int, st_int, exclude_int )
16  end
17  return (act::Z)
18 end

```

Referenced In: DetermineMultiIntruderAction(p.254)

ARBITRATECONFLICTINGSENSES (Algorithm 231) is utilized to select the global action when a multi-threat level-off (MTLO) advisory is not issued and there are multiple threats. In this case the actions selected for each of the multiple threats have conflicting senses.

For the special cases where MTLO was prohibited despite the presence of multiple individual actions with conflicting senses, action arbitration amounts to selecting the action that has the highest cost (indicating that the corresponding intruder is the highest threat).

In order to keep the action selection from changing every cycle, there must be a compelling reason to change from the previous action. A new worst case is selected if the new worst case cost is above *C_differential_worst_case_threshold*. The possibility of a change in action is allowed only if conflicting senses were arbitrated on the last cycle and a different intruder has the worst case cost this cycle.

There are two online cost state data structures associated with this cost. ACTIONARBITRATION-GLOBALCSTATE stores information about the previous worst case action. ACTIONARBITRATIONCSTATE stores state information for the individual intruder in *st_int*. Because the list of intruders can change from one cycle to the next, the intruder with the worst case action for this cycle is flagged using *st_int.st_arbitrate.was_worst_case*.

The *exclude_int* variable is used to indicate when an intruder should be excluded as a threat and from consideration for the global action.

This algorithm takes as input *act_indiv*, *cost_indiv*, *st_int*, *exclude_int*, and *s_c*. This algorithm returns *act*. This algorithm updates *s_c::ActionArbitrationGlobalCState*.

Algorithm 231 ArbitrateConflictingSenses

```

1 function ArbitrateConflictingSenses( act_indiv::Vector{Z}, cost_indiv::Vector{R},
2                                     st_int::Vector{TRMIIntruderState(p.E-33)}, exclude_int::Vector{Bool},
3                                     s_c::ActionArbitrationGlobalCState(p.E-41) )
4   const C_differential_threshold::R =
5     params().threat_resolution.C_differential_threshold
6   const C_differential_worst_case_threshold::R =
7     params().threat_resolution.action_arbitration.C_differential_worst_case_threshold
8   act::Z = COC
9   worst_case_cost::R      = -Inf
10  worst_case_idx::Z       = 0
11  worst_case_idx_prev::Z = 0
12  for i = 1:length( cost_indiv )
13    if !exclude_int[i]
14      if (worst_case_cost < (cost_indiv[i] - C_differential_threshold))
15        act           = act_indiv[i]
16        worst_case_cost = cost_indiv[i]
17        worst_case_idx = i
18    end
19    if s_c.mtlo_prohibited_prev && st_int[i].st_arbitrate.was_worst_case
20      worst_case_idx_prev = i
21    end
22  end
23  st_int[i].st_arbitrate.was_worst_case = false
24 end
25 if (0 == worst_case_idx_prev)
26   s_c.mtlo_prohibited_prev = false
27 end
28 if s_c.mtlo_prohibited_prev && (worst_case_idx != worst_case_idx_prev)
29   if (worst_case_cost <= (s_c.worst_case_cost_prev + C_differential_worst_case_threshold)) ||
30     (worst_case_cost <= (cost_indiv[worst_case_idx_prev] + C_differential_worst_case_threshold))
31   worst_case_idx = worst_case_idx_prev
32   act = s_c.worst_case_action_prev
33 end
34 end
35 if (0 < worst_case_idx)
36   st_int[worst_case_idx].st_arbitrate.was_worst_case = true
37   s_c.mtlo_prohibited_prev = true
38   s_c.worst_case_cost_prev = cost_indiv[worst_case_idx]
39   s_c.worst_case_action_prev = act
40 else
41   s_c.mtlo_prohibited_prev = false
42   s_c.worst_case_cost_prev = -Inf
43   s_c.worst_case_action_prev = COC
44 end
45 return (act::Z)
46 end

```

Referenced In: ActionArbitration(p.264)
--

ARBITRATEMATCHINGSENSES (Algorithm 232) is utilized to select the global action when a multi-threat level-off (MTLO) advisory is not issued and there are multiple threats. In this case the actions selected for each of the multiple threats have a common sense. This algorithm selects the strongest individual action being considered as the global action. As an example, if the best actions for two intruders are Climb 1500 and Climb 2500, respectively, the global action will be selected to be Climb 2500. In the case where two different actions have the same vertical rate limits (dz_{min} and dz_{max}), the acceleration (ddz) is used as a tie-breaker. For example, initial Climb 1500 and subsequent Climb 1500 have the same vertical rate limits, but different accelerations. The subsequent Climb 1500 has a higher acceleration and is selected for the global action.

The initial best action (*act*) is set as Clear of Conflict. If any unequipped threats exists, the un-equipped action is used as the initial *act*.

The vertical rate limits associated with the action for each valid, equipped intruder are determined with a call to ACTIONTORATES (Algorithm 273). The vertical rate limits of the new action are compared to the current best action. If the vertical rate limits of the new action are more restrictive (i.e., permit a smaller range of vertical rates) than the current best action, the best action is replaced with the new action.

The ACTIONARBITRATIONCSTATE online cost state structure (*st_int.st_arbitrate*) stores advisory information for the individual intruder. It is updated after each comparison to ensure consistent vertical rate limits are used for Maintain actions.

The *exclude_int* variable is used to indicate when an intruder should be excluded as a threat and from consideration for the global action.

This algorithm takes as input *act_unequipped*, *act_indiv*, *dz_own_ave*, *num_unequipped_threats*, *equip_int*, *st_int*, and *exclude_int*. This algorithm returns *act*. The algorithm outputs updates to *st_int::Vector<TRMIntruderState>*.

Algorithm 232 ArbitrateMatchingSenses

```

1 function ArbitrateMatchingSenses( act_unequipped::Z, act_indiv::Vector{Z}, dz_own_ave::R,
2                               num_unequipped_threats::Z, equip_int::Vector{Bool},
3                               st_int::Vector{TRMIIntruderState(p.E-33)}, exclude_int::Vector{Bool} )
4   act::Z = COC
5   if (0 < num_unequipped_threats)
6     act = act_unequipped
7   end
8   (dz_min_best::R, dz_max_best::R, ddz_best::R) = ActionToRates(p.325)( act, dz_own_ave, -1, NaN, NaN, NaN )
9   for i = 1:length( st_int )
10    if equip_int[i] && !exclude_int[i]
11      (dz_min::R, dz_max::R, ddz::R) = ActionToRates(p.325)( act_indiv[i], dz_own_ave,
12                                              st_int[i].st_arbitrate.action,
13                                              st_int[i].st_arbitrate.dz_min,
14                                              st_int[i].st_arbitrate.dz_max,
15                                              st_int[i].st_arbitrate.ddz )
16      if (dz_min_best < dz_min) || (dz_max < dz_max_best) ||
17        ((dz_min_best == dz_min) && (dz_max == dz_max_best) && (ddz_best < ddz))
18        dz_min_best = dz_min
19        dz_max_best = dz_max
20        ddz_best = ddz
21        act = act_indiv[i]
22      end
23      if (act_indiv[i] != st_int[i].st_arbitrate.action)
24        st_int[i].st_arbitrate.action = act_indiv[i]
25        (st_int[i].st_arbitrate.dz_min, st_int[i].st_arbitrate.dz_max,
26         st_int[i].st_arbitrate.ddz) =
27          ActionToRates(p.325)( act_indiv[i], dz_own_ave, -1, NaN, NaN, NaN )
28      end
29    end
30  end
31  return (act::Z)
32 end

```

Referenced In: ActionArbitration(p.264)

3.4 Coordination Selection

An overview of the COORDINATIONSELECTION component of the Threat Resolution Module is shown in Figure 3-8.

COORDINATIONSELECTION (Algorithm 233) returns the coordination message settings (*vrc*, *cvc*, *vsb*) to be sent to the intruder as well as the vertical sense of the advisory (*sense_indiv*). The coordination message settings are generated using the individual and global advisory outputs from ACTIONSELECTION, together with ownership and intruder state information. When there is a global resolution advisory (RA), COORDINATIONSELECTION uses the optimal action that would be issued if the intruder were considered in isolation.

The global RA is given by *act_global*, *dz_min_global*, *dz_max_global*, and *ddz_global*. The optimal individual action is given by *act_indiv*. The previous global and individual advisories are given by *a_global_prev* and *a_indiv_prev*, respectively.

The algorithm first determines what individual vertical rates to use for generating the coordination message. ACTIONTORATES is used to obtain the vertical rates associated with *act_indiv*, as described in the following bulleted list. Adjustments to the individual vertical rates are needed when the global advisory is clear of conflict (COC) or a Maintain advisory.

- There is no global RA, (*act_global* is COC). Any advisory for the individual intruder is

cleared; the vertical rates associated with COC are used.

- The global RA on the previous cycle was a Maintain advisory. When the individual advisory is also a Maintain advisory, the same vertical rates as the global Maintain advisory are used. Otherwise, the vertical rates associated with the individual advisory are used.
- The global RA on the current cycle is a Maintain advisory. When the individual advisory is also a Maintain advisory, the same vertical rates as the global Maintain advisory are used. Otherwise, the vertical rates associated with the individual advisory are used.
- The global advisory is something other than COC or a Maintain advisory. The vertical rates associated with the individual advisory are used.

Next, CROSSLINK is called to get the values of *vrc*, *cvc*, *vsb*, and *sense_indiv* based on the individual vertical rates, previous coordination message sent to this intruder (*a_indiv_prev.vrc* and *a_indiv_prev.cvc*), and the equipage of this intruder (*equipage_int*).

Finally, after CROSSLINK is run, the current values are recorded to be used as the previous values when CROSSLINK is run in the next cycle. The individual action (*a_indiv_prev.action*) on the previous cycle is used to record whether an RA was issued on the previous cycle (*a_indiv_prev.ra_prev*).

This algorithm takes as input *act_indiv*, *act_global*, *dz_min_global*, *dz_max_global*, *ddz_global*, *dz_own_ave*, *a_global_prev*, *a_indiv_prev*, and *equipage_int*.

A description of the output variables is found in Table 3-8.

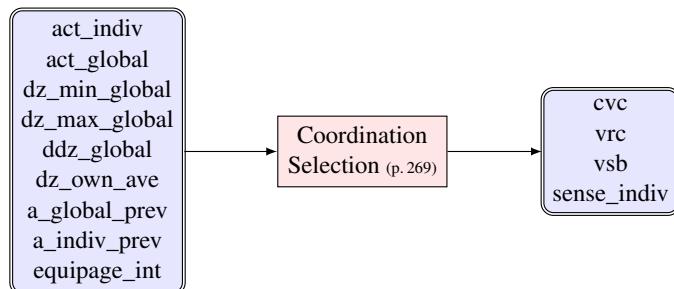


Figure 3-8. CoordinationSelection overview.

Table 3-8. TRM Algorithm Output Variables - CoordinationSelection

Variable	Units	Type	Description
cvc	N/A	uint32	Cancel Vertical Resolution Advisory Complement
vrc	N/A	uint32	Vertical Resolution Advisory Complement
vsb	N/A	uint32	Vertical Sense Bits
sense_indiv	N/A	Symbol	Resolution sense (up, down, none) for this intruder

Algorithm 233 CoordinationSelection

```

1 function CoordinationSelection( act_indiv::Z, act_global::Z, dz_min_global::R, dz_max_global::R,
2                               ddz_global::R, dz_own_ave::R, a_global_prev::GlobalAdvisory(p.E-35),
3                               a_indiv_prev::IndividualAdvisory(p.E-36), equipage_int::Z )
4   dz_min_indiv::R = -Inf
5   dz_max_indiv::R = Inf
6   ddz_indiv::R = 0.0
7   if (COC == act_global)
8     (dz_min_indiv, dz_max_indiv, ddz_indiv) = (dz_min_global, dz_max_global, 0.0)
9   elseif IsMaintain(p.335)( a_global_prev.dz_min, a_global_prev.dz_max )
10    (dz_min_indiv, dz_max_indiv, ddz_indiv) =
11      ActionToRates(p.325)( act_indiv, dz_own_ave, a_indiv_prev.action,
12                            a_global_prev.dz_min, a_global_prev.dz_max, ddz_global )
13   elseif IsMaintain(p.335)( dz_min_global, dz_max_global )
14    (dz_min_indiv, dz_max_indiv, ddz_indiv) =
15      ActionToRates(p.325)( act_indiv, dz_own_ave, a_indiv_prev.action,
16                            dz_min_global, dz_max_global, ddz_global )
17   else
18    (dz_min_indiv, dz_max_indiv, ddz_indiv) =
19      ActionToRates(p.325)( act_indiv, dz_own_ave, -1, NaN, NaN, NaN )
20   end
21   (cvc::UInt32, vrc::UInt32, vsb::UInt32, sense_indiv::Symbol) =
22     Crosslink(p.270)( dz_min_indiv, dz_max_indiv, a_indiv_prev.vrc, a_indiv_prev.cvc, equipage_int )
23   if (COC == a_indiv_prev.action)
24     a_indiv_prev.ra_prev = false
25   else
26     a_indiv_prev.ra_prev = true
27   end
28   a_indiv_prev.action = act_indiv
29   a_indiv_prev.dz_min = dz_min_indiv
30   a_indiv_prev.dz_max = dz_max_indiv
31   a_indiv_prev.ddz = ddz_indiv
32   a_indiv_prev.sense = sense_indiv
33   a_indiv_prev.vrc = vrc
34   a_indiv_prev.cvc = cvc
35   return (cvc::UInt32, vrc::UInt32, vsb::UInt32, sense_indiv::Symbol)
36 end

```

Referenced In: VerticalTRMUpdate(p. 152)

CROSSLINK (Algorithm 234) selects the appropriate values for *vrc* (Vertical Resolution Advisory Complement), *cvc* (Cancel Vertical Resolution Advisory Complement), *vsb* (Vertical Sense Bits), and the individual sense *sense_indiv*. CROSSLINK makes no determination about whether VRCs need to be sent, but includes the information needed to make the decision. The responsibility rests with the transponder or front end system. CROSSLINK accepts the current individual advisory, expressed as vertical rates (*dz_min_indiv* and *dz_max_indiv*), the VRC and CVC of the previous coordination message issued (*vrc_prev* and *cvc_prev*), and the equipage of the intruder (*equipage_int*). CROSSLINK calls RATESTOSENSE with the vertical rates to obtain the sense of the current individual resolution advisory (*sense_indiv*). Next, *sense_indiv* is used to determine *vrc* and *cvc*. The VRC transmitted to the intruder on the previous cycle is used by VRCTOSENSE to determine the sense of the RA for the previous cycle. That sense is used for determining *cvc*.

CROSSLINK calls ENCODEVRC (Algorithm 236) to determine *vrc*, then ENCODECVC (Algorithm 235) to determine *cvc*. It calls ENCODEVSB (Algorithm 237) with those results to determine *vsb*. Those three values are used to encode the coordination message.

Downstream processing will ensure that no coordination message is sent to the intruder if *cvc* = 0 and *vrc* = 0.

If the intruder is not capable of receiving resolution advisory coordination messages, the values of *vrc*, *cvc*, and *vsb* will be 0. Intruders that are equipped with TCAS systems, ACAS X systems, Collision Avoidance Systems (CAS) with Responsive capability, and Detect And Avoid (DAA) Responsive systems are all considered capable of receiving coordination messages.

This algorithm takes as input *dz_min_indiv*, *dz_max_indiv*, *vrc_prev*, *cvc_prev*, and *equippage_int*. This algorithm returns *cvc*, *vrc*, *vsb* and *sense_indiv*.

Algorithm 234 Crosslink

```

1 function Crosslink( dz_min_indiv::R, dz_max_indiv::R, vrc_prev::UInt32, cvc_prev::UInt32, equippage_int::Z )
2   sense_indiv::Symbol = RatesToSense(p. 342)( dz_min_indiv, dz_max_indiv )
3   vrc::UInt32 = EncodeVRC(p. 272)( sense_indiv, equippage_int )
4   cvc::UInt32 = EncodeCVC(p. 271)( sense_indiv, VRCToSense(p. 344)( vrc_prev ), cvc_prev, equippage_int )
5   vsb::UInt32 = EncodeVSB(p. 272)( vrc, cvc )
6   return (cvc::UInt32, vrc::UInt32, vsb::UInt32, sense_indiv::Symbol)
7 end
```

Referenced In: SetInvalidIntruder(p. 324), CoordinationSelection(p. 269), SetDroppedIntruder(p. 322), ConvertAdvisory(p. 296)

ENCODECVC (Algorithm 235) outputs the setting for the Cancel Vertical Resolution Advisory Complement (CVC) value in the outgoing coordination message to an individual intruder. It persists the CVC from one cycle to the next, until there is no longer an advisory, to account for possible loss of messages. When there is a clear of conflict, the CVC cancelling the previous Vertical Resolution Advisory Complement (VRC) will be sent only once.

If the individual intruder is not equipped to receive collision avoidance coordination messages, as determined from *equippage_int*, then *cvc* is set to 0. An individual intruder is considered equipped if it is equipped with a TCAS system, ACAS X system, Collision Avoidance System (CAS) with Responsive capability, or Detect And Avoid (DAA) Responsive system.

If the individual intruder is equipped to receive coordination messages, the sense of the current individual resolution advisory (RA) (*sense_indiv*) and the sense of the individual RA on the previous cycle (*sense_indiv_prev*) are examined to see if there has been a change. The value of *sense_indiv_prev* is derived from the VRC sent to this intruder on the previous cycle. A sense of *:None* indicates no advisory. A *sense_indiv* value of *:None* indicates clear of conflict when following an RA, indicated by a *sense_indiv_prev* value of *:Up* or *:Down*.

For an intruder equipped to receive coordination messages, the value of *cvc* is set as follows:

- The sense is *:None* for both; there is no RA to be cancelled. Therefore, *cvc* is set to 0.
- A down sense RA cleared; the previous VRC is cancelled. Therefore, *cvc* is set to 1.
- An up sense RA cleared; the previous VRC is cancelled. Therefore, *cvc* is set to 2.
- There is no change in the sense; the previous value of *cvc* is persisted.
- There is a reversal from a down sense RA to an up sense RA; the previous VRC is cancelled. Therefore, *cvc* is set to 1.
- There is a reversal from an up sense RA to a down sense RA; the previous VRC is cancelled. Therefore, *cvc* is set to 2.

- This is an initial RA, either up or down sense; there is no RA to be cancelled. Therefore, *cvc* is set to 0.

This algorithm takes as input *sense_indiv*, *sense_indiv_prev*, *cvc_prev*, and *equipage_int*. This algorithm returns *cvc*.

Algorithm 235 EncodeCVC

```

1 function EncodeCVC( sense_indiv::Symbol, sense_indiv_prev::Symbol, cvc_prev::UInt32, equipage_int::Z )
2   cvc::UInt32 = 0
3   if (equipage_int != EQUIPAGE_CASRA) &&
4     (equipage_int != EQUIPAGE_CASTA) &&
5     (equipage_int != EQUIPAGE_CASRESP) &&
6     (equipage_int != EQUIPAGE_DAARESP)
7     cvc = 0
8   elseif (sense_indiv == :None) && (sense_indiv_prev == :None)
9     cvc = 0
10  elseif (sense_indiv == :None) && (sense_indiv_prev == :Down)
11    cvc = 1
12  elseif (sense_indiv == :None) && (sense_indiv_prev == :Up)
13    cvc = 2
14  elseif (sense_indiv == sense_indiv_prev)
15    cvc = cvc_prev
16  elseif (sense_indiv == :Up) && (sense_indiv_prev == :Down)
17    cvc = 1
18  elseif (sense_indiv == :Down) && (sense_indiv_prev == :Up)
19    cvc = 2
20  else
21    cvc = 0
22  end
23  return cvc::UInt32
24 end
```

Referenced In: Crosslink(p.270)
--

ENCODEVRC (Algorithm 236) outputs the setting for the Vertical Resolution Advisory Complement (VRC) value in the outgoing coordination message to an individual intruder.

If the individual intruder is not equipped to receive collision avoidance coordination messages, as determined from *equipage_int*, then *vrc* is set to 0. An individual intruder is considered equipped if it is equipped with a TCAS system, ACAS X system, Collision Avoidance System (CAS) with Responsive capability, or Detect And Avoid (DAA) Responsive system.

If the individual intruder is equipped to receive coordination messages, the sense of the current individual resolution advisory (RA) (*sense_indiv*) is used to determine what value to give *vrc*. When the sense of the individual RA is down, *vrc* is set to 1 indicating ‘Do not pass below.’ When the sense of the individual RA is up, *vrc* is set to 2 indicating ‘Do not pass above.’ A sense of :None indicates no advisory and *vrc* is set to 0 indicating no intent.

This algorithm takes as input *sense_indiv* and *equipage_int*. This algorithm returns *vrc*.

Algorithm 236 EncodeVRC

```

1 function EncodeVRC( sense_indiv::Symbol, equipage_int::Z )
2     vrc::UInt32 = 0
3     if (equipage_int != EQUIPAGE_CASRA)    &&
4         (equipage_int != EQUIPAGE_CASTA)    &&
5         (equipage_int != EQUIPAGE_CASRESP) &&
6         (equipage_int != EQUIPAGE_DAARESP)
7         vrc = 0
8     elseif (sense_indiv == :Up)
9         vrc = 2
10    elseif (sense_indiv == :Down)
11        vrc = 1
12    else
13        vrc = 0
14    end
15    return vrc::UInt32
16 end

```

Referenced In: UpdateIntruderInputs(p. 211), TRMIIntruderData(p. E-39), Crosslink(p. 270)

ENCODEVSB (Algorithm 237) outputs the setting for the Vertical Sense Bits (VSB) value in the outgoing coordination message to an individual intruder.

The *vsb* field is a parity field used to protect the *vrc* and *cvc* fields. The value of *vsb* is encoded directly from *vrc* and *cvc* using a parity table. The parity table (*X_parity_table*) is an input parameter, the use of which is described in Section 2.5.1.

When *vrc* and *cvc* are both 0, *vsb* will be 0 as well.

This algorithm takes as input *vrc* and *cvc*. This algorithm returns *vsb*.

Algorithm 237 EncodeVSB

```

1 function EncodeVSB( vrc::UInt32, cvc::UInt32 )
2     const X_parity_table::Vector{Z} = params().coordination.parity_table
3     vsb::UInt32 = uint32( X_parity_table[ (cvc * 4) + vrc + 1 ] )
4     return vsb::UInt32
5 end

```

Referenced In: Crosslink(p. 270)

3.5 Track Threat Assessment

An overview of the TRACKTHREATASSESSMENT component of the Threat Resolution Module is shown in Figure 3-9.

TRACKTHREATASSESSMENT (Algorithm 238) is responsible for creating a code that indicates the status of the track and a Track Display Score (TDS) for the onboard pilot display. The track status code indicates whether an intruder is involved in a Proximity Advisory (PA), Traffic Advisory (TA), or Resolution Advisory (RA). The TDS is a real-valued quantity that allows the intruder tracks to be prioritized in case of limited display resources.

First, DETERMINEINTRUDERALERT (Algorithm 239) determines whether a TA should be active or inactive. Next, DETERMINECODE (Algorithm 240) is used to determine the track status code.

Finally, DETERMINESCORE (Algorithm 241) is used to determine the Track Display Score.

This algorithm takes as input *mode_int*, *height_own*, *cost_ta*, *sense_indiv*, *degraded_surveillance*, *is_proximate*, *is_designated*, *equip_int*, and *st_int*.

The output of TRACKTHREATASSESSMENT is summarized in Table 3-9.

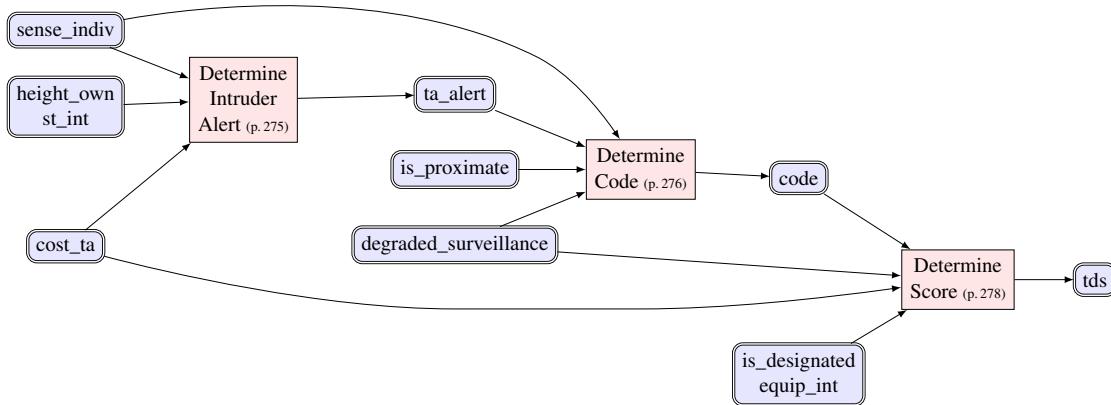


Figure 3-9. TrackThreatAssessment overview.

Table 3-9. TRM Algorithm Output Variables - TrackThreatAssessment

Variable	Units	Type	Description
code	N/A	uint8	Track advisory code
tds	N/A	real	Track display score

Algorithm 238 TrackThreatAssessment

```

1 function TrackThreatAssessment( mode_int::Z, height_own::R, cost_ta::Vector{R}, sense_indiv::Symbol,
2                                 degraded_surveillance::UInt16, is_proximate::Bool, is_designated::Bool,
3                                 equip_int::Bool, st_int::TRMIIntruderState(p. E-33) )
4     ta_alert::Bool =
5         DetermineIntruderAlert(p. 275)( mode_int, height_own, cost_ta, sense_indiv, st_int )
6     code::UInt8 =
7         DetermineCode(p. 276)( ta_alert, sense_indiv, degraded_surveillance, is_proximate )
8     tds::R =
9         DetermineScore(p. 278)( cost_ta[COC], code, degraded_surveillance, is_designated, equip_int )
10    return (code::UInt8, tds::R)
11 end
  
```

Referenced In: VerticalTRMUpdate(p. 152)

3.5.1 Determine Intruder Alert

DETERMINEINTRUDERALERT (Algorithm 239) determines whether a Traffic Advisory (TA) should

be active or inactive for an individual intruder.

DETERMINEINTRUDERALERT uses the vector *cost_ta*, output by INDIVIDUALCOSTESTIMATION, to obtain the costs of the possible actions. The altitude of own aircraft, in feet, is input in the variable *height_own*. The value is determined by GETOWNHEIGHT. It will be radio altitude (AGL) when own aircraft is relatively close to the ground. It will be the barometric altitude otherwise. The algorithm is initialized with an inactive TA status for the intruder.

It next determines whether a resolution advisory (RA) has just terminated. When an RA terminates, a TA will continue to be displayed for a minimum number of seconds, represented by *T_maintain_ta_after_ra*. An RA on the previous cycle is indicated by a previous individual sense (*st_int.sense_prev*) of :Up or :Down for this intruder. A clear of conflict (COC) on the current cycle is indicated by an individual sense (*sense_indiv*) of :None for this intruder. The value of *coc_occurred* is set to TRUE when there is a clear of conflict (COC) on the current cycle following an RA on the previous cycle for the individual intruder. Otherwise, *coc_occurred* is set to FALSE. If there is a COC on the current cycle indicating an RA is terminated, the TA timer, *st_int.time_since_ta*, is started.

In addition to issuing TAs following an RA, TAs are also issued to warn of a potential threat in advance of an RA. If an RA has not just terminated, then the TA status is set based on the adjusted cost of the COC action, *cost_ta[COC]*. A high cost for the COC action relative to all the other RA actions indicates a higher threat potential for the intruder. The costs for the RA actions are stored in *costs_ra_actions*. The cost thresholds for activating a TA are dependent on the altitude band (*H_ta_threshold*) containing the altitude of own aircraft (*height_own*). The cost thresholds associated with each altitude band are given by *C_ta_threshold_on* and *C_ta_differential_threshold*. The settings for *alt_lo*, *alt_hi*, *c_on_lo*, *c_on_hi*, *c_diff_lo*, and *c_diff_hi* are determined from the altitude band containing the altitude of own aircraft. The variables *alt_lo* and *alt_hi* are used to determine a scale factor (*scale_factor*) based on own aircraft's position within the altitude band. This scale factor is used to determine the altitude-dependent cost thresholds (*c_threshold_on* and *c_differential_threshold*). To activate a TA, the cost of COC must be both larger than the altitude-dependent TA activation threshold (*c_threshold_on*), and the cost of COC plus an altitude-dependent epsilon value (*c_differential_threshold*) must be larger than the minimum of the costs for all other actions.

The TA is required to stay active for a some minimum duration specified by a parameter, in this case *T_ta_min_alert*. The TA will be deactivated when the cost of COC is less than a specified deactivation threshold (*C_ta_threshold_off*), or the minimum active duration is achieved.

The variable *st_int.time_since_ta* is a multiuse counter that is incremented every second that a TA is active. It is set to zero when the TA status is FALSE or the minimum alert time is reached.

This algorithm takes as input *mode_int*, *height_own*, *cost_ta*, *sense_indiv*, and *st_int*. This algorithm returns *ta_alert*. This algorithm updates *st_int::TRMIntruderState*.

Algorithm 239 DetermineIntruderAlert

```

1 function DetermineIntruderAlert( mode_int::Z, height_own::R, cost_ta::Vector{R}, sense_indiv::Symbol,
2                               st_int::TRMIIntruderState(p.E-33) )
3   const C_ta_threshold_on::Vector{R} =
4     params().modes[mode_int].track_threat.ta_cost_threshold_on
5   const H_ta_threshold::Vector{R} =
6     params().modes[mode_int].track_threat.ta_altitude_threshold
7   const C_ta_threshold_off::R =
8     params().modes[mode_int].track_threat.ta_cost_threshold_off
9   const C_ta_differential_threshold::Vector{R} =
10    params().modes[mode_int].track_threat.ta_differential_threshold
11   const T_ta_min_alert::Z =
12     params().modes[mode_int].track_threat.ta_min_alert_time
13   const T_maintain_ta_after_ra::Z =
14     params().modes[mode_int].track_threat.maintain_ta_after_ra
15   ta_alert::Bool = false
16   coc_occurred::Bool = (st_int.sense_prev != :None) && (sense_indiv == :None)
17   if coc_occurred && (0 < T_maintain_ta_after_ra)
18     ta_alert = true
19     st_int.time_since_ta = T_ta_min_alert - T_maintain_ta_after_ra + 1
20   else
21     costs_ra_actions::Vector{R} = [ cost_ta[1:COC-1], cost_ta[COC+1:end] ]
22     idx::Z = 1
23     while ((idx+1) < length( H_ta_threshold )) && (H_ta_threshold[idx+1] < height_own)
24       idx = idx + 1
25     end
26     alt_lo::R = H_ta_threshold[idx]
27     alt_hi::R = alt_lo
28     c_on_lo::R = C_ta_threshold_on[idx]
29     c_on_hi::R = c_on_lo
30     c_diff_lo::R = C_ta_differential_threshold[idx]
31     c_diff_hi::R = c_diff_lo
32     if ((idx+1) <= length( H_ta_threshold ))
33       alt_hi = H_ta_threshold[idx+1]
34       c_on_hi = C_ta_threshold_on[idx+1]
35       c_diff_hi = C_ta_differential_threshold[idx+1]
36     end
37     scale_factor::R = CalculateThresholdRampUpFactor(p.326)( height_own, alt_lo, (alt_hi - alt_lo) )
38     c_threshold_on::R = c_on_lo + ((c_on_hi - c_on_lo) * scale_factor)
39     c_differential_threshold::R = c_diff_lo + ((c_diff_hi - c_diff_lo) * scale_factor)
40     if ( ((c_threshold_on < cost_ta[COC]) &&
41           (minimum( costs_ra_actions ) < (cost_ta[COC] + c_differential_threshold))) ||
42           ((st_int.time_since_ta != 0) && (st_int.time_since_ta < T_ta_min_alert)) ||
43           ((st_int.time_since_ta != 0) && (C_ta_threshold_off <= cost_ta[COC])) )
44       ta_alert = true
45       st_int.time_since_ta = st_int.time_since_ta + 1
46     else
47       ta_alert = false
48       st_int.time_since_ta = 0
49     end
50   end
51   return ta_alert::Bool
52 end

```

Referenced In: TrackThreatAssessment(p.273)

3.5.2 Determine Code

DETERMINECODE (Algorithm 240), used to determine the track status code for an individual intruder, is derived from TCAS II. The track status code is assigned based on the most critical advisory for the individual intruder.

First, the sense of the advisory for the individual intruder is examined to determine if there is a

Resolution Advisory (RA). If so, the value is set to *TACODE_RA* (4) to indicate that an RA is present.

If not, the *ta_alert* flag from DETERMINEINTRUDERALERT is used to determine if there is a Traffic Advisory (TA). DETERMINECODE uses *degraded_surveillance*, obtained from the intruder input from the STM, to assign a value to the target status code for the TA. When the surveillance is Non-Altitude Reporting (NAR), the value is set to *TACODE_TA_DEGRADED* (2). When the surveillance is ADS-B only, without active validation, the value is also set to *TACODE_TA_DEGRADED* (2). TCAS II assigns a track status code for the TA based on altitude reporting; it does not support ADS-B only surveillance. ACAS X differs from TCAS II in that regard.

If the TA alert is due to a threat with non-degraded surveillance, the TA code of *TACODE_TA_NOMINAL* (3) is used.

If there is no RA or TA, the *is_proximate* intruder input from the STM is used to determine if a Proximity Advisory (PA) should be issued. In that case, the value is set to *TACODE_PA* (1).

If none of the other conditions hold, the code is set to *TACODE_CLEAR* (0) to indicate no advisory is in effect.

This algorithm takes as input *ta_alert*, *sense_indiv*, *degraded_surveillance*, and *is_proximate*. This algorithm returns *code*.

Algorithm 240 DetermineCode

```

1 function DetermineCode( ta_alert::Bool, sense_indiv::Symbol, degraded_surveillance::UInt16,
2   is_proximate::Bool )
3   code::UInt8 = TACODE_CLEAR
4   if (sense_indiv != :None)
5     code = TACODE_RA
6   elseif ta_alert
7     if ((degraded_surveillance & DEGRADED_SURVEILLANCE_NAR) != 0)
8       code = TACODE_TA_DEGRADED
9     elseif ((degraded_surveillance & DEGRADED_SURVEILLANCE_ADSB_ONLY) != 0)
10      code = TACODE_TA_DEGRADED
11    else
12      code = TACODE_TA_NOMINAL
13    end
14   elseif is_proximate
15     code = TACODE_PA
16   else
17     code = TACODE_CLEAR
18   end
19   return code::UInt8
20 end
```

Referenced In: SetInvalidIntruder(p. 324), TrackThreatAssessment(p. 273), ConvertAdvisory(p. 296)
--

3.5.3 Determine Score

DETERMINESCORE (Algorithm 241) determines the Track Display Score for the individual intruder, to be used for prioritizing tracks for the onboard pilot display. It uses the cost of clear of conflict (i.e., not alerting) when determining the Track Display Score. An intruder with a higher associated cost for the clear of conflict action (*cost_ta_coc*) poses a greater threat, and so should be assigned a higher priority for display purposes.

When the surveillance is ADS-B only, without active validation, ACAS X treats it the same as

Non-Altitude Reporting, assigning a lower track display score for anything other than a resolution advisory.

The remainder of the algorithm assigns ranks based on the advisory codes as follow:

- Intruders with resolution advisories receive high scores and go to the top of the list. Within this category, intruders with which own aircraft is coordinating resolution advisories receive the highest scores.
- Traffic advisories are broken into two groups: *TACODE_TA_NOMINAL*, which does not have degraded surveillance, and *TACODE_TA_DEGRADED*, which has degraded surveillance. *TACODE_TA_NOMINAL* advisories are ranked above *TACODE_TA_DEGRADED* advisories.
- Intruders with proximity advisories, *TACODE_PA*, receive lower scores than any intruder with a traffic advisory (*TACODE_TA_NOMINAL* and *TACODE_TA_DEGRADED*). Within this category, intruders with degraded surveillance receive lower scores because *scorefactor* is smaller for degraded surveillance.
- Any intruders that have no advisory (*TACODE_CLEAR*) receive the lowest score. Within this category, intruders with degraded surveillance receive lower scores because *scorefactor* is smaller for degraded surveillance.
- Finally, if an intruder has been designated by the pilot to an Xo mode (*is_designated*) and has no active resolution advisory or traffic advisory, then that intruder is ranked above all intruders with proximity advisories (*TACODE_PA*). There is no adjustment based on surveillance quality in this category.

The clear of conflict cost used for determining the TDS is the same one used in DETERMINEINTRUDERALERT for determining whether a TA should be issued.

This algorithm takes as input *cost_ta_coc*, *code*, *degraded_surveillance*, *is_designated*, and *equip_int*. This algorithm returns *tds*.

Algorithm 241 DetermineScore

```

1 function DetermineScore( cost_ta_coc::R, code::UInt8, degraded_surveillance::UInt16,
2           is_designated::Bool, equip_int::Bool )
3   const X_scorfactr_good::R      = params().display.scorfactr_good
4   const X_scorfactr_degraded::R = params().display.scorfactr_degraded
5   const X_hiscore::R          = params().display.hiscore
6   const X_medscore::R         = params().display.medscore
7   const X_loscore::R          = params().display.loscore
8   const X_tinyscore::R        = params().display.tinyscore
9   const X_designatedscore::R  = params().display.designatedscore
10  tds::R                   = 0.0
11  scorefactor::R = X_scorfactr_good
12  if ((degraded_surveillance & DEGRADED_SURVEILLANCE_NAR) != 0) || 
13    ((degraded_surveillance & DEGRADED_SURVEILLANCE_ADSB_ONLY) != 0)
14    scorefactor = X_scorfactr_degraded
15  end
16  if (code == TACODE_RA)
17    if equip_int
18      tds = X_hiscore + X_medscore + cost_ta_coc
19    else
20      tds = X_hiscore + cost_ta_coc
21    end
22  elseif (code == TACODE_TA_NOMINAL)
23    tds = (X_medscore * X_scorfactr_good) + cost_ta_coc
24  elseif (code == TACODE_TA_DEGRADED)
25    tds = (X_medscore * X_scorfactr_degraded) + cost_ta_coc
26  elseif (code == TACODE_PA)
27    tds = (X_loscore * scorefactor) + cost_ta_coc
28  elseif (code == TACODE_CLEAR)
29    tds = (X_tinyscore * scorefactor) + cost_ta_coc
30  end
31  if is_designated
32    tds = max( tds, (X_designatedscore + cost_ta_coc) )
33  end
34  return tds::R
35 end

```

Referenced In: SetInvalidIntruder(p. 324), TrackThreatAssessment(p. 273), ConvertAdvisory(p. 296)

3.6 Display Logic

An overview of the DISPLAYLOGICDETERMINATION component of the Threat Resolution Module is shown in Figure 3-10.

DISPLAYLOGICDETERMINATION (Algorithm 242) converts the action and rates for the current advisory into parameters that drive the pilot display, including the visual indicators and aural alarms.

DETERMINECROSSING (Algorithm 243) is used to determine if the global advisory is a crossing advisory and to set the crossing flag appropriately.

DETERMINEDIPLAYDATA (Algorithm 244) is used to determine the Label 270 code and target rate for the global advisory given own aircraft altitude and vertical rate, the advisory state, and the crossing flag.

DETERMINEAURALINHIBIT (Algorithm 247) is used to determine if aural output to the cockpit should be inhibited completely. It sets the aural inhibit flag accordingly. Note that the aural inhibit is different from the altitude inhibits. The altitude inhibits change the advisory and the aural annunciation (e.g., from 'Level-Off' to 'Monitor Vertical Speed'), but they don't inhibit the aural from being output.

DETERMINETRAFFICALERT (Algorithm 248) is used to determine whether a global traffic advisory is being issued and set a flag accordingly. It uses per-intruder inputs produced by DETERMINECODE to determine the global advisory state.

DISPLAYLOGICDETERMINATION takes as input h_own , z_own_ave , dz_own_ave , z_int_ave , $action$, dz_min , dz_max , $sense_int$, $code_int$, and st_own .

The outputs are summarized in Table 3-10.

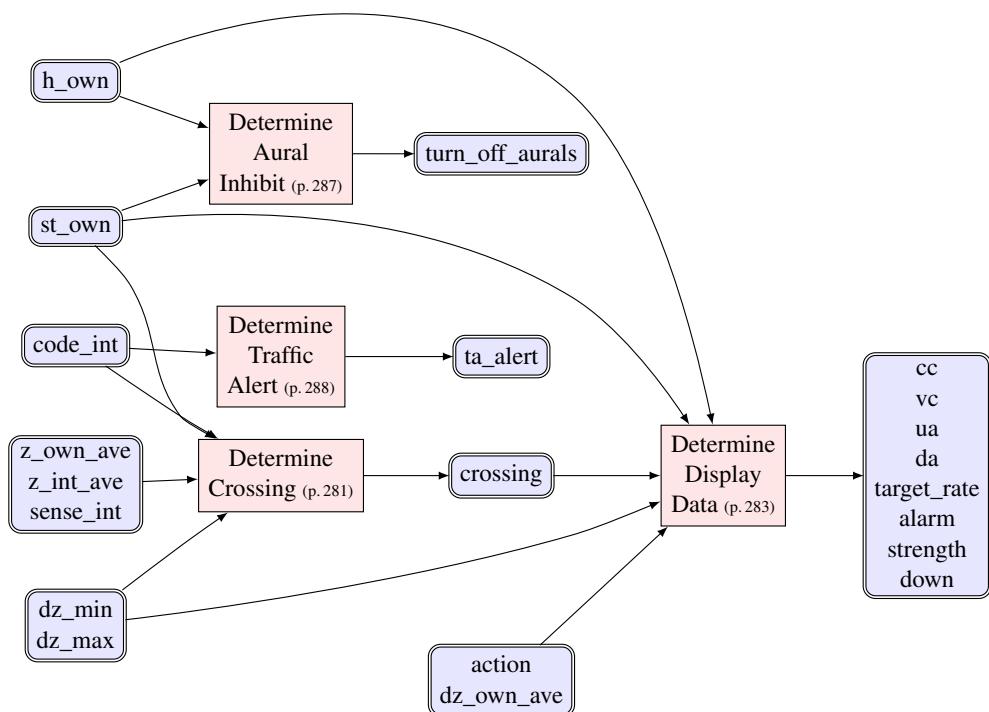


Figure 3-10. DisplayLogicDetermination overview.

Table 3-10. TRM Algorithm Output Variables - DisplayLogicDetermination

Variable	Units	Type	Description
alarm	N/A	bool	Resolution Advisory changed
cc	N/A	uint8	Label270 Combined Control
crossing	N/A	bool	Crossing flag
da	N/A	uint8	Label270 Down Advisory
strength	N/A	uint8	Vertical ARA strength code
ta_alert	N/A	bool	Traffic Advisory is active
target_rate	ft/s	real	Target rate
turn_off_aurals	N/A	bool	Turn off aurals
ua	N/A	uint8	Label270 Up Advisory
up	N/A	bool	Vertical ARA Up flag
vc	N/A	uint8	Label270 Vertical Control

Algorithm 242 DisplayLogicDetermination

```

1 function DisplayLogicDetermination( h_own::R, z_own_ave::R, dz_own_ave::R,
2                                     z_int_ave::Vector{R}, action::Z, dz_min::R, dz_max::R,
3                                     sense_int::Vector{Symbol}, code_int::Vector{UInt8}, st_own::TRMOwnState(p.E-32) )
4     crossing::Bool =
5         DetermineCrossing(p.281)( dz_min, dz_max, z_own_ave, z_int_ave, sense_int, code_int, st_own )
6     (cc::UInt8, vc::UInt8, ua::UInt8, da::UInt8, target_rate::R, alarm::Bool,
7      strength::UInt8, down::Bool) =
8         DetermineDisplayData(p.283)( action, crossing, dz_min, dz_max, h_own, dz_own_ave, st_own )
9     turn_off_aurals::Bool =
10        DetermineAuralInhibit(p.287)( h_own, st_own )
11    ta_alert::Bool =
12        DetermineTrafficAlert(p.288)( code_int )
13    display_logic::DisplayLogic(p.E-35) =
14        DisplayLogic(p.E-35)( crossing, cc, vc, ua, da, target_rate, alarm, turn_off_aurals, ta_alert,
15                           strength, down )
16    return display_logic::DisplayLogic(p.E-35)
17 end

```

Referenced In: VerticalTRMUpdate(p.152)

3.6.1 Determine Crossing

DETERMINECROSSING (Algorithm 243) predicts whether there will be an altitude crossing with an intruder against which there is an active resolution advisory. The resulting crossing flag is used in certain situations to adjust the aural advisory annunciation in the cockpit. The crossing flag is also output in messages to the ground station.

DETERMINECROSSING sets the crossing flag (*is_crossing*) when it determines the global resolution advisory (RA) is a crossing advisory. For this check, an intruder is only considered if there is an active resolution advisory against it, as indicated by *code_int* having a value of *TACODE_RA* for that intruder. Furthermore, only those intruders with an individual RA sense (*sense_int*) in the same direction as the global sense (*sense_own*) can lead to the crossing flag being set. When there is a single threat, the sense of the RA for the individual threat will always be the same as the sense of the global RA. If the the global RA is a Multi-Threat Level-Off (MTLO), this check is considered

passed as MTLO RAs are neither up sense nor down sense.

DETERMINECROSSING then checks if the threat is above and the individual RA sense is up, or if the threat is below and the individual RA sense is down. If the individual RA directs ownship towards the threat and the vertical separation is greater than h_cross_thresh feet, then a crossing is indicated. Whenever the separation is less than or equal to h_cross_thresh feet, no crossing is indicated, regardless of the individual RA sense. h_cross_thresh is a vertical threshold with hysteresis. If there was a crossing on the previous cycle, the minimum vertical separation for a crossing will be lower ($H_cross_thresh_lo$ instead of $H_cross_thresh_hi$). This hysteresis allows the crossing flag to remain set until after the altitudes have crossed.

The crossing flag is used to determine the appropriate aural advisory and may be encoded in the "vertical control" field of Label 270. It is used, without modification, to set the crossing bit in the output to the ground station.

This algorithm takes as input dz_min , dz_max , z_own_ave , z_int_ave , $sense_int$, $code_int$ and st_own . This algorithm returns $is_crossing$. This algorithm updates $st_own::TRMOwnState$.

Algorithm 243 DetermineCrossing

```

1 function DetermineCrossing( dz_min::R, dz_max::R, z_own_ave::R, z_int_ave::Vector{R},
2                               sense_int::Vector{Symbol}, code_int::Vector{UInt8}, st_own::TRMOwnState(p. E-32) )
3   const H_cross_thresh_lo::R = params().display.crosstrhllo
4   const H_cross_thresh_hi::R = params().display.crosstrhlhi
5   is_crossing::Bool = false
6   sense_own::Symbol = RatesToSense(p.342)( dz_min, dz_max )
7   h_cross_thresh::R = H_cross_thresh_hi
8   if st_own.crossing_prev
9     h_cross_thresh = H_cross_thresh_lo
10  end
11  for i = 1:length( code_int )
12    if (code_int[i] == TACODE_RA) &&
13      (sense_int[i] == sense_own) || IsMTLO(p.336)( dz_min, dz_max )
14      if (sense_int[i] == :Up) && (h_cross_thresh < (z_int_ave[i] - z_own_ave))
15        is_crossing = true
16      elseif (sense_int[i] == :Down) && (h_cross_thresh < (z_own_ave - z_int_ave[i]))
17        is_crossing = true
18      end
19    end
20  end
21  st_own.crossing_prev = is_crossing
22  return is_crossing::Bool
23 end
```

Referenced In: DisplayLogicDetermination(p.280)
--

3.6.2 Determine Display Data

DETERMINEDIPLAYDATA (Algorithm 244) produces the values needed to display the global resolution advisory to the pilot. In addition, it produces the strength and up values needed to convey the global resolution advisory to the ground station.

DETERMINEDIPLAYDATA first determines if an altitude inhibit should be in effect. When own aircraft is at or below $H_descend_thr$, it may be necessary to modify the annunciation, and therefore the Label 270 word. For example, when own aircraft is below 1000' above ground level and transitioning from an increase descend or descend resolution advisory (RA) to a Do Not Climb RA, the annunciation should be 'Monitor Vertical Speed' (Label 270 word 6002) instead of 'Level-Off,

'Level-Off' (Label 270 word 5002), as would be customary. In this case, a Label 270 rule can be constructed to produce the desired behavior based on the value of *alt_inhibit* being TRUE.

Next, a clear of conflict (COC) annunciation is suppressed whenever there was an RA on the previous cycle and that RA was suppressed in the output to the pilot display. An RA is suppressed in output to the pilot display when there is a single threat and it is designated to Designated No Alerts (DNA) Xo mode. Suppression of the COC annunciation is accomplished by setting the previous action value (*action_prev*) to *COC*.

GETMATCHINGLABEL270RULE (Algorithm 245) is then called to obtain the Label 270 values (*cc*, *vc*, *ua*, and *da*), the *force_alarm* flag, and the *strength* and *up* values associated with the inputs. A Label 270 word is formed by combining *cc*, *vc*, *ua*, and *da*. If *cc* is 5, *vc* is 0, *ua* is 0, and *da* is 2, the Label 270 word is 5002.

The *alarm* flag indicates whether or not there should be an RA annunciation in the cockpit. Whenever the RA changes, as denoted by a change in action, there will be an annunciation. But if the RA was suppressed to the pilot display on the previous cycle, then an annunciation will be forced whenever there is a global RA.

In some cases, a change in the RA should not be annunciated, such as the transition between an initial Climb 1500 action and a subsequent Climb 1500 action. Because they are two different actions, the logic will set the *alarm* flag to TRUE. However, since these two Climb actions produce the same Label 270 word value (4010 or 4110) and the same aural message, there should be no re-annunciation. In other cases, a change in the Label 270 word within an action, that would not otherwise be annunciated, should be annunciated because the aural message has changed. An example of the latter is the transition between 6002 (Monitor Vertical Speed) and 5002 (Level-Off, Level-Off). The *force_alarm* variable is used to provide this finer tuning to the setting of *alarm*.

The target rate (ft/s) (*target_rate*) is derived from the vertical rate range for the global RA. Whenever the vertical rates for the global RA indicate a level-off or a clear of conflict, the target rate will be 0.0 ft/s. The target rate, when converted to ft/min, may appear on the pilot display as an arrow depicting the vertical advisory rate to maintain.

This algorithm takes as input *action*, *crossing*, *dz_min*, *dz_max*, *h_own*, *dz_own_ave*, and *st_own*. This algorithm returns *cc*, *vc*, *ua*, *da*, *target_rate*, *alarm*, *strength*, and *up*.

Algorithm 244 DetermineDisplayData

```

1 function DetermineDisplayData( action::Z, crossing::Bool, dz_min::R, dz_max::R,
2                               h_own::R, dz_own_ave::R, st_own::TRMOwnState(p_E-32) )
3   const H_descend_thr::R = params().display.hdescendthr
4   action_prev::Z = st_own.action_prev
5   alt_inhibit::Bool = false
6   if !isnan( h_own ) && (h_own <= H_descend_thr)
7     alt_inhibit = true
8   end
9   if st_own.ra_suppressed_prev && IsCOC(p_332)( dz_min, dz_max )
10    action_prev = COC
11  end
12  (cc::UInt8, vc::UInt8, ua::UInt8, da::UInt8, force_alarm::Z, strength::UInt8, down::Bool) =
13    GetMatchingLabel270Rule(p_285)( action, crossing, alt_inhibit, dz_own_ave,
14                                   action_prev, st_own.word_prev, st_own.strength_prev )
15  if st_own.ra_suppressed_prev
16    force_alarm = FORCE_ALARM_ON
17  end
18  alarm::Bool = false
19  if !IsCOC(p_332)( dz_min, dz_max ) && (FORCE_ALARM_OFF != force_alarm) &&
20    ((action_prev != action) || (FORCE_ALARM_ON == force_alarm))
21    alarm = true
22  end
23  target_rate::R = min( abs( dz_min ), abs( dz_max ) )
24  if ( abs( dz_min ) == abs( dz_max ) )
25    target_rate = 0.0
26  elseif (RatesToSense(p_342)( dz_min, dz_max ) == :Down)
27    target_rate = -target_rate
28  end
29  return (cc::UInt8, vc::UInt8, ua::UInt8, da::UInt8, target_rate::R, alarm::Bool,
30           strength::UInt8, down::Bool)
31 end

```

Referenced In: DisplayLogicDetermination(p_280)

GETMATCHINGLABEL270RULE (Algorithm 245) returns the cc, vc, ua, da, strength, and up values associated with the global resolution advisory. It also returns a flag indicating if an annunciation should be forced or suppressed.

The current advisory and dynamic state for comparison are specified using the following inputs: the global action for the current cycle (*action*), whether there is a crossing (*crossing*), whether altitude inhibit is activated (*alt_inhibit*), the current vertical rate of ownership (ft/s) (*dz_own_ave*), the global action on the previous cycle (*action_prev*), the Label 270 code on the previous cycle (*word_prev*), and the RA strength code on the previous cycle (*strength_prev*).

The parameters file encodes a set of rules for mapping the current action, previous action, previous Label 270 word, ownership vertical rate, crossing, and altitude inhibit status into the various display outputs. The rules are represented as arrays for the various input and output variables.

The logic rules are specified using the following parameters: a global action for the current cycle (*L_act_in*), a global action for the previous cycle (*L_act_in_prev*), a Label 270 code for the previous cycle (*L_word_in_prev*), a lower bound on ownership vertical rate (ft/s) (*R_dz_lo_in*), an upper bound on ownership vertical rate (ft/s) (*R_dz_hi_in*), a crossing dependency indication (*L_crossing_in*), an altitude inhibit dependency indication (*L_alt_inhibit_in*), and an RA strength code for the previous cycle (*L_strength_in_prev*).

Each logic rule also contains the results from a match specified by the following parameters: a Combined Control (*L_cc_out*), a Vertical Control (*L_vc_out*), an Up Advisory (*L_ua_out*), a Down

Advisory (L_{da_out}), whether to force an annunciation on or off ($L_{force_alarm_out}$), and an RA strength code ($L_{strength_out}$).

The loop iterates over all of the Label 270 rules until it finds a rule that is consistent with the current action, previous action, previous Label 270 word, ownship vertical rate, crossing, and altitude inhibit status. CONSISTENT (Algorithm 246) is used to determine if the input values are consistent with the given rule. GETMATCHINGLABEL270RULE returns output variables set to values associated with that rule. It also uses cc , ua , and da to determine whether the advisory has an up sense or not (*up*). If there is no match to any of the logic rules, the outputs all retain their initial values.

This algorithm takes as input *action*, *crossing*, *alt_inhibit*, *dz_own_ave*, *action_prev*, *word_prev*, and *strength_prev*. This algorithm returns cc , vc , ua , da , *force_alarm*, *strength*, and *up*.

Algorithm 245 GetMatchingLabel270Rule

```

1 function GetMatchingLabel270Rule( action::Z, crossing::Bool, alt_inhibit::Bool, dz_own_ave::R,
2   action_prev::Z, word_prev::Z, strength_prev::Uint8 )
3   const L_act_in::Vector{R} = params().display.label270rules.action
4   const L_act_in_prev::Vector{R} = params().display.label270rules.prevaction
5   const L_word_in_prev::Vector{R} = params().display.label270rules.prevword
6   const R_dz_lo_in::Vector{R} = params().display.label270rules.lodz
7   const R_dz_hi_in::Vector{R} = params().display.label270rules.hidz
8   const L_crossing_in::Vector{R} = params().display.label270rules.crossing
9   const L_alt_inhibit_in::Vector{R} = params().display.label270rules.altinhibit
10  const L_strength_in_prev::Vector{R} = params().display.label270rules.prevstrength
11 #
12  const L_cc_out::Vector{Z} = params().display.label270rules.cc
13  const L_vc_out::Vector{Z} = params().display.label270rules.vc
14  const L_ua_out::Vector{Z} = params().display.label270rules.ua
15  const L_da_out::Vector{Z} = params().display.label270rules.da
16  const L_force_alarm_out::Vector{Z} = params().display.label270rules.forcealarm
17  const L_strength_out::Vector{Z} = params().display.label270rules.strength
18 #
19  num_rules::Z = length( L_act_in )
20  cc::Uint8 = 0
21  vc::Uint8 = 0
22  ua::Uint8 = 0
23  da::Uint8 = 0
24  force_alarm::Z = FORCE_ALARM_NONE
25  strength::Uint8 = 0
26  down::Bool = false
27  for i in 1:num_rules
28    if Consistent(p.286)( dz_own_ave, action, action_prev, word_prev, crossing,
29      alt_inhibit, strength_prev,
30      R_dz_lo_in[i], R_dz_hi_in[i], L_act_in[i], L_act_in_prev[i],
31      L_word_in_prev[i], L_crossing_in[i], L_alt_inhibit_in[i], L_strength_in_prev[i] )
32      cc = uint8( L_cc_out[i] )
33      vc = uint8( L_vc_out[i] )
34      ua = uint8( L_ua_out[i] )
35      da = uint8( L_da_out[i] )
36      if (4 == cc)
37        down = false
38      elseif (5 == cc)
39        down = true
40      elseif (6 == cc) && (0 == ua) && (0 != da)
41        down = true
42      else
43        down = false
44      end
45      strength = uint8( L_strength_out[i] )
46      force_alarm = L_force_alarm_out[i]
47      break
48    end
49  end
50  return (cc::Uint8, vc::Uint8, ua::Uint8, da::Uint8, force_alarm::Z, strength::Uint8, down::Bool)
51 end

```

Referenced In: DetermineDisplayData(p.283)

CONSISTENT (Algorithm 246) determines whether the current advisory and dynamic state is consistent with the specified logic rule.

The logic rule being tested is specified using a lower bound on ownship vertical rate (R_{-dz_lo}), an upper bound on ownship vertical rate (ft/s) (R_{dz_hi}), a global action for the current cycle (L_{act}), a global action for the previous cycle (L_{act_prev}), a Label 270 code for the previous cycle (L_{word_prev}), a crossing dependency indication ($L_{crossing}$), an altitude inhibit dependency

indication (*L_alt_inhibit*), and a strength code for the previous cycle (*L_strength_prev*).

A value of *NaN* (Not a Number) for a logic rule variable functions as a wildcard. It will match any input value.

The advisory and dynamic state for comparison is specified using the current vertical rate of ownship (ft/s) (*dz_own*), the global action for the current cycle (*action*), the global action on the previous cycle (*action_prev*), the Label 270 code on the previous cycle (*word_prev*), whether there is a crossing (*crossing*), whether altitude inhibit is activated (*alt_inhibit*), and the strength code on the previous cycle (*strength_prev*).

If any of the inputs fail to match a rule, *is_consistent* is set to FALSE. If all the inputs match all of the rules, *is_consistent* is TRUE.

This algorithm takes as input *dz_own*, *action*, *action_prev*, *word_prev*, *crossing*, *alt_inhibit*, *strength_prev*, *R_dz_lo*, *R_dz_hi*, *L_act*, *L_act_prev*, *L_word_prev*, *L_crossing*, *L_alt_inhibit*, and *L_strength_prev*. This algorithm returns *is_consistent*.

Algorithm 246 Consistent

```

1 function Consistent( dz_own::R, action::Z, action_prev::Z, word_prev::Z, crossing::Bool,
2           alt_inhibit::Bool, strength_prev::Uint8,
3           R_dz_lo::R, R_dz_hi::R, L_act::R, L_act_prev::R,
4           L_word_prev::R, L_crossing::R, L_alt_inhibit::R, L_strength_prev::R )
5   is_consistent::Bool = true
6   if ( (!isnan( R_dz_lo ) && (dz_own < R_dz_lo) )
7     || (!isnan( R_dz_hi ) && (R_dz_hi < dz_own))
8     || (!isnan( L_act ) && (action != int( L_act )))
9     || (!isnan( L_act_prev ) && (action_prev != int( L_act_prev )))
10    || (!isnan( L_word_prev ) && (word_prev != int( L_word_prev )))
11    || (!isnan( L_crossing ) && (crossing != bool( L_crossing )))
12    || (!isnan( L_alt_inhibit ) && (alt_inhibit != bool( L_alt_inhibit ))))
13    || (!isnan( L_strength_prev ) && (strength_prev != uint8( L_strength_prev ))))
14   is_consistent = false
15 end
16 return is_consistent::Bool
17 end
```

Referenced In: GetMatchingLabel270Rule(p. 285)

3.6.3 Determine Aural Inhibit

DETERMINEAURALINHIBIT (Algorithm 247) provides the logic for determining whether to turn off aural advisories. The ACAS X logic is similar to the existing TCAS logic. The setting of the *turn_off_aurals* flag is a function of the own altitude above ground according to the radar altimeter, *h_own*, and whether the aurals were disabled at the previous time step, *st_own.turn_off_aurals_prev*. DETERMINEAURALINHIBIT uses two thresholds for hysteresis, *H_no_aural_lo* and *H_no_aural_hi*.

Aural advisories shall be inhibited when own aircraft radio altitude (*h_own*) is less than or equal to the upper aural advisory inhibit limit altitude (*H_no_aural_hi*) and the aurals were previously inhibited (*st_own.turn_off_aurals_prev*), or whenever the own aircraft radio altitude is less than or equal to the lower aural advisory inhibit altitude (*H_no_aural_lo*).

This algorithm takes as input *h_own* and *st_own*. This algorithm returns *turn_off_aurals*. This algorithm updates *st_own*::**TRMOwnState**.

Algorithm 247 DetermineAuralInhibit

```

1 function DetermineAuralInhibit( h_own::R, st_own::TRMOwnState(p. E-32) )
2   const H_no_aural_lo::R = params().display.hnoaurallo
3   const H_no_aural_hi::R = params().display.hnoauralhi
4   turn_off_aurals::Bool = st_own.turn_off_aurals_prev
5   if isnan( h_own )
6     turn_off_aurals = false
7   elseif st_own.turn_off_aurals_prev
8     if (H_no_aural_hi <= h_own)
9       turn_off_aurals = false
10    end
11   elseif (h_own <= H_no_aural_lo)
12     turn_off_aurals = true
13   end
14   st_own.turn_off_aurals_prev = turn_off_aurals
15   return turn_off_aurals::Bool
16 end

```

Referenced In: DisplayLogicDetermination(p. 280)

3.6.4 Determine Traffic Alert

DETERMINETRAFFICALERT (Algorithm 248) determines whether to issue a traffic alert based on the display codes of the various intruders. The return value is set to TRUE whenever there is a traffic advisory (TA) on at least one intruder and there are no resolution advisories (RAs) on any other intruders.

The input, *code_int*, is a vector indicating the type of advisory for each intruder.

- *TACODE_RA* indicates a resolution advisory. The output value of *ta_alert* is always FALSE.
- *TACODE_TA_NOMINAL* indicates a traffic advisory based on nominal surveillance. The output value of *ta_alert* will be TRUE unless another intruder has an RA.
- *TACODE_TA_DEGRADED* indicates a traffic advisory based on degraded surveillance. The output value of *ta_alert* will be TRUE unless another intruder has an RA.
- *TACODE_PA* indicates a proximity advisory. This does not change the value of *ta_alert*.
- *TACODE_CLEAR* means no advisory. This does not change the value of *ta_alert*.

The value of *ta_alert* will be TRUE unless any intruder code is *TACODE_RA* or unless all intruder codes are *TACODE_CLEAR* or *TACODE_PA*.

A value of TRUE for *ta_alert* indicates a Traffic Advisory is in effect. The traffic advisory should be annunciated to the pilot only when the advisory is first initiated.

This algorithm takes as input *code_int*. This algorithm returns *ta_alert*.

Algorithm 248 DetermineTrafficAlert

```

1 function DetermineTrafficAlert( code_int::Vector<UInt8> )
2     ta_alert::Bool = false
3     for i = 1:length( code_int )
4         if (code_int[i] == TACODE_RA)
5             ta_alert = false
6             break
7         elseif (code_int[i] != TACODE_CLEAR) && (code_int[i] != TACODE_PA)
8             ta_alert = true
9         end
10    end
11    return ta_alert::Bool
12 end

```

Referenced In: DisplayLogicDetermination(p. 280)

3.7 Generate TRM Output

The GENERATETRMOUTPUT component of the Threat Resolution Module is shown in Figure 3-11.

The type of output processing to be performed during generation of the TRM Report is determined by a combination of the own aircraft Operational Mode and intruder-specific processing flags set in TRMINTRUDERSTATE, variable *st_int*. Two of the *st_int* fields that are important for TRM output processing are *st_int.no_alerts*, which indicates active processing for Designated No Alerts (DNA), and *st_int.processing*, which indicates processing for resolution advisories, traffic advisories only, or degraded surveillance. These fields in *st_int* are set by INTRUDERPREP (Algorithm 145), during preparation for TRM processing.

GENERATETRMOUTPUT (Algorithm 249) prepares the TRM results for output and generates a TRM Report with display, coordination, designation, and advisory information.

GENERATETRMOUTPUT uses four algorithms to prepare the TRM results for output in the TRM Report and to update the TRM state data structures. Within each algorithm intruder-specific advisory processing flags are used to further adjust the TRM Report outputs. These flags include *st_int.no_alerts*, which indicates active processing for Designated No Alerts (DNA), and *st_int.processing*, which indicates processing for resolution advisories, traffic advisories only, or degraded surveillance.

GENERATETAONLYMODEOUTPUT (Algorithm 250) is used when own aircraft is in TA-Only Operational Mode as indicated by a value of *OPMODE_TA* for *input_own.opmode*. In this mode, only traffic advisories can be produced by the ACAS X system. All resolution advisories produced during normal threat processing are converted to traffic advisories. There are no resolution advisories output to the cockpit, to other aircraft, or to the ground station. The exception being if an intruder is designated to the Designated No Alerts (DNA) Xo mode. In that case, no traffic advisories are produced for that intruder, but proximity advisories will still be produced when appropriate. If there was a resolution advisory on the previous cycle, all VRCs are cancelled and an RA terminated message is sent to the ground station. The global advisory state is reset. The individual intruder advisory state for each intruder is reset, except for the initialization cost. The initial state for the next processing cycle will be no advisory.

GENERATETARAMODEOUTPUT (Algorithm 253) is used when own aircraft is in TA/RA Opera-

tional Mode as indicated by a value of *OPMODE_RA* for *input_own.opmode*. In this mode, resolution advisories and traffic advisories can be produced by the ACAS X system. One exception to that is when an intruder has degraded surveillance that precludes generation of resolution advisories for that intruder. Another exception is when an intruder is designated to the Designated No Alerts (DNA) Xo mode. In that case, special processing for DNA Xo mode is in effect for that intruder. The initial state for the next processing cycle will be the current advisory, as adjusted for DNA Xo mode.

GENERATESTANDBYMODEOUTPUT (Algorithm 257) is used when own aircraft is in Standby Operational Mode as indicated by a value of *OPMODE_STANDBY* for *input_own.opmode*. In this mode, only proximity advisories can be produced by the ACAS X system. If there was a resolution advisory on the previous cycle, an RA terminated message is sent to the ground station. All intruders are invalid for TRM processing and are handled outside this algorithm. The global advisory state is completely reset. The initial state for the next processing cycle will be no advisory.

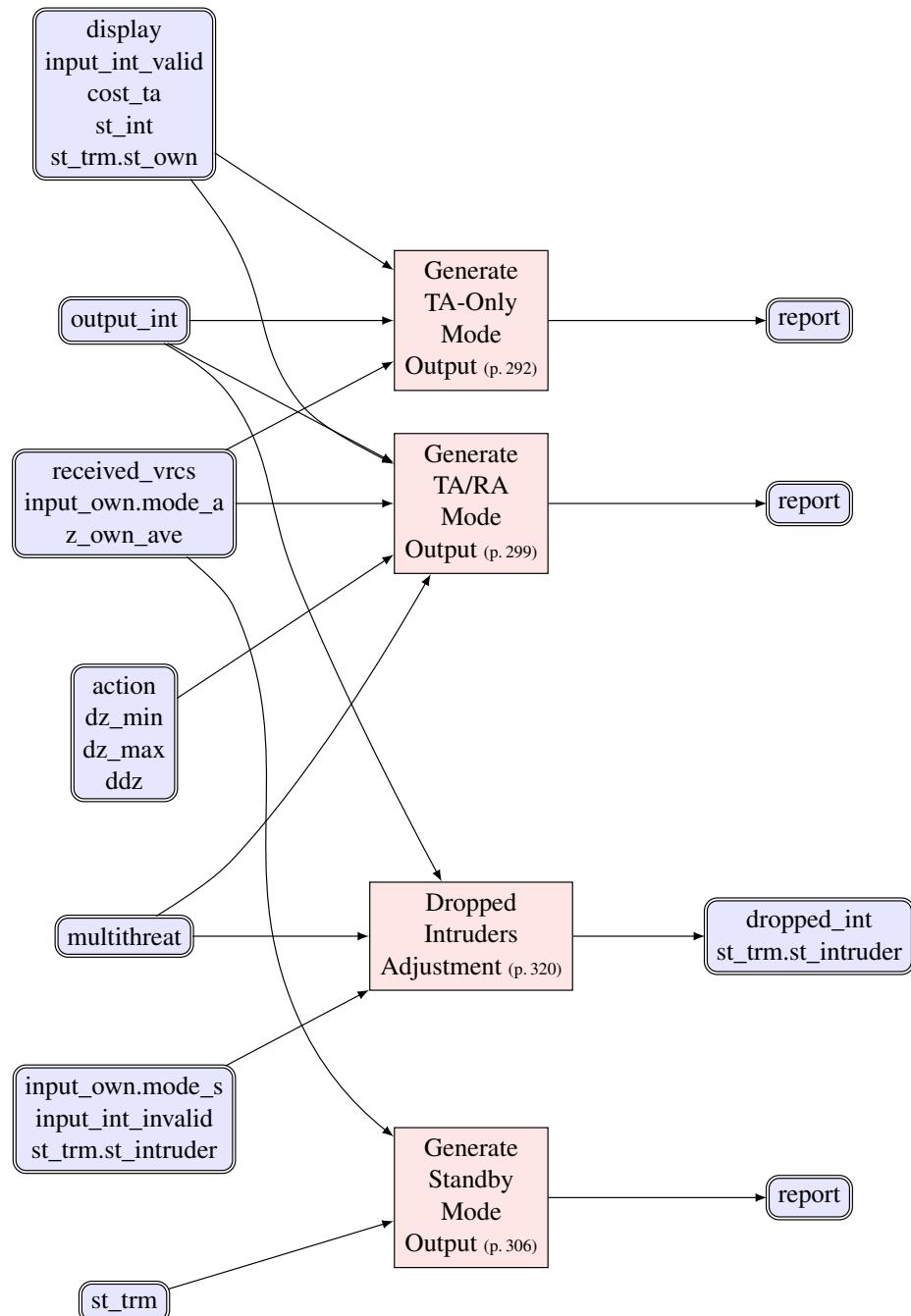
At this point, the **TRMREPORT** contains information about the global advisory and intruders that were evaluated for advisories.

DROPPEDINTRUDERSADJUSTMENT (Algorithm 270) generates report information for intruders that were not valid for TRM TA or RA processing on this cycle. Such intruders include intruders that were omitted by the STM from input to the TRM and intruders that were marked as invalid for TRM processing. If there was a resolution advisory on the previous cycle, VRCs for invalid intruders are cancelled. The individual intruder advisory state for each invalid and dropped intruder is reset. When an intruder is dropped, its state information is deleted unless that intruder is designated to an Xo mode.

After **DROPPEDINTRUDERSADJUSTMENT** is called, each dropped intruder output in the list is added to the TRM Report.

Finally, the ownership operational mode (*input_own.opmode*) is stored for use in the next cycle.

GENERATETRMOUTPUT takes as input *input_own*, *display*, *multithreat*, *action*, *dz_min*, *dz_max*, *ddz*, *z_own_ave*, *input_int_invalid*, *input_int_valid*, *output_int*, *cost_ta*, *received_vrcs*, *st_int*, and *st_trm*. Descriptions of the **GENERATETRMOUTPUT** output variables are found in Table 3-11.

**Figure 3-11. GenerateTRMOutput overview.****Table 3-11. TRM Algorithm Output Variables - GenerateTRMOutput**

Variable	Units	Type	Description
report	N/A	TRMReport(p. E-23)	Data for coordination and the onboard pilot display

Algorithm 249 GenerateTRMOutput

```

1 function GenerateTRMOutput( input_own::TRMOwnInput(p.E-19), display::DisplayLogic(p.E-35),
2                               multithreat::Bool, action::Z, dz_min::R, dz_max::R, ddz::R,
3                               z_own_ave::R,
4                               input_int_invalid::Vector{TRMIIntruderInput(p.E-20)},
5                               input_int_valid::Vector{TRMIIntruderInput(p.E-20)},
6                               output_int::Vector{TRMIIntruderData(p.E-39)},
7                               cost_ta::Matrix{R}, received_vrcs::Vector{Bool},
8                               st_int::Vector{TRMIIntruderState(p.E-33)}, st_trm::TRMState(p.E-31) )
9   report::TRMReport = TRMReport(p.E-23)()
10  if (OPMODE_TA == input_own.opmode)
11    report =
12      GenerateTAOnlyModeOutput(p.292)( display, input_int_valid, output_int,
13                                         cost_ta, received_vrcs, input_own.mode_a, z_own_ave,
14                                         st_int, st_trm.st_own )
15  elseif (OPMODE_RA == input_own.opmode)
16    report =
17      GenerateTARAModeOutput(p.299)( display, action, dz_min, dz_max, ddz,
18                                     input_int_valid, output_int, cost_ta, received_vrcs,
19                                     multithreat, input_own.mode_a, z_own_ave,
20                                     st_int, st_trm.st_own )
21  else
22    report =
23      GenerateStandbyModeOutput(p.306)( received_vrcs, input_own.mode_a, z_own_ave,
24                                         st_trm )
25  end
26  report.designation.availability = copy( input_own.xo_availability )
27  (dropped_int::Vector{TRMIIntruderData(p.E-39)}, st_trm.st_intruder) =
28    DroppedIntrudersAdjustment(p.320)( input_own.mode_s, multithreat, input_int_invalid,
29                                       output_int, st_trm.st_intruder )
30  for i in 1:length( dropped_int ) do
31    if (dropped_int[i].id == dropped_int[i].designation.id)
32      push!( report.display.intruder, dropped_int[i].display )
33      push!( report.coordination, dropped_int[i].coordination )
34      push!( report.debug.intruder, dropped_int[i].debug )
35    end
36    push!( report.designation.intruder, dropped_int[i].designation )
37  end
38  st_trm.st_own.opmode_prev = input_own.opmode
39  return (report::TRMReport(p.E-23))
40 end

```

Referenced In: VerticalTRMUpdate(p.152)
--

3.7.1 TA-Only Mode Processing

GENERATETAONLYMODEOUTPUT (Algorithm 250) is used when own aircraft is in TA-Only Operational Mode, and after the individual and global advisories are produced, to update and package the results for output in the TRM Report. It takes into account Xo mode designations and surveillance quality for each intruder.

In TA-Only Operational Mode, all resolution advisories (RAs) are converted to traffic advisories (TAs) for the pilot display. Coordination of resolution advisories with all intruders is disabled. In addition, when an individual intruder is designated to Designated No Alerts (DNA) Xo mode, any TA is converted to a proximity advisory or no advisory, as appropriate.

First, ADJUSTTAONLYMODEINTRUDEROUTPUT (Algorithm 251) is called to update the display, coordination, and designation settings for each intruder.

Once the outputs for each individual intruder have been adjusted, the outputs for the onboard pilot display are adjusted.



For the onboard pilot display, all RA output including the global RA, is disabled. The aural inhibit flag (*display.turn_off_aurals*) and information about presence of TAs (*ta_alert*) are preserved. The global RA is disabled by setting the Label 270 word to 0000 (*uint8(0)*) arguments to the DISPLAYLOGIC constructor).

Next, the TRMREPORT output (*report*) is created. The updated global display settings (*display*) and the vertical rates (*dz_min* and *dz_max*) associated with no advisory are used to initialize *report*. Then, for each intruder, the adjusted intruder entries for display, coordination, designation, and debug are added to *report*.

The state for own aircraft state (*st_own*) is then reset to remove RA history for this cycle. The history in *st_own.ra_output_prev* is preserved for use in SETRAMESSAGEOUTPUT.

Finally, SETRAMESSAGEOUTPUT is used to set the values in *report.ground_msg* and *report.broadcast*.

This algorithm takes as input *display*, *input_int_invalid*, *output_int*, *cost_ta*, *received_vrcs*, *mode_a*, *z_own_ave*, *st_int*, and *st_trm*.

The outputs are the TRM Report (*report*), modified output settings for each intruder (*output_int*), modified state settings for each intruder (*st_int*), and modified own state (*st_own*).

Algorithm 250 GenerateTAOnlyModeOutput

```

1 function GenerateTAOnlyModeOutput( display::DisplayLogic(p.E-35),
2                               input_int_valid::Vector{TRMIIntruderInput(p.E-20)},
3                               output_int::Vector{TRMIIntruderData(p.E-39)}, cost_ta::Matrix{R},
4                               received_vrcs::Vector{Bool}, mode_a::UInt32, z_own_ave::R,
5                               st_int::Vector{TRMIIntruderState(p.E-33)}, st_own::TRMOwnState(p.E-32 ) )
6   ta_alert::Bool =
7     AdjustTAOnlyModeIntruderOutput(p.294)( display, input_int_valid, output_int,
8                                         cost_ta, st_int )
9   display = DisplayLogic(p.E-35)( false, uint8(0), uint8(0), uint8(0), uint8(0),
10                           0.0, false, display.turn_off_aurals, ta_alert,
11                           uint8(0), false )
12  (dz_min::R, dz_max::R, ddz::R) =
13    ActionToRates(p.325)( COC, NaN, COC, NaN, NaN, NaN )
14  report::TRMReport = TRMReport(p.E-23)( display, dz_min, dz_max, ddz )
15  for i in 1:length(output_int) do
16    push!( report.display.intruder,      output_int[i].display )
17    push!( report.coordination,          output_int[i].coordination )
18    push!( report.designation.intruder, output_int[i].designation )
19    push!( report.debug.intruder,        output_int[i].debug )
20  end
21  st_own.action_prev = COC
22  st_own.word_prev = 0
23  st_own.crossing_prev = false
24  st_own.strength_prev = display.strength
25  st_own.ra_suppressed_prev = true
26  st_own.st_multithreat_cost_balancing = MultithreatCostBalancingCState(p.E-45)()
27  st_own.st_arbitrate = ActionArbitrationGlobalCState(p.E-41)()
28  (report.ground_msg::TRMGroundMsgData(p.E-27), report.broadcast::TRMRABroadcastData(p.E-27) ) =
29    SetRAMessageOutput(p.308)( -1, display, false, false, received_vrcs, false,
30                               input_int_valid, mode_a, z_own_ave, st_own, st_int )
31  st_own.a_prev = GlobalAdvisory(p.E-35)()
32  return (report::TRMReport(p.E-23))
33 end
```

Referenced In: GenerateTRMOutput(<i>p.291</i>)

ADJUSTTAONLYMODEINTRUDEROUTPUT (Algorithm 251) is used to adjust the intruder outputs when own aircraft is in TA-Only Operational Mode. It is used after the individual and global advisories are produced to update the intruder results in preparation for output in the TRM Report. It takes into account Xo mode designations and surveillance quality for each intruder.

In Global TA-Only processing mode, all resolution advisories (RAs) are converted to traffic advisories (TAs) for the pilot display. Coordination with all intruders is disabled. In addition, when an individual intruder is designated to Designated No Alerts (DNA) Xo mode, any TA is converted to a proximity advisory or no advisory, as appropriate.

The algorithm loops over all intruders in the output (*output_int*) and treats each intruder individually.

First, **CONVERTADVISORY** is called to adjust coordination settings, adjust RAs and TAs for the onboard pilot display, and initialize the traffic advisory timer for the individual intruder. If the intruder is DNA, **CONVERTADVISORY** downgrades both RAs and TAs. Otherwise, it converts RAs to TAs; TAs are not converted.

If there is a TA for any intruder (*is_ta* is TRUE), *ta_alert* will be set to TRUE. The return variable *isConverted_ra* is not used in Global TA-Only processing mode.

Next, for the individual intruder, the **TRMINTRUDERSTATEUPDATE** algorithm is used to update **TRMINTRUDERSTATE** for this intruder. Previous resolution advisories and the online cost state are cleared. However, the initialization delay counter (*T_count*) in **INITIALIZATIONCSTATE** is preserved to allow advisories to be generated on the next processing cycle. The initialization delay counter will be reset in **INTRUDERPREP** when ACAS X transitions out of Global TA-Only processing mode; there will be an enforced delay before any advisories are produced following that transition.

For the final step in the loop over intruders, the outputs for that intruder are set appropriately for TA-Only Operational Mode. There are no RAs, so all output variables associated with RAs are cleared (*st_int.is_threat*, *designation.active_ra*, *designation.suppressed_ra*, *designation.multithreat*, and *coordination.mtb*). The validity and status information for designation (*designation.valid* and *designation.status*) are set directly from the inputs.

This algorithm takes as input *display*, *input_int_valid*, *output_int*, *cost_ta*, and *st_int*. The outputs are a flag indicating whether there is a TA (*ta_alert*), modified output settings for each intruder (*output_int*), and modified state settings for each intruder (*st_int*).

Algorithm 251 AdjustTAOnlyModeIntruderOutput

```

1 function AdjustTAOnlyModeIntruderOutput( display::DisplayLogic(p. E-35),
2                                         input_int_valid::Vector{TRMIIntruderInput(p. E-20)},
3                                         output_int::Vector{TRMIIntruderData(p. E-39)}, cost_ta::Matrix{R},
4                                         st_int::Vector{TRMIIntruderState(p. E-33)} )
5   ta_alert::Bool = display.alert
6   for i in 1:length( output_int )
7     (isConverted_ra::Bool, is_ta::Bool) =
8       ConvertAdvisory(p.296)( cost_ta[i,COC], input_int_valid[i].degraded_surveillance,
9                               input_int_valid[i].is_proximate,
10                             (DESIGNATION_NONE != input_int_valid[i].designated_mode),
11                             output_int[i], false, st_int[i], input_int_valid[i].equipage )
12     if is_ta
13       ta_alert = true
14     end
15     TRMIIntruderStateUpdate(p.344)( st_int[i], output_int[i].coordination.vrc,
16                                     output_int[i].coordination.cvc, input_int_valid[i].equipage,
17                                     output_int[i].coordination.coordination_msg,
18                                     true, false )
19     st_int[i].is_identified_threat           = false
20     output_int[i].designation.active_ra     = false
21     output_int[i].designation.suppressed_ra = false
22     output_int[i].designation.multithreat   = false
23     output_int[i].designation.valid[Xo_IDX_DNA] = input_int_valid[i].xo_valid[Xo_IDX_DNA]
24     output_int[i].designation.valid[Xo_IDX_CSP03k] = input_int_valid[i].xo_valid[Xo_IDX_CSP03k]
25     output_int[i].designation.status        = input_int_valid[i].xo_status
26     output_int[i].coordination.mtb         = false
27   end
28   return (ta_alert::Bool)
29 end

```

Referenced In: GenerateTAOnlyModeOutput(p. 292)

CONVERTADVISORY (Algorithm 252) is used by ADJUSTTAONLYMODEINTRUDEROUTPUT and ADJUSTTARAMODEINTRUDEROUTPUT to adjust the display, coordination, and designation output values for the individual intruder based on the TRM processing settings. CONVERTADVISORY returns flags to indicate whether a resolution advisory was converted (*isConverted_ra*) and whether the resulting advisory is a traffic advisory (*is_ta*).

Whenever the intruder is flagged as designated to Designated No Alerts (DNA) Xo mode (*st_int.no_alerts* is TRUE), both resolution advisories (RAs) and traffic advisories (TAs) are suppressed in the output. For both TA-Only Operational Mode (*RA_PROCESSING_GLOBAL_TA_Only*) and degraded surveillance (*RA_PROCESSING_DEGRADED_SURVEILLANCE*) processing modes, RAs are converted to TAs in the output. The active advisory processing mode is indicated using *st_int.processing*. Since DNA is more restrictive, the *st_int.no_alerts* flag is checked before *st_int.processing*.

When the intruder is flagged for DNA processing and own aircraft is not coordinating with the intruder (*not st_int.is_dna_coordination*), CROSSLINK is used to generate a CANCEL coordination message whenever there is an active VRC. Canceling the previous VRC allows the equipped intruder flexibility in choosing its own advisory; it also helps the equipped intruder with bookkeeping. Otherwise, the coordination message is disabled. Coordination messages are disabled by setting the values of both CVC and VRC to 0.

Next, if the advisory was an RA or a TA, it is downgraded to a proximity advisory or no advisory. That means the track display code and track display score are both downgraded to prevent TAs and RAs from being shown on the onboard pilot display and annunciated to the pilot. DETERMINECODE

and DETERMINESCORE are used to reset the intruder code and score values for the onboard pilot display. The call to DETERMINECODE sets the *ta_alert* argument to FALSE and the RA sense to *:None* to indicate no TA and no RA. When an RA is downgraded, *isConverted_ra* is set to TRUE. There is never a TA in this case, so *is_ta* is set to FALSE and the TA timer (*st_int.time_since_ta*) is reset.

When the intruder is not flagged for DNA processing but the active processing mode is either TA-Only Operational Mode or degraded surveillance, an RA is downgraded to a TA. As when the intruder is flagged for DNA processing, CROSSLINK is called to CANCEL any pre-existing VRC, allowing the equipped intruder flexibility in choosing its own advisory and helping it with book-keeping. If there was no pre-existing VRC, the coordination message is disabled. Next, an RA on the individual intruder is converted to a TA. That means the track display code and track display score are both downgraded to force TAs to be shown on the onboard pilot display and annunciated to the pilot instead of RAs. The call to DETERMINECODE sets the *ta_alert* argument to TRUE and the RA sense to *:None* to indicate a TA and no RA. When an RA is converted to a TA, the TA timer (*st_int.time_since_ta*) is initialized if it was previously unset. The TA timer limits the frequency at which 'Traffic, Traffic' will be annunciated to the pilot by persisting TAs for a fixed length of time.

At the end, the *is_ta* return value is set based on whether the resulting intruder output contains a TA.

This algorithm takes as input *cost_ta_indiv_coc*, *degraded_surveillance_int*, *is_proximate_int*, *is_designated_int*, *output_int*, *equip_int*, and *st_int*. This algorithm outputs *isConverted_ra* and *is_ta*.

Algorithm 252 ConvertAdvisory

```

1 function ConvertAdvisory( cost_ta_indiv_coc::R, degraded_surveillance_int::UInt16,
2                           is_proximate_int::Bool, is_designated_int::Bool, output_int::TRMIntruderData(p.E-39),
3                           equip_int::Bool, st_int::TRMIntruderState(p.E-33), equipage_int::Z )
4   isConverted_ra::Bool = false
5   is_ta::Bool          = false
6   output_int.designation.logic_mode.processing = st_int.processing
7   if st_int.no_alerts
8     if !st_int.is_dna_coordination
9       (output_int.coordination.cvc, output_int.coordination.vrc, output_int.coordination.vsb) =
10      Crosslink(p.270)( 0.0, 0.0, st_int.vrc_prev, st_int.cvc_prev, equipage_int )
11    end
12    if (TACODE_RA      == output_int.display.code) ||
13      (TACODE_TA_NOMINAL == output_int.display.code) ||
14      (TACODE_TA_DEGRADED == output_int.display.code)
15      if (TACODE_RA == output_int.display.code)
16        isConverted_ra = true
17      end
18      is_ta = false
19      output_int.display.code =
20      DetermineCode(p.276)( is_ta, :None, degraded_surveillance_int, isProximate_int )
21      output_int.display.tds =
22      DetermineScore(p.278)( cost_ta_indiv_coc, output_int.display.code, degraded_surveillance_int,
23                            isDesignated_int, equip_int )
24    end
25    st_int.time_since_ta = 0
26  elseif (RA_PROCESSING_GLOBAL_TA_Only == st_int.processing) ||
27    (RA_PROCESSING_DEGRADED_SURVEILLANCE == st_int.processing)
28    (output_int.coordination.cvc, output_int.coordination.vrc, output_int.coordination.vsb) =
29    Crosslink(p.270)( 0.0, 0.0, st_int.vrc_prev, st_int.cvc_prev, equipage_int )
30    if (TACODE_RA == output_int.display.code)
31      isConverted_ra = true
32      is_ta          = true
33      output_int.display.code =
34      DetermineCode(p.276)( is_ta, :None, degraded_surveillance_int, isProximate_int )
35      output_int.display.tds =
36      DetermineScore(p.278)( cost_ta_indiv_coc, output_int.display.code, degraded_surveillance_int,
37                            isDesignated_int, equip_int )
38    if (0 == st_int.time_since_ta)
39      st_int.time_since_ta = 1
40    end
41  end
42  if (TACODE_TA_NOMINAL == output_int.display.code) ||
43    (TACODE_TA_DEGRADED == output_int.display.code)
44    is_ta = true
45  end
46  return (isConverted_ra::Bool, is_ta::Bool)
47 end

```

Referenced In: AdjustTARAModeIntruderOutput(p.302), AdjustTAOnlyModeIntruderOutput(p.294)

3.7.2 TA/RA Mode Processing

GENERATETARAMODEOUTPUT (Algorithm 253) is used when own aircraft is in TA/RA Operational Mode, and after the individual and global advisories are produced, to update and package the results for output in the TRM Report. It takes into account Xo mode designation and surveillance quality for each intruder.

The ACAS X TA/RA Operational Mode produces resolution advisories (RAs) as well as traffic advisories (TAs). Within this mode of operation, individual intruders can be processed differently, depending on their Xo mode designation and surveillance quality. When an individual intruder is

designated to Designated No Alerts (DNA) Xo mode, any TA or RA is converted to a proximity advisory or no advisory, as appropriate. When an individual intruder has degraded surveillance insufficient for producing RAs, any RA is converted to a TA, or a proximity advisory or no advisory, if the intruder is also designated to DNA Xo mode.

First, ADJUSTTARAMODEINTRUDEROUTPUT (Algorithm 254) is called to update the display, coordination, and designation settings for each intruder.

Once all the intruders have been updated individually, the global outputs are updated.

The threat index (*idx_tid*) is assigned based on the threat information in *st_adj* in the following order:

1. A threat that wasn't previously reported; *st_adj.idx_threat_new* is greater than 0.
2. A threat that is designated to an Xo mode; *st_adj.idx_threat_designated* is greater than 0.
3. The threat that was last reported; *st_adj.idx_threat_last* is greater than 0.
4. Any other threat; in this case, *st_adj.idx_threat_upd* is greater than 0.
5. No threat; in this case, *st_adj.idx_threat_upd* has the value -1.

Next, the outputs for the onboard pilot display are adjusted by one of the following:

- All of the individual RAs were downgraded for intruders included in determination of the global RA (*converted_all_ras* is TRUE). The global RA is suppressed by setting the Label 270 word to 0000 (*uint8(0)* arguments to the DISPLAYLOGIC constructor). The aural inhibit flag (*display.turn_off_aurals*) and information about the presence of TAs (*ta_alert*) are preserved. The RA suppressed state flag (*st_own.ra_suppressed_prev*) is set to indicate this RA was suppressed.
- The global advisory is Clear of Conflict and either the previous RA was suppressed or all the intruders with individual RAs on the previous cycle were excluded from determination of this global Clear of Conflict (*excluded_all_cocs* is TRUE). The Clear of Conflict annunciation is disabled. The aural inhibit flag (*display.turn_off_aurals*) and information about presence of TAs (*ta_alert*) are preserved so TAs will be annunciated. The Clear of Conflict is disabled by converting the Label 270 word of 1000 to 0000 so 'Clear of Conflict' won't be annunciated. The advisory is not considered suppressed in this case, since there is no RA, so *st_own.ra_suppressed_prev* is reset to FALSE.
- Neither the global RA nor Clear of Conflict are suppressed. Only the TA alert flag (*display.alert*) is updated to suppress annunciation of 'Traffic, Traffic' when there is no global RA and all TAs were downgraded, as indicated by *ta_alert*. The outputs from DISPLAYLOGICDETERMINATION and TRACKTHREATASSESSMENT are otherwise unchanged. Since there is no suppressed RA in this case, *st_own.ra_suppressed_prev* is reset to FALSE.

In all cases, the stored state variables for the previous action and word are updated to reflect the actual outputs.

Next, the TRMREPORT output (*report*) is created. The updated global display settings (*display*) and the vertical rates (*dz_min* and *dz_max*) associated with the global advisory are used to initialize *report*. Then, for each intruder, the adjusted intruder entries for display, coordination, designation, and debug are added to *report*.

Finally, SETRAMESSAGEOUTPUT is used to set the values in *report.ground_msg* and *report.broadcast*.

This algorithm takes as input *display*, *action*, *dz_min*, *dz_max*, *ddz*, *input_int_valid*, *output_int*, *cost_ta*, *received_vrcs*, *multithreat*, *mode_a*, *z_own_ave*, *st_int*, and *st_own*. The outputs are the TRM Report (*report*), modified output settings for each intruder (*output_int*), modified state settings for each intruder (*st_int*), and modified own state (*st_own*).

Algorithm 253 GenerateTARAModeOutput

```

1 function GenerateTARAModeOutput( display::DisplayLogic(p.E-35), action::Z, dz_min::R, dz_max::R, ddz::R,
2                               input_int_valid::Vector{TRMIIntruderInput(p.E-20)},
3                               output_int::Vector{TRMIIntruderData(p.E-39)}, cost_ta::Matrix{R},
4                               received_vrcs::Vector{Bool}, multithreat::Bool, mode_a::UInt32,
5                               z_own_ave::R, st_int::Vector{TRMIIntruderState(p.E-33)}, st_own::TRMOwnState(p.E-32) )
6   (st_adj::TRMIndivAdjustState(p.E-39), ta_alert::Bool) =
7     AdjustTARAModeIntruderOutput(p.302)( input_int_valid, output_int, cost_ta,
8                                         st_int, st_own )
9   if (0 < st_adj.idx_threat_new)
10    idx_tid = st_adj.idx_threat_new
11  elseif (0 < st_adj.idx_threat_designated)
12    idx_tid = st_adj.idx_threat_designated
13  elseif (0 < st_adj.idx_threat_last)
14    idx_tid = st_adj.idx_threat_last
15  else
16    idx_tid = st_adj.idx_threat_upd
17  end
18  is_ra_suppressed::Bool = false
19  converted_all_ras::Bool = (0 < st_adj.ra_change_count) &&
20    (st_adj.ra_change_count == st_adj.ra_count)
21  excluded_all_cocs::Bool = (0 < st_adj.force_silent_count) &&
22    (st_adj.force_silent_count == st_adj.ra_prev_count)
23  if converted_all_ras
24    display = DisplayLogic(p.E-35)( false, uint8(0), uint8(0), uint8(0), uint8(0),
25                                0.0, false, display.turn_off_aurals, ta_alert,
26                                display.strength, display.down )
27    is_ra_suppressed = true
28  elseif (COC == action) && (st_own.ra_suppressed_prev || excluded_all_cocs)
29    display = DisplayLogic(p.E-35)( false, uint8(0), uint8(0), uint8(0), uint8(0),
30                                0.0, false, display.turn_off_aurals, ta_alert,
31                                display.strength, display.down )
32    is_ra_suppressed = false
33  else
34    display.alert = ta_alert
35    is_ra_suppressed = false
36  end
37  st_own.action_prev = action
38  st_own.word_prev = (display.cc * 1000) + (display.vc * 100) + (display.ua * 10) + display.da
39  st_own.strength_prev = display.strength
40  report::TRMReport = TRMReport(p.E-23)( display, dz_min, dz_max, ddz )
41  for i in 1:length( output_int )
42    push!( report.display.intruder,      output_int[i].display )
43    push!( report.coordination,        output_int[i].coordination )
44    push!( report.designation.intruder, output_int[i].designation )
45    push!( report.debug.intruder,       output_int[i].debug )
46  end
47  conflicting_senses::Bool = (st_adj.upsense && st_adj.downsense)
48  (report.ground_msg::TRMGroundMsgData(p.E-27), report.broadcast::TRMRABroadcastData(p.E-27)) =
49    SetRAMessageOutput(p.308)( idx_tid, display, multithreat, !conflicting_senses, received_vrcs,
50                               is_ra_suppressed, input_int_valid, mode_a, z_own_ave, st_own, st_int )
51  st_own.a_prev.action = action
52  st_own.a_prev.dz_min = dz_min
53  st_own.a_prev.dz_max = dz_max
54  st_own.a_prev.ddz = ddz
55  st_own.a_prev.multithreat = multithreat
56  st_own.ra_suppressed_prev = is_ra_suppressed
57  return (report::TRMReport(p.E-23))
58 end

```

Referenced In: GenerateTRMOutput(p.291)
--

ADJUSTTARAMODEINTRUDEROUTPUT (Algorithm 254) is used to adjust the intruder outputs when own aircraft is in TA/RA Operational Mode. It is used, after the individual and global advi-

sories are produced, to update the intruder results in preparation for output in the TRM Report. It takes into account Xo mode designations and surveillance quality for each intruder.

Global TA/RA processing mode is the nominal processing mode for the TRM. However, when an individual intruder is designated to Designated No Alerts (DNA) Xo mode, any TA is converted to a proximity advisory or no advisory, as appropriate.

The algorithm loops over all intruders in the output (*output_int*) and treats each intruder individually. The number of intruders in *output_int* is given by *N_intruders*.

For each individual intruder, UPDATEINDIVADJUSTTHREATINFO is first called to determine if the intruder should have DNA Xo mode processing applied based on the current resolution advisory (RA) state. For example, when there is a multithreat RA that includes the DNA intruder, its designation will be temporarily suspended. If not, the DNA state and status settings (*st_int.no_alerts* and *output_int.designation.status*) are updated. UPDATEINDIVADJUSTTHREATINFO is also used to report whether the intruder was excluded from determination of the global RA (*was_excluded*) and to set the indices and flags in *st_adj* used within SETRAMESSAGEOUTPUT.

Next, when the individual intruder was excluded from determination of the global RA or is DNA, adjustments are needed to the intruder outputs. CONVERTADVISORY is called to adjust coordination settings, adjust RAs and TAs for the onboard pilot display, and initialize the traffic advisory timer for the individual intruder. If the intruder is DNA, CONVERTADVISORY downgrades both RAs and TAs. Otherwise, if the intruder was excluded from production of the global RA and is not DNA, CONVERTADVISORY converts RAs to TAs; TAs are not converted. If an RA was downgraded, *isConverted_ra* is TRUE. If there is a TA for any intruder, *is_ta* is TRUE. A traffic advisory (*ta_alert*) can be issued only if there is at least one TA and no RAs (*ra_found* is FALSE). If any RAs are found that are not converted, *ta_alert* is set to FALSE. Until an RA is found, any TA that is found results in the *ta_alert* flag being set to TRUE. The *ta_alert* flag will be used later to update the global advisory issued for the onboard pilot display.

UPDATEINDIVADJUSTCOUNTS is called next for the individual intruder to update the counts used to determine whether the global RA should be suppressed as a result of designation or if a Clear of Conflict annunciation should be suppressed due to a previously suppressed RA or degraded surveillance. UPDATEINDIVADJUSTCOUNTS returns whether the individual intruder advisory state should be reset at the end of the intruder processing loop (*advisory_reset*).

The intruder outputs for the individual intruder are next updated as appropriate. If the intruder was excluded from the global RA, it cannot have an active RA (*designation.active_ra* is FALSE) and the logic mode processing type (*designation.logic_mode_processing*) original setting (*RA_PROCESSING_DEGRADED_SURVEILLANCE*) remains in effect. Otherwise, the logic mode processing was global TA/RA processing, even if the intruder was DNA; *designation.logic_mode_processing* is set to *RA_PROCESSING_GLOBAL_TARA*. DNA is indicated in the logic mode (*designation.logic_mode.dna* is TRUE), as is the protection mode (*designation.logic_mode.protection_mode*). If the intruder is DNA and the RA was downgraded, it is indicated in the designation output (*designation.ra_suppressed* is TRUE). The designation validity outputs (*designation.xo_valid[Xo_IDX_DNA]* and *designation.xo_valid[Xo_IDX_CSPO3k]*) are adjusted to prevent undesignation by the ASSAP function in the ASA System when the specified designation mode was actually active during this processing cycle. Otherwise, the input validity value is stored as the output validity.

For the final step in the loop over the intruders, TRMINTRUDERSTATEUPDATE is used to update TRMINTRUDERSTATE for each intruder. Whenever *advisory_reset* is TRUE, the previous resolution advisory and the online cost state are cleared for this intruder. However, the initialization

delay counter (T_count) in INITIALIZATIONSTATE is preserved to allow advisories to be generated on the next processing cycle. The initialization delay counter will be reset in INTRUDERPREP when the surveillance is no longer degraded; there will be an enforced delay before any advisories are produced following that transition. T_count is not reset for DNA so an RA can be produced immediately if DNA is undesignated.

This algorithm takes as input $input_int_valid$, $output_int$, $cost_ta$, st_int , and st_own . The outputs are state variables for GENERATETARAMODEOUTPUT (st_adj), a flag indicating whether there is a TA (ta_alert), modified output settings for each intruder ($output_int$), and modified state settings for each intruder (st_int).

Algorithm 254 AdjustTARAModeIntruderOutput

```

1 function AdjustTARAModeIntruderOutput( input_int_valid::Vector{TRMIIntruderInput(p.E-20)}, 
2                               output_int::Vector{TRMIIntruderData(p.E-39)}, cost_ta::Matrix{R}, 
3                               st_int::Vector{TRMIIntruderState(p.E-33)}, st_own::TRMOwnState(p.E-32) )
4   const S_standard_protection_mode::Uint8 = uint8(params().target_designation.S_standard_protection_mode)
5   const N_intruders::Z = length( output_int )
6   st_adj::TRMIndivAdjustState = TRMIndivAdjustState(p.E-39)( N_intruders )
7   ta_alert::Bool = false
8   ra_found::Bool = false
9   for i in 1:N_intruders
10    (was_excluded::Bool, st_int[i].no_alerts, output_int[i].designation.status) =
11      UpdateIndivAdjustThreatInfo(p.304)( st_own, i, input_int_valid[i], output_int[i], st_int[i], st_adj )
12    isConverted_ra::Bool = false
13    is_ta::Bool = false
14    if was_excluded || st_int[i].no_alerts
15      equip_int::Bool = (EQUIPAGE_CASRA == input_int_valid[i].equipage)
16      (isConverted_ra, is_ta) =
17        ConvertAdvisory(p.296)( cost_ta[i,COC], input_int_valid[i].degraded_surveillance,
18                                input_int_valid[i].is_proximate,
19                                (DESIGNATION_NONE != input_int_valid[i].designated_mode),
20                                output_int[i], equip_int, st_int[i], input_int_valid[i].equipage )
21      if !ra_found && is_ta
22        ta_alert = true
23      end
24    elseif (TACODE_RA == output_int[i].display.code)
25      ra_found = true
26      ta_alert = false
27    elseif !ra_found &&
28      ((TACODE_TA_NOMINAL == output_int[i].display.code) ||
29       (TACODE_TA_DEGRADED == output_int[i].display.code))
30      ta_alert = true
31    end
32    is_advisory_reset::Bool =
33      UpdateIndivAdjustCounts(p.305)( was_excluded, isConverted_ra, st_int[i], st_adj )
34    if was_excluded
35      output_int[i].designation.active_ra = false
36    else
37      output_int[i].designation.logic_mode.processing = RA_PROCESSING_GLOBAL_TARA
38      if isConverted_ra
39        output_int[i].designation.suppressed_ra = true
40      end
41    end
42    output_int[i].designation.logic_mode.dna = st_int[i].no_alerts
43    if output_int[i].designation.logic_mode.dna
44      output_int[i].designation.valid[Xo_IDX_DNA] = true
45    else
46      output_int[i].designation.valid[Xo_IDX_DNA] = input_int_valid[i].xo_valid[Xo_IDX_DNA]
47    end
48    if (output_int[i].designation.logic_mode.protection_mode != S_standard_protection_mode)
49      output_int[i].designation.valid[Xo_IDX_CSP03k] = true
50    else
51      output_int[i].designation.valid[Xo_IDX_CSP03k] = input_int_valid[i].xo_valid[Xo_IDX_CSP03k]
52    end
53    TRMIIntruderStateUpdate(p.344)( st_int[i], output_int[i].coordination.vrc,
54                                    output_int[i].coordination.cvc, input_int_valid[i].equipage,
55                                    output_int[i].coordination.coordination_msg,
56                                    is_advisory_reset, false )
57  end
58  return ( st_adj::TRMIndivAdjustState(p.E-39), ta_alert::Bool)
59 end

```

Referenced In:	GenerateTARAModeOutput(p.299)
----------------	-------------------------------

UPDATEINDIVADJUSTTHREATINFO (Algorithm 255) sets the indices and flags needed for prepar-

...
...

ing TRM Report content in GENERATETARAMODEOUTPUT.

First, UPDATEINDIVADJUSTTHREATINFO determines whether this intruder should receive degraded surveillance output processing and sets *was_excluded* accordingly.

Next, the algorithm determines if this intruder is one of the threats in a multi-threat RA on the current cycle or on the previous cycle. If it is and the intruder is designated to Designated No Alerts (DNA) Xo mode, DNA Xo mode processing is temporarily suspended for this intruder (*no_alerts* is set to FALSE). The status output is set to indicate the reason DNA Xo mode processing is suspended for this intruder for this cycle, *xo_status* is set to *Xo_STATUS_SUSPENDED_MULTITHREAT*.

The remainder of algorithm gathers information to be used when preparing the resolution advisory output to the ground station.

If there is an individual resolution advisory for this intruder, the intruder is flagged as a threat and the sense of the advisory is recorded. Next, indexing information is recorded for use determining which threat will be reported in the Threat Identifier (TID) in the output to the ground station. The index into the intruder array is registered as being for a previously unidentified threat (*st_adj.idx_threat_new*), the threat reported in the threat identifier (TID) on the last cycle (*st_adj.idx_threat_last*), or a threat that was previously reported (*st_adj.idx_threat_upd*). If this intruder is designated to an Xo mode and the designation is in force, the index of the threat is registered as being for a designated intruder (*st_adj.idx_threat_designated*). Whenever an intruder is identified as a new threat, it is flagged as an identified threat. Only one intruder can be flagged as a new threat each TRM cycle.

If there is no individual resolution advisory for this intruder, the intruder is flagged as not being a threat and the identified threat flag is reset.

UPDATEINDIVADJUSTTHREATINFO takes as input *st_own*, *idx*, *input_int_valid*, *output_int*, *st_int*, and *st_adj*. This algorithm outputs a flag for degraded surveillance processing (*was_excluded*), a flag for DNA processing (*no_alerts*), the xo status (*xo_status*), and the individual adjust state (*st_adj*).

Algorithm 255 UpdateIndivAdjustThreatInfo

```

1 function UpdateIndivAdjustThreatInfo( st_own::TRMOwnState(p. E-32), idx::Z,
2                               input_int_valid::TRMIintruderInput(p. E-20),
3                               output_int::TRMIintruderData(p. E-39), st_int::TRMIintruderState(p. E-33),
4                               st_adj::TRMIndivAdjustState(p. E-39) )
5   is_multithreat_ra::Bool = false
6   xo_status::UInt8        = input_int_valid.xo_status
7   no_alerts::Bool          = st_int.no_alerts
8   was_excluded::Bool       = (RA_PROCESSING_DEGRADED_SURVEILLANCE == st_int.processing)
9   if (output_int.designation.multithreat && output_int.designation.active_ra) ||
10    ((COC != st_int.a_prev.action) && st_own.a_prev.multithreat)
11    is_multithreat_ra = true
12  end
13  if input_int_valid.dna && is_multithreat_ra
14    no_alerts = false
15    xo_status = Xo_STATUS_SUSPENDED_MULTITHREAT
16  end
17  if (:None != output_int.sense) && !was_excluded
18    st_int.is_threat = true
19    if (:Up == output_int.sense)
20      st_adj.upsense = true
21    else
22      st_adj.downsense = true
23    end
24    if (st_adj.idx_threat_new <= 0) && !st_int.is_identified_threat
25      st_adj.idx_threat_new = idx
26      st_int.is_identified_threat = true
27    elseif (output_int.id == st_own.ra_output_prev.tgtid_tid)
28      st_adj.idx_threat_last = idx
29    else
30      st_adj.idx_threat_upd = idx
31    end
32    if IsDesignationInForce(p. 311)( output_int.designation.multithreat, input_int_valid, st_int )
33      st_adj.idx_threat_designated = idx
34    end
35  else
36    st_int.is_threat          = false
37    st_int.is_identified_threat = false
38  end
39  return (was_excluded::Bool, no_alerts::Bool, xo_status::UInt8)
40 end

```

Referenced In: AdjustTARAModeIntruderOutput(p. 302)

UPDATEINDIVADJUSTCOUNTS (Algorithm 256) updates counters used when updating DISPLAYLOGIC values in GENERATETARAMODEOUTPUT.

The counters set in this algorithm are stored in a local data structure, *st_adj* of type TRMINDIVADJUSTSTATE.

The *ra_change_count* field is incremented whenever a resolution advisory (RA) that had been included in the global RA determination has been downgraded. This counter is exclusively associated with DNA intruders that have surveillance of nominal quality. The *ra_change_count* field is used in conjunction with the *ra_count* field to determine when the global resolution advisory should be suppressed at the onboard pilot display. *ra_count* is the number of RAs included in the global RA determination. It is initialized in GENERATETARAMODEOUTPUT to the number of intruders. *ra_count* is decremented for every intruder with an individual Clear of Conflict (COC) action and for every intruder that was excluded from the global RA determination (*was_excluded*). When *ra_change_count* and *ra_count* both have the same value, all RAs were downgraded and the RA will

be suppressed in the display output.

A reset of the individual intruder advisory state (*advisory_state*) is indicated whenever either this intruder was excluded from the global RA determination or DNA is currently active for this intruder and there is no coordination between own aircraft and this intruder. When own aircraft is coordinating with the intruder, the coordination state must be maintained from one cycle to the next and an advisory reset is counterindicated.

At the end of the algorithm, the fields *ra_prev_count* and *force_silent_count* are set. *ra_prev_count* is used in conjunction with *force_silent_count* to determine when annunciation of Clear of Conflict should be disabled. The field *ra_prev_count* is incremented whenever there was an RA for this intruder on the previous cycle. When there was an RA for this intruder on the previous cycle and this intruder was excluded from determination of the global RA on this cycle, *force_silent_count* is also incremented. When *ra_prev_count* and *force_silent_count* have the same value, all previous RAs were excluded from determination of the current global Clear of Conflict.

UPDATEINDIVADJUSTCOUNTS takes as input *was_excluded*, *isConverted_ra*, *st_int*, and *st_adj*. This algorithm returns whether an advisory reset is needed (*advisory_reset*) due to output processing for degraded surveillance or Designated No Alerts (DNA) and the adjust state data (*st_adj*).

Algorithm 256 UpdateIndivAdjustCounts

```

1 function UpdateIndivAdjustCounts( was_excluded::Bool, isConverted_ra::Bool,
2                                st_int::TRMIintruderState(p. E-33), st_adj::TRMIndivAdjustState(p. E-39 ) )
3     if isConverted_ra && !was_excluded
4         st_adj.ra_change_count = st_adj.ra_change_count + 1
5     end
6     is_advisory_reset::Bool = false
7     if was_excluded || (st_int.no_alerts && !st_int.is_dna_coordination)
8         is_advisory_reset = true
9     end
10    if IsCOC(p.332)( st_int.a_prev.dz_min, st_int.a_prev.dz_max ) || was_excluded
11        st_adj.ra_count = st_adj.ra_count - 1
12        if st_int.a_prev.ra_prev
13            st_adj.ra_prev_count = st_adj.ra_prev_count + 1
14            if was_excluded
15                st_adj.force_silent_count = st_adj.force_silent_count + 1
16            end
17        end
18    end
19    return (is_advisory_reset::Bool)
20 end
```

Referenced In: AdjustTARAModeIntruderOutput(<i>p. 302</i>)
--

3.7.3 Standby Mode Processing

GENERATESTANDBYMODEOUTPUT (Algorithm 257) is used when own aircraft is in Standby Operational Mode to package results for the TRM Report.

In Standby Operational Mode, only proximity advisories can be produced by the ACAS X system. There is no resolution advisory input to this algorithm. There are no intruder inputs to this algorithm, all intruders are invalid for TRM processing and are handled within DROPPEDINTRUDERSADJUSTMENT.

SETRAMESSAGEOUTPUT is used to set the values in *report.ground_msg* and *report.broadcast*.

.....

The inputs to SETRAMESSAGEOUTPUT indicate there is no resolution advisory for this cycle. If there was a resolution advisory on the previous cycle, an RA terminated message is sent to the ground station.

The global advisory state is completely reset. The initial state for the next processing cycle will be no advisory.

For every intruder state, the traffic advisory timer (*st_trm.st_intruder.time_since_ta*) and the previous resolution advisory (*st_trm.st_intruder.sense_prev*) are cleared. Setting these variables in this way ensures that no traffic advisories will be produced in SETINVALIDINTRUDER.

This algorithm takes as input *received_vrcs*, *mode_a*, *z_own_ave*, and *st_trm*. The outputs are the TRM Report (*report*), modified state settings for each intruder (*st_int*), and modified own state (*st_own*).

Algorithm 257 GenerateStandbyModeOutput

```

1 function GenerateStandbyModeOutput( received_vrcs::Vector{Bool}, mode_a::UInt32, z_own_ave::R,
2                                     st_trm::TRMState(p.E-31) )
3   report::TRMReport           = TRMReport(p.E-23)()
4   display::DisplayLogic        = DisplayLogic(p.E-35)()
5   input_int_valid::Vector{TRMIntruderInput} = Array( TRMIntruderInput(p.E-20), 0 )
6   st_int::Vector{TRMIntruderState} = Array( TRMIntruderState(p.E-33), 0 )
7   (report.ground_msg::TRMGroundMsgData(p.E-27), report.broadcast::TRMRABroadcastData(p.E-27)) =
8     SetRAMessageOutput(p.308)( -1, display, false, false, received_vrcs, false,
9                               input_int_valid, mode_a, z_own_ave, st_trm.st_own, st_int )
10  st_trm.st_own.a_prev = GlobalAdvisory(p.E-35)()
11  for i in 1:length( st_trm.st_intruder )
12    st_trm.st_intruder[i].time_since_ta = 0
13    st_trm.st_intruder[i].sense_prev   = :None
14  end
15  return (report::TRMReport(p.E-23))
16 end
```

Referenced In: GenerateTRMOutput(p.291)

3.7.4 Resolution Advisory Output Messages

Collision Avoidance Systems employ four types of messages to communicate resolution advisories:

- TCAS Resolution Advisory Messages used for air-to-air coordination. The data needed for these messages is contained in TRMCOORDINATIONINTERROGATIONDATA. The data are transmitted when *vrc* or *cvc* contain a value other than 0. The fields in this data structure are populated using CROSSLINK (Algorithm 234). The CROSSLINK algorithm is used in several different sections in this document.
- Resolution Advisory (RA) Report Messages used to provide resolution advisory information to a Mode S ground sensor. The data needed for these messages is contained in TRM-GROUNDMSGDATA. The data are transmitted when *ra_data.avra_strength* contains a value other than 0 or the *ra_data.rat* bit is set. The fields in this data structure are populated using SETGROUNDMSGDATA (Algorithm 262).
- Broadcast Interrogation Messages used to indicate the presence and identity of own aircraft and to allow monitoring of resolution advisory information. The data needed for these messages is contained in TRMRABROADCASTDATA. The data are transmitted when *ra_data.avra_strength* contains a value other than 0 or the *ra_data.rat* bit is set. The fields in this data



structure are populated using SETRABROADCASTDATA (Algorithm 268).

- Coordination Reply Messages used as a handshake and to allow monitoring of resolution advisory information. The data needed for these messages is contained in TRMGROUNDMSGDATA. The data are transmitted when *ra_data.avra_strength* contains a value other than 0 or the *ra_data.rat* bit is set. The fields in this data structure are populated using SETGROUNDMSGDATA (Algorithm 262).

TRMRACOORDINATIONDATA (Type 42) encapsulates the Active Vertical Resolution Advisory (AVRA), Resolution Advisory Message Format (RMF), Resolution Advisory Complements (RAC), Resolution Advisory Terminated indicator (RAT), and Multiple Threat Encounter (MTE) values used in both TRMGROUNDMSGDATA and TRMRABROADCASTDATA.

This section describes how the fields in TRMRACOORDINATIONDATA, TRMGROUNDMSGDATA, and TRMRABROADCASTDATA are populated.

SETRAMESSAGEOUTPUT (Algorithm 258) populates the TRMGROUNDMSGDATA and TRMRABROADCASTDATA data structures.

This algorithm calls three algorithms to populate the TRMGROUNDMSGDATA and TRMRABROADCASTDATA data structures for the TRM Report.

DETERMINERACOORDINATIONDATA (Algorithm 259) populates the TRMRACOORDINATIONDATA data structure (*ra_data*) that is used in both TRMGROUNDMSGDATA and TRMRABROADCASTDATA.

SETGROUNDMSGDATA (Algorithm 262) populates the TRMGROUNDMSGDATA data structure, *ground_msg*.

SETRABROADCASTDATA (Algorithm 268) populates the TRMRABROADCASTDATA data structure, *broadcast*.

This algorithm takes as input *idx_tid*, *display*, *multithreat*, *single_sense*, *received_vrcs*, *is_ra_suppressed*, *input_int_valid*, *mode_a*, *z_own_ave*, *st_own*, and *st_int*. The outputs from SETRAMESSAGEOUTPUT are populated data structures for TRMGROUNDMSGDATA (*ground_msg*) and TRMRABROADCASTDATA (*broadcast*).

Algorithm 258 SetRAMessageOutput

```

1 function SetRAMessageOutput( idx_tid::Z, display::DisplayLogic(p. E-35), multithreat::Bool, single_sense::Bool,
2                               received_vrcs::Vector{Bool}, is_ra_suppressed::Bool,
3                               input_int_valid::Vector{TRMIIntruderInput(p. E-20)}, mode_a::UInt32, z_own_ave::R,
4                               st_own::TRMOwnState(p. E-32), st_int::Vector{TRMIIntruderState(p. E-33)} )
5   ra_data::TRMRACoordinationData(p. E-28) =
6     DetermineRACoordinationData(p. 309)( st_own, display, multithreat,
7                                         single_sense, received_vrcs )
8   ground_msg::TRMGroundMsgData(p. E-27) =
9     SetGroundMsgData(p. 312)( ra_data, idx_tid, input_int_valid, z_own_ave, is_ra_suppressed,
10                           st_own, st_int )
11  broadcast::TRMRABroadcastData(p. E-27) =
12    SetRABroadcastData(p. 317)( ra_data, is_ra_suppressed, input_int_valid, mode_a, z_own_ave,
13                            st_own, st_int )
14  return (ground_msg::TRMGroundMsgData(p. E-27), broadcast::TRMRABroadcastData(p. E-27))
15 end

```

Referenced In: GenerateTAOnlyModeOutput(p. 292), GenerateTARAModeOutput(p. 299), GenerateStandbyModeOutput(p. 306)

3.7.4.1 Common RA Output Message Data

DETERMINERACOORDINATIONDATA (Algorithm 259) populates the TRMRACOORDINATIONDATA data structure with information about the active resolution advisory. This data structure is used in both TRMGROUNDMSGDATA and TRMRABROADCASTDATA in the TRM Report.

The inputs are own aircraft state (*st_own*), the resolution advisory information for display (*display*), whether or not the advisory is a multi-threat advisory (*multithreat*), whether or not the advisory has a single sense or conflicting senses (*single_sense*), and the received VRCs to be used for RAC (*received_vrcs*).

The low-level descend inhibit value (*ra_data.ldi*) is taken from the ownship altitude inhibit settings (*st_own.st_alt_inhibit*) for the Xa protection mode.

The remaining fields in the data structure are set based on the resolution advisory state as follows:

- There is an active resolution advisory. The fields are set directly from the inputs. The new field settings are recorded in *st_own.ra_output_prev.ra_data* for use when the RA is terminated.
- There was an active resolution advisory on the previous TRM cycle. The fields settings from the previous cycle are used except RA termination is indicated by setting *rat* to TRUE. Cleared field settings are recorded in *st_own.ra_output_prev.ra_data*.
- There is no resolution advisory present or past. The field settings are cleared except the *rac* field is set from *received_vrcs*. The new field settings are recorded in *st_own.ra_output_prev.ra_data*.

This algorithm takes as input *st_own*, *display*, *multithreat*, *single_sense*, and *received_vrcs*. The output of DETERMINERACOORDINATIONDATA (Algorithm 259) is a populated TRMRACOORDINATIONDATA data structure (*ra_data*).

Algorithm 259 DetermineRACoordinationData

```

1 function DetermineRACoordinationData( st_own::TRMOwnState(p. E-32), display::DisplayLogic(p. E-35),
2                                     multithreat::Bool, single_sense::Bool, received_vrcs::Vector{Bool} )
3   ra_data::TRMRACoordinationData = TRMRACoordinationData(p. E-28)()
4   ra_data.rmf = RMF_ACAS_Xa
5   mode_idx::Z = GetProtectionModeIndex(p. 328)( PROTECTION_MODE_Xa )
6   ra_data.ldi = st_own.st_alt_inhibit[mode_idx].ldi
7   if (0 < display.strength)
8     ra_data.avra_single_intent = single_sense
9     ra_data.avra_down = display.down
10    ra_data.avra_strength = display.strength
11    ra_data.avra_crossing = display.crossing
12    ra_data.rat = false
13    ra_data.mte = multithreat
14    ra_data.rac = copy(received_vrcs)
15    st_own.ra_output_prev.ra_data = deepcopy(ra_data)
16   elseif (0 < st_own.ra_output_prev.ra_data.avra_strength)
17     ra_data = deepcopy(st_own.ra_output_prev.ra_data)
18     ra_data.rat = true
19     st_own.ra_output_prev.ra_data = TRMRACoordinationData(p. E-28)()
20     st_own.ra_output_prev.ra_data.rmf = RMF_ACAS_Xa
21   else
22     ra_data.avra_single_intent = false
23     ra_data.avra_down = false
24     ra_data.avra_strength = 0
25     ra_data.avra_crossing = false
26     ra_data.rat = false
27     ra_data.mte = false
28     ra_data.rac = copy(received_vrcs)
29     st_own.ra_output_prev.ra_data = deepcopy(ra_data)
30   end
31   return ra_data::TRMRACoordinationData(p. E-28)
32 end

```

Referenced In: SetRAMessageOutput(p. 308)

DETERMINESPI (Algorithm 260) determines the Suppression Indicator (SPI) setting.

The inputs are whether the resolution advisory is suppressed (*is_ra_suppressed*), the setting of the multithreat bit (*mte*), the threat index for the most recent threat (*idx_tid*), the intruder intputs from the STM for all valid intruders (*input_int_valid*), and the intruder state for all valid intruders (*st_int*). The setting of the multithreat bit indicates whether or not the advisory is a multi-threat advisory. The most recent threat is the one identified in the Threat ID (TID), when applicable.

When the resolution advisory is suppressed, SPI is set to TRUE.

Otherwise, if the resolution advisory is a multi-threat advisory, SPI is set to TRUE only if a threat is designated to an Xo mode and that designation is in force. The threat associated with the threat index is excluded from this determination because the designation information is captured elsewhere in the message. The threat index is set to -1 in the inputs to indicate designation information is not captured elsewhere in the message, as well as to indicate there is no threat.

If neither of those two cases apply, SPI is FALSE.

This algorithm takes as inputs *is_ra_suppressed*, *mte*, *idx_tid*, *input_int_valid*, and *st_int*. The output of DETERMINESPI is the suppression bit setting (*spi*).

Algorithm 260 DetermineSPI

```

1 function DetermineSPI( is_ra_suppressed::Bool, mte::Bool, idx_tid::Z,
2           input_int_valid::Vector{TRMIintruderInput(p.E-20)}, st_int::Vector{TRMIintruderState(p.E-33)} )
3   spi::Bool = false
4   if is_ra_suppressed
5     spi = true
6   elseif mte
7     for i in 1:length( input_int_valid )
8       if !spi && (i != idx_tid) && st_int[i].is_threat
9         spi = IsDesignationInForce(p.311)( mte, input_int_valid[i], st_int[i] )
10      end
11    end
12  end
13  return spi::Bool
14 end
```

Referenced In: SetGroundMsgData(*p.312*), SetRABroadcastData(*p.317*)

IsDESIGNATIONINFORCE (Algorithm 261) determines whether this intruder is designated to an Xo mode and that designation is in force. This algorithm is used when setting the Suppression Indicator (SPI) and Designation Indicator (DSI) bits in the TRMGROUNDMSGDATA and TRMRABROADCASTDATA data structures. It is inappropriate for use when processing Xo modes and shouldn't be used for that purpose.

The inputs are whether or not the advisory is a multi-threat advisory (*multithreat*), the intruder inputs from the STM (*intruder*), and the intruder state (*st_int*).

An Xo designation is considered to be in force, when setting the SPI and DSI bits, when one of the following conditions is met.

- Whenever a protection mode other than the standard Xa protection mode is active.
- Whenever traffic advisories and resolution advisories are deactivated via *st_int.no_alerts*.
- Whenever the Designated No Alerts (DNA) Xo mode is valid and there is a multi-threat resolution advisory while this intruder is designated to the DNA Xo mode. During a multi-threat resolution advisory, DNA Xo mode processing is suspended but the intruder is not undesigned from DNA Xo mode. DNA Xo designation is considered to be in force only when setting the SPI and DSI bits. After that intruder becomes the sole threat, DNA Xo mode processing will resume.

Otherwise, Xo designation is not in force.

This algorithm takes as input *multithreat*, *intruder*, and *st_int*. The output of **IsDESIGNATIONINFORCE** is whether this intruder is designated to an Xo mode and that designation is in force (*in_force*).

Algorithm 261 IsDesignationInForce

```

1 function IsDesignationInForce( multithreat::Bool, intruder::TRMIIntruderInput(p.E-20),
2                               st_int::TRMIIntruderState(p.E-33) )
3   const S_standard_protection_mode::Z = params().target_designation.S_standard_protection_mode
4   in_force::Bool = false
5   if (S_standard_protection_mode != st_int.protection_mode_curr)
6     in_force = true
7   elseif st_int.no_alerts
8     in_force = true
9   elseif multithreat && intruder.xo_valid[Xo_IDX_DNA]
10    if (DESIGNATION_DESIGNATED_NO_ALERTS == intruder.designated_mode)
11      in_force = true
12    end
13  end
14  return in_force::Bool
15 end

```

Referenced In: DetermineSPI(p.310), DetermineDSI(p.316), UpdateIndivAdjustThreatInfo(p.304)

3.7.4.2 RA Report Message

SETGROUNDMSGDATA (Algorithm 262) populates the TRMGROUNDMSGDATA data structure for the TRM Report.

The inputs are TRMRACOORDINATIONDATA (*ra_data*), the threat index for the most recent threat (*idx_tid*), the intruder inputs from the STM for all valid intruders (*input_int_valid*), the own aircraft altitude (*z_own_ave*), whether the resolution advisory is suppressed (*is_ra_suppressed*), the state for own aircraft (*st_own*), and the intruder state for all valid intruders (*st_int*). The most recent threat is the one to be identified in the Threat ID (TID).

First, the TRMRACOORDINATIONDATA is copied.

Next, the Threat Type Indicator (TTI), Threat TID (TID), Designation Indication (DSI), and Suppression Indication (SPI) values are set. If the active resolution advisory (RA) from the previous cycle was terminated, the values for TTI, TID, DSI, and SPI from the previous cycle are used. If there is a threat identified, DETERMINETTIANDTID (Algorithm 263) is used to set the TTI and TID values, DETERMINEDSI (Algorithm 267) is used to set the DSI value, DETERMINESPI (Algorithm 260) is used to set the SPI value. Otherwise, the TTI, TID, DSI, and SPI values are cleared.

After all the fields in TRMGROUNDMSGDATA are populated, the TRM state data structures are updated. For each intruder, the *st_int.is_identified_threat* intruder state value is cleared when that intruder isn't a threat.

The algorithm finishes by storing the TTI, TID, DSI, and SPI values for use on the next cycle. The threat index is also stored for use in setting the threat index on the next cycle. When the index of the most recent threat is a valid index (greater than 0) and is less than or equal to the length of the input array (*input_int_valid*), the TID is represented as either an address or a coded altitude-range-bearing value.

This algorithm takes as input *ra_data*, *idx_tid*, *input_int_valid*, *z_own_ave*, *is_ra_suppressed*, *st_own*, and *st_int*.

The output from SETGROUNDMSGDATA is a populated TRMGROUNDMSGDATA data structure (*ground_msg*).

Algorithm 262 SetGroundMsgData

```

1 function SetGroundMsgData( ra_data::TRMRACoordinationData(p. E-28), idx_tid::Z,
2           input_int_valid::Vector{TRMIIntruderInput(p. E-20)}, z_own_ave::R,
3           is_ra_suppressed::Bool, st_own::TRMOwnState(p. E-32), st_int::Vector{TRMIIntruderState(p. E-33)}
4           )
5   ground_msg::TRMGroundMsgData = TRMGroundMsgData(p. E-27)()
6   ground_msg.ra_data = deepcopy( ra_data )
7   if ra_data.rat
8     ground_msg.tti = st_own.ra_output_prev.tti
9     ground_msg.tid = st_own.ra_output_prev.tid
10    ground_msg.dsi = st_own.ra_output_prev.dsi
11    ground_msg.spi = st_own.ra_output_prev.gnd_msg_spi
12  elseif (0 < idx_tid) && (idx_tid <= length( input_int_valid ))
13    (ground_msg.tti, ground_msg.tid) =
14      DetermineTTIAndTID(p. 313)( idx_tid, input_int_valid, z_own_ave )
15    ground_msg.dsi =
16      DetermineDSI(p. 316)( ra_data.mte, idx_tid, input_int_valid, st_int )
17    ground_msg.spi =
18      DetermineSPI(p. 310)( is_ra_suppressed, ra_data.mte, idx_tid, input_int_valid, st_int )
19  else
20    ground_msg.tti = TTI_ADDRESS
21    ground_msg.tid = TIDAddress(p. E-28)()
22    ground_msg.dsi = false
23    ground_msg.spi = false
24  end
25  for i in 1:length( st_int )
26    if !st_int[i].is_threat
27      st_int[i].is_identified_threat = false
28    end
29  end
30  st_own.ra_output_prev.tti      = ground_msg.tti
31  st_own.ra_output_prev.tid      = ground_msg.tid
32  st_own.ra_output_prev.dsi      = ground_msg.dsi
33  st_own.ra_output_prev.gnd_msg_spi = ground_msg.spi
34  if (0 < idx_tid)
35    st_own.ra_output_prev.tgtid_tid = input_int_valid[idx_tid].id
36  else
37    st_own.ra_output_prev.tgtid_tid = 0
38  end
39 return (ground_msg::TRMGroundMsgData(p. E-27))

```

Referenced In: SetRAMessageOutput(p. 308)
--

DETERMINETTIANDTID (Algorithm 263) sets the Threat Type Indicator (TTI) and Threat ID (TID) of the most recent threat. The TTI and TID values are fields in the TRMGROUNDMMSGDATA data structure in TRM Report.

DETERMINETTIANDTID takes the vector of intruder inputs (*input_int_valid*), the index for the most recent threat (*idx_tid*), and the barometric altitude of own aircraft (*z_own_ave*) as inputs.

When the index of the most recent threat is a valid index (greater than 0) and is less than or equal to the length of the input array (*input_int_valid*), the TID is represented as either an address or a coded altitude-range-bearing value.

If an ICAO address is supplied (*is_icao* is TRUE), the TID is set to that address and the TTI is set to *TTI_ADDRESS*.

Otherwise, the TID is a coded altitude-range-bearing. The TTI is set to *TTI_ALT_RNG_BRG*. The altitude, range, and bearing information for the threat is obtained from the STMDISPLAYSTRUCT

output from the STM. The coded altitude is obtained from ENCODETIDALTITUDE, using whether or not the intruder aircraft is reporting altitude (*alt_reporting*) and the barometric altitude in feet (*altitude*). The barometric altitude of the threat is obtained by subtracting the vertical separation distance (*z_rel*) from own barometric altitude (*z_own_ave*). The coded range is obtained from ENCODETIDRANGE, using the ground range in feet (*r_ground*). The coded relative bearing is obtained from ENCODETIDBEARING, using the relative bearing, in radians (*Chi_rel*). The coded altitude is 13 bits, the coded range is 7 bits, and the coded relative bearing is 6 bits. DETERMINETTIANDTID protects against overflow of each of these three bit fields.

If there is no threat available, the TID and TTI information is cleared to the default setting. The default setting is a TID address of 0 with TTI set to *TTI_ADDRESS*.

This algorithm takes as input *input_int_valid*, *idx_tid*, and *z_own_ave*.

The outputs from DETERMINETTIANDTID are the TTI (*tti*) and the TID (*tid*).

Algorithm 263 DetermineTTIAndTID

```

1 function DetermineTTIAndTID( idx_tid::Z, input_int_valid::Vector{TRMIntruderInput(p.E-20)}, z_own_ave::R )
2   tti::UInt8 = TTI_ADDRESS
3   tid::TID(p.E-28) = TIDAddress(p.E-28)()
4   if (0 < idx_tid) && (idx_tid <= length( input_int_valid ))
5     if input_int_valid[idx_tid].is_icao
6       tti = TTI_ADDRESS
7       tid = TIDAddress(p.E-28)( input_int_valid[idx_tid].address )
8     else
9       tti = TTI_ALT_RNG_BRG
10      stm_display_data::StmDisplayStruct(p.E-1) = input_int_valid[idx_tid].stm_display
11      altitude::R = z_own_ave - input_int_valid[idx_tid].stm_display.z_rel
12      tida::UInt16 = min( 0x07ff, EncodeTIDAltitude(p.314)( stm_display_data.alt_reporting, altitude ) )
13      tidr::UInt8 = min( 0x0007f, EncodeTIDRange(p.315)( stm_display_data.r_ground ) )
14      tidb::UInt8 = min( 0x0003f, EncodeTIDBearing(p.315)( stm_display_data.Chi_rel ) )
15      tid = TIDAltRngBrg(p.E-29)( tida, tidr, tidb )
16    end
17  end
18  return (tti::UInt8, tid::TID(p.E-28))
19 end
```

Referenced In: SetGroundMsgData(p.312)

ENCODETIDALTITUDE (Algorithm 264) returns an 11-bit encoding of altitude to be used for the Threat ID. The altitude is input in feet. The coded altitude is also in feet.

First, the maximum altitude that can be encoded is determined using *TIDA_MAX_CODE* and *TIDA_OFFSET*. Then the altitude is encoded.

When the intruder aircraft is not altitude reporting (*alt_reporting* is FALSE) or the altitude is not set (it is *Nan*), the coded altitude returned by ENCODETIDALTITUDE is 0.

Otherwise, ENCODETIDALTITUDE checks the value of the altitude to make sure it stays within the range limits for TID altitude encoding. When the value of the altitude is less than *TIDA_MIN_ALTITUDE*, the coded altitude is 1. When the value of the altitude is greater than *max_altitude*, the coded altitude is *TIDA_MAX_CODE*.

When the altitude is within the range limits, the altitude is coded in 100.0 foot increments offset from *TIDA_OFFSET*.

The coded altitude is returned as an unsigned integer.

This algorithm takes as input *alt_reporting*, and *altitude*. This algorithm returns *coded_altitude*.

Algorithm 264 EncodeTIDAltitude

```

1 function EncodeTIDAltitude( alt_reporting::Bool, altitude::R )
2     coded_altitude::UInt32 = 0
3     max_altitude::R = ((TIDA_MAX_CODE - 1) * 100.0) + TIDA_OFFSET
4     if !alt_reporting || isnan( altitude )
5         coded_altitude = 0
6     elseif (altitude < TIDA_MIN_ALTITUDE)
7         coded_altitude = 1
8     elseif (max_altitude < altitude)
9         coded_altitude = TIDA_MAX_CODE
10    else
11        coded_altitude = uint32( 1 + round( (altitude - TIDA_OFFSET) * 0.01 ) )
12        if (coded_altitude < 1)
13            coded_altitude = 1
14        elseif (TIDA_MAX_CODE < coded_altitude)
15            coded_altitude = TIDA_MAX_CODE
16        end
17    end
18    return coded_altitude::UInt32
19 end
```

Referenced In: DetermineTTIAndTID(p. 313)

ENCODETIDRANGE (Algorithm 265) returns the coded range. The range is input in feet. The coded range is in nautical miles.

The range is first converted from feet to nautical miles. If the range value is not set (it is *NaN*), the coded range is 0.

Since the coded range is represented as 7 bits, the coded values must be between 1 and 127, inclusive. If the intruder is within 0.05 nautical miles in the horizontal plane, the coded range is 1. If the intruder is outside 12.55 nautical miles in the horizontal plane, the coded range is 127.

Otherwise, the range is coded in 0.1 nautical mile increments, so a range of 0.1 is coded as 2 and a range of 11.2 is coded as 113.

The coded range is returned as an unsigned integer.

This algorithm takes as input *range*. This algorithm returns *coded_range*.

Algorithm 265 EncodeTIDRange

```

1 function EncodeTIDRange( range::R )
2   coded_range::UInt32 = 0
3   const ft2nmi::R = geoutils.feet_to_nautical_miles
4   range = abs( range ) * ft2nmi
5   if isnan( range )
6     coded_range = 0
7   elseif (range < 0.05)
8     coded_range = 1
9   elseif (12.55 < range)
10    coded_range = 127
11   else
12     coded_range = uint32( 1 + round( range * 10.0 ) )
13     if (coded_range < 2)
14       coded_range = 2
15     elseif (126 < coded_range)
16       coded_range = 126
17     end
18   end
19   return coded_range::UInt32
20 end
```

Referenced In: DetermineTTIAndTID(p. 313)

ENCODETIDBEARING (Algorithm 266) returns the coded relative bearing. The relative bearing is input in radians. The coded relative bearing is in degrees.

If the relative bearing value is not set (it is *Nan*), the coded bearing is 0.

Otherwise, the relative bearing is first converted from radians to degrees. WRAPTOPI is used to ensure the radian value used for the conversion is between $-\pi$ and $+\pi$. After adjustment, the resulting relative bearing will be between 0 and 360 degrees.

The coded relative bearing is 6 bits, supporting values from 1 to 63, inclusive. However, the values 61, 62, and 63 are not used. A relative bearing of 5.0 degrees is coded as 1. A relative bearing of 359.0 degrees is coded as 60.

The coded bearing is returned as an unsigned integer.

This algorithm takes as input *bearing*. This algorithm returns *coded_bearing*.

Algorithm 266 EncodeTIDBearing

```

1 function EncodeTIDBearing( bearing::R )
2   coded_bearing::UInt32 = 0
3   if !isnan( bearing )
4     bearing = rad2deg( WrapToPi(p. H-3)( bearing ) )
5     if (bearing < 0.0)
6       bearing = bearing + 360.0
7     end
8     coded_bearing = uint32( 1 + floor( bearing / 6.0 ) )
9   end
10  return coded_bearing::UInt32
11 end
```

Referenced In: DetermineTTIAndTID(p. 313)

DETERMINEDSI (Algorithm 267) determines the Designation Indicator (DSI) setting.

The inputs are the setting of the multithreat bit (*mte*), the threat index for the most recent threat (*idx_tid*), the intruder intputs from the STM for all valid intruders (*input_int_valid*), and the intruder state for all valid intruders (*st_int*). The setting of the multithreat bit indicates whether or not the advisory is a multi-threat advisory. The most recent threat is the one identified in the Threat ID (TID).

The DSI is set to TRUE only if the most recent threat is designated to an Xo mode and that designation is in force. If the threat index is not a valid index, there is no threat.

Otherwise, the DSI is FALSE.

This algorithm takes as input *mte*, *idx_tid*, *input_int_valid*, and *st_int*. The output of DETERMINEDSI is the designation indicator setting (*dsi*).

Algorithm 267 DetermineDSI

```

1 function DetermineDSI( mte::Bool, idx_tid::Z, input_int_valid::Vector{TRMIntruderInput(p.E-20)},
2           st_int::Vector{TRMIntruderState(p.E-33)} )
3   dsi::Bool = false
4   if (0 < idx_tid) && (idx_tid <= length( input_int_valid ))
5     dsi = IsDesignationInForce(p.311)( mte, input_int_valid[idx_tid], st_int[idx_tid] )
6   end
7   return dsi::Bool
8 end
```

Referenced In: SetGroundMsgData(<i>p.312</i>)
--

3.7.4.3 Broadcast Interrogation Message

SETRABROADCASTDATA (Algorithm 268) populates the TRMRABROADCASTDATA data structure for the TRM Report.

The inputs are TRMRACOORDINATIONDATA (*ra_data*), whether the resolution advisory is suppressed (*is_ra_suppressed*), the intruder intputs from the STM for all valid intruders (*input_int_valid*), the Mode A identifier for own aircraft (*mode_a*), the barometric altitude in feet for own aircraft (*z_own_ave*), the state for own aircraft (*st_own*), and the intruder state for all valid intruders (*st_int*). The identified threat is the one to be identified in the Threat ID (TID).

First, the TRMRACOORDINATIONDATA is copied.

Next, the Suppression Indication (SPI) value is set. If the active resolution advisory (RA) from the previous cycle was terminated, the value for SPI from the previous cycle is used. If there is a threat identified, DETERMINESPI (Algorithm 260) is used to set the SPI value. Otherwise, the SPI value has the default value of FALSE.

The Mode A identifier field (*aid*) is set directly from the input. The Mode C code field (*cac*) is set using ENCODECAC (Algorithm 269). That algorithm is provided with the barometric altitude for own aircraft (*z_own_ave*).

The algorithm finishes by storing the SPI value for use on the next cycle.

This algorithm takes as input *ra_data*, *is_ra_suppressed*, *input_int_valid*, *mode_a*, *z_own_ave*, *st_own*, and *st_int*.

The output from SETRABROADCASTDATA is a populated TRMRABROADCASTDATA data structure (*broadcast*).

Algorithm 268 SetRBroadcastData

```

1 function SetRBroadcastData( ra_data::TRMRACoordinationData(p.E-28),
2                               is_ra_suppressed::Bool, input_int_valid::Vector{TRMIIntruderInput(p.E-20)},
3                               mode_a::UInt32, z_own_ave::R,
4                               st_own::TRMOwnState(p.E-32), st_int::Vector{TRMIIntruderState(p.E-33)} )
5   broadcast = TRMRABroadcastData(p.E-27)()
6   broadcast.ra_data = deepcopy( ra_data )
7   if ra_data.rat
8     broadcast.spi = st_own.ra_output_prev.broadcast_spi
9   else
10    broadcast.spi =
11      DetermineSPI(p.310)( is_ra_suppressed, ra_data.mte, -1, input_int_valid, st_int )
12  end
13  broadcast.aid = mode_a
14  broadcast.cac = EncodeCAC(p.318)( z_own_ave )
15  st_own.ra_output_prev.broadcast_spi = broadcast.spi
16  return (broadcast::TRMRABroadcastData(p.E-27))
17 end
```

Referenced In: SetRAMessageOutput(p.308)

ENCODECAC (Algorithm 269) returns the Gillham coded altitude. GILLHAMENCODE is used to obtain the Gillham code for the altitude.

When the altitude is not set (it is *NaN*), the coded altitude returned by ENCODECAC is 0.

Otherwise, ENCODECAC checks the value of the altitude to make sure it stays within the range limits for the Gillham encoding. When the value of the altitude is outside the range limits (*NARS_THRESHOLD_MIN* and *NARS_THRESHOLD_MAX*), the Gillham code for the relevant range limit is stored in *gillham_value*. Otherwise, the Gillham code for *altitude* is stored in *gillham_value*.

If the resulting Gillham code in *gillham_value* is not valid, the coded altitude will default to 0. Otherwise, the coded altitude will be an unsigned integer representation of *gillham_value*.

This algorithm takes as input *altitude*. This algorithm returns *coded_altitude*.

Algorithm 269 EncodeCAC

```

1 function EncodeCAC( altitude::R )
2     coded_altitude::UInt32 = 0
3     if !isnan( altitude )
4         gillham_value::Z = 0
5         if (NARS_THRESHOLD_MAX <= altitude)
6             gillham_value = GillhamEncode(p.26)( NARS_THRESHOLD_MAX-1.0 )
7         elseif (altitude < NARS_THRESHOLD_MIN)
8             gillham_value = GillhamEncode(p.26)( NARS_THRESHOLD_MIN )
9         else
10            gillham_value = GillhamEncode(p.26)( altitude   )
11        end
12        if (GILLHAM_INVALID_VALUE != gillham_value)
13            coded_altitude = uint32( gillham_value )
14        end
15    end
16    return coded_altitude::UInt32
17 end

```

Referenced In: SetRBroadcastData(p. 317)

3.7.5 Intruder Management

The intruder inputs to the TRM (*input.intruder*) correspond to the set of active tracks produced by the ACAS X Surveillance and Tracking Module (STM) within the threat volume around own aircraft. The TRM must handle cases when no tracks are available, as might happen when ACAS X is first starting up or the aircraft is enroute. It must also handle cases when the STM drops a track or a received intruder is not valid, as might happen with degraded surveillance. When that happens, the TRM must ensure the appropriate output is produced so the display and coordination messages will reflect the correct advisory state.

Additionally, the TRM must properly handle outputs to the Xo mode designation interface. ACAS Xo traffic designation support is provided via the Aircraft Surveillance Applications (ASA) System described in DO-317B [15]. The Aircraft Surveillance and Separation Assurance Processing (ASSAP) function in the ASA System provides the interface to the ACAS X system. Two Xo designation modes are currently defined for ACAS X: Designated No Alerts (DNA) and CSPO-3000.

DROPPEDINTRUDERSADJUSTMENT (Algorithm 270) processes all intruder states and produces an array of intruder outputs for dropped and invalid intruders. It also produces an updated array containing intruder states for all intruders that were not dropped.

After initializing output arrays, this algorithm creates a dictionary of intruder outputs (*output_int_dict*). The dictionary is composed of intruders that were processed for advisories by the vertical TRM (*output_int*). The key used to find intruders in this dictionary is the target id (*output_int[i].id*).

DROPPEDINTRUDERSADJUSTMENT next creates a dictionary of invalid intruders input to the TRM (*input_invalid_int_dict*). The dictionary is composed of intruders that were skipped over for processing by the vertical TRM (*input_invalid_int*). Only intruder inputs to the TRM flagged as valid are processed by the TRM. Intruder inputs to the TRM that are flagged as invalid are processed only by DROPPEDINTRUDERSADJUSTMENT. The key used to find intruders in this dictionary also is the target id (*input_invalid_int[i].id*).

The remainder of the algorithm is a loop over the list of intruder states. For each state, the first check is whether the intruder associated with this state was in the output. If it was in the output and

it wasn't *MarkedForDeletion*, the *status* is marked as *InUse* and the intruder state (*st_int_next*) is stored for use in the next TRM processing cycle in *st_int_out*.

If there is no intruder output and the intruder wasn't found in the list of invalid intruders, the intruder was dropped from the STM outputs. This situation is handled by **SETDROPPEDINTRUDER**. The state information for this intruder is not retained.

When the intruder is found in the list of invalid intruders, it could be an invalid input for one of three reasons:

1. The intruder is designated and would otherwise have been dropped entirely from the STM output due to lack of surveillance.
2. The intruder has degraded surveillance that is inadequate for use in producing advisories.
3. The ACAS X system is in Standby operational Mode.

The first is distinguished from the other two by the bit settings in *input_invalid_int_dict.degraded_surveillance*.

The first condition is indicated when all the bits are set (*DEGRADED_SURVEILLANCE_ALL*). In this case, **SETDROPPEDINTRUDER** is called. Additional information available in the intruder input (*designated_mode* and *xo_status*) is provided to **SETDROPPEDINTRUDER**. The state information for the dropped intruder is retained only if the designation is on a timer as indicated by *DesignationStateTimeout*.

The other conditions are handled by **SETINVALIDINTRUDER**. In this case, the state information is always retained. The intruder state (*st_int_next*) is stored for use in the next TRM processing cycle in *st_int_out*.

Whenever the intruder was not found in the output intruder dictionary, an output is created for that intruder (*dropped_int*) and added to the list (*dropped_int_out*).

This algorithm takes as input *mode_s_own*, *multithreat*, *input_int_invalid*, *output_int*, and *st_int*. The list of intruder states (*st_int_out*) and the list of dropped intruder outputs (*dropped_int_out*) are returned from **DROPPEDINTRUDERSADJUSTMENT**.

Algorithm 270 DroppedIntrudersAdjustment

```

1 function DroppedIntrudersAdjustment( mode_s_own::UInt32, multithreat::Bool,
2                                     input_int_invalid::Vector{TRMIntruderInput(p.E-20)},
3                                     output_int::Vector{TRMIntruderData(p.E-39)},
4                                     st_int::Vector{TRMIntruderState(p.E-33)} )
5 #
6   dropped_int_out::Vector{TRMIntruderData} = Array( TRMIntruderData(p.E-39), 0 )
7   st_int_out::Vector{TRMIntruderState}      = Array( TRMIntruderState(p.E-33), 0 )
8   output_int_dict::Dict{Z,TRMIntruderData} = Dict{Z, TRMIntruderData(p.E-39)}()
9   input_invalid_int_dict::Dict{Z,TRMIntruderInput} = Dict{Z, TRMIntruderInput(p.E-20)}()
10  for i in 1:length( output_int )
11    output_int_dict[output_int[i].id] = output_int[i]
12  end
13  for i in 1:length( input_int_invalid )
14    input_invalid_int_dict[input_int_invalid[i].id] = input_int_invalid[i]
15  end
16  for st_int_next in st_int
17    id::UInt32 = st_int_next.id
18    if !haskey( output_int_dict, id ) && (:MarkedForDeletion != st_int_next.status)
19      dropped_int::TRMIntruderData = TRMIntruderData(p.E-39)()
20      if !haskey( input_invalid_int_dict, id )
21        st_int_next.designated_mode = DESIGNATION_NONE
22        dropped_int = SetDroppedIntruder(p.322)( id, mode_s_own, multithreat,
23                                         Xo_STATUS_NONE, st_int_next )
24      elseif (DEGRADED_SURVEILLANCE_ALL == input_invalid_int_dict[id].degraded_surveillance)
25        st_int_next.designated_mode = input_invalid_int_dict[id].designated_mode
26        dropped_int = SetDroppedIntruder(p.322)( id, mode_s_own, multithreat,
27                                         input_invalid_int_dict[id].xo_status,
28                                         st_int_next )
29        if (:DesignationStateTimeout == st_int_next.status)
30          push!( st_int_out, st_int_next )
31        end
32      else
33        dropped_int = SetInvalidIntruder(p.324)( id, mode_s_own, multithreat,
34                                         input_invalid_int_dict[id], st_int_next )
35        push!( st_int_out, st_int_next )
36      end
37      push!( dropped_int_out, dropped_int )
38    elseif (:MarkedForDeletion != st_int_next.status)
39      st_int_next.status = :InUse
40      push!( st_int_out, st_int_next )
41    end
42  end
43  return (dropped_int_out::Vector{TRMIntruderData(p.E-39)}, st_int_out::Vector{TRMIntruderState(p.E-33)})
44 end

```

Referenced In: GenerateTRMOOutput(p.291)

SETDROPPEDINTRUDER (Algorithm 271) handles clean up functions for a dropped intruder. As no intruder inputs are available for a dropped intruder, the intruder state information (*st_int*) is used for preparing the outputs.

This algorithm first adjusts the display and coordination settings. For a dropped intruder, any advisories are cleared from the onboard pilot display (*code* is set to *TACODE_CLEAR*) and this intruder is given the lowest display priority (*tds* is set to 0.0). Next, CROSSLINK is used to produce a CANCEL coordination message for this intruder whenever a coordination message was sent to the intruder on the previous cycle. The coordination settings for the previous cycle are stored in *st_int.vrc_prev* and *st_int.cvc_prev*.

The processing mode is set to *RA_PROCESSING_DROPPED_TRACK* and the resolution advisory state variables are reset.

Next, the Xo validity and status for this intruder are set based on whether the intruder is designated and, if so, whether the timer is still active. The intruder state (*st_int.designated_mode*) is examined to determine the designation status and to set *is_designated* accordingly. When *designated_mode* has the value *DESIGNATION_UNDESIGNATE_PROTECTION_MODE* or *DESIGNATION_UNDESIGNATE_NO_ALERTS* the intruder is being undesignated from an Xo mode; *is_designated* is still TRUE. When the intruder is being undesignated from an Xo mode, the intruder state is marked for deletion after this processing cycle. Otherwise when *is_designated* is TRUE, *designated_mode* indicates an active designation. When the intruder is designated to an Xo mode, the intruder state is marked as being on a timer and the state will be retained for another processing cycle. The Xo validity flag is set to TRUE so the intruder won't be undesignated by the ASSAP function in the ASA System. A value of FALSE for *is_designated* indicates the intruder is not designated. When the intruder is not designated, the intruder state is marked for deletion after this processing cycle.

After that, the TRM state for that intruder is set appropriately using `TRMINTRUDERSTATEUPDATE`. The remaining designation state values are reset.

Finally, the output is prepared depending on whether the outputs will include all intruder data or only the intruder designation data. The full set of intruder data is output when `SETDROPPEDINTRUDER` is called for this intruder the first time, as indicated by *is_full_output*. Otherwise, the intruder designation data is the only valid output. `SETDROPPEDINTRUDER` can be called more than once on an intruder designated to an Xo mode because the intruder state is retained until the Xo mode becomes valid or the intruder is no longer designated. In either case, the xo valid and xo status settings are updated last.

The intruder outputs are returned in `TRMINTRUDERDATA` (*dropped_int*). Whenever this is an initial dropped track, the display, coordination, and debug information will be output in addition to the designation information (*is_full_output* is TRUE). In this case, all the intruder outputs are stored in *dropped_int*. Otherwise, only the designation outputs are relevant for storage in *dropped_int*.

This algorithm takes as input *id_int*, *mode_s_own*, *multithreat*, *xo_status_int*, and *st_int*. This algorithm returns *dropped_int*.

Algorithm 271 SetDroppedIntruder

```

1 function SetDroppedIntruder( id_int::UInt32, mode_s_own::UInt32, multithreat::Bool,
2                               xo_status_int::UInt8, st_int::TRMIntruderState(p.E-33) )
3   const S_standard_protection_mode::UInt8 = uint8(params().target_designation.S_standard_protection_mode)
4   code::UInt8 = TACODE_CLEAR
5   tds::R      = 0.0
6   (cvc::UInt32, vrc::UInt32, vsb::UInt32, sense_indiv::Symbol) =
7     Crosslink(p.270)( 0.0, 0.0, st_int.vrc_prev, st_int.cvc_prev, st_int.equipage_prev )
8   st_int.processing          = RA_PROCESSING_DROPPED_TRACK
9   st_int.time_since_ta       = 0
10  st_int.a_prev.sense       = sense_indiv
11  st_int.is_dna_coordination = false
12  is_designated::Bool = (DESIGNATION_NONE != st_int.designated_mode)
13  xo_status::UInt8 = Xo_STATUS_NONE
14  is_full_output::Bool = !st_int.is_desonly_output
15  if is_designated
16    if (DESIGNATION_UNDESIGNATE_PROTECTION_MODE == st_int.designated_mode) ||
17      (DESIGNATION_UNDESIGNATE_NO_ALERTS == st_int.designated_mode)
18      st_int.status           = :MarkedForDeletion
19      st_int.is_desonly_output = true
20      xo_valid               = false
21      xo_status              = xo_status_int
22    else
23      st_int.status           = :DesignationStateTimeout
24      st_int.is_desonly_output = true
25      xo_valid               = true
26      xo_status              = Xo_STATUS_SUSPENDED_DROPPED
27    end
28  else
29    st_int.status           = :MarkedForDeletion
30    st_int.is_desonly_output = false
31    is_full_output          = true
32    xo_valid               = false
33  end
34  TRMIntruderStateUpdate(p.344)( st_int, vrc, cvc, st_int.equipage_prev,
35                                st_int.coordination_msg_prev, true, true )
36  st_int.protection_mode_curr = S_standard_protection_mode
37  st_int.no_alerts          = false
38  st_int.is_identified_threat = false
39  dropped_int::TRMIntruderData = TRMIntruderData(p.E-39)()
40  if is_full_output
41    dropped_int =
42      TRMIntruderData(p.E-39)( id_int, :None, mode_s_own, st_int.address,
43                                cvc, vrc, vsb, multithreat, st_int.coordination_msg_prev,
44                                tds, code, st_int.designated_mode, st_int.processing,
45                                st_int.no_alerts, st_int.protection_mode_curr,
46                                false, st_int.address, st_int.is_icao )
47  else
48    dropped_int.designation =
49      TRMIntruderDesignationData(p.E-26)( id_int, st_int.address, st_int.is_icao,
50                                         st_int.designated_mode, st_int.processing,
51                                         st_int.no_alerts, st_int.protection_mode_curr,
52                                         false, false, multithreat )
53  end
54  dropped_int.designation.valid[Xo_IDX_DNA] = xo_valid
55  dropped_int.designation.valid[Xo_IDX_CSP03k] = xo_valid
56  dropped_int.designation.status           = xo_status
57  return (dropped_int::TRMIntruderData(p.E-39))
58 end

```

Referenced In: DroppedIntrudersAdjustment(p.320)

SETINVALIDINTRUDER (Algorithm 272) handles clean up functions for an intruder flagged as invalid. Intruder inputs (*intruder*) are available when an intruder is flagged as invalid. These inputs

are used for updating the state information and providing more informative outputs.

This algorithm takes as input *id_int*, *mode_s_own*, *multithreat*, *intruder*, and *st_int*.

The TA status (*ta_alert*) is initialized to TRUE.

SETINVALIDINTRUDER first uses CROSSLINK to produce a CANCEL coordination message for this intruder when a coordination message was sent to the intruder previously.

It next determines the status of the traffic advisory.

- When the intruder is Designated No Alerts (DNA) (*intruder.dna* is TRUE), no traffic advisories (TAs) are output. the TA timer is reset and the TA status is set to FALSE.
- When there was previously a resolution advisory, the TA timer is started and the TA status is set to TRUE.
- When there was previously a TA and the TA timer hasn't expired, the TA timer is updated and the TA status is set to TRUE.
- Otherwise, the TA timer is reset and the TA status is set to FALSE.

It next determines the type of advisory that should be sent to the onboard pilot display depending on the current advisory and designation state. DETERMINECODE sets the advisory code (*code*) based on the TA status, surveillance quality (*degraded_surveillance*), and proximity (*is_proximate*). DETERMINESCORE sets the track display score (*tds*) based on the code, surveillance quality and whether the intruder is designated *is_designated*.

The processing state is set to *RA_PROCESSING_NONE* and designation state information is reset. Finally, the TRM state for that intruder is set appropriately, using **TRMINTRUDERSTATEUPDATE**. The intruder outputs are stored and returned in **TRMINTRUDERDATA** (*invalid_int*).

This algorithm takes as input *id_int*, *mode_s_own*, *multithreat*, *intruder*, and *st_int*. This algorithm returns *invalid_int*.

Algorithm 272 SetInvalidIntruder

```

1 function SetInvalidIntruder( id_int::UInt32, mode_s_own::UInt32, multithreat::Bool,
2                               intruder::TRMIIntruderInput(p.E-20), st_int::TRMIIntruderState(p.E-33) )
3     mode_idx::Z = GetProtectionModeIndex(p.328)( st_int.protection_mode_prev )
4     const T_ta_min_alert::Z           = params().modes[mode_idx].track_threat.ta_min_alert_time
5     const T_maintain_ta_after_ra::Z = params().modes[mode_idx].track_threat.maintain_ta_after_ra
6     is_designated::Bool            = (DESIGNATION_NONE != intruder.designated_mode)
7     ta_alert::Bool                = false
8     (cvc::UInt32, vrc::UInt32, vsb::UInt32, sense_indiv::Symbol) =
9         Crosslink(p.270)( 0.0, 0.0, st_int.vrc_prev, st_int.cvc_prev, intruder.equipage )
10    if intruder.dna
11        st_int.time_since_ta = 0
12        ta_alert            = false
13    elseif (:None != st_int.sense_prev)
14        st_int.time_since_ta = T_ta_min_alert - T_maintain_ta_after_ra + 1
15        ta_alert            = true
16    elseif (0 < st_int.time_since_ta < T_ta_min_alert)
17        st_int.time_since_ta = st_int.time_since_ta + 1
18        ta_alert            = true
19    else
20        st_int.time_since_ta = 0
21        ta_alert            = false
22    end
23    code::UInt8 =
24        DetermineCode(p.276)( ta_alert, sense_indiv, intruder.degraded_surveillance, intruder.is_proximate )
25    tds::R =
26        DetermineScore(p.278)( 0.0, code, intruder.degraded_surveillance, is_designated, false )
27    st_int.status          = :NotValid
28    st_int.processing      = RA_PROCESSING_NONE
29    st_int.a_prev.sense   = sense_indiv
30    st_int.no_alerts       = intruder.dna
31    st_int.is_dna_coordination = false
32    st_int.designated_mode = intruder.designated_mode
33    st_int.protection_mode_curr = intruder.protection_mode
34    st_int.is_identified_threat = false
35    TRMIIntruderStateUpdate(p.344)( st_int, vrc, cvc, intruder.equipage,
36                                    intruder.coordination_msg, true, true )
37    invalid_int::TRMIIntruderData(p.E-39) =
38        TRMIIntruderData(p.E-39)( id_int, :None, mode_s_own, intruder.address,
39                                cvc, vrc, vsb, multithreat, intruder.coordination_msg,
40                                tds, code, intruder.designated_mode,
41                                st_int.processing, st_int.no_alerts,
42                                st_int.protection_mode_curr,
43                                false, intruder.address, intruder.is_icao )
44    invalid_int.designation.valid[Xo_IDX_DNA] = intruder.xo_valid[Xo_IDX_DNA]
45    invalid_int.designation.valid[Xo_IDX_CSP03k] = intruder.xo_valid[Xo_IDX_CSP03k]
46    invalid_int.designation.status = intruder.xo_status
47    return (invalid_int::TRMIIntruderData(p.E-39))
48 end

```

Referenced In: DroppedIntrudersAdjustment(p.320)

3.8 Standard States and Conversions

These algorithms provide definitions for standard ACAS X conversions and behavior. They are used by other algorithms throughout this document.

ACTIONTORATES (Algorithm 273) maps actions (RAs) to their corresponding vertical rate limits and vertical acceleration. Three tables from the parameters file are used to perform the mapping. The first table contains the minimum vertical rate for each possible action and is stored in the variable *R_dz_min_adv*. The second table contains the maximum vertical rate for each possible

action and is stored in the variable $R_dz_max_adv$. The third table contains the vertical acceleration for each possible action and is stored in the variable R_ddz_adv .

The proper vertical rate limits and vertical acceleration for a given action are extracted from the parameter file tables using act . Vertical rate limits that are defined as Nan in the tables correspond to the Maintain RA and require special processing. If the Maintain RA is continuing from the previous cycle, the vertical rate limits and vertical acceleration are set to the values used in the previous cycle: dz_min_prev , dz_max_prev , and ddz_prev . New Maintain RAs use MAINTAINRATES (Algorithm 297) to determine the vertical rate limits while ddz is retrieved from the parameter file tables.

The algorithm takes as inputs the action from the current cycle (act), the current vertical rate of the ownship (dz_own_ave), the action index from the previous cycle (act_prev), the minimum vertical rate of the action from the previous cycle (dz_min_prev), the maximum vertical rate of the action from the previous cycle (dz_max_prev), and the vertical acceleration of the action from the previous cycle (ddz_prev). The algorithm outputs the minimum vertical rate of the input action (dz_min), the maximum vertical rate of the input action (dz_max), and the vertical acceleration of the input action (ddz).

Algorithm 273 ActionToRates

```

1 function ActionToRates( act::Z, dz_own_ave::R, act_prev::Z, dz_min_prev::R, dz_max_prev::R, ddz_prev::R )
2   const R_dz_min_adv::Vector{R} = params().actions.min_rates
3   const R_dz_max_adv::Vector{R} = params().actions.max_rates
4   const R_ddz_adv::Vector{R} = params().actions.accelerations
5   dz_min::R = 0.0
6   dz_max::R = 0.0
7   ddz::R = 0.0
8   if isnan( R_dz_min_adv[act] ) || isnan( R_dz_max_adv[act] )
9     if (act_prev == act)
10      (dz_min, dz_max, ddz) = (dz_min_prev, dz_max_prev, ddz_prev)
11    else
12      (dz_min, dz_max) = MaintainRates(p. 339)( dz_own_ave )
13      ddz = R_ddz_adv[act]
14    end
15  else
16    (dz_min, dz_max, ddz) = (R_dz_min_adv[act], R_dz_max_adv[act], R_ddz_adv[act])
17  end
18  return (dz_min::R, dz_max::R, ddz::R)
19 end
```

Referenced In: CoordinationSelection([p. 269](#)), SandwichPreventionCost([p. 257](#)), GenerateTAOnlyModeOutput([p. 292](#)), NoIntrudersAction([p. 251](#)), OnlineUncoordinatedCostEstimation([p. 191](#)), ArbitrateMatchingSenses([p. 267](#)), PersistMTLO([p. 341](#)), ActionSelection([p. 250](#)), IndividualCostEstimation([p. 248](#)), OnlineCostEstimation([p. 214](#))

CALCULATETHRESHOLDRAMPDOWNFACTOR (Algorithm 274) computes a multiplicative factor by mapping an input value onto a threshold/roll-off set.

The algorithm takes as inputs the current value to test against the threshold/roll-off set ($value$), the value below which the factor will be 1.0 ($threshold$), and the range over which the ramp goes from 1.0 to 0.0 ($rolloff$). The algorithm outputs a value between 0.0 and 1.0 ($factor$).

Algorithm 274 CalculateThresholdRampDownFactor

```

1 function CalculateThresholdRampDownFactor( value::R, threshold::R, rolloff::R )
2   factor::R = 0.0
3   if (value <= threshold)
4     factor = 1.0
5   elseif (value < (threshold + rolloff))
6     factor = 1.0 - ((value - threshold) / rolloff)
7   end
8   return factor::R
9 end
```

Referenced In: DetermineHorizontalWorstCaseNearMeanFactor(p. 171), DetermineAltitudeDependentCOCFactor(p. 181), DetermineOwnNonComplianceFactor(p. 246), DetermineVerticalProximateFactor(p. 183), DetermineHorizontalWorstCasePhiSpreadFactor(p. 172), DetermineVerticalRateReductionFactor(p. 182), AltitudeDependentCOCCost(p. 196), UpdateTimeBasedNonComplianceCState(p. 244), UpdateLowAltitudeParallelRADeferralCState(p. 231)

CALCULATETHRESHOLDRAMPUPFACTOR (Algorithm 275) computes a multiplicative factor by mapping an input value onto a threshold/roll-off set.

The algorithm takes as inputs the current value to test against the threshold/roll-off set (*value*), the value below which the factor will be 0.0 (*threshold*), and the range over which the ramp goes from 0.0 to 1.0 (*rolloff*). The algorithm outputs a value between 0.0 and 1.0 (*factor*).

Algorithm 275 CalculateThresholdRampUpFactor

```

1 function CalculateThresholdRampUpFactor( value::R, threshold::R, rolloff::R )
2   factor::R = 1.0
3   if (value <= threshold)
4     factor = 0.0
5   elseif (value < (threshold + rolloff))
6     factor = (value - threshold) / rolloff
7   end
8   return factor::R
9 end
```

Referenced In: DetermineHorizontalWorstCaseNearMeanFactor(p. 171), DetermineAltitudeDependentCOCFactor(p. 181), DetermineOwnNonComplianceFactor(p. 246), UpdateSafeCrossingRADeferralCState(p. 209), DetermineVerticalRateReductionFactor(p. 182), DetermineIntruderAlert(p. 275), UpdateCoordinatedRADeferralCState(p. 226), UpdateLowAltitudeParallelRADeferralCState(p. 231)

EXPECTEDTAU (Algorithm 276) calculates the weighted average value for tau. To do this the algorithm loops through each belief in *b_tau_int* and adds the weight to *w_cumulative* and the tau value multiplied by the weight to *tau_cumulative*. If a tau value has no weight, it has no impact on the calculation of tau. The computation of *tau_expected* is done by taking the cumulative weighted time, *tau_cumulative* and dividing it by the cumulative weight, *w_cumulative*. If *w_cumulative* is 0.0 then *tau_expected* is set to the horizon value, which is located in the final bin.

The algorithm takes as inputs the current tau beliefs (*b_tau_int*), and a flag on whether to exclude the horizon tau (the last tau bin) or not (*exclude_tau_horizon*). The algorithm outputs the weighted average value for tau (*tau_expected*).

Algorithm 276 ExpectedTau

```

1 function ExpectedTau( b_tau_int::Vector{TauBelief(p. E-38)}, exclude_tau_horizon::Bool )
2   w_cumulative::R = 0.0
3   tau_cumulative::R = 0.0
4   t::Z = length( b_tau_int )
5   if exclude_tau_horizon
6     t = t - 1
7   end
8   for i = 1:t
9     w_cumulative = w_cumulative + b_tau_int[i].weight
10    tau_cumulative = tau_cumulative + (b_tau_int[i].tau * b_tau_int[i].weight)
11  end
12  tau_expected::R = b_tau_int[end].tau
13  if (0.0 < w_cumulative)
14    tau_expected = tau_cumulative / w_cumulative
15  end
16  return tau_expected::R
17 end

```

Referenced In:	StateAndCostEstimation(p. 161), OnlineUncoordinatedCostEstimation(p. 191), OnlineCostEstimation(p. 214)
-----------------------	---

GETOWNHEIGHT (Algorithm 277) determines the current height of ownship. If the radio altitude is defined (i.e., not *NaN*) and is less than the maximum reporting height for radio altimeters, then the radio altitude is returned. Otherwise, the tracked barometric altitude for ownship is used.

The algorithm takes as inputs ownship radio altitude (*h_own*), and ownship barometric altitude (*z_own_ave*). The algorithm outputs current ownship height (*height*).

Algorithm 277 GetOwnHeight

```

1 function GetOwnHeight( h_own::R, z_own_ave::R )
2   const H_radalt_limit::R = params().threat_resolution.H_radalt_limit
3   height::R = z_own_ave
4   if !isnan( h_own ) && (h_own <= H_radalt_limit)
5     height = h_own
6   end
7   return height::R
8 end

```

Referenced In:	GetVTRMOwnData(p. 153), UpdateLowAltitudeParallelRADeferralCState(p. 231)
-----------------------	---

GETPROTECTIONMODEINDEX (Algorithm 278) returns the index into the protection modes array of the parameter files whose code matches the input protection mode code.

The algorithm takes as input the code for the protection mode to retrieve (*protect_mode_int*). The algorithm outputs the index of the protection mode that corresponds to the input code (*mode_int*).

Algorithm 278 GetProtectionModeIndex

```

1 function GetProtectionModeIndex( protect_mode_int::UInt8 )
2     mode_int::Z = -1
3     for i = 1:length( params().modes )
4         n_protect_mode_code = int( params().modes[i].protection_mode_code )
5         if (n_protect_mode_code == protect_mode_int)
6             mode_int = i
7             break
8         end
9     end
10    return mode_int::Z
11 end

```

Referenced In: SetInvalidIntruder(p. 324), VerticalTRMUpdatePrep(p. 1-2), IntruderPrep(p. 165), DetermineRACoordinationData(p. 309)

INTERPOLANTS (Algorithm 279) converts input points into a weighted distribution of points that are represented in a table. This process, known as multi-linear interpolation, is done by weighting the indices in the table that are closest to the input point in such a way that their weighted sum is equivalent to the input point. Each dimension of the interpolation can be done independently using traditional linear interpolation and then combined to produce the final weighting.

Initial processing is done to determine the start and end indices of one dimension of the table to be interpolated. These indices are used to extract a vector of the table on which traditional linear interpolation can be done with INTERPOLATEONEDIMENSION (Algorithm 280). The process repeats along each dimension until the final weights across all dimensions are obtained.

The algorithm takes as inputs the multi-dimensional state to be represented with table indices (*point*), a table format parameter that indicates the number of indices along each dimension of the table (*cut_counts*), the table dimension on which interpolation should begin (*start_dim*), the total number of dimensions over which interpolation is done (*num_dimensions*), and the indices of the data table (*cuts*). The algorithm outputs a vector of indices of the data table that can be used to represent the input point (*indices*), a vector of weights that corresponds to the indices of the data table that can be used to represent the input point (*weights*), and the number of elements in the portion of the table over which interpolation was done (*block_size*).

Additional information related to the data table formats can be found in Appendix G.

Algorithm 279 Interpolants

```

1 function Interpolants( point::Vector{R}, cut_counts::Vector{Z}, start_dim::Z, num_dimensions::Z,
2   cuts::Vector{R} )
3   indices::Vector{Z} = Z[1]
4   weights::Vector{R} = R[1..0]
5   block_size::Z      = 1
6   idx_cut_start::Z = 1
7   for j = 1:(start_dim-1)
8     idx_cut_start = idx_cut_start + cut_counts[j]
9   end
10  for j = start_dim:(start_dim-1+num_dimensions)
11    idx_cut_end::Z = idx_cut_start + cut_counts[j] - 1
12    cuts_subset::Vector{R} = cuts[ idx_cut_start:idx_cut_end ]
13    (indices, weights, block_size) =
14      InterpolateOneDimension(p.331)( indices, weights, point[j], cuts_subset, block_size )
15    idx_cut_start = idx_cut_start + cut_counts[j]
16  end
17  return (indices::Vector{Z}, weights::Vector{R}, block_size::Z)
18 end

```

Referenced In: InterpolateBlocks(p. 177), LookupOfflineCost(p. 188)

INTERPOLATEONEDIMENSION (Algorithm 280) performs linear interpolation across a single dimension and combines it with the results from previous dimensions through multi-linear interpolation. Initial processing is performed to determine the indices of the table that are directly below (*idx_lo*) and above (*idx_hi*) the input value. Values that are less than the first index value or greater than the last index value are capped at the end value by setting both the upper and lower indices to be the first or last index, respectively. If a value falls between the table end points values, it is compared to the value at each index in turn until an index value is not less than the input value. While generally unlikely, if the input point is exactly equal to an index value, both the upper and lower indices will be set to that index. Otherwise, the upper index is set to the current index and the lower index is set to the previous index.

The upper and lower indices are used to perform the linear interpolation along the single dimension and the results are then combined with the indices from previous dimensions for the multi-linear interpolation. If the upper and lower indices are the same, the total number of indices from the input to the output does not change. However, the location of each input index must be offset by the product of the single dimension index (decremented by 1) and the block size to incorporate the information from the new dimension. The weights of the indices are not affected, which indicates that no interpolation was done. The more likely case, that the upper and lower indices are distinct, is slightly more complex. The total number of indices received as input is doubled for the output to accommodate the new interpolation. The indices of the first half are offset by the product of the lower index (decremented by 1) and the block size, while the indices of the second half are offset by the product of the upper index (decremented by 1) and the block size. The number of weights is similarly increased to match the number of indices. A weighting factor for the new dimension is calculated with standard linear interpolation methods using the input value and the values at the upper and lower indices. The first half of the new, larger weights vector is multiplied by the weighting factor while the second half is multiplied by the 1.0 minus the weighting factor. This operation applies the results of the one dimensional linear interpolation to the set of previous indices completing the multi-linear interpolation.

Before completion, the new, larger block size is calculated for use in subsequent calls to INTERPOLATEONEDIMENSION.

The algorithm takes as inputs an array of indices from other dimensions used in multi-linear interpolation (*indices*), a vector of weights that corresponds to the indices from other dimensions used in multi-linear interpolation (*w*), the one dimensional value to be interpolated (*point*), the indices of the table along the dimension being interpolated (*cuts*), and the offset needed to index into the table along this dimension (*subblock_size*). The algorithm outputs an array of indices, including this dimension, that can be used to represent a data point (*indices_out*), a vector of weights that corresponds to the indices, including this dimension, that can be used to represent data point (*weights*), and the offset needed to index into the table along the next dimension (*subblock_size_out*).

Algorithm 280 InterpolateOneDimension

```

1 function InterpolateOneDimension( indices::Vector{Z}, w::Vector{R}, point::R, cuts::Vector{R},
2                                     subblock_size::Z )
3     const L::Z = length( cuts )
4     const D::Z = length( w )
5     idx_lo::Z = 0
6     idx_hi::Z = 0
7     if (point <= cuts[1])
8         idx_lo = 1
9         idx_hi = 1
10    elseif (cuts[L] <= point)
11        idx_lo = L
12        idx_hi = L
13    else
14        i = 1
15        while (cuts[i] < point)
16            i = i + 1
17        end
18        if (cuts[i] == point)
19            idx_lo = i
20            idx_hi = i
21        else
22            idx_lo = (i-1)
23            idx_hi = i
24        end
25    end
26    indices_out::Vector{Z} = Z[]
27    w_out::Vector{R}      = R[]
28    D_out::Z              = 0
29    if (idx_lo == idx_hi)
30        indices_out = Array( Z, D )
31        for i = 1:D
32            indices_out[i] = indices[i] + ((idx_lo - 1) * subblock_size)
33        end
34        w_out = copy( w )
35        D_out = D
36    else
37        D_out = D + D
38        indices_out = Array( Z, D_out )
39        for i = 1:D
40            indices_out[i] = indices[i] + ((idx_lo - 1) * subblock_size)
41            indices_out[i+D] = indices[i] + ((idx_hi - 1) * subblock_size)
42        end
43        w_out = Array( R, D_out )
44        w_lo::R = 1.0 - ((point - cuts[idx_lo]) / (cuts[idx_hi] - cuts[idx_lo]))
45        for i = 1:D
46            w_out[i] = w[i] * w_lo
47            w_out[i+D] = w[i] * (1.0 - w_lo)
48        end
49    end
50    subblock_size_out::Z = subblock_size * L
51    return (indices_out::Vector{Z}, w_out::Vector{R}, subblock_size_out::Z)
52 end

```

Referenced In: Interpolants(p. 329)

IsCOC (Algorithm 281) determines if the input action corresponds to the clear of conflict advisory.

The algorithm takes as inputs the maximum vertical rate of the action (dz_{max}), and the minimum vertical rate of the action (dz_{min}). The algorithm outputs a boolean flag that is TRUE if the input action is clear of conflict and FALSE otherwise (is_coc).

Algorithm 281 IsCOC

```

1 function IsCOC( dz_min::R, dz_max::R )
2   is_coc::Bool = false
3   if (dz_min == -Inf) && (dz_max == Inf)
4     is_coc = true
5   end
6   return is_coc::Bool
7 end
```

Referenced In: DetermineDisplayData(p. 283), UpdateAdvisoryRestartCState(p. 195), IsPreventive(p. 336), LowAltitude-ParallelRADeferralCost(p. 229), UpdateMaxReversalCState(p. 236), AdvisoryRestartCost(p. 194), RestrictCOCDueToReversal(p. 238), UpdateRestrictCOCDueToReversalCState(p. 238), CrossingNoAlertCost(p. 228), CriticalIntervalProtection-Cost(p. 201), GetModifiedGlobalRates(p. 192), InitializationCost(p. 205), UpdatePreventEarlyCOCCState(p. 207), SafeCrossingRADeferralCost(p. 208), MaxReversalCost(p. 235), UpdateBadTransitionCState(p. 199), PreventEarlyCOCCost(p. 206), AltitudeInhibitCost(p. 197), UpdateCriticalIntervalProtectionCState(p. 203), SA01Heuristic(p. 239), AltitudeDependent-COCCost(p. 196), CompatibilityCost(p. 223), TimeBasedNonComplianceCost(p. 242), UpdateTimeBasedNonComplianceC-State(p. 244), UpdateIndivAdjustCounts(p. 305), CoordinatedRADeferralCost(p. 225), CoordinationDelayCost(p. 227)

ISCORRECTIVE (Algorithm 282) examines the vertical rate limits of an input action (*dz_min*, *dz_max*) to determine whether that action is corrective. This algorithm considers only the vertical rate limits of the input action when determining if an action is corrective. Consequently, Level-off RAs are not considered correctives by this algorithm because they have the same vertical rate limits as Monitor Vertical Speed RAs. Additionally, Maintain RAs are not considered to be correctives by this algorithm.

The algorithm takes as inputs the maximum vertical rate of the action (*dz_max*), and the minimum vertical rate of the action (*dz_min*). The algorithm outputs a boolean flag that is TRUE if the input action is corrective and FALSE otherwise (*is_corrective*).

Algorithm 282 IsCorrective

```

1 function IsCorrective( dz_min::R, dz_max::R )
2   is_corrective::Bool = false
3   if (((-Inf == dz_min) && (dz_max < 0.0)) || ((0.0 < dz_min) && (Inf == dz_max))) &&
4     !IsMaintain(p. 335)( dz_min, dz_max )
5     is_corrective = true
6   end
7   return is_corrective::Bool
8 end
```

Referenced In: PreventEarlyWeakeningCost(p. 237)

ISCROSSING (Algorithm 283) determines if the sense of an intruder action will result in an altitude crossing between ownship and intruder. For example, if the intruder is below the ownship and the intruder action sense is :Up, the algorithm will return TRUE. If the aircraft are closer than the altitude buffer, (*CROSSING_ALTITUDE_BUFFER*) the *is_crossing* flag is always set to FALSE.

The algorithm takes as inputs current barometric altitude of ownship (*z_ac1_ave*), current barometric altitude of the intruder (*z_ac2_ave*), and current sense of the intruder RA (*sense_ac2*). The algorithm outputs a boolean flag that is TRUE if following the intruder ra sense indicates a desire to cross altitude (*is_crossing*).

Algorithm 283 IsCrossing

```

1 function IsCrossing( z_ac1_ave::R, z_ac2_ave::R, sense_ac2::Symbol )
2   global CROSSING_ALTITUDE_BUFFER
3   is_crossing::Bool = false
4   if (CROSSING_ALTITUDE_BUFFER <= abs( z_ac2_ave - z_ac1_ave ))
5     if ((z_ac2_ave < z_ac1_ave) && (sense_ac2 == :Up)) || 
6       ((z_ac1_ave < z_ac2_ave) && (sense_ac2 == :Down))
7       is_crossing = true
8     end
9   end
10  return is_crossing::Bool
11 end
```

Referenced In: CompatibilityCost(p.223), UpdateCrossingNoAlertCState(p.229)
--

IsDIVERGING (Algorithm 284) determines whether the sense of an intruder action is directed away from the ownship altitude. For example, if the intruder is below the ownship and the intruder action sense is :*Down*, the algorithm will return TRUE.

The algorithm takes as inputs current barometric altitude of ownship (*z_own_ave*), current barometric altitude of the intruder (*z_int_ave*), and current sense of the intruders RA (*sense_int*). The algorithm outputs a boolean flag that is TRUE if the intruder RA sense is directed away from the ownship altitude (*is_diverging*).

Algorithm 284 IsDiverging

```

1 function IsDiverging( z_own_ave::R, z_int_ave::R, sense_int::Symbol )
2   is_diverging::Bool = false
3   if ((z_int_ave < z_own_ave) && (sense_int == :Down)) ||
4     ((z_own_ave < z_int_ave) && (sense_int == :Up))
5     is_diverging = true
6   end
7   return is_diverging::Bool
8 end
```

Referenced In: UpdateCoordinatedRADeferralCState(p.226)
--

IsDNC (Algorithm 285) uses the vertical rate limits of a given action to determine whether the action is Do Not Climb. This is done by comparing the input vertical rate limits to the pre-defined vertical limits of Do Not Climb.

The algorithm takes as inputs the maximum vertical rate of the action (*dz_max*) and the minimum vertical rate of the action (*dz_min*). The algorithm outputs a boolean flag that is TRUE if the input action is Do Not Climb (*is_dnc*).

Algorithm 285 IsDNC

```

1 function IsDNC( dz_min::R, dz_max::R )
2   is_dnc::Bool = false
3   if (dz_min == -Inf) && (dz_max == 0)
4     is_dnc = true
5   end
6   return is_dnc::Bool
7 end
```

Referenced In: PersistMTLO(p. 341), PreventEarlyWeakeningCost(p. 237)

IsDND (Algorithm 286) uses the vertical rate limits of a given action to determine whether the action is Do Not Descend. This is done by comparing the input vertical rate limits to the pre-defined vertical limits of Do Not Descend.

The algorithm takes as inputs the maximum vertical rate of the action (*dz_max*) and the minimum vertical rate of the action (*dz_min*). The algorithm outputs a boolean flag that is TRUE if the input action is Do Not Descend (*is_dnd*).

Algorithm 286 IsDND

```

1 function IsDND( dz_min::R, dz_max::R )
2   is_dnd::Bool = false
3   if (dz_min == 0) && (dz_max == Inf)
4     is_dnd = true
5   end
6   return is_dnd::Bool
7 end
```

Referenced In: PersistMTLO(p. 341), PreventEarlyWeakeningCost(p. 237)

ISINTRUDERMASTER (Algorithm 287) determines whether or not the intruder is the master aircraft in an equipped-equipped encounter. In order for intruder to be the master aircraft, both aircraft must be in TA/RA mode and the intruder Mode S address must be lower than the Mode S address of the ownship.

The algorithm takes as inputs input data for ownship (*input_own*) and input data for the intruder (*input_int*). The algorithm outputs a boolean flag that is TRUE if the intruder is master (*is_master_int*).

Algorithm 287 IsIntruderMaster

```

1 function IsIntruderMaster( input_own::TRMOwnInput(p. E-19), input_int::TRMIIntruderInput(p. E-20) )
2   is_master_int::Bool = false
3   if (EQUIPAGE_CASRA == input_int.equipage) &&
4     (OPMODE_RA == input_own.opmode) &&
5     (input_int.address < input_own.mode_s)
6     is_master_int = true
7   end
8   return is_master_int::Bool
9 end
```

Referenced In: IntruderPrep(p. 165)

IsMAINTAIN (Algorithm 288) determines whether an input action is a Maintain advisory. This is done by comparing the vertical rates of the input action against all non-maintain vertical rates in the parameter file. If there is no match, the action is a Maintain RA.

The algorithm takes as inputs the maximum vertical rate of the action (*dz_max*) and the minimum vertical rate of the action (*dz_min*). The algorithm outputs a boolean flag that is TRUE if the input action is a Maintain RA (*is_maintain*).

Algorithm 288 IsMaintain

```

1 function IsMaintain( dz_min::R, dz_max::R )
2   const R_dz_min_adv::Vector{R} = params().actions.min_rates
3   const R_dz_max_adv::Vector{R} = params().actions.max_rates
4   is_maintain::Bool = true
5   for act = 1:length( R_dz_min_adv )
6     if (R_dz_min_adv[act] == dz_min) && (R_dz_max_adv[act] == dz_max)
7       is_maintain = false
8     end
9   end
10  return is_maintain::Bool
11 end
```

Referenced In: CoordinationSelection(p. 269), IsCorrective(p. 332), LowAltitudeParallelRADeferralCost(p. 229), Sandwich-PreventionCost(p. 257), UpdateBadTransitionCState(p. 199), BadTransitionCost(p. 198), MaintainRates(p. 339), AltitudeDependentCOCCost(p. 196)

ISMASTERFORCINGREVERSAL (Algorithm 289) determines if the intruder is forcing ownship to issue a reversal. A prerequisite for the intruder to force a reversal is that it must be master. The determination of whether the reversal is forced is done by examining whether the sense of the coordination message received from intruder is the same as the sense selected by ownship on the previous cycle.

The algorithm takes as inputs a flag for if the current intruder is the master (*master_int*), the sense of the action selected by ownship on the previous cycle (*sense_own_prev*), and the sense of the current intruder RA (*sense_int*). The algorithm outputs a boolean flag that is TRUE if the intruder is forcing ownship to reverse (*is_forced_reversal*).

Algorithm 289 IsMasterForcingReversal

```

1 function IsMasterForcingReversal( master_int::Bool, sense_own_prev::Symbol, sense_int::Symbol )
2   is_forced_reversal::Bool = false
3   if master_int && (sense_own_prev == sense_int)
4     is_forced_reversal = true
5   end
6   return( is_forced_reversal::Bool )
7 end
```

Referenced In: UpdateMaxReversalCState(p. 236), RestrictCOCDueToReversal(p. 238), MaxReversalCost(p. 235)

ISMTO (Algorithm 290) determines whether an action is a Multi-Threat Level-Off (MTLO). This is done by comparing the vertical rates of the input action against the standard MTLO rates in the parameter file. If the vertical rates match, the action is MTLO.

The algorithm takes as inputs the maximum vertical rate of the action (dz_max) and the minimum vertical rate of the action (dz_min). The algorithm outputs a boolean flag that is TRUE if the input action is an MTLO RA (is_mtlo).

Algorithm 290 IsMTLO

```

1 function IsMTLO( dz_min::R, dz_max::R )
2   act_mtlo::Z = MTLOAction(p. 340)()
3   const R_dz_min_L0::R = params().actions.min_rates[act_mtlo]
4   const R_dz_max_L0::R = params().actions.max_rates[act_mtlo]
5   is_mtlo::Bool = false
6   if (dz_min == R_dz_min_L0) && (dz_max == R_dz_max_L0)
7     is_mtlo = true
8   end
9   return is_mtlo::Bool
10 end
```

Referenced In: GetModifiedGlobalRates(p. 192), DetermineCrossing(p. 281)

ISPREVENTIVE (Algorithm 291) determines if the current action is a preventive RA. This is done by inspecting the vertical rate limits of an input action. If the action is not clear of conflict and either dz_min is equal to negative infinity and dz_max is greater than or equal to zero or dz_max is equal to positive infinity and dz_min is less than or equal to zero, the action is considered preventive. It should be noted that this algorithm does not return $is_preventive$ as TRUE for Maintain RAs.

The algorithm takes as inputs the maximum vertical rate of the action (dz_max) and the minimum vertical rate of the action (dz_min). The algorithm outputs a boolean flag that is TRUE if the input action is a preventive RA ($is_preventive$).

Algorithm 291 IsPreventive

```

1 function IsPreventive( dz_min::R, dz_max::R )
2   is_preventive::Bool = false
3   if !IsCOC(p. 332)( dz_min, dz_max ) &&
4     ((-Inf == dz_min) && (0.0 <= dz_max)) || ((dz_min <= 0.0) && (Inf == dz_max))
5   is_preventive = true
6   end
7   return is_preventive::Bool
8 end
```

Referenced In: LowAltitudeParallelRADeferralCost(p. 229), SandwichPreventionCost(p. 257), BadTransitionCost(p. 198), AltitudeDependentCOCCost(p. 196), IndividualCostEstimation(p. 248), CoordinatedRADeferralCost(p. 225)

ISPROJECTEDCROSSING (Algorithm 292) uses the current and projected relative altitude between ownship and intruder to determine if an altitude crossing will occur. This is done by comparing the sign of the two relative altitude inputs, with explicit checks for zero relative altitude.

The algorithm takes as inputs the current altitude of intruder relative to ownship (z_{rel}) and the projected altitude of intruder relative to ownship ($z_{rel_projected}$). The algorithm outputs a boolean flag that is TRUE if ownship is projected to cross altitudes with the intruder ($is_projected_crossing$).

Algorithm 292 IsProjectedCrossing

```

1 function IsProjectedCrossing( z_rel::R, z_rel_projected::R )
2   is_projected_crossing::Bool = false
3   if (sign( z_rel ) != sign( z_rel_projected )) || 
4     (0 == sign( z_rel )) || (0 == sign( z_rel_projected ))
5   is_projected_crossing = true
6 end
7 return is_projected_crossing::Bool
8 end
```

Referenced In: UpdateSafeCrossingRADeferralCState(p. 209), DetermineVerticalProximateFactor(p. 183), UpdateCriticalIntervalProtectionCState(p. 203), UpdateCoordinatedRADeferralCState(p. 226), UpdateTimeBasedNonComplianceC-State(p. 244)

ISREVERSAL (Algorithm 293) determines if the current RA sense is a reversal of the RA sense from the previous cycle.

The algorithm takes as inputs the RA sense from the previous cycle (*sense_prev*) and the sense of the current RA (*sense_curr*). The algorithm outputs a boolean flag that is TRUE when the current RA is a reversal (*is_reversal*).

Algorithm 293 IsReversal

```

1 function IsReversal( sense_prev::Symbol, sense_curr::Symbol )
2   is_reversal::Bool = false
3   if (:Up == sense_prev) && (:Down == sense_curr) || ((:Down == sense_prev) && (:Up == sense_curr))
4     is_reversal = true
5   end
6   return( is_reversal::Bool )
7 end
```

Referenced In: UpdateMaxReversalCState(p. 236), MaxReversalCost(p. 235), BadTransitionCost(p. 198)

ISSLOWCLOSURE (Algorithm 294) determines if the ownship and intruder are in a slow, horizontal convergence geometry where vertical separation still exists. This is done by comparing the relative ground range and horizontal closure rate to protection-mode-dependent thresholds obtained from the parameters file. While not directly tied to horizontal geometry, the altitude of the intruder relative to ownship is also compared to a threshold to see if sufficient vertical separation still exists.

The algorithm takes as inputs the intruder protection mode (*mode_int*), the ground range between intruder and ownship (*r_ground*), the magnitude of the relative velocity between intruder and ownship (*s_ground*), and the altitude of intruder relative to ownship (*z_rel*). The algorithm outputs a boolean flag that is TRUE if the current intruder is in a horizontal slow closure with ownship (*is_slow_closure*).

Algorithm 294 IsSlowClosure

```

1 function IsSlowClosure( mode_int::Z, r_ground::R, s_ground::R, z_rel::R )
2   const D_range_thres::R = params().modes[mode_int].state_estimation.tau.D_range_thres
3   const R_speed_thres::R = params().modes[mode_int].state_estimation.tau.R_speed_thres
4   const H_alt_rel_thres::R = params().modes[mode_int].state_estimation.tau.H_alt_rel_thres
5   is_slow_closure::Bool = false
6   if (r_ground < D_range_thres) && (s_ground < R_speed_thres) &&
7     (H_alt_rel_thres < abs( z_rel ))
8     is_slow_closure = true
9   end
10  return is_slow_closure::Bool
11 end

```

Referenced In: TauEstimation(p. 174)

ISSTRENGTHENING (Algorithm 295) determines if the current action is a strengthening of the previous action. This is done by testing whether the sense of the current action is the same sense as the previous action and the target rate of the current action has a greater magnitude than the previous action. The sense of the current and previous action are determined with a call to RATESTOSENSE (Algorithm 302).

The algorithm takes as inputs the maximum vertical rate of the previous action (*dz_max_prev*), the minimum vertical rate of the previous action (*dz_min_prev*), the maximum vertical rate of the current action (*dz_max*), and the minimum vertical rate of the current action (*dz_min*). The algorithm outputs a boolean flag that is TRUE if the current action is a strengthening (*is_strengthening*).

Algorithm 295 IsStrengthening

```

1 function IsStrengthening( dz_min_prev::R, dz_max_prev::R, dz_min::R, dz_max::R )
2   is_strengthening::Bool = false
3   sense_own_prev::Symbol = RatesToSense(p.342)( dz_min_prev, dz_max_prev )
4   sense_own_curr::Symbol = RatesToSense(p.342)( dz_min, dz_max )
5   if (sense_own_prev == sense_own_curr)
6     if (sense_own_curr == :Down)
7       is_strengthening = (dz_max < dz_max_prev)
8     elseif (sense_own_curr == :Up)
9       is_strengthening = (dz_min > dz_min_prev)
10    end
11  end
12  return is_strengthening::Bool
13 end

```

Referenced In: BadTransitionCost(p. 198)

ISVERTICALLYCONVERGING (Algorithm 296) determines if ownship and an intruder are converging vertically. This is done by determining if the altitude and vertical rate of the intruder relative to ownship indicate the aircraft are closing in altitude. Explicit checks are performed to ensure that the magnitude of the vertical rate of the intruder relative to ownship is above a threshold. This prevents divide by zero failures and ensures sufficient relative vertical rate exists to determine convergence.

The algorithm takes as inputs the protection mode of the intruder (*mode_int*), current barometric altitude of ownship (*z_own_ave*), current vertical rate of the ownship (*dz_own_ave*), average of the belief states of intruder barometric altitude (*z_int_ave*), and average of the belief states of intruder

vertical rate (dz_int_ave). The algorithm outputs a boolean flag that is TRUE if ownship and intruder are converging vertically (*is_vertically_converging*).

Algorithm 296 IsVerticallyConverging

```

1 function IsVerticallyConverging( mode_int::Z, z_own_ave::R, dz_own_ave::R, z_int_ave::R, dz_int_ave::R )
2   const R_converge_thres_vert::R =
3     params().modes[mode_int].state_estimation.tau.R_converge_thres_vert
4   is_vertically_converging::Bool = false
5   if (R_converge_thres_vert < abs( dz_own_ave - dz_int_ave )) &&
6     (0.0 < ((z_int_ave - z_own_ave) / (dz_own_ave - dz_int_ave)))
7   is_vertically_converging = true
8   end
9   return is_vertically_converging::Bool
10 end
```

Referenced In: TauEstimation(p. 174)

MAINTAINRATES (Algorithm 297) uses the current vertical rate of the ownship to determine the minimum and maximum vertical rates limits for a Maintain RA. If dz_own_ave is zero or less, dz_min is set to dz_own_ave and dz_max is set to infinity. If dz_own_ave is greater than zero, dz_max is set to dz_own_ave and dz_min is set to negative infinity. IsMaintain is called with the calculated rates to protect against the possibility that the ownship vertical rate is exactly equal to a standard RA target rate. If this occurs, *epsilon*, a small value, is either added to dz_max or subtracted from dz_min , as appropriate, to prevent the Maintain RA from being indistinguishable with a standard RA. Additionally, if the ownship vertical rate is exactly zero, dz_max is set to positive infinity and dz_min is set to *epsilon*.

The algorithm takes as input the current ownship vertical rate (dz_own_ave). The algorithm outputs the maximum vertical rate of the Maintain RA (dz_max) and the minimum vertical rate of the Maintain RA (dz_min).

Algorithm 297 MaintainRates

```

1 function MaintainRates( dz_own_ave::R )
2   dz_min::R = Nan
3   dz_max::R = Nan
4   epsilon::R = 2e-4
5   if (0 <= dz_own_ave)
6     (dz_min, dz_max) = (dz_own_ave, Inf)
7     if !IsMaintain(p.335)( dz_min, dz_max )
8       if (0 < dz_min)
9         dz_min = dz_min - epsilon
10      else
11        dz_min = epsilon
12      end
13    end
14  else
15    (dz_min, dz_max) = (-Inf, dz_own_ave)
16    if !IsMaintain(p.335)( dz_min, dz_max )
17      dz_max = dz_max + epsilon
18    end
19  end
20  return (dz_min::R, dz_max::R)
21 end
```

Referenced In: DetermineSenses(p. 260), ActionToRates(p. 325)
--

MINCOSTINDEX (Algorithm 298) returns the index of the lowest cost action in *costs*. The lowest cost index is initialized to the first action in the *costs* array before beginning a comparison through the remaining elements. If an element is found to have a cost less than the current lowest cost by more than *epsilon*, a small value, the lowest cost index is updated to the current element. Action costs that are within *epsilon* are considered equivalent, with the lower indexed action being preferred.

The algorithm takes as inputs a vector containing the weighted cost of each potential action (*costs*) and the minimum difference between two actions required to consider them distinct (*epsilon*). The algorithm outputs the index of the action that has the lowest cost (*cost_min_idx*).

Algorithm 298 MinCostIndex

```

1 function MinCostIndex( costs::Vector{R}, epsilon::R )
2   cost_min_idx::Z = 1
3   cost_min::R = costs[1]
4   for i = 2:length( costs )
5     if ((costs[i] + epsilon) < cost_min)
6       cost_min_idx = i
7       cost_min = costs[i]
8     end
9   end
10  return (cost_min_idx::Z)
11 end
```

Referenced In: DetermineMinimumCostAction(p. 252), IndividualSelectionCostEstimation(p. 255), UnequippedCostFusion(p. 258), IndividualCostEstimation(p. 248)

MTLOACTION (Algorithm 299) returns the index for MTLO in the action vector. The algorithm takes no inputs. The algorithm outputs the index of MTLO in the action vector (*action_mtlo*).

Algorithm 299 MTLOAction

```

1 function MTLOAction()
2   const N_actions::Z = params().actions.num_actions
3   action_mtlo::Z = N_actions
4   return action_mtlo::Z
5 end
```

Referenced In: IsMTLO(p. 336), DetermineMinimumCostAction(p. 252), IndividualSelectionCostEstimation(p. 255), Un-equippedCostFusion(p. 258), DetermineMultiIntruderAction(p. 254), PersistMTLO(p. 341), IndividualCostEstimation(p. 248)

PERSISTMTLO (Algorithm 300) determines if a multi-threat level-off (MTLO) action issued last cycle should continue this cycle. An MTLO should persist if the advisory in the previous cycle was an MTLO, if multiple threats still exist, and if the current action would be considered a corrective alert in an encounter with a single intruder. This last condition is evaluated by checking the compatibility between the ownship's vertical rate in the previous cycle and the sense of the current advisory, which in a multithreat encounter can only be DND or DNC; an advisory that restricts the vertical rate of ownship to be in the opposite sense of that of the previous cycle would be considered a corrective alert in an encounter with a single intruder. If one of these two alerts was selected by the logic through single-threat processing and that action would be a corrective based on *dz_own_ave_prev*, the *persist_mtlo* flag is set to TRUE.

The algorithm takes as inputs the action selected for the current cycle (*action*), a boolean flag that is TRUE if there are multiple threats (*multithreat*), current vertical rate of the ownship (*dz_own_ave*), vertical rate of the ownship from the previous cycle (*dz_own_ave_prev*), and the action selected during the previous cycle (*a_global_prev*). The algorithm outputs a boolean flag that is TRUE if the MTLO should be persisted (*persist_mtlo*).

Algorithm 300 PersistMTLO

```

1 function PersistMTLO( action::Z, multithreat::Bool,
2           dz_own_ave::R, dz_own_ave_prev::R,
3           a_global_prev::GlobalAdvisory(p.E-35 ) )
4   persist_mtlo::Bool = false
5   if multithreat && (a_global_prev.action == MTLOAction(p.340()))
6     (dz_min::R, dz_max::R) =
7       ActionToRates(p.325)( action, dz_own_ave, a_global_prev.action,
8                           a_global_prev.dz_min, a_global_prev.dz_max, a_global_prev.ddz )
9     if IsDNC(p.334)( dz_min, dz_max ) && (0 <= dz_own_ave_prev)
10      persist_mtlo = true
11    elseif IsDND(p.334)( dz_min, dz_max ) && (dz_own_ave_prev < 0)
12      persist_mtlo = true
13    end
14  end
15  return persist_mtlo::Bool
16 end
```

Referenced In: MTLODetermination(p.259) , DetermineMultiIntruderAction(p.254)
--

RATESTOACTION (Algorithm 301) maps the vertical rate limits and vertical acceleration of an action to the corresponding action number. Three tables from the parameters file are used to perform the mapping. The first table contains the minimum vertical rate for each possible action and is stored in the variable *R_dz_min_adv*. The second table contains the maximum vertical rate for each possible action and is stored in the variable *R_dz_max_adv*. The third table contains the vertical acceleration for each possible action and is stored in the variable *R_ddz_adv*.

The input vertical rates and vertical acceleration are compared to those corresponding to the standard RAs. If a match is found, the *action* variable is set to the action number for that RA. Since Maintain RAs are represented in the vertical rate parameter file tables with *Nan*, comparison for Maintain will always return FALSE. Explicit checks for *Nan* are performed to determine the action number of Maintain during the standard RA comparison step. If no action was matched using the standard values, additional processing is done to check for Maintain RAs. This consists of checking for the potential representations of Maintain: *dz_min* as negative infinity with a negative *dz_max*, *dz_max* as positive infinity with a positive *dz_min*, or either *dz_min* or *dz_max* being defined as *Nan*.

The algorithm takes as inputs the minimum vertical rate of the action (*dz_min*), the maximum vertical rate of the action (*dz_max*), and the vertical acceleration of the action (*ddz*). The algorithm outputs the action that corresponds to the input vertical rate limits and vertical acceleration (*action*).

Algorithm 301 RatesToAction

```

1 function RatesToAction( dz_min::R, dz_max::R, ddz::R )
2   const R_dz_min_adv::Vector{R} = params().actions.min_rates
3   const R_dz_max_adv::Vector{R} = params().actions.max_rates
4   const R_ddz_adv::Vector{R} = params().actions.accelerations
5   action::Z = 0
6   maintain::Z = 0
7   for i = 1:length(R_dz_min_adv)
8     if (dz_min == R_dz_min_adv[i]) && (dz_max == R_dz_max_adv[i]) && (ddz == R_ddz_adv[i])
9       action = i
10      break
11    elseif isnan(R_dz_min_adv[i] ) || isnan(R_dz_max_adv[i])
12      maintain = i
13    end
14  end
15  if (action == 0) &&
16    (((dz_min == -Inf) && (dz_max < 0)) ||
17     ((0 <= dz_min) && (dz_max == Inf)) ||
18     isnan(dz_min) ||
19     isnan(dz_max))
20    action = maintain
21  end
22  return action::Z
23 end

```

Referenced In: DetermineMultiIntruderAction(p.254)

RATESTOSENSE (Algorithm 302) determines the sense of an RA from the vertical rate limits of that RA. This is done by comparing the magnitudes of *dz_min* and *dz_max*. If the magnitude of *dz_min* is greater than the magnitude of *dz_max* the sense is :Down. If the magnitude of *dz_max* is greater than the magnitude of *dz_min* the sense is :Up. The only remaining option, that the two vertical rate limits are of equal magnitude, does not have a defined sense and set as :None. The rates of each action used in this comparison can be found in Table F-2.

The algorithm takes as inputs the minimum vertical rate of the action (*dz_min*) and the maximum vertical rate of the action (*dz_max*). The algorithm outputs (*sense*), which indicates if the inputs correspond to an up sense, down sense, or no sense RA.

Algorithm 302 RatesToSense

```

1 function RatesToSense( dz_min::R, dz_max::R )
2   sense::Symbol = :undef
3   if (abs( dz_min ) > abs( dz_max ))
4     sense = :Down
5   elseif (abs( dz_min ) < abs( dz_max ))
6     sense = :Up
7   else
8     sense = :None
9   end
10  return sense::Symbol
11 end

```

Referenced In: DetermineDisplayData(p.283), OwnResponseEstimation(p.219), UpdateMaxReversalCState(p.236), SandwichPreventionCost(p.257), UpdateSA01HeuristicCState(p.240), UpdateRestrictCOCDueToReversalCState(p.238), DetermineSenses(p.260), GetModifiedGlobalRates(p.192), Crosslink(p.270), MaxReversalCost(p.235), UpdateBadTransitionCState(p.199), DetermineCrossing(p.281), BadTransitionCost(p.198), IsStrengthening(p.338), UpdateCriticalIntervalProtectionCState(p.203), SA01Heuristic(p.239), CompatibilityCost(p.223)

TRMINTRUDERSTATEUPDATE (Algorithm 303) updates an internal data structure of the TRM related to intruder state, TRMINTRUDERSTATE (Type 50). Using the inputs provided, the algorithm updates the values in the data structure appropriately.

First, information about the resolution advisory coordination messages sent on this cycle is preserved. Then the advisory reset and initialization reset flags are used to determine how much of the current advisory state should be preserved for the next cycle.

The advisory reset flag (*is_advisory_reset*) indicates whether or not any state information about the current advisory for this intruder should be preserved for the next cycle. The argument *is_advisory_reset* is set to TRUE under any of these conditions: ownship is in TA-Only operational mode, the intruder is designated to DNA and there is no coordination of resolution advisories, there is degraded surveillance for this intruder, the intruder is invalid for TRM processing, or the intruder was dropped. The initialization reset flag (*is_initialization_reset*) indicates whether or not the online cost state should be fully reset when the advisory state is reset. The argument *is_initialization_reset* is set to TRUE whenever the intruder is dropped or invalid. The initialization state is may also be reset in INTRUDERPREP when entering TA/RA operational mode.

When resetting for advisories but preserving the initialization state, the INITIALIZATIONCSTATE *t_count*, CROSSINGNOALERTCSTATE *vrc_int_prev*, LOWALTITUDEPARALLELRADEFERRALC-STATE *range* and *speed_rel* are preserved so traffic and resolution advisories will not be prohibited on the next cycle. When fully resetting for advisories and initialization, the algorithm resets all data values relevant to determining and issuing advisories.

The algorithm takes as inputs current state of the intruder (*st_int*), vrc coordination message from the intruder (*vrc*), cvc coordination message from the intruder (*cvc*), equipage code from the intruder (*equipage*), coordination message type from the intruder (*coordination_msg*), a boolean flag that is TRUE if an advisory reset should occur (*is_advisory_reset*), and a boolean flag that is true if an initialization reset should occur (*is_initialization_reset*). The algorithm outputs no explicit return values; the algorithm makes modifications to the values stored in *st_int*.

Algorithm 303 TRMIIntruderStateUpdate

```

1 function TRMIIntruderStateUpdate( st_int::TRMIIntruderState(p. E-33), vrc::UInt32, cvc::UInt32,
2                               equipage::Z, coordination_msg::UInt32,
3                               is_advisory_reset::Bool, is_initialization_reset::Bool )
4   st_int.protection_mode_prev = st_int.protection_mode_curr
5   st_int.sense_prev = st_int.a_prev.sense
6   st_int.vrc_prev = vrc
7   st_int.cvc_prev = cvc
8   st_int.equipage_prev = equipage
9   st_int.coordination_msg_prev = coordination_msg
10  st_int.a_prev.vrc = vrc
11  st_int.a_prev.cvc = cvc
12  if is_advisory_reset || is_initialization_reset
13    st_int.a_prev = IndividualAdvisory(p. E-36)()
14    st_int.b_prev = AdvisoryBeliefState(p. E-34)()
15    st_int.st_arbitrate = ActionArbitrationCState(p. E-41)()
16    st_int.is_threat = false
17    if !is_initialization_reset
18      t_count_init::Z = st_int.st_cost_on.initialization.t_count
19      range_prev::R = st_int.st_cost_on.low_alt_parallel_ra_deferral.range_prev
20      range_prev_ta::R = st_int.st_cost_on.ta_low_alt_parallel_ra_deferral.range_prev
21      vrc_int_prev::UInt32 = st_int.st_cost_on.crossing_no_alert.vrc_int_prev
22      st_int.st_cost_on = OnlineCostState(p. E-37)()
23      st_int.st_cost_on.initialization.t_count = t_count_init
24      st_int.st_cost_on.low_alt_parallel_ra_deferral.range_prev = range_prev
25      st_int.st_cost_on.ta_low_alt_parallel_ra_deferral.range_prev = range_prev_ta
26      st_int.st_cost_on.crossing_no_alert.vrc_int_prev = vrc_int_prev
27    else
28      st_int.st_cost_on = OnlineCostState(p. E-37)()
29    end
30  end
31 end
```

Referenced In: SetInvalidIntruder(p. 324), SetDroppedIntruder(p. 322), AdjustTARAModeIntruderOutput(p. 302), AdjustTAOnlyModeIntruderOutput(p. 294)

VRCTOSENSE (Algorithm 304) maps the VRC message from an intruder to the appropriate sense symbol.

The algorithm takes as input the VRC coordination message from the intruder (*vrc*). The algorithm outputs the RA sense based on the intruder VRC message (*sense*).

Algorithm 304 VRCToSense

```

1 function VRCToSense( vrc::UInt32 )
2   sense::Symbol = :undef
3   if (vrc == 0)
4     sense = :None
5   elseif (vrc == 1)
6     sense = :Down
7   else
8     sense = :Up
9   end
10  return sense::Symbol
11 end
```

Referenced In: UpdateIntruderInputs(p. 211), UpdateCoordinationDelayCState(p. 227), UpdateMaxReversalCState(p. 236), RestrictCOCDueToReversal(p. 238), IntruderResponseEstimation(p. 218), ShouldReverse(p. 241), Crosslink(p. 270), UpdateCoordinatedRADeferralCState(p. 226), PreventEarlyWeakeningCost(p. 237), UpdateCompatibilityCState(p. 224), CompatibilityCost(p. 223), UpdateCrossingNoAlertCState(p. 229)

4 Acknowledgments

Many people have contributed to this document by providing content or participating in analysis, review, and editing. Thanks goes out to those individuals from MIT Lincoln Laboratory, JHU Applied Physics Laboratory, the FAA Technical Center, Aurora Sciences, and Regulus Group that have provided input. Special recognition is given to the following individuals that have provided significant contributions.

Last Name	First Name	Organization
Alvarez	Luis	MIT Lincoln Laboratory
Asmar	Dylan	MIT Lincoln Laboratory
Bender	Walter	JHU Applied Physics Laboratory
Bezanson	Jeffrey	Julia Computing Inc.
Billingsley	Thomas	MIT Lincoln Laboratory
Brush	Jeff	JHU Applied Physics Laboratory
Capuder	Lawrence	MIT Lincoln Laboratory
Chryssanthacopoulos	James	MIT Lincoln Laboratory
Deal	Forest	JHU Applied Physics Laboratory
Dean	Christopher	MIT Lincoln Laboratory
Drumm	Ann	MIT Lincoln Laboratory
Elder	Tomas	MIT Lincoln Laboratory
Eyler	Alexander	MIT Lincoln Laboratory
Gardner	Ryan	JHU Applied Physics Laboratory
Greaves	Jesse	MIT Lincoln Laboratory
Havener	Robin	MIT Lincoln Laboratory
Jessen	Ian	MIT Lincoln Laboratory
Johnson	David	MIT Lincoln Laboratory
Klaus	Robert	MIT Lincoln Laboratory
Kobzik-Juul	Barbara	JHU Applied Physics Laboratory
Kochenderfer	Mykel	MIT Lincoln Laboratory
Lempka	Matthew	JHU Applied Physics Laboratory
Lopez	Jessica	JHU Applied Physics Laboratory
Lorenzo	Ed	JHU Applied Physics Laboratory
Mackay	Justin	MIT Lincoln Laboratory
Martin	Sean	JHU Applied Physics Laboratory
McLain	Cynthia	MIT Lincoln Laboratory
Monath	Nicholas	MIT Lincoln Laboratory
Moss	Robert	MIT Lincoln Laboratory
Owen	Michael	MIT Lincoln Laboratory
Pagano	Tom	FAA Technical Center
Panken	Adam	MIT Lincoln Laboratory
Perez	Christian	MIT Lincoln Laboratory
Pierce	Mark	MIT Lincoln Laboratory
Puntin	Brendon	MIT Lincoln Laboratory
Schmidt	Aurora	JHU Applied Physics Laboratory
Silbermann	Joshua	JHU Applied Physics Laboratory
Thober	Mark	JHU Applied Physics Laboratory
Treleaven	Kyle	MIT Lincoln Laboratory
Walters	Ethan	Aurora Sciences
Wikle	Jared	MIT Lincoln Laboratory
Williams	Richard	MIT Lincoln Laboratory
Young	Tyler	JHU Applied Physics Laboratory

This page intentionally left blank.

Appendix A STM Housekeeping

At the end of each processing cycle, the STM performs housekeeping tasks through a call to STM-HOUSEKEEPING (Algorithm 305) that are required to happen after the TRM has updated.

Algorithm 305 StmHousekeeping

```
1 function StmHousekeeping( report_trm::TRMReport(p. E-23) )
2   StmHousekeepingTargetDesignation(p. A-2)( report_trm )
3 end
```

STMHOUSEKEEPINGTARGETDESIGNATION (Algorithm 306) updates the Target DESIGNATION-STATE using information in the TRMREPORT input.

First, a dictionary of TRMINTRUDERDESIGNATIONDATA outputs (*intruder_dict*) is constructed. The Target identifier (*id*) is the key to entries in the dictionary. This dictionary is used to match an entry in the target database with a designation output from the TRM.

Next, the designation state (DESIGNATIONSTATE) for each entry in the target database (*target_db*) is updated. The Target associated with each id in the target database is retrieved using RETRIEVEWITHID. If there is a TRMINTRUDERDESIGNATIONDATA output associated with that id, the DESIGNATIONSTATE is updated from *intruder*. Otherwise, the DESIGNATIONSTATE settings that indicate an active designation (*active_dna* and *active_protection_mode*) and an active advisory (*active_ra* and *multithreat_ra*) are reset.

Once the active settings in DESIGNATIONSTATE have been updated, additional settings are updated depending on whether the Target should remain designated. ISTARGETDESIGNATIONACTIVE is used to determine whether the target designation is still active. It is called with *check_timer* set to FALSE so the result will be based on the designation state settings for the next processing cycle, not on the target designation timers (*timer_dna* and *timer_protection_mode*). The target designation timers are updated during GENERATESTMREPORT. If target designation is no longer active, as indicated by ISTARGETDESIGNATIONACTIVE, the Target is marked as undesignated (via *is_designated*, *designated_mode*, and *status*), the timers (*timer_dna* and *timer_protection_mode*) are reset, and state variables for active DNA (*is_landing*, *is_proximate*, *r_ground_history*) are reset.

This algorithm takes as input *report*. This algorithm updates *tgt.designation_state*.

Algorithm 306 StmHousekeepingTargetDesignation

```

1 function StmHousekeepingTargetDesignation( report::TRMReport(p.E-23) )
2   const S_standard_protection_mode::UInt8 = uint8(params().target_designation.S_standard_protection_mode)
3   intruder_dict::Dict<UInt32,TRMIIntruderDesignationData> = Dict<UInt32,TRMIIntruderDesignationData(p.E-26){}()
4   for intruder::TRMIIntruderDesignationData(p.E-26) in report.designation.intruder
5     intruder_dict[intruder.id] = intruder
6   end
7   for id::UInt32 in keys( target_db )
8     tgt::Target(p.E-14) = RetrieveWithID(p.20)( target_db, id )
9     if haskey( intruder_dict, id )
10       intruder = intruder_dict[id]
11       tgt.designation_state.active_ra      = intruder.active_ra
12       tgt.designation_state.multithreat_ra = intruder.multithreat && intruder.active_ra
13       tgt.designation_state.active_dna     = intruder.logic_mode.dna
14       tgt.designation_state.active_protection_mode = intruder.logic_mode.protection_mode
15     else
16       tgt.designation_state.active_ra      = false
17       tgt.designation_state.multithreat_ra = false
18       tgt.designation_state.active_dna     = false
19       tgt.designation_state.active_protection_mode = S_standard_protection_mode
20     end
21     if !IsTargetDesignationActive(p.141)( tgt.designation_state, false )
22       tgt.designation_state.is_designated      = false
23       tgt.designation_state.designated_mode    = DESIGNATION_NONE
24       tgt.designation_state.status            = Xo_STATUS_NONE
25       tgt.designation_state.timer_dna         = 0
26       tgt.designation_state.timer_protection_mode = 0
27       tgt.designation_state.is_landing        = false
28       tgt.designation_state.was_proximate     = false
29       tgt.designation_state.r_ground_history  = Array( R, 0 )
30   end
31 end
32 end

```

Referenced In: StmHousekeeping(p.A-1)

Appendix B Mode C Processing Implementation

The entry point to the suggested Mode C surveillance processing implementation is given by RECEIVEMODECREPLIES (Algorithm 307). This algorithm takes as input the variables described in Table B-1, which includes an array of Mode C replies that have been derived from a single Whisper Shout sequence. It is important that the array of Mode C replies is collected for each interval before calling RECEIVEMODECREPLIES.

Each individual Mode C reply contains the measured slant-range, relative bearing, reported altitude code, altitude confidence bits, and TOA associated with a single Mode C reply. RECEIVEMODECREPLIES ensures that an estimate of ownship barometric altitude and ownship heading are available (i.e., they do not evaluate to *Nan*) as both are needed in the subsequent algorithms.

A call to MERGE MODECREPLIES identifies and merges any replies that have been elicited from a common transponder. This is an important step as often a Whisper Shout sequence will elicit two or more replies from a single transponder and the tracking algorithms can only function on a single reply message. This algorithm returns an array (*merged_replies*) that will always have a length less than or equal to the array *replies*.

An attempt to associate each element of the *merged_replies* array to an existing Target is then performed through a call to ASSOCIATE MODECTOTARGET. This algorithm returns two values. The first is an array of MODECTRACKFILES (*associated_tracks*) that correspond to the associated tracks for each element of *merged_replies*. An unsuccessful association is represented in the *associated_tracks* array as a value of *nothing*. The second value returned from ASSOCIATE MODECTOTARGET is an array of those replies that were not successfully associated to a Target (*unassociated_replies*), which is used in a later step to initialize new MODECTRACKFILES.

Next, each element of the array *associated_tracks* is checked to see if association was successful (i.e., the element in *associated_tracks* does not evaluate to *nothing*). If association was successful, the corresponding Target is retrieved from memory and UPDATEREADY is called to update the MODECTRACKFILE with the measurements in *merged_replies*.

Next, all unassociated replies are used to potentially promote hypothetical Mode C Track Files into established Mode C Track Files. This promotion is done by PROMOTEMODECTRACKS and newly promoted tracks are established through ESTABLISHMODECTRACK.

In the last two steps of RECEIVEMODECREPLIES, new hypothetical tracks are formed using FORMHYPOTHETICALTRACKS, and interval maintenance is performed with a call to INTERVALMAINTENANCE.

This algorithm takes as input the variables in the following table. This algorithm updates *promoted_tracks* and calls algorithms ESTABLISHMODECTRACK, FORMHYPOTHETICALTRACKS, and INTERVALMAINTENANCE.

Table B-1. STM Input Variables - Mode C Replies Message

Variable	Units	Type	Description
replies	N/A	N/A	Array of Mode C replies derived from a single Whisper Shout sequence

Algorithm 307 ReceiveModeCReplies

```

1 function ReceiveModeCReplies(replies::Vector{ModeCReply(p. E-7)})  

2   if (isnan(own.toa_h) || (!own.heading_initialized)  

3     return  

4   end  

5   merged_replies = MergeModeCReplies(p. B-3)(replies)  

6   (associated_tracks, unassociated_replies) = AssociateModeCtoTarget(p. B-5)(merged_replies)  

7   for i in 1:length(associated_tracks)  

8     if (associated_tracks[i] != nothing)  

9       UpdateModeCTrack(p. 24)(associated_tracks[i], merged_replies[i])  

10      EvaluateOnGroundModeC(p. B-17)(associated_tracks[i], merged_replies[i].toa, merged_replies[i].  

           coded_alt)  

11    end  

12  end  

13  promoted_tracks = PromoteModeCTracks(p. B-10)(unassociated_replies)  

14  EstablishModeCTrack(p. B-11)(promoted_tracks)  

15  FormHypotheticalTracks(p. B-12)  

16  IntervalMaintenance(p. B-16)  

17 end

```

MERGE MODE C REPLIES (Algorithm 308) takes as input a vector of Mode C replies generated during a single Whisper Shout sequence and merges together all replies believed to have been generated by a common intruder. Each reply is compared to all others and assigned a probabilistic score based on how well they correlate in range, altitude and bearing.

Prior to scoring the replies are sorted by range for more efficient iteration. The difference in range between a pair of replies must be below the *range_gate* in order to proceed. Altitude codes from each reply are decoded into barometric altitude using Gillham altitude decoding.

The range component of the score is calculated using the Mahalanobis distance for the slant ranges of each reply and the measurement's slant range error.

The altitude component of the score takes into account whether or not each reply is NAR. If both replies in a pairing are NAR, the altitude dimension is disregarded by not accumulating the altitude score and subtracting *merge_alt_threshold* from the total score. If only one reply is NAR, a failed association is triggered by setting the altitude component equal to double the *merge_threshold*. If neither reply is NAR, the altitude component of the score is calculated using the Mahalanobis distance for the slant ranges of each reply and the measurement's altitude error.

If either reply has no bearing, the scoring threshold is subtracted by the *merge_az_threshold*. Otherwise, the bearing component of the score is calculated using the Mahalanobis distance for the ANGLE DIFFERENCE between each reply and the measurement's bearing error.

When the sum of the range, altitude, and bearing scores is below the *merge_threshold* the replies are flagged as a duplicate pairing and their position indices are stored in *merge_idx*. Once all duplicates for a given reply are located, the *merge_duplicate_selector* parameter is used to select which reply should be provided to STM as a reply report.

The algorithm returns a vector of replies that have not been flagged as duplicates.

This algorithm takes as input *replies*. This algorithm returns a subset of *replies* determined to be non-duplicates.

Algorithm 308 MergeModeCReplies

```

1 function MergeModeCReplies(replies::Vector{ModeCReply(p.E-7)})
2   const range_gate::Z = params().surveillance.mode_c.merging.range_gate
3   const r_sigma::Z = params().surveillance.mode_c.merging.range_sigma
4   const alt_sigma::Z = params().surveillance.mode_c.merging.altitude_sigma
5   const chi_sigma::R = params().surveillance.mode_c.merging.chi_sigma
6   const merge_threshold::Z = params().surveillance.mode_c.merging.merging_threshold
7   const merge_alt_threshold::Z = params().surveillance.mode_c.merging.merging_alt_threshold
8   const merge_az_threshold::Z = params().surveillance.mode_c.merging.merging_az_threshold
9   const r_bias::Z = params().surveillance.mode_c.merging.range_bias
10  const merge_duplicate_selector::Z = params().surveillance.mode_c.merging.merging_duplicate_selector
11  CompareRange(a::ModeCReply(p.E-7), b::ModeCReply(p.E-7)) = a.r_slant < b.r_slant
12  replies = sort(replies, lt = CompareRange)
13  is_duplicate::Vector{Bool} = fill(false, length(replies))
14  decoded_altitude::Vector{R} = [ GillhamDecode(p.25)(replies[i].coded_alt) for i in 1:length(replies)]
15  for i in 1:length(replies)
16    if (decoded_altitude[i] == GILLHAM_INVALID_VALUE) && (replies[i].conf == 0)
17      is_duplicate[i] = true
18    elseif (is_duplicate[i] == false)
19      merge_idx = Z[]
20      push!(merge_idx,i)
21      for j in i+1:length(replies)
22        if (is_duplicate[j] == false) && (decoded_altitude[i] != GILLHAM_INVALID_VALUE) && (
23          decoded_altitude[j] != GILLHAM_INVALID_VALUE)
24          threshold = merge_threshold
25          range_score = altitude_score = bearing_score = 0.0
26          range_score = mahal(p.H-1)(replies[i].r_slant, (r_sigma + r_bias)^2, replies[j].
27          r_slant, r_sigma^2)
27          if (replies[i].coded_alt == GILLHAM_NAR_CODE) && (replies[j].coded_alt ==
28            GILLHAM_NAR_CODE)
29            threshold = threshold - merge_alt_threshold
30            elseif (replies[i].coded_alt == GILLHAM_NAR_CODE) || (replies[j].coded_alt ==
31              GILLHAM_NAR_CODE)
32              altitude_score = merge_threshold + merge_threshold
33            else
34              altitude_score = mahal(p.H-1)(decoded_altitude[i], alt_sigma^2, decoded_altitude[j].
35              alt_sigma^2)
36            end
37            if (isnan(replies[i].Chi_rel)) || (isnan(replies[j].Chi_rel))
38              threshold = threshold - merge_az_threshold
39            else
40              bearing_score = mahal(p.H-1)(abs(WrapToPi(p.H-3)(AngleDifference(p.H-2)(replies[i].
41                Chi_rel, replies[j].Chi_rel))), chi_sigma^2, 0, chi_sigma^2)
42            end
43            merge_score = range_score + altitude_score + bearing_score
44            if (merge_score <= threshold)
45              push!(merge_idx,j)
46              is_duplicate[j] = true
47            end
48            selected_reply = floor((length(merge_idx)-1)/merge_duplicate_selector)+1
49            is_duplicate[merge_idx[selected_reply]] = false
50        end
51      end
52    return replies[!is_duplicate]:Vector{ModeCReply(p.E-7)}
53 end

```

Referenced In: ReceiveModeCReplies(p.B-2)

ASSOCIATEMODECTOTARGET (Algorithm 309) takes as input a vector of unique replies and uses

Gillham altitude decoding to decode the *coded_alt* of each reply to a barometric altitude. The algorithm iterates through each MODECTRACKFILE in each TARGET in range order as dictated by the call to SORTBYRANGESQUARED (Algorithm 312) and evaluates whether each input MODECREPLY should be associated to that MODECTRACKFILE. If the decoded altitude of the reply is valid (i.e., doesn't evaluate to *Nan*), then the selected reply flag is initialized to *MODE_C_FAILED_ASSOCIATION*, indicating that no association of reply to track has yet been identified.

For efficiency, a range gate is used to skip association when the difference in range between the track and the reply is large. The same range gate is used to reject certain tracks for association when the difference in range between the reply and the track is large. Similarly, a check ensures that the relative altitude is within an altitude gate before proceeding.

When the sum of the range, altitude, and bearing scores returned by a call to DETERMINEMODECREPLYASSOCIATIONSCORE (Algorithm 310) are below the *merge_threshold* and no other track received a lower eligible score (saved in the *scores* array), then the track is eligible to associate with the reply. The eligible track may be selected as the best associating track if the reply has not received a lower association score from an eligible track. If the reply and track pair have been selected as the best association, the track is stored in *associated_tracks* to be updated with the reply. If the reply had been selected to associate with a previous track, that association is removed from *associated_tracks*. The *association_score* is stored in *scores*, the reply is marked as having been associated in *is_associated*, and the reply index, *i*, is stored in *selected_reply* for future use in the association process. The algorithm returns a vector of tracks to be updated by the associated replies along with a vector of unassociated replies.

This algorithm takes as input *replies*. This algorithm returns *associated_tracks* and a subset of *replies* determined to be unassociated.

Algorithm 309 AssociateModeCtoTarget

```

1 function AssociateModeCtoTarget(replies::Vector{ModeCReply(p.E-7)})
2   const range_gate::Z = params().surveillance.mode_c.association.range_gate
3   const altitude_gate::Z = params().surveillance.mode_c.association.altitude_gate
4   scores = fill(Inf, length(replies))
5   is_associated = fill(false, length(replies))
6   decoded_altitude::Vector{R} = Array(R, length(replies))
7   for i in 1:length(replies)
8     decoded_altitude[i] = GillhamDecode(p.25)(replies[i].coded_alt)
9   end
10  associated_tracks::Vector{Union(ModeCTrackFile(p.E-8), Nothing)} = fill(nothing, length(replies))
11  modec_tracks::Vector{ModeCTrackFile} = Array(ModeCTrackFile(p.E-8), 0)
12  for id in keys(target_db)
13    for modec_track in target_db[id].modec_tracks
14      push!(modec_tracks, modec_track)
15    end
16  end
17  modec_tracks = SortByRangeSquared(p.B-8)(modec_tracks)
18  low_index::Z = 1
19  for modec_track in modec_tracks
20    selected_reply::Z = MODE_C_FAILED_ASSOCIATION
21    for i in low_index:length(replies)
22      if (decoded_altitude[i] != GILLHAM_INVALID_VALUE)
23        if (modec_track.mu_rng[1] - replies[i].r_slant > range_gate)
24          low_index += 1
25          continue
26        end
27        if (replies[i].r_slant - modec_track.mu_rng[1] > range_gate)
28          break
29        end
30        if (modec_track.valid_vert) && (abs(modec_track.mu_vert[1] - decoded_altitude[i]) >
31          altitude_gate)
32          continue
33        end
34        association_score = DetermineModeCReplyAssociationScore(p.B-7)(modec_track, replies[i],
35          decoded_altitude[i])
36        if (association_score != MODE_C_FAILED_ASSOCIATION) && ((associated_tracks[i] == nothing) ||
37          (association_score < scores[i]))
38          if (selected_reply == MODE_C_FAILED_ASSOCIATION) || (association_score < scores[
39            selected_reply])
40            associated_tracks[i] = modec_track
41            scores[i] = association_score
42            is_associated[i] = true
43            if (selected_reply != MODE_C_FAILED_ASSOCIATION)
44              associated_tracks[selected_reply] = nothing
45            end
46            selected_reply = i
47          end
48        end
49      end
50    end
51  end
52  return (associated_tracks::Vector{Union(ModeCTrackFile(p.E-8), Nothing)}, replies[!is_associated]::Vector{
53    ModeCReply(p.E-7)})
54 end

```

Referenced In: ReceiveModeCReplies(p. B-2)

DETERMINEMODECREPLYASSOCIATIONSCORE (Algorithm 310) compares a Mode C Reply to a TRACKSUMMARY, and gives a probabilistic score based on how well it associates range, altitude and bearing

CONVERTTOCARTESIAN and GROUNDRANGE are used to transform the measurement to the coor-

dinate system of the track's TRACKSUMMARY, then EXTRAPOLATETRACK is used to extrapolate the MODECTRACKFILE to the time of the Whisper Shout sequence.

The range component of the score is calculated using the Mahalanobis distance for the ground ranges of the reply and track, and the range error/covariance of the reply and track. The observation noise of the vertical, range, and bearing measurements are accounted for in the Mahalanobis calculation using *altitude_inflation*, *range_inflation*, and *bearing_inflation*.

The bearing component can handle the case when either measurement is bearingless. If both the reply and the track are bearingless, and the range in the Track Summary is less than the *nar_cpa_range*, then the scoring threshold is subtracted by an *association_az_threshold*. When either the reply or track are bearingless a *bearingless_penalty* is added to the overall *penalty*. Otherwise, if both measurements have bearing, then the bearing component of the score is calculated using the Mahalanobis distance for the Cartesian position of the reply and track, and the Cartesian error/covariance's of the reply and track.

Similarly, the altitude component can handle the case of NAR intruders. If both the reply and the track are NAR, and the range in the Track Summary is less than the *nar_cpa_range*, then the scoring threshold is subtracted by an *association_az_threshold* and the bearing component is set to zero. If both are still NAR, the scoring threshold is subtracted by an *association_alt_threshold*. If either the reply or track is a NAR, the altitude component is set to double the *association_threshold*. Otherwise, if both measurements have altitude, then the altitude component of the score is calculated using the Mahalanobis distance for the altitudes of the reply and track, and the altitude error/covariance of the reply and track. The greater of the calculated *altitude_score* and *min_alt_score* is used to ensure that a low *altitude_score* due to level quantized tracks does not excessively enlarge the association window for range and bearing.

A penalty term is added to *association_score* using *coast_penalty_multiplier* and the time the MODECTRACKFILE has coasted to limit the growth of the association windows while coasting. To prevent replies associating with a multipath image track as checked by ISIMAGETRACK (Algorithm 311) rather than the real track, a penalty cost (*image_penalty*) is added to the association score of image tracks, and an *image_threshold* is subtracted from the scoring threshold.

This algorithm takes as input *modec_track*, *reply*, and *decoded_altitude*. This algorithm returns *association_score*.



Algorithm 310 DetermineModeCReplyAssociationScore

```

1 function DetermineModeCReplyAssociationScore(modec_track::ModeCTrackFile(p.E-8), reply::ModeCReply(p.E-7),
2     decoded_altitude::R )
3     const range_inflation::Z      = params().surveillance.mode_c.association.range_inflation
4     const association_threshold::Z = params().surveillance.mode_c.association.association_threshold
5     const nar_cpa_range::Z       = params().surveillance.mode_c.association.nar_cpa_range
6     const bearing_inflation::R   = params().surveillance.mode_c.association.bearing_inflation
7     const association_az_threshold::Z = params().surveillance.mode_c.association.association_az_threshold
8     const association_alt_threshold::Z = params().surveillance.mode_c.association.association_alt_threshold
9     const altitude_inflation::Z   = params().surveillance.mode_c.association.altitude_inflation
10    const min_alt_score::Z       = params().surveillance.mode_c.association.min_alt_score
11    const coast_penalty_multiplier::R = params().surveillance.mode_c.association.coast_penalty_multiplier
12    const image_penalty::Z       = params().surveillance.mode_c.association.image_penalty
13    const image_threshold::Z     = params().surveillance.mode_c.association.image_threshold
14    const bearingless_penalty::Z = params().surveillance.mode_c.association.bearingless_penalty
15    (Chi_abs, z_rel) = ConvertMeasurements(p.19)(decoded_altitude, reply.Chi_rel, HeadingAtToa(p.85)(reply.toa),
16        BaroAltAtToa(p.78)(reply.toa))
17    o = ConvertToCartesian(p.29)(reply.r_slant, Chi_abs, z_rel)
18    range = GroundRange(p.35)(reply.r_slant,z_rel)
19    ExtrapolateTrack(p.C-2)(modec_track, reply.toa)
20    TS::TrackSummary(p.E-16) = modec_track.trk_summary
21    range_score = mahal(p.H-1)(TS.mu_range[1], TS.Sigma_range[1,1], range, range_inflation^2)
22    altitude_score = 0.0
23    bearing_score = 0.0
24    penalty = 0.0
25    threshold = association_threshold
26    if (isnan(reply.Chi_rel)) || (!modec_track.valid_cart)
27        if (TS.mu_range[1] > nar_cpa_range)
28            threshold = threshold - association_az_threshold
29        end
30        penalty += bearingless_penalty
31    else
32        bearing_score = mahal(p.H-1)(WrapToPi(p.H-3)(AngleDifference(p.H-2)(Chi_abs,TS.mu_rng_az[2])),TS.
33            Sigma_rng_az[2,2], 0, bearing_inflation^2)
34    end
35    if (reply.coded_alt == GILLHAM_NAR_CODE) && (!modec_track.valid_vert)
36        if (TS.mu_range[1] < nar_cpa_range)
37            threshold = threshold - association_az_threshold
38        bearing_score = 0.0
39    end
40    threshold = threshold - association_alt_threshold
41    elseif (reply.coded_alt == GILLHAM_NAR_CODE) || (!modec_track.valid_vert)
42        altitude_score = 2*association_threshold
43    else
44        altitude_score = mahal(p.H-1)(TS.mu_vert[1], TS.Sigma_vert[1,1], decoded_altitude, altitude_inflation
45            ^2)
46        altitude_score = max(altitude_score,min_alt_score)
47    end
48    association_score::R = range_score + altitude_score + bearing_score
49    association_score = association_score + ( coast_penalty_multiplier * ( reply.toa - modec_track.toa ) );
50    if (IsImageTrack(p.B-8)(modec_track))
51        penalty += image_penalty
52        threshold -= image_threshold
53    end
54    if (association_score < threshold)
55        association_score += penalty
56    else
57        association_score = MODE_C_FAILED_ASSOCIATION
58    end
59    return association_score::R
60 end

```

Referenced In: AssociateModeCtoTarget(p.B-5)
--

Algorithm 311 IsImageTrack

```

1 function IsImageTrack(trk:TrackFile(p. E-3))
2   if (typeof(trk) == ModeCTrackFile(p. E-8))
3     return trk.is_image
4   else
5     return false
6   end
7 end
```

Referenced In: HasValidModeCTrack(p. 99), DecorrelateTargets(p. C-6), DetermineModeCReplyAssociationScore(p. B-7), AddModeCTrackToReport(p. 114), RemoveStaleModeCTracks(p. 118)

Algorithm 312 SortByRangeSquared

```

1 function SortByRangeSquared(list:Union(Vector{TrackMap(p. E-15)}, Vector{ModeCTrackFile(p. E-8)}))
2   if (typeof(list) == Vector{TrackMap(p. E-15)})
3     CompareRangeSquared(a:TrackMap(p. E-15), b:TrackMap(p. E-15)) = RangeSquared(a.track) <
      RangeSquared(p. B-8)(b.track)
4   else
5     CompareRangeSquared(a:ModeCTrackFile(p. E-8), b:ModeCTrackFile(p. E-8)) = RangeSquared(a) <
      RangeSquared(p. B-8)(b)
6   end
7   return sort(list, lt = CompareRangeSquared)
8 end
```

Referenced In: AssociateModeCtoTarget(p. B-5), PromoteModeCTracks(p. B-10)

Algorithm 313 RangeSquared

```

1 function RangeSquared(track:Union(HypotheticalModeCTrackFile(p. E-7), ModeCTrackFile(p. E-8)))
2   if (typeof(track) == ModeCTrackFile(p. E-8))
3     mu_vert = track.mu_vert[1]
4     mu_rng = track.mu_rng[1]
5     valid = track.valid_vert
6   elseif (typeof(track) == HypotheticalModeCTrackFile(p. E-7))
7     mu_vert = track.altitude
8     mu_rng = track.range
9     valid = !isnan(mu_vert)
10  end
11  own_h:R = BaroAltAtToa(p. 78)(track.toa)
12  if (valid)
13    alt = own_h - mu_vert
14  else
15    alt = 0.0
16  end
17  return (alt2 + mu_rng2):R
18 end
```

Referenced In: SortByRangeSquared(p. B-8)

Any MODECREPLY that does not associate to an established track is used to initialize new MODECTRACKFILES. This process begins in PROMOTEMODECTRACKS (Algorithm 314), where replies are correlated together to form HYPOTHETICALMODECTRACKFILES and used to mark the HYPOTHETICALMODECTRACKFILES for promotion into established MODECTRACKFILES. The HYPO-

HYPOTHETICALMODECTRACKFILES are stored in a global DATABASE, *hyp_track_db*. Gillham altitude decoding is used to decode each reply's coded altitude (*coded_alt*) into a barometric altitude. The algorithm then iterates through each existing HYPOTHETICALMODECTRACKFILE in range sorted order through a call to `SORTBYRANGESQUARED` and attempts to promote the track using each reply contained in *unassociated_replies*. The hypothetical track's slant range is extrapolated to the time of the reply.

For efficiency, a *range_gate* is used to skip promotion when the difference in range between the prediction and the reply is large. The same *range_gate* is used to reject certain hypothetical tracks for promotion when the difference in range between the reply and the hypothetical track is large. Similarly, a check ensures that the absolute difference in range between the prediction and the reply is smaller than the *range_gate*, otherwise the promotion score will be set to double the promotion threshold in order to force promotion to fail.

If the gating checks have passed, the reply is then compared to the HYPOTHETICALMODECTRACKFILE, and given a probabilistic score based on how well it associates range, altitude and bearing. The range component of the score is calculated using the Mahalanobis distance for the slant range of the reply and the predicted track slant range using the range standard deviation (*r_sigma*) of the reply and predicted track. The altitude component of the score can handle the case for NAR intruders. If both the reply and the hypothetical track are NAR, then the scoring threshold is subtracted by the *promote_alt_threshold*. If either the reply or hypothetical track is a NAR, then the altitude component of the score is set to double the *promote_threshold*. Otherwise, if both measurements have altitude, then the altitude component of the score is calculated using the Mahalanobis distance of the reply and the predicted altitude of the track, using altitude error derived from standard deviation *alt_sigma* for the reply and track. The bearing component of the score can handle the case when either the reply or the hypothetical track have invalid bearing, i.e. are bearingless. If either have invalid bearing, then the scoring threshold will be subtracted by the *promote_az_threshold*. Otherwise, the bearing component of the score is calculated using the Mahalanobis distance for the ANGLEDIFFERENCE between the bearing of the reply and predicted track using the bearing uncertainty (derived from standard deviation *chi_sigma*) of the reply and track. When the sum of the range, altitude, and bearing scores is below the *promote_threshold*, the four replies (three from the hypothetical track and the one establishing reply) are used later to initialize a MODECTRACKFILE through a call to `ESTABLISHMODECTRACK`.

This algorithm takes as input *unassociated_replies*. This algorithm returns *promoted_tracks*.

Algorithm 314 PromoteModeCTracks

```

1 function PromoteModeCTracks(unassociated_replies::Vector{ModeCReply(p.E-7)})  

2     const range_gate::Z = params().surveillance.mode_c.promote.range_gate  

3     const r_sigma::Z = params().surveillance.mode_c.promote.range_sigma  

4     const alt_sigma::R = params().surveillance.mode_c.promote.altitude_sigma  

5     const chi_sigma::R = params().surveillance.mode_c.promote.chi_sigma  

6     const promote_threshold::R = params().surveillance.mode_c.promote.promote_threshold  

7     const promote_alt_threshold::R = params().surveillance.mode_c.promote.promote_alt_threshold  

8     const promote_az_threshold::R = params().surveillance.mode_c.promote.promote_az_threshold  

9     modecIntervals.In1 = unassociated_replies  

10    promoted_tracks = Vector{ModeCReply(p.E-7)}[]  

11    scores = fill(Inf, length(unassociated_replies))  

12    low_index::Z = 1  

13    hyp_track_list::Vector{TrackMap} = Array(TrackMap(p.E-15), 0)  

14    for id in keys(hyp_track_db)  

15        push!(hyp_track_list, TrackMap(p.E-15)(hyp_track_db[id], id))  

16    end  

17    hyp_track_list = SortByRangeSquared(p.B-8)(hyp_track_list)  

18    reply_altitude::Vector{R} = Array(R, length(unassociated_replies))  

19    for i in 1:length(unassociated_replies)  

20        reply_altitude[i] = GillhamDecode(p.25)(unassociated_replies[i].coded_alt)  

21    end  

22    for hyp_track_map in hyp_track_list  

23        (hyp_track, id) = (hyp_track_map.track, hyp_track_map.db_id)  

24        predicted_range = [hyp_track.rangerate * (unassociated_replies[k].toa - hyp_track.toa) + hyp_track.  

                           range for k in 1:length(unassociated_replies) ]  

25        for j in low_index:length(unassociated_replies)  

26            if (reply_altitude[j] != GILLHAM_INVALID_VALUE)  

27                if (predicted_range[j] - unassociated_replies[j].r_slant > 2*range_gate)  

28                    low_index += 1  

29                    continue  

30                end  

31                if (unassociated_replies[j].r_slant - predicted_range[j] > 2*range_gate)  

32                    break  

33                end  

34                threshold = promote_threshold  

35                range_score = altitude_score = bearing_score = 0.0  

36                if (abs(predicted_range[j] - unassociated_replies[j].r_slant) > range_gate)  

37                    promote_score = 2 * promote_threshold  

38                else  

39                    predicted_range_sigma = (2*(unassociated_replies[j].toa-hyp_track.toa)+1)*r_sigma*  

                           r_sigma  

40                    range_score = mahal(p.H-1)(predicted_range[j], predicted_range_sigma, unassociated_replies  

                           [j].r_slant, r_sigma^2)  

41                    if (unassociated_replies[j].coded_alt == GILLHAM_NAR_CODE) && (hyp_track.replies[1].  

                           coded_alt == GILLHAM_NAR_CODE)  

42                        threshold = threshold - promote_alt_threshold  

43                    elseif (unassociated_replies[j].coded_alt == GILLHAM_NAR_CODE) || (hyp_track.replies[1].  

                           coded_alt == GILLHAM_NAR_CODE)  

44                        altitude_score = promote_threshold+promote_threshold  

45                    else  

46                        altitude_score = mahal(p.H-1)(hyp_track.altitude, alt_sigma^2, reply_altitude[j],  

                           alt_sigma^2)  

47                    end  

48                    if (isnan(unassociated_replies[j].Chi_rel)) || (isnan(hyp_track.Chi_rel))  

49                        threshold = threshold - promote_az_threshold  

50                    else  

51                        bearing_score = mahal(p.H-1)(0, chi_sigma^2, abs(WrapToPi(p.H-3)(AngleDifference(p.H-2)(  

                           hyp_track.Chi_rel,unassociated_replies[j].Chi_rel))), chi_sigma^2)  

52                    end  

53                    promote_score = range_score + altitude_score + bearing_score  

54                end  

55                if (promote_score <= threshold) && (promote_score < scores[j])  

56                    promoted_replies = vcat(deepcopy(unassociated_replies[j]), hyp_track.replies)  

57                    push!(promoted_tracks, promoted_replies)  

58                    scores[j] = promote_score  

59                    break  

60                end  

61            end  

62        end  

63    end  

64    return promoted_tracks::Vector{Vector{ModeCReply(p.E-7)}}  


```

65 end

Referenced In: ReceiveModeCReplies(p. B-2)
--

ESTABLISHMODECTRACK (Algorithm 315) takes as input a collection of tracks to be initialized (where an array of Mode C Replies indicates a track). The tracks, which have been promoted, are to be initialized into a new MODECTRACKFILE. The algorithm loops over each set of tracks and checks the estimated range of the hypothetical track and terminates if the estimate is invalid (i.e., equal to NaN). Another check ensures that the altitude is valid. For valid estimates, a new MODECTRACKFILE is created and the latest reply is used to initialize. The new track file is initialized as being on the ground (indicated by *trk.on_ground*) and then its on-ground state is evaluated (EVALUATEONGROUNDMODEC). Next, the remaining replies are used to update the track in time order. After each update, the on-ground state is evaluated (EVALUATEONGROUNDMODEC). Finally, the new Track File is added to the database through ADDMODECTRACKTODB (Algorithm 13).

This algorithm takes as input *promoted_tracks*. This algorithm calls algorithm ADDMODECTRACKTODB.

Algorithm 315 EstablishModeCTrack

```

1 function EstablishModeCTrack(promoted_tracks::Vector{Vector{ModeCReply(p. E-7)}})
2   for replies in promoted_tracks
3     N::Z = length(replies)
4     if (isnan(replies[N].r_slant))
5       continue
6     end
7     if (replies[N].coded_alt != GILLHAM_NAR_CODE)
8       z_baro::R = GillhamDecode(p.25)(replies[N].coded_alt)
9       if (z_baro == GILLHAM_INVALID_VALUE)
10         continue
11       end
12     end
13     trk = ModeCTrackFile(p. E-8)()
14     InitializeModeCTrack(p. 23)(trk, replies[N])
15     trk.on_ground = true
16     EvaluateOnGroundModeC(p. B-17)(trk, replies[N].toa, replies[N].coded_alt)
17     for i in reverse(1:N-1)
18       UpdateModeCTrack(p. 24)(trk, replies[i])
19       EvaluateOnGroundModeC(p. B-17)(trk, replies[i].toa, replies[i].coded_alt)
20     end
21     AddModeCTrackToDB(p. 23)(trk)
22   end
23 end

```

Referenced In: ReceiveModeCReplies(p. B-2)
--

Unassociated Mode C replies received in the three previous intervals are used to form new HYPOTHETICALMODECTRACKFILES using FORMHYPOTHETICALTRACKS (Algorithm 316). Hypothetical tracks form when a reply from each of the three intervals correlate.

The algorithm first begins iteration through the most recent replies to identify and initialize hypothetical tracks from each of the replies in the second and third most recent intervals. The replies for each interval are stored, sorted by range, in the members of the global variable *modecIntervals*,

which is of type MODECINTERVALS. Replies from the most recent interval are stored in the *In1* member; from the oldest interval, the *In3* member.

The algorithm then iterates through the second interval of Mode C Replies using a range gate to prevent unnecessary evaluations, and an *index* that's used as a starting point for each subsequent iteration. If the difference in slant-range from the static reply and the interval in questions is larger than the *range_gate*, then the starting *index* is set to *j* and the loop continues. If the difference in slant-range from the interval and the static reply is larger than the *range_gate*, then the algorithm breaks the loop and returns the current *index*.

Otherwise, the algorithm repeats the process for the third interval of Mode C Replies. Again, the *range_gate* is checked for efficiency. If all tests pass, then the selected replies from all three intervals are used to call HYPOTHETICALTRACKTEST.

Algorithm 316 FormHypotheticalTracks

```

1 function FormHypotheticalTracks()
2   const range_gate::Z = params().surveillance.mode_c.initialize.range_gate
3   low_index::Z = 1
4   for i in 1:length(modecIntervals.In1)
5     static_reply::ModeCReply(p.E-7) = modecIntervals.In1[i]
6     inner_index::Z = 1
7     for j in low_index:length(modecIntervals.In2)
8       if (static_reply.r_slant - modecIntervals.In2[j].r_slant > range_gate)
9         low_index = j
10        continue
11      end
12      if (modecIntervals.In2[j].r_slant - static_reply.r_slant > range_gate)
13        break
14      end
15      for k in inner_index:length(modecIntervals.In3)
16        if (modecIntervals.In2[j].r_slant - modecIntervals.In3[k].r_slant > range_gate)
17          inner_index = k
18          continue
19        end
20        if (modecIntervals.In3[k].r_slant - modecIntervals.In2[j].r_slant > range_gate)
21          break
22        end
23        HypotheticalTrackTest(p.B-14)(static_reply, modecIntervals.In2[j], modecIntervals.In3[k])
24      end
25    end
26  end
27 end
```

Referenced In: ReceiveModeCReplies(p.B-2)
--

HYPOTHETICALTRACKTEST (Algorithm 317) takes as input three Mode C replies received in previous surveillance intervals, and attempts to correlate the replies to form HYPOTHETICALMODECTRACKFILES. First, the *rangerate* and *predicted_range* are estimated using the slant range of the most recent and oldest replies. If either estimate are outside of their respective gating window (*rangerate_gate* and *range_gate*), then the algorithm terminates. Otherwise, the algorithm assigns a probabilistic score based on how well the replies associate to one another in range, altitude and bearing. The range component of the score is calculated using the Mahalanobis distance for the slant range of the reply and the estimated track slant range using the slant range uncertainty (given by standard deviation *r_sigma*) of the reply and predicted range. The algorithm can handle replies from NAR aircraft. If all three replies are NAR, then the scoring threshold is subtracted by the *init_-*

alt_threshold. Otherwise, if any of the replies are NAR, then the altitude component of the score is set to double the *init_threshold*. Otherwise, if all replies are reporting altitude, then the number of disagreements in the coded altitudes of the three replies is summed for the DAB bits and C bits using ALTCODEAGREEMENT, yielding *sumDAB* and *sumC*. These values are weighted according to *weight_DAB* and *weight_C* to provide the altitude component of the score. This algorithm can also handle replies from bearingless intruders. If all of the replies are bearingless, then the scoring threshold is subtracted by the *init_az_threshold*. Otherwise, the bearing component of the score is calculated by the Mahalanobis distance of the bearing differences between the three replies. This is calculated using the average ANGLEDIFFERENCE of each pair of replies and the bearing uncertainty (given by standard deviation *chi_sigma*). When the sum of the range, altitude, and bearing scores is below the *init_threshold*, the three replies are used to initialize a new HYPOTHETICALMODECTRACKFILE. The estimated altitude of the hypothetical track must be calculated through a call to ALTCODEESTIMATE. The new hypothetical track can then be added to the hypothetical track database through a call to ADDTODB.

This algorithm takes as input *replyA*, *replyB*, and *replyC*. This algorithm updates *trk* and calls algorithm ADDTODB.

Algorithm 317 HypotheticalTrackTest

```

1 function HypotheticalTrackTest(replyA::ModeCReply, replyB::ModeCReply, replyC::ModeCReply(p.E-7))
2   const rangerate_gate::Z = params().surveillance.mode_c.initialize.rangerate_gate
3   const range_gate::Z      = params().surveillance.mode_c.initialize.range_gate
4   const DAB::UInt16        = params().surveillance.mode_c.initialize.DABbits
5   const C::UInt16          = params().surveillance.mode_c.initialize.Cbits
6   const weight_DAB::R     = params().surveillance.mode_c.initialize.weight_DAB
7   const weight_C::R       = params().surveillance.mode_c.initialize.weight_Cbit
8   const init_threshold::R = params().surveillance.mode_c.initialize.initialize_threshold
9   const init_alt_threshold::R = params().surveillance.mode_c.initialize.initialize_alt_threshold
10  const init_az_threshold::R = params().surveillance.mode_c.initialize.initialize_az_threshold
11  const r_sigma::R         = params().surveillance.mode_c.promote.range_sigma
12  const chi_sigma::R       = params().surveillance.mode_c.promote.chi_sigma
13  rangerate::R = (replyA.r_slant - replyC.r_slant)/(replyA.toa - replyC.toa)
14  if (abs(rangerate) > rangerate_gate)
15    return
16  end
17  predicted_range::R = replyA.r_slant - rangerate * (replyA.toa - replyB.toa)
18  if (predicted_range - replyB.r_slant > range_gate)
19    return
20  end
21  range_score = altitude_score = bearing_score = 0.0
22  threshold = init_threshold
23  range_score      = mahal(p.H-1)(predicted_range, r_sigma^2, replyB.r_slant, r_sigma^2)
24  if (replyA.coded_alt == GILLHAM_NAR_CODE) && (replyB.coded_alt == GILLHAM_NAR_CODE) && (replyC.coded_alt
25    == GILLHAM_NAR_CODE)
26    threshold = threshold - init_alt_threshold
27  elseif (replyA.coded_alt == GILLHAM_NAR_CODE) || (replyB.coded_alt == GILLHAM_NAR_CODE) || (replyC.
28    coded_alt == GILLHAM_NAR_CODE)
29    altitude_score = init_threshold+init_threshold
30  else
31    (sumDAB, sumC) = AltCodeAgreement(p.B-15)(replyA.coded_alt, replyB.coded_alt, replyC.coded_alt, DAB, C)
32    altitude_score = weight_DAB*sumDAB + weight_C*sumC
33  end
34  if (isnan(replyA.Chi_rel)) && (isnan(replyB.Chi_rel)) && (isnan(replyC.Chi_rel))
35    threshold = threshold - init_az_threshold
36  else
37    delta_chi = (abs(WrapToPi(p.H-3)(AngleDifference(replyA.Chi_rel, replyB.Chi_rel))) + abs(
38      WrapToPi(p.H-3)(AngleDifference(replyB.Chi_rel, replyC.Chi_rel))) + abs(WrapToPi(p.H-3)(
39        AngleDifference(p.H-2)(replyA.Chi_rel, replyC.Chi_rel))))/3
40    bearing_score = mahal(p.H-1)(0, chi_sigma^2, delta_chi, chi_sigma^2)
41  end
42  initialize_score = range_score + altitude_score + bearing_score
43  if (initialize_score < threshold)
44    estimated_altitude = AltCodeEstimate(p.B-15)(replyA.coded_alt, replyB.coded_alt, replyC.coded_alt,
45      replyA.conf, replyB.conf)
46    if (estimated_altitude != GILLHAM_INVALID_VALUE)
47      trk      = HypotheticalModeCTrackFile(p.E-7)()
48      trk.replies = [deepcopy(replyA), deepcopy(replyB), deepcopy(replyC)]
49      trk.rangerate = (replyA.r_slant - replyC.r_slant)/(replyA.toa - replyC.toa)
50      trk.altitude = GillhamDecode(p.25)(estimated_altitude)
51      trk.range   = replyA.r_slant
52      trk.Chi_rel = replyA.Chi_rel
53      trk.toa     = replyA.toa
54      trk.age     = 0.0
55      trk.replies[1].coded_alt = trk.replies[2].coded_alt = trk.replies[3].coded_alt =
56        estimated_altitude
57      AddToDB(p.20)(hyp_track_db, trk)
58    end
59  end
60 end
```

Referenced In: FormHypotheticalTracks(^{p.B-12})

ALTCODEAGREEMENT (Algorithm 318) takes as input the altitude codes of three Mode C replies and the DAB and C bit masks. Bit positions that do not agree across all three replies are marked and stored into *bitAgree*. The sum of bits which do not agree for each bit mask is accumulated into *sumDAB* and *sumC* and returned to the calling function.

This algorithm takes as input *codeA*, *codeB*, *codeC*, *DAB*, and *C*. This algorithm returns *sumDAB* and *sumC*.

Algorithm 318 AltCodeAgreement

```

1 function AltCodeAgreement(codeA::Z, codeB::Z, codeC::Z, DAB::UInt16, C::UInt16)
2   bitAgree = (~codeA $ codeB) & (~codeB $ codeC)
3   DABbits = DAB & bitAgree
4   Cbits = C & bitAgree
5   sumDAB = 8
6   sumC = 3
7   while (DABbits > 0)
8     sumDAB -= 1
9     DABbits &= (DABbits - 1)
10  end
11  while (Cbits > 0)
12    sumC -= 1
13    Cbits &= (Cbits - 1)
14  end
15  return (sumDAB, sumC)
16 end
```

Referenced In: HypotheticalTrackTest(p. B-14)
--

ALTCODEESTIMATE (Algorithm 319) takes as input the altitude codes and the confidence fields of three correlating replies, which resolves conflicting or possible garbled bits to estimate an altitude. If all three codes match, no alterations are necessary. Otherwise, the algorithm begins with the oldest code, and marks bits received as a zero or with high confidence as *trusted_bits* (i.e., they have not been garbled), which are then stored in *code*. The evaluation is repeated for each subsequent altitude code, and the trusted *code* field is updated with new un-garbled decoded bits. If the trusted *code* can be decoded into a valid altitude by the Gillham altitude decoding algorithm, it is returned to the calling function. Otherwise, *GILLHAM_INVALID_VALUE* is returned.

This algorithm takes as input *code3*, *code2*, *code1*, *conf3*, and *conf2*. This algorithm returns *code*.

Algorithm 319 AltCodeEstimate

```

1 function AltCodeEstimate(code3::Z, code2::Z, code1::Z, conf3::Z, conf2::Z)
2   code::Z = code1
3   if (code1 != code2) || (code2 != code3)
4     trusted_bit = (~code2)|(~conf2)
5     code = (code & (~trusted_bit)) | (code2 & trusted_bit)
6     trusted_bit = (~code3)|(~conf3)
7     code = (code & (~trusted_bit)) | (code3 & trusted_bit)
8   end
9   if (GillhamDecode(p.25)(code) == GILLHAM_INVALID_VALUE)
10    code = convert( Z, GILLHAM_INVALID_VALUE )
11  end
12  return code::Z
13 end
```

Referenced In: HypotheticalTrackTest(p. B-14)
--

INTERVALMAINTENANCE (Algorithm 320) performs the housekeeping tasks for the Mode C initialization process at the end of each processing interval. HYPOTHETICALMODECTRACKFILES are removed from memory (stored in *hyp_track_db*) if their *age* exceeds the *promote_limit*. The oldest stored interval of Mode C replies (*In3*) is deleted and all stored intervals are shuffled to make room for a future interval of replies.

Algorithm 320 IntervalMaintenance

```
1 function IntervalMaintenance()
2     const promote_limit::Z = params().surveillance.mode_c.promote.promote_limit
3     modecIntervals.In3 = modecIntervals.In2
4     modecIntervals.In2 = modecIntervals.In1
5     modecIntervals.In1 = Array(ModeCReply(p.E-7), 0)
6     for id in keys(hyp_track_db)
7         if (hyp_track_db[id].age >= promote_limit)
8             delete!(hyp_track_db, id)
9         else
10            hyp_track_db[id].age += 1
11        end
12    end
13 end
```

Referenced In: ReceiveModeCReplies(p.B-2)

EVALUATEONGROUNDDEC (Algorithm 321) evaluates whether or not a given Mode C Track File is on the ground or in the air. First, the reply's coded altitude is decoded with GILLHAMDECODE. If this value indicates NAR, the track is assumed to be in air. Additionally, if the decoded altitude is invalid, the rest of the algorithm is skipped over. The existing Vertical Track is then extrapolated to the time of the observation through PREDICTVERTICALTRACKER. The measurement and the mean of the predicted track are then stored into *trk.vert_history* using UPDATEHISTORY.

The ability to determine the ground level is only credible if own radio altitude is under *max_rad_alt* feet. This is allowed to have +/-50 feet of hysteresis depending on own vertical rate. The next segment of code evaluates this hysteresis (set in *hysteresis*). Next, the algorithm evaluates whether own radio altitude is valid, whether the own radio altitude is under the credibility threshold, and whether the intruder vertical track is valid. Without these conditions, the intruder's height above the ground cannot be reliably calculated and therefore *trk.on_ground* is automatically set to FALSE.

If the above conditions are satisfied, the ground level is calculated by subtracting the most recent own radio altitude (*own.radalt*) from the own barometric altitude track at the time of the most recent reply (output by BAROALTATTOA). The intruder's current height above the ground is calculated by subtracting the ground level from the predicted intruder altitude.

If *trk.on_ground* is currently TRUE, a transition from the ON-GROUND state to the IN-AIR state is evaluated in a call to EVALUATEGROUNDTOAIRTRANSITIONDEC (Algorithm 322). If *trk.on_ground* is instead currently FALSE, then the transition from the IN-AIR state to the ON-GROUND state is evaluated in a call to EVALUATEAIRTOGROUNDTRANSITIONDEC (Algorithm 323).

This algorithm takes as input *trk*, *reply_toa*, and *reply_coded_alt*. This algorithm updates *trk.on_ground*.

Algorithm 321 EvaluateOnGroundModeC

```

1 function EvaluateOnGroundModeC(trk::ModeCTrackFile(p. E-8), reply_toa::R, reply_coded_alt::Z)
2   const vert_history_window::Z = params().surveillance.mode_c.on_ground.vert_history_window
3   const max_rad_alt::R = params().surveillance.mode_c.on_ground.max_rad_alt
4   const rad_alt_hysteresis::R = params().surveillance.mode_c.on_ground.rad_alt_hysteresis
5   z_baro::R = GillhamDecode(p. 25)(reply_coded_alt)
6   if (isnan(z_baro))
7     trk.on_ground = false
8   elseif (z_baro != GILLHAM_INVALID_VALUE)
9     dt = reply_toa - trk.toa_vert
10    (mu_pred_vert::Vector{R}, nothing) = PredictVerticalTracker(p. 45)(trk.mu_vert, trk.Sigma_vert, dt)
11    UpdateHistory(p. 77)(trk.vert_history.meas, z_baro, reply_toa, vert_history_window)
12    UpdateHistory(p. 77)(trk.vert_history.track, trk.mu_vert[1], reply_toa, vert_history_window)
13    hysteresis::R = 0.0
14    if (own.mu_h[2] > 0)
15      hysteresis = rad_alt_hysteresis
16    elseif (own.mu_h[2] < 0)
17      hysteresis = -rad_alt_hysteresis
18    end
19    if ((!isnan(own.radalt)) && (own.radalt < (max_rad_alt + hysteresis)) && (trk.valid_vert))
20      ground_level::R = BaroAltAtToa(p. 78)(reply_toa) - own.radalt
21      height_above_ground::R = mu_pred_vert[1] - ground_level
22      if (trk.on_ground) && (EvaluateGroundToAirTransitionModeC(p. B-18)(trk, z_baro, height_above_ground))
23        trk.on_ground = false
24      elseif (!trk.on_ground) && (EvaluateAirToGroundTransitionModeC(p. B-19)(trk, ground_level,
25        height_above_ground, reply_toa))
26        trk.on_ground = true
27      end
28    else
29      trk.on_ground = false
30    end
31 end
31 end

```

Referenced In: ReceiveModeCReplies(p. B-2), EstablishModeCTrack(p. B-11)

EVALUATEGROUNDTOAIRTRANSITIONMODEC (Algorithm 322) is an implementation of section 2.2.4.6.2.2.3.1 from the ACAS Xa/Xo MOPS Volume I [16]. The first condition from the MOPS evaluated is that "there has been at least one positive altitude transition that has persisted for *min_num_after_pos_transition* consecutive reports" (set by *had_pos_transition*, which is initialized as FALSE). In order for this to be TRUE, there needs to be at least *min_num_after_pos_transition* reports in the history. If that is true, the algorithm then finds the point immediately after the last negative transition (put into *idx_neg_last*). If there is none, the full history is used (*idx_neg_last* is set to 1). If the negative transition is within the last *min_num_after_pos_transition* reports, the MOPS condition is FALSE. Otherwise, the algorithm finds if there was any altitude in the observation history (*h_hist*) less than the minimum of the last *min_num_after_pos_transition* observations (*h_first_of_end*) since the last negative transition. If so, this indicates that a positive transition happened and therefore *had_pos_transition* is set to TRUE. Next, the intruder observation's height above the minimum (*height_above_min*) is calculated by subtracting the minimum of the observation history from the current intruder observation. If either the tracked height above the ground is greater than or equal to *max_alt_on_ground*, or if the intruder's tracked barometric altitude is *alt_evaluate_in_air* feet above the ground level estimate, the intruder's reported barometric altitude is greater than *max_height_above_min_on_ground* feet above its lowest reported barometric altitude, and *had_pos_transition* is TRUE, then an indication that intruder has transitioned to the IN-AIR state is returned by the algorithm.

This algorithm takes as input *trk*, *z_baro*, and *height_above_ground*. This algorithm returns *ground_to_air_transition*.

Algorithm 322 EvaluateGroundToAirTransitionModeC

```

1 function EvaluateGroundToAirTransitionModeC(trk::ModeCTrackFile(p.E-8), z_baro::R, height_above_ground::R)
2   const min_num_after_pos_transition::Z = params().surveillance.mode_c.on_ground.min_num_after_pos-
    transition
3   const max_alt_on_ground::R = params().surveillance.mode_c.on_ground.max_alt_on_ground
4   const alt_evaluate_in_air::R = params().surveillance.mode_c.on_ground.alt_evaluate_in_air
5   const max_height_above_min_on_ground::R = params().surveillance.mode_c.on_ground.max_height_above_min-
    on_ground
6   ground_to_air_transition::Bool = false
7   h_hist = trk.vert_history.meas.value
8   had_pos_transition::Bool = false
9   if (length(h_hist) > min_num_after_pos_transition)
10    idx_neg_last = 1
11    for i in 2:length(h_hist)
12      if ((h_hist[i] - h_hist[i-1]) < 0)
13        idx_neg_last = i
14    end
15  end
16  if (idx_neg_last <= (length(h_hist) - min_num_after_pos_transition))
17    h_first_of_end = h_hist[length(h_hist) - min_num_after_pos_transition + 1]
18    for i in idx_neg_last:length(h_hist)
19      if (h_hist[i] < h_first_of_end)
20        had_pos_transition = true
21      end
22    end
23  end
24 end
25 height_above_min::R = z_baro - minimum(h_hist)
26 if ((height_above_ground >= max_alt_on_ground) || ((height_above_ground >= alt_evaluate_in_air) &&
    (height_above_min > max_height_above_min_on_ground) && had_pos_transition))
27   ground_to_air_transition = true
28 end
29 return ground_to_air_transition::Bool
30 end

```

Referenced In: EvaluateOnGroundModeC(p.B-17)

EVALUATEAIRTOGROUNDTRANSITIONMODEC (Algorithm 323) is an implementation of section 2.2.4.6.2.2.3.1 from the ACAS Xa/Xo MOPS Volume I [16]. In order to evaluate this transition, there needs to be a full *vert_history_window* seconds of history (set in *is_trk_history_full*). It is possible to have a *max_coasts* at the front of the history and still have a track that has existed longer than *vert_history_window* seconds. The MOPS conditional that is evaluated next is "the intruder's barometric altitude has been less than *alt_evaluate_on_ground* feet above the ground level estimate for *vert_history_window* seconds". This is set in *is_track_above_ground*, which is initially set to FALSE. The intruder tracked vertical history is looped over and checked if it is greater than the sum of the ground level (*ground_level*) and *alt_evaluate_on_ground*. If it is at any point, then *is_track_above_ground* is set to TRUE. The next MOPS condition evaluated is whether "reported barometric altitude has not increased by more than *max_height_above_min_on_ground* feet from its lowest reported barometric altitude for the same *vert_history_window*-second period". This is set in *is_meas_static*. If the tracked height above the ground is less than *min_alt_in_air*, the algorithm return value is set to indicate the ON-GROUND state. Otherwise, if there is enough history, the vertical track is not too high above the ground for the history, and the observations have not gone too high above their minimum, then again the algorithm return value is set to indicate the ON-GROUND

state.

This algorithm takes as input *trk*, *ground_level*, *height_above_ground*, and *reply_toa*. This algorithm returns *air_to_ground_transition*.

Algorithm 323 EvaluateAirToGroundTransitionModeC

```
1 function EvaluateAirToGroundTransitionModeC(trk::ModeCTrackFile(p.E-8), ground_level::R, height_above_ground::R,  
    reply_toa::R)  
2     const max_coasts::Z = params().surveillance.max_normal_coasts  
3     const vert_history_window::Z = params().surveillance.mode_c.on_ground.vert_history_window  
4     const alt_evaluate_on_ground::R = params().surveillance.mode_c.on_ground.alt_evaluate_on_ground  
5     const max_height_above_min_on_ground::R = params().surveillance.mode_c.on_ground.max_height_above_min_-  
        on_ground  
6     const min_alt_in_air::R = params().surveillance.mode_c.on_ground.min_alt_in_air  
7     air_to_ground_transition::Bool = false  
8     h_hist = trk.vert_history.meas.value  
9     is_trk_history_full = (reply_toa - minimum(trk.vert_history.track.time)) >= (vert_history_window -  
        max_coasts)  
10    is_track_above_ground = false  
11    for i = 1:length(h_hist)  
12        if (h_hist[i] > (ground_level + alt_evaluate_on_ground))  
13            is_track_above_ground = true  
14        end  
15    end  
16    is_meas_static = maximum(h_hist) - minimum(h_hist) <= max_height_above_min_on_ground  
17    if ((height_above_ground < min_alt_in_air) || (is_trk_history_full && (!is_track_above_ground) &&  
        is_meas_static))  
18        air_to_ground_transition = true  
19    end  
20    return air_to_ground_transition::Bool  
21 end
```

Referenced In: EvaluateOnGroundModeC(p.B-17)

This page intentionally left blank.

Appendix C Correlation Processing Implementation

C.1 Track Extrapolation

PREDICTTRACKSUMMARY (Algorithm 324) iterates through every TARGET in memory and obtains each MODECTRACKFILE, MODESTRACKFILE, and ADSBTRACKFILE contained within the target through a call to GETTRACKSTOEXTRAPOLATE (Algorithm 325). The TRACKSUMMARY for each track is then extrapolated to time T using EXTRAPOLATETRACK (Algorithm 326), which calls either EXTRAPOLATEACTIVETRACK or EXTRAPOLATEADSBTRACK depending on the track type.

This algorithm takes as input T . This algorithm calls EXTRAPOLATETRACK.

Algorithm 324 PredictTrackSummary

```

1 function PredictTrackSummary( $T: \mathbb{R}$ )
2   for id in keys(target_db)
3     target = target_db[id]
4     tracks = GetTracksToExtrapolate(p.C-1)(target)
5     for trk in tracks
6       ExtrapolateTrack(p.C-2)(trk, T)
7     end
8   end
9 end

```

Referenced In: CorrelationProcessing(p.95)

Algorithm 325 GetTracksToExtrapolate

```

1 function GetTracksToExtrapolate(target::Target(p.E-14))
2   tracks_to_extrap = Array(TrackFile(p.E-3), 0)
3   for modec_track in target.modec_tracks
4     push!(tracks_to_extrap, modec_track)
5   end
6   if (TrackExists(p.18)(target.modes_track))
7     push!(tracks_to_extrap, target.modes_track)
8   end
9   if (TrackExists(p.18)(target.adsb_track))
10    push!(tracks_to_extrap, target.adsb_track)
11   end
12   if (TrackExists(p.18)(target.adsr_track))
13     push!(tracks_to_extrap, target.adsr_track)
14   end
15   return tracks_to_extrap::Vector{TrackFile(p.E-3)}
16 end

```

Referenced In: PredictTrackSummary(p.C-1)
--

Algorithm 326 ExtrapolateTrack

```

1 function ExtrapolateTrack(trk::TrackFile(p.E-3), T::R)
2   track_type = typeof(trk)
3   if (track_type <: Union(ModeCTrackFile(p.E-8), ModeSTrackFile(p.E-9)))
4     ExtrapolateActiveTrack(p.C-2)(trk, T)
5   elseif (track_type == ADSBTrackFile(p.E-4))
6     ExtrapolateADSBTrack(p.C-3)(trk, T)
7   end
8   trk.trk_summary.toa = T
9 end
```

Referenced In: DetermineModeCReplyAssociationScore(p.B-7), PredictTrackSummary(p.C-1)

To update the TRACKSUMMARY of the active track to the current time, EXTRAPOLATEACTIVETRACK (Algorithm 327) will retrieve the means and covariances of the vertical, range, and Cartesian state estimates. If *dt* is less than *min_extrap_toa_step*, the state estimates must first be predicted forward using PREDICTVERTICALTRACKER, PREDICTRANGETRACKER, and PREDICTCARTESSIANTRACKER. The updated Cartesian estimate (*mu_xy* and *Sigma_xy*) is transformed to a polar estimate using LINEARTRANSFORM.

This algorithm takes as input *trk* and *T*. This algorithm updates the input *trk*.

Algorithm 327 ExtrapolateActiveTrack

```

1 function ExtrapolateActiveTrack(trk::Union(ModeCTrackFile(p.E-8), ModeSTrackFile(p.E-9)), T::R)
2   const I_xy::Vector{Z} = [1,2]
3   dt_vert = T - trk.toa_vert
4   dt_rng = T - trk.toa_rng
5   dt_cart = T - trk.toa_cart
6   (mu_vert, Sigma_vert) = (copy(trk.mu_vert), copy(trk.Sigma_vert))
7   (mu_rng, Sigma_rng) = (copy(trk.mu_rng), copy(trk.Sigma_rng))
8   (mu_cart, Sigma_cart) = (copy(trk.mu_cart), copy(trk.Sigma_cart))
9   TS::TrackSummary(p.E-16) = trk.trk_summary
10  (TS.mu_vert, TS.Sigma_vert) = PredictVerticalTracker(p.45)(mu_vert, Sigma_vert, dt_vert)
11  (mu_r, Sigma_r) = PredictRangeTracker(p.39)(mu_rng, Sigma_rng, dt_rng)
12  (mu_xy, Sigma_xy) = PredictCartesianTracker(p.33)(mu_cart, Sigma_cart, dt_cart)
13  (TS.mu_rng_az, TS.Sigma_rng_az) = LinearTransform(p.32)(mu_xy[I_xy], Sigma_xy[I_xy, I_xy])
14  TS.mu_range = mu_r[1:2]
15  TS.Sigma_range = Sigma_r[1:2, 1:2]
16  TS.valid_rng = trk.valid_rng
17  TS.valid_vert = trk.valid_vert
18  TS.valid_rng_az = trk.valid_cart
19 end
```

Referenced In: ExtrapolateTrack(p.C-2)

To update the TRACKSUMMARY of the ADSBTRACKFILE to the current time, EXTRAPOLATEADSBTRACK (Algorithm 328) will retrieve the means and covariances of the vertical, and Cartesian state estimates. The latitude and longitude of the ownship is estimated with PROPAGATEOWNSHIPTO-TOA (Algorithm 53) to account for ownship motion in the time between the desired extrapolation time and the most recent ownship WGS84 state input. The relative horizontal track is then redefined in a rotated coordinate frame based on the predicted latitude and longitude of the ownship. If *dt* is less than *min_extrap_toa_step*, the state estimates must first be predicted forward using PREDICTVERTICALTRACKER and PREDICTADSBTRACKER. In order to complete the *mu_range* field

of the TRACKSUMMARY, the updated Cartesian estimate (*mu_xy* and *Sigma_xy*) is transformed to a ground range estimate using CONVERTCARTESIANTOPOLAR2D.

This algorithm takes as input *trk* and *T*. This algorithm updates the input *trk*.

Algorithm 328 ExtrapolateADSBTrack

```

1 function ExtrapolateADSBTrack(trk::ADSBTrackFile(p.E-4), T::R)
2   const m2ft::R = geoutils.meters_to_feet
3   const I_rng_az::Vector{Z} = [1, 3]
4   TS::TrackSummary(p.E-16)           = trk.trk_summary
5   mu_vert::Vector{R}                = copy(trk.mu_vert)
6   Sigma_vert::Matrix{R}             = copy(trk.Sigma_vert)
7   mu_ra::Vector{R}                 = zeros(4)
8   Sigma_ra::Matrix{R}              = zeros(4,4)
9   dt_vert = T - trk.toa_vert
10  dt_hor = T - trk.toa_hor
11  (olat, olon, nothing, nothing, nothing, nothing, nothing) = PropagateOwnershipToToa(p.56)(T)
12  (mu_xy_meters, Sigma_xy_meters) = RedefineEstimateInRotatedFrame(p.58)(trk, olat, olon)
13  (TS.mu_vert, TS.Sigma_vert) = PredictVerticalTracker(p.45)(mu_vert, Sigma_vert, dt_vert)
14  (mu_xy_meters, Sigma_xy_meters) = PredictADSBTracker(p.68)(mu_xy_meters, Sigma_xy_meters, dt_hor)
15  mu_xy = mu_xy_meters * m2ft
16  Sigma_xy = Sigma_xy_meters * (m2ft^2)
17  (mu_ra, Sigma_ra) = ConvertCartesianToPolar2D(p.C-4)(mu_xy, Sigma_xy)
18  TS.mu_rng_az = mu_ra[I_rng_az]
19  TS.Sigma_rng_az = Sigma_ra[I_rng_az, I_rng_az]
20  TS.mu_range    = mu_ra[1:2]
21  TS.Sigma_range = Sigma_ra[1:2,1:2]
22  TS.valid_rng = true
23  TS.valid_vert = trk.valid_vert
24  TS.valid_rng_az = true
25 end
```

Referenced In: ExtrapolateTrack(p.C-2)

CONVERTCARTESIANTOPOLAR2D (Algorithm 329) takes the two dimensional mean and covariance of a Cartesian state estimate and performs a coordinate transformation into a polar state estimate. The coordinate transformation is performed by sigma point sampling the Cartesian distribution using SIGMAPONTSAMPLE and calculating the position of the samples in polar coordinates. The samples are combined to form a polar mean and covariance using WEIGHTEDMEANANDCOVARIANCE, which is returned to the calling function.

This algorithm takes as input *mu_xy* and *Sigma_xy*. This algorithm returns *mu_ra* and *Sigma_ra*.

Algorithm 329 ConvertCartesianToPolar2D

```
1 function ConvertCartesianToPolar2D(mu_xy::Vector{R}, Sigma_xy::Matrix{R})
2     (S_xy::Matrix{R}, weights_xy::Vector{R}) = SigmaPointSample(p.28)(mu_xy, Sigma_xy, 2.0)
3     S_ra::Matrix{R} = zeros(size(S_xy))
4     for i in 1:length(weights_xy)
5         S_ra[1,i] = sqrt(S_xy[1,i]^2 + S_xy[3,i]^2)
6         S_ra[3,i] = atan2(S_xy[1,i], S_xy[3,i])
7         S_ra[2,i] = (S_xy[2,i] * sin(S_ra[3,i])) + (S_xy[4,i] * cos(S_ra[3,i]))
8         S_ra[4,i] = (S_xy[2,i] * cos(S_ra[3,i])) - (S_xy[4,i] * sin(S_ra[3,i]))
9     end
10    (mu_ra, Sigma_ra) = WeightedMeanAndCovariance(p.29)(S_ra, weights_xy)
11    return (mu_ra::Vector{R}, Sigma_ra::Matrix{R})
12 end
```

Referenced In: ExtrapolateADSBTrack(p.C-3)



C.2 Track Decorrelation

DECORRELATETARGETS (Algorithm 330) iterates through all the *lead_tracks* in the list returned by GENERATELEADTRACKLIST. New targets are formed for all Track Files within a target that have diverged from the *lead_track* of that target. If the difference between the time T and the initialization time of the Target is greater than the parameter *unique_time*, then a unique scale factor will be set to *unique_multiplier*. A list of all Mode C tracks is obtained through a call to GETTRACKSTODECORRELATE (Algorithm 331) and evaluated for correlation to the *lead_track* using CORRELATEBYTYPE (as a real or image track as determined by ISIMAGETRACK). During evaluation, the algorithm will skip correlation for the tracks whose *track_id* matches the *lead_track_id*. CORRELATIONHISTORYUPDATE (Algorithm 332) is called to update the correlation history if CORRELATEBYTYPE is successful. An M of N check is performed on the correlation history to determine whether the track should be decorrelated using CORRELATIONHISTORYCHECK (Algorithm 333), in which case the track is added to a new target using ADDDECORRELATEDTRACKTOTARGET (Algorithm 334) and removed from the previous target using REMOVEDECORRELATEDTRACKFROMTARGET (Algorithm 335). The new targets are marked as unique (previously established) and real (non-multipath), and stored in *decorrelated_targets*, which are then saved in memory using ADDTODB.

This algorithm takes as input T . This algorithm calls ADDTODB.

Algorithm 330 DecorrelateTargets

```

1 function DecorrelateTargets(T::R)
2   const max_coast_time::Z = params().surveillance.decorrelation.max_coast_time
3   const M::Z = params().surveillance.decorrelation.M
4   const N::Z = params().surveillance.decorrelation.N
5   const unique_time::Z = params().surveillance.decorrelation.unique_time
6   const unique_multiplier::R = params().surveillance.decorrelation.unique_multiplier
7   history::CorrelationHistory(p.E-5) = own.decorr_history
8   (history.M, history.N) = (M, N)
9   decorrelated_targets::Vector{Target} = Target(p.E-14) []
10  lead_track_list::Vector{TrackMap(p.E-15)} = GenerateLeadTrackList(p.C-9)()
11  for lead_track_mapping in lead_track_list
12    lead_track = lead_track_mapping.track
13    tgt_id = lead_track_mapping.db_id
14    lead_track_id = GetTrackID(p.C-8)(lead_track, tgt_id)
15    if (T - target_db[tgt_id].init_time > unique_time)
16      unique_scale = unique_multiplier
17    else
18      unique_scale = 1.0
19    end
20    tracks = GetTracksToDecorrelate(p.C-7)(tgt_id)
21    removed_tracks = TrackFile(p.E-3) []
22    for trk in tracks
23      track_id = GetTrackID(p.C-8)(trk, tgt_id)
24      if (lead_track_id == track_id) || (T - trk.toa > max_coast_time)
25        continue
26      end
27      as_image::Bool = IsImageTrack(p.B-8)(trk)
28      if (!CorrelateByType(p.C-17)(lead_track, trk, tgt_id, as_image, unique_scale))
29        CorrelationHistoryUpdate(p.C-7)(lead_track_id, track_id, history, T)
30        if (CorrelationHistoryCheck(p.C-7)(lead_track_id, track_id, history))
31          tgt = Target(p.E-14)()
32          tgt.init_time = trk.toa - unique_time
33          if (typeof(trk) == ModeCTrackFile(p.E-8)) && (as_image)
34            trk.is_image = false
35          end
36          AddDecorrelatedTrackToTarget(p.C-8)(tgt, trk)
37          push!(decorrelated_targets, tgt)
38          push!(removed_tracks, trk)
39        end
40      end
41    end
42    for rm_trk in removed_tracks
43      RemoveDecorrelatedTrackFromTarget(p.C-8)(target_db[tgt_id], rm_trk)
44    end
45  end
46  for dtgt in decorrelated_targets
47    AddToDB(p.20)(target_db, dtgt)
48  end
49 end

```

Referenced In: CorrelationProcessing(p.95)

Algorithm 331 GetTracksToDecorrelate

```

1 function GetTracksToDecorrelate(target_id::UInt32)
2     tracks = Array(TrackFile(p.E-3),0)
3     for modec_track in target_db[target_id].modec_tracks
4         push!(tracks, modec_track)
5     end
6     if (TrackExists(p.18)(target_db[target_id].adsr_track)) && (target_db[target_id].adsr_track.non_icao)
7         push!(tracks, target_db[target_id].adsr_track)
8     end
9     return tracks::Array{TrackFile(p.E-3)}
10 end
```

Referenced In: DecorrelateTargets(p.C-6)

Algorithm 332 CorrelationHistoryUpdate

```

1 function CorrelationHistoryUpdate(lead_track_id::Union(UInt32, String), track_id::Union(UInt32, String),
2     history::CorrelationHistory(p.E-5), T::R)
3     h1 = history.id_pairs
4     h2 = history.timing
5     v = [lead_track_id, track_id]
6     if (!isempty(h2))
7         while T - h2[end] >= history.N
8             pop!(h2)
9             pop!(h1)
10            if (isempty(h2))
11                break
12            end
13        end
14        unshift!(h1, v)
15        unshift!(h2, T)
16 end
```

Referenced In: CorrelateIndividualTracks(p.C-14), DecorrelateTargets(p.C-6)
--

Algorithm 333 CorrelationHistoryCheck

```

1 function CorrelationHistoryCheck(lead_track_id::Union(UInt32, String), track_id::Union(UInt32, String), history
2     ::CorrelationHistory(p.E-5))
3     count::Z = 0
4     h1 = history.id_pairs
5     v = [lead_track_id, track_id]
6     for k in 1:length(h1)
7         if (v == h1[k])
8             count += 1
9         end
10    end
11    return (count >= history.M)::Bool
12 end
```

Referenced In: CorrelateIndividualTracks(p.C-14), DecorrelateTargets(p.C-6)
--

Algorithm 334 AddDecorrelatedTrackToTarget

```

1 function AddDecorrelatedTrackToTarget(tgt::Target(p.E-14), trk::TrackFile(p.E-3))
2   if (typeof(trk) == ModeCTrackFile(p.E-8))
3     push!(tgt.modec_tracks, trk)
4   elseif (typeof(trk) == ADSBTrackFile(p.E-4)) && (trk.non_icao)
5     tgt.adsr_track = trk
6   end
7 end
```

Referenced In: DecorrelateTargets(<i>p.</i> C-6)
--

Algorithm 335 RemoveDecorrelatedTrackFromTarget

```

1 function RemoveDecorrelatedTrackFromTarget(tgt::Target(p.E-14), trk::TrackFile(p.E-3))
2   for i in 1:length(tgt.modec_tracks)
3     if (tgt.modec_tracks[i] == trk)
4       splice!(tgt.modec_tracks, i)
5       return
6     end
7   end
8   if (TrackExists(p.18)(tgt.adsr_track)) && (tgt.adsr_track == trk) && (tgt.adsr_track.non_icao)
9     tgt.adsr_track = nothing
10    return
11  end
12 end
```

Referenced In: DecorrelateTargets(<i>p.</i> C-6)
--

In order to uniquely identify tracks, an internal ID is given to each track by **GETTRACKID** (Algorithm 336). The algorithm retrieves the *target* from memory using the *target_id* and will find the index of the *track* within that *target* (denoted by *track_index*). The returned *track_id* is the concatenation of the target ID in memory (*target_id*), the type of track (*track_type*), and the index of the track within the specified *target* (*track_index*). An example ID would be “12345ModeSTrackFile1”.

This algorithm takes as input *track* and *target_id*. This algorithm returns *track_id*.

Algorithm 336 GetTrackID

```

1 function GetTrackID(track::TrackFile(p.E-3), target_id::UInt32)
2   track_type = string(typeof(track))
3   target::Target(p.E-14) = target_db[target_id]
4   if (track_type == "ModeCTrackFile(p.E-8)")
5     track_index = findfirst(target.modec_tracks, track)
6   else
7     track_index = 1
8   end
9   if (track_type == "ADSBTrackFile(p.E-4)") && (track.rebroadcast)
10    track_type = "ADSRTrackFile"
11  end
12  track_id::String = string(target_id, track_type, track_index)
13  return track_id::String
14 end
```

Referenced In: DecorrelateTargets(<i>p.</i> C-6)
--

GENERATELEADTRACKLIST (Algorithm 337) returns a listing of tracks selected from the Target database which best represent each target for correlation processing purposes. Mode S tracks are the preferred correlation track type followed by ADS-B, then Mode C, and finally ADS-R. Mode C image tracks are not eligible to be returned as lead tracks.

This algorithm takes no input. This algorithm returns *lead_track_list*.

Algorithm 337 GenerateLeadTrackList

```

1 function GenerateLeadTrackList()
2     lead_track_list::Vector{TrackMap} = Array(TrackMap(p.E-15), 0)
3     for id in keys(target_db)
4         lead_track::Union(TrackFile(p.E-3), Nothing) = nothing
5         target = target_db[id]
6         if (TrackExists(p.18)(target.modes_track))
7             lead_track = target.modes_track
8         elseif (TrackExists(p.18)(target.adsb_track))
9             lead_track = target.adsb_track
10        elseif (TrackExists(p.18)(target.modec_tracks))
11            found_non_img_trk::Bool = false
12            for trk in target.modec_tracks
13                if (trk.is_image == false)
14                    lead_track = trk
15                    found_non_img_trk = true
16                    break
17                end
18            end
19            if (!found_non_img_trk)
20                lead_track = target.modec_tracks[1]
21            end
22        elseif (TrackExists(p.18)(target.adsr_track))
23            lead_track = target.adsr_track
24        end
25        if (TrackExists(p.18)(lead_track))
26            track_map::TrackMap = TrackMap(p.E-15)(lead_track, id)
27            push!(lead_track_list, track_map)
28        end
29    end
30    return lead_track_list::Vector{TrackMap(p.E-15)}
31 end
```

Referenced In: DecorrelateTargets(p.C-6) , CorrelateTargets(p.C-12)
--

C.3 Track Correlation

CORRELATETARGETS (Algorithm 338) iterates through all targets in memory and merges any pairs of targets that have correlating lead tracks into one target. A list of lead tracks is generated using GENERATELEADTRACKLIST and stored in *lead_track_list*. The list of lead tracks is sorted by range using SORTBYRANGE (Algorithm 339), which gets the range from the *TrackSummary*. The algorithm iterates through the *lead_track_list* and examines every combination of target pairs for correlation. Before proceeding, an efficiency check is performed to skip correlation when the difference in range between the two tracks is larger than the *range_gate*. If correlation is not skipped, the process continues by passing the tracks and TARGET IDs to CORRELATEINDIVIDUALTRACKS (Algorithm 340). If an ID is returned, a target has been merged and is flagged as deleted in *pending_deletion*. If the returned ID is equal to *CORRELATION_PENDING*, then we want to skip the following steps to avoid correlating by image on a track that has already passed at least 1 of *N* tests. The process is repeated for targets that have Mode C lead tracks by setting the *is_image* argument of CORRELATEINDIVIDUALTRACKS to TRUE.

This algorithm takes as input *T*. This algorithm updates *pending_deletion*.

Algorithm 338 CorrelateTargets

```

1 function CorrelateTargets(T::R)
2   const range_gate::Z      = params().surveillance.correlation.range_gate
3   const image_range_gate::Z = params().surveillance.correlation.image_range_gate
4   status::Z = CORRELATION_FAILED
5   lead_track_list::Vector{TrackMap(p.E-15)} = GenerateLeadTrackList(p.C-9)()
6   lead_track_list = SortByRange(p.C-13)(lead_track_list)
7   pending_deletion::Vector{Bool} = fill(false, length(lead_track_list))
8   lowIndexB::Z = 1
9   for indexA in 1:length(lead_track_list)
10    if (pending_deletion[indexA])
11      continue
12    end
13    trackA = lead_track_list[indexA].track
14    for indexB in lowIndexB:length(lead_track_list)
15      if (pending_deletion[indexB]) || (indexA == indexB)
16        continue
17      end
18      trackB = lead_track_list[indexB].track
19      if (trackA.trk_summary.mu_range[1] - trackB.trk_summary.mu_range[1] > range_gate)
20        lowIndexB = indexB
21        continue
22      end
23      targetA_id = lead_track_list[indexA].db_id
24      targetB_id = lead_track_list[indexB].db_id
25      if (trackB.trk_summary.mu_range[1] - trackA.trk_summary.mu_range[1] <= range_gate)
26        (merged_id::UInt32, status) = CorrelateIndividualTracks(p.C-14)(trackA, targetA_id, trackB,
27                           targetB_id, false, T)
28        if (status != CORRELATION_FAILED)
29          if (merged_id == targetB_id)
30            pending_deletion[indexA] = true
31            break
32          elseif (merged_id == targetA_id)
33            pending_deletion[indexB] = true
34            continue
35          elseif (status == CORRELATION_PENDING)
36            continue
37          end
38        end
39        if (trackA.trk_summary.mu_range[1] > trackB.trk_summary.mu_range[1])
40          continue
41        end
42        adj_image_range_gate = image_range_gate
43        if (trackA.trk_summary.valid_vert && trackB.trk_summary.valid_vert)
44          adj_image_range_gate = min(image_range_gate, (trackA.trk_summary.mu_vert[1] + trackB.
45                                         trk_summary.mu_vert[1]))
46        end
47        if (trackB.trk_summary.mu_range[1] - trackA.trk_summary.mu_range[1] > adj_image_range_gate)
48          break
49        end
50        (merged_image_id::UInt32, status) = CorrelateIndividualTracks(p.C-14)(trackA, targetA_id, trackB,
51                           targetB_id, true, T)
52        if (status != CORRELATION_FAILED)
53          if (merged_image_id == targetB_id)
54            pending_deletion[indexA] = true
55            break
56          elseif (merged_image_id == targetA_id)
57            pending_deletion[indexB] = true
58            continue
59          end
60        end
61 end

```

Referenced In: CorrelationProcessing(p.95)
--

Algorithm 339 SortByRange

```
1 function SortByRange(list::Vector{TrackMap(p. E-15)})  
2     CompareRange(a,b) = a.track.trk_summary.mu_range[1] < b.track.trk_summary.mu_range[1]  
3     return sort(list, lt = CompareRange)  
4 end
```

Referenced In: CorrelateTargets(p. C-12)

CORRELATEINDIVIDUALTRACKS (Algorithm 340) takes as input two tracks and their associated TARGET ID and merges the two targets together if they correlate. The tracks are evaluated for correlation using CORRELATEBYTYPE (as a real or image track based on the *as_image* argument). CORRELATIONHISTORYUPDATE is called to update the correlation history if CORRELATEBYTYPE is successful. A call to CORRELATIONHISTORYCHECK performs an M of N check on the correlation history to determine whether the targets should be merged, in which case the tracks of the most recently initialized target are merged into the older target through a call to MERGETARGETS. The TARGET ID of the merged target is returned to the calling function.

This algorithm takes as input *trackA*, *targetA_id*, *trackB*, *targetB_id*, *as_image*, and *T*. This algorithm returns *target_id* and *status*.

Algorithm 340 CorrelateIndividualTracks

```

1 function CorrelateIndividualTracks(trackA::TrackFile, targetA_id::UInt32, trackB::TrackFile(p.E-3), targetB_id::UInt32, as_image::Bool, T::R)
2   const M::Z = params().surveillance.correlation.M
3   const N::Z = params().surveillance.correlation.N
4   const unique_time::Z = params().surveillance.correlation.unique_time
5   const unique_multiplier::R = params().surveillance.correlation.unique_multiplier
6   target_id::UInt32 = 0
7   status::Z = CORRELATION_FAILED
8   history = own.corr_history
9   if (as_image)
10     history = own.corr_image_history
11   end
12   (history.M, history.N) = (M, N)
13   if (T - max(target_db[targetA_id].init_time, target_db[targetB_id].init_time) > unique_time)
14     unique_scale = unique_multiplier
15   else
16     unique_scale = 1.0
17   end
18   if (CorrelateByType(p.C-17)(trackA, trackB, targetA_id, as_image, unique_scale))
19     CorrelationHistoryUpdate(p.C-7)(targetA_id, targetB_id, history, T)
20     if (CorrelationHistoryCheck(p.C-7)(targetA_id, targetB_id, history))
21       if (typeof(trackB) == ModeCTrackFile(p.E-8)) && (as_image)
22         trackB.is_image = as_image
23       end
24       targetA = target_db[targetA_id]
25       targetB = target_db[targetB_id]
26       if (targetB.init_time < targetA.init_time)
27         MergeTargets(p.C-15)(targetB_id, targetA_id)
28         target_id = targetB_id
29         status = CORRELATION_SUCCEEDED
30       else
31         MergeTargets(p.C-15)(targetA_id, targetB_id)
32         target_id = targetA_id
33         status = CORRELATION_SUCCEEDED
34       end
35     else
36       target_id = 0
37       status = CORRELATION_PENDING
38     end
39   else
40     target_id = 0
41     status = CORRELATION_FAILED
42   end
43   return (target_id::UInt32, status::Z)
44 end

```

Referenced In: CorrelateTargets(p.C-12)

MERGETARGETS (Algorithm 341) is used during correlation to merge two targets together. If the Target to be deleted has a Mode S Track File, and the Target to be merged does not, then the Mode S Track File, active validation history, and active validation state from the Target to be deleted are copied over to the merging Target. After that, the Mode S Track File of the Target to be deleted is removed. All Mode C Track Files within the Target to be deleted are copied over to the Target to be merged and removed. If the Target to be deleted has an ADS-B Track File or ADS-R Track File, and the Target to be merged does not, then the ADS-B Track File/ADS-R Track File, ADS-B quality history, and ADS-B quality override flag is copied over to the merging Target. After that the ADS-B Track File/ADS-R Track File of the Target to be deleted is removed. Next, the most up-to-date coordination data between the two Targets will be saved. The Xo designation states of the two Targets are merged using MERGETARGETDESIGNATIONS. Finally, if TARGETISEMPTY

returns TRUE, then the Target is deleted from memory.

This algorithm takes as input *id_to_merge* and *id_to_delete*. This algorithm calls MERGETARGETDESIGNATIONS and deletes the target from memory if TARGETISEMPTY returns TRUE.

Algorithm 341 MergeTargets

```

1 function MergeTargets(id_to_merge::UInt32, id_to_delete::UInt32)
2   global target_db
3   target_to_merge = target_db[id_to_merge]
4   target_to_delete = target_db[id_to_delete]
5   if (TrackExists(target_to_delete.modes_track)) && (!TrackExists(p.18)(target_to_merge.modes_track))
6     target_to_merge.modes_track = target_to_delete.modes_track
7     target_to_merge.av_history = deepcopy(target_to_delete.av_history)
8     target_to_merge.av_state = target_to_delete.av_state
9   end
10  target_to_delete.modes_track = nothing
11  for i in reverse(1:length(target_to_delete.modec_tracks))
12    trk = shift!(target_to_delete.modec_tracks)
13    push!(target_to_merge.modec_tracks, trk)
14  end
15  if (TrackExists(target_to_delete.adsb_track)) && (!TrackExists(p.18)(target_to_merge.adsb_track))
16    target_to_merge.adsb_track = target_to_delete.adsb_track
17    target_to_merge.adsb_qual_history = target_to_delete.adsb_qual_history
18    target_to_merge.passive_only_adsb_qual_history = target_to_delete.passive_only_adsb_qual_history
19    target_to_merge.adsb_qual_override = target_to_delete.adsb_qual_override
20  end
21  target_to_delete.adsb_track = nothing
22  if (TrackExists(target_to_delete.adsr_track)) && (!TrackExists(p.18)(target_to_merge.adsr_track))
23    target_to_merge.adsr_track = target_to_delete.adsr_track
24    target_to_merge.adsb_qual_history = target_to_delete.adsb_qual_history
25    target_to_merge.passive_only_adsb_qual_history = target_to_delete.passive_only_adsb_qual_history
26    target_to_merge.adsb_qual_override = target_to_delete.adsb_qual_override
27  end
28  target_to_delete.adsr_track = nothing
29  if (target_to_delete.coord_data.toa > target_to_merge.coord_data.toa)
30    target_to_merge.coord_data = deepcopy(target_to_delete.coord_data)
31    target_to_merge.bad_uf16uds30 = target_to_delete.bad_uf16uds30
32  end
33  MergeTargetDesignations(p.C-16)(target_to_merge, target_to_delete)
34  if (TargetIsEmpty(p.119)(target_to_delete))
35    delete!(target_db, id_to_delete)
36  end
37 end
```

Referenced In: CorrelateIndividualTracks(p.C-14)

Algorithm 342 MergeTargetDesignations

```

1 function MergeTargetDesignations(target_to_merge::Target, target_to_delete::Target(p. E-14))
2     mode_s::UInt32 = 0
3     if (TrackExists(p.18)(target_to_merge.modes_track))
4         mode_s = target_to_merge.modes_track.modes
5     elseif (TrackExists(p.18)(target_to_merge.adsb_track))
6         mode_s = target_to_merge.adsb_track.modes
7     elseif (TrackExists(p.18)(target_to_merge.adsr_track)) && (!target_to_merge.adsr_track.non_icao)
8         mode_s = target_to_merge.adsr_track.modes
9     elseif (IsTargetDesignationActive(p.141)(target_to_merge.designation_state,true))
10        mode_s = target_to_merge.designation_state.mode_s
11    elseif (IsTargetDesignationActive(p.141)(target_to_delete.designation_state,true))
12        mode_s = target_to_delete.designation_state.mode_s
13    end
14    active_ra::Bool = target_to_merge.designation_state.active_ra ||
15                    target_to_delete.designation_state.active_ra
16    multithreat_ra::Bool = target_to_merge.designation_state.multithreat_ra ||
17                    target_to_delete.designation_state.multithreat_ra
18    if (0 == mode_s)
19        if (!IsTargetDesignationActive(p.141)(target_to_merge.designation_state,false))
20            if (IsTargetDesignationActive(p.141)(target_to_delete.designation_state,false))
21                target_to_merge.designation_state = target_to_delete.designation_state
22            else
23                target_to_merge.designation_state = DesignationState(p. E-6)()
24            end
25        end
26        target_to_delete.designation_state = DesignationState(p. E-6)()
27    elseif (!IsTargetDesignationActive(p.141)(target_to_merge.designation_state,true))
28        if (mode_s == target_to_delete.designation_state.mode_s)
29            target_to_merge.designation_state = target_to_delete.designation_state
30            target_to_delete.designation_state = DesignationState(p. E-6)()
31        elseif (mode_s != target_to_merge.designation_state.mode_s)
32            target_to_merge.designation_state = DesignationState(p. E-6)(mode_s)
33        end
34    elseif (mode_s != target_to_merge.designation_state.mode_s)
35        if (mode_s == target_to_delete.designation_state.mode_s)
36            tmp::DesignationState(p. E-6) = target_to_merge.designation_state
37            target_to_merge.designation_state = target_to_delete.designation_state
38            target_to_delete.designation_state = tmp
39        else
40            target_to_delete.designation_state = target_to_merge.designation_state
41            target_to_merge.designation_state = DesignationState(p. E-6)(mode_s)
42        end
43    end
44    target_to_merge.designation_state.active_ra = active_ra
45    target_to_merge.designation_state.multithreat_ra = multithreat_ra
46 end
```

Referenced In: MergeTargets(p.C-15)
--

CORRELATEBYTYPE (Algorithm 343) takes as input two track files, *lead_track* and *trk*, and executes the proper correlation algorithm for that pair of tracks types. If *is_image* is set, the tracks are evaluated for correlation as image tracks using CORRELATEIMAGE. If both tracks have ICAO addresses (e.g. Mode S and ADS-B tracks), CORRELATEID is used. For all other combinations, spatial correlation is performed in CORRELATEPOSITION.

This algorithm takes as input *lead_track*, *trk*, *target_id*, *as_image*, and *unique_scale*. This algorithm returns the outputs of either CORRELATEID or CORRELATEPOSITION.

Algorithm 343 CorrelateByType

```

1 function CorrelateByType(lead_track::TrackFile, trk::TrackFile(p.E-3), target_id::UInt32, as_image::Bool,
2   unique_scale::R)
3   id_types = Union(ADSBTrackFile(p.E-4), ModeSTrackFile(p.E-9))
4   lead_non_icao::Bool = ((typeof(lead_track) == ADSBTrackFile(p.E-4)) && (lead_track.non_icao))
5   trk_non_icao::Bool = ((typeof(trk) == ADSBTrackFile(p.E-4)) && (trk.non_icao))
6   if (as_image)
7     if (typeof(lead_track) == ModeCTrackFile) && (typeof(trk) == ModeCTrackFile(p.E-8))
8       return CorrelateImage(p.C-18)(lead_track, trk, unique_scale)
9     else
10    return false
11  end
12  elseif (typeof(lead_track) <: id_types) && (typeof(trk) <: id_types) && (lead_non_icao == false) && (
13    trk_non_icao == false)
14  return CorrelateID(p.C-20)(trk, target_id)
15  else
16  return CorrelatePosition(p.C-21)(lead_track, trk, unique_scale)
17  end
18 end

```

Referenced In: CorrelateIndividualTracks(p.C-14), DecorrelateTargets(p.C-6)

CORRELATEIMAGE (Algorithm 344) takes two TRACKSUMMARY data structures and determines whether a track correlates with another as an image track using spatial techniques. To correlate as an image track, the track must first be further in range than the real track as a multipath reply cannot be detected as closer in range. The altitude estimates of each track are compared and will fail correlation if the distance exceeds the *altitude_gate* to avoid unnecessary computation. The estimated range rate of the image track is calculated for both the single reflection case and the double reflection case using CALCULATEIMAGERANGERATE and stored in *imageRR1* and *imageRR2*. The track summaries of Mode C tracks are inflated with *altitude_inflation* to account for quantization bias. The algorithm calculates a probabilistic score based on how well the two tracks correlate in range, altitude and range rate. The range rate component of the score is calculated using the Mahalanobis distance for the mean and covariance of the tracked range rate estimate and each calculated range rate estimate. A scaling factor is applied to the rangerate covariance during the Mahalanobis distance calculation (*variance_scale*). This algorithm can handle when tracks are NAR. If both tracks are NAR (i.e. invalid vertical track), then the scoring threshold is subtracted by the *correlation_alt_threshold*. If either of the tracks are NAR, then the altitude component of the score is set to double the *correlation_threshold*. Otherwise, if both tracks are reporting altitude, then the altitude component of the score is calculated using the Mahalanobis distance for the mean and covariance of the tracked vertical estimates of the two tracks. If ownship is nominally receiving heading (*ownship.heading_state* is valid and not degraded) and the TRACKSUMMARY for both tracks have valid bearing (*valid_rng_az* is TRUE), the algorithm will calculate a probabilistic score based on how well the two tracks correlate in bearing. The bearing component of the score is calculated using the Mahalanobis distance for the mean and covariance of the tracked bearing estimates of the two tracks.

A final correlation *score* is calculated using the sum of the altitude, altitude rate, and the smaller of *imageRR1* and *imageRR2* scores. The bearing score is used in place of the altitude rate score if the ownship is nominally receiving heading. If the bearing score is used with tracks that are NAR, the score threshold is increased by the *correlation_az_threshold* to account for the additional uncertainty. If the final *score* is below the *correlation_threshold* scaled by the incoming *unique_scale*, the algorithm returns a Boolean value indicating the two tracks correlate.

This algorithm takes as input *trackA*, *trackB*, and *unique_scale*. This algorithm returns TRUE or FALSE.

Algorithm 344 CorrelateImage

```

1 function CorrelateImage(trackA::TrackFile, trackB::TrackFile(p.E-3), unique_scale::R)
2     const altitude_gate::Z          = params().surveillance.correlation.type_selection.image.altitude_gate
3     const rangerate_gate::Z         = params().surveillance.correlation.type_selection.image.rangerate_gate
4     const altitude_inflation::R    = params().surveillance.correlation.type_selection.image.altitude_
5         inflation
6     const correlation_threshold::R = params().surveillance.correlation.type_selection.image.correlation_
7         threshold
8     const correlation_alt_threshold::R = params().surveillance.correlation.type_selection.image.correlation_alt_threshold
9     const correlation_az_threshold::R = params().surveillance.correlation.type_selection.image.correlation_az_threshold
10    const variance_scale::R        = params().surveillance.correlation.type_selection.image.variance_scale
11    TS1 = deepcopy(trackA.trk_summary)
12    TS2 = deepcopy(trackB.trk_summary)
13    if (TS1.mu_range[1] > TS2.mu_range[1])
14        return false
15    elseif (TS1.valid_vert) && (TS2.valid_vert) && (abs(TS1.mu_vert[1] - TS2.mu_vert[1]) > altitude_gate)
16        return false
17    end
18    imageRR1::R = CalculateImageRangeRate(p.C-19)(TS1, TS2.mu_range[1], 1)
19    imageRR2::R = CalculateImageRangeRate(p.C-19)(TS1, TS2.mu_range[1], 2)
20    if (min(abs(TS2.mu_range[2] - imageRR1), abs(TS2.mu_range[2] - imageRR2)) > rangerate_gate)
21        return false
22    end
23    score::R = altitude_score::R = altituderate_score::R = bearing_score::R = 0.0
24    threshold = correlation_threshold
25    rangerate_score1::R = mahal(p.H-1)(imageRR1, TS1.Sigma_range[2,2]/variance_scale, TS2.mu_range[2], TS2.
26        Sigma_range[2,2]/variance_scale)
27    rangerate_score2::R = mahal(p.H-1)(imageRR2, TS1.Sigma_range[2,2]/variance_scale, TS2.mu_range[2], TS2.
28        Sigma_range[2,2]/variance_scale)
29    if (!TS1.valid_vert) && (!TS2.valid_vert)
30        threshold = threshold - correlation_alt_threshold
31    elseif (!TS1.valid_vert) || (!TS2.valid_vert)
32        altitude_score = 2*correlation_threshold
33    else
34        TS1.Sigma_vert[1,1] = TS1.Sigma_vert[1,1] + altitude_inflation^2
35        altitude_score = mahal(p.H-1)(TS1.mu_vert[1], TS1.Sigma_vert[1,1], TS2.mu_vert[1], TS2.Sigma_vert
36            [1,1])
37        altituderate_score = mahal(p.H-1)(TS1.mu_vert[2], TS1.Sigma_vert[2,2], TS2.mu_vert[2], TS2.Sigma_vert
38            [2,2])
39    end
40    use_bearing = (own.heading_state == OWN_HEADING_NOMINAL) && (TS1.valid_rng_az) && (TS2.valid_rng_az)
41    if (use_bearing)
42        bearing_score = mahal(p.H-1)(WrapToPi(p.H-3)(AngleDifference(p.H-2)(TS1.mu_rng_az[2], TS2.mu_rng_az[2])), TS1
43            .Sigma_rng_az[2,2], 0, TS2.Sigma_rng_az[2,2])
44        if (threshold < correlation_threshold)
45            threshold = threshold + correlation_az_threshold
46        end
47        score = bearing_score + altitude_score + min(rangerate_score1, rangerate_score2)
48    else
49        score = altituderate_score + altitude_score + min(rangerate_score1, rangerate_score2)
50    end
51    if (score < threshold*unique_scale)
52        return true
53    else
54        return false
55    end
56 end

```

Referenced In: CorrelateByType(p. C-17)

Algorithm 345 CalculateImageRangeRate

```

1 function CalculateImageRangeRate(trk::TrackSummary(p.E-16), im_ground_range::R, reflections::Z)
2   own_alt::R           = own.mu_h[1]
3   own_alt_rate::R      = own.mu_h[2]
4   real_im_ground_range::R = trk.mu_range[1]
5   real_im_ground_rangerate::R = trk.mu_range[2]
6   real_alt::R          = own_alt
7   real_alt_rate::R     = 0.0
8   if trk.valid_vert
9     real_alt           = trk.mu_vert[1]
10    real_alt_rate      = trk.mu_vert[2]
11  end
12  alt1::R            = own_alt + real_alt
13  alt2::R            = own_alt - real_alt
14  alt3::R            = own_alt_rate + real_alt_rate
15  alt4::R            = own_alt_rate - real_alt_rate
16  im_range             = sqrt(im_ground_range2 + alt22)
17  real_im_range::R    = sqrt(real_im_ground_range2 + alt22)
18  real_im_rangerate::R = (real_im_ground_range*real_im_ground_rangerate + alt2*alt4) / (real_im_range)
19  im_ground_rangerate::R = 0.0
20  if (reflections == 1)
21    tm1::R           = 2*im_range - real_im_range
22    im_rangerate = (real_im_rangerate + ((sqrt(abs(tm12 - real_im_range2 + alt22)) * alt3) +
23      real_im_range*real_im_rangerate - alt2*alt4) / (tm1) / 2
24    im_ground_rangerate = (im_range*im_rangerate - alt2*alt4) / (im_ground_range)
25  else (reflections == 2)
26    im_ground_rangerate = ((sqrt(abs(im_ground_range2 - real_im_ground_range2 + alt22)) * alt3) +
27      real_im_ground_range*real_im_ground_rangerate - alt2*alt4) / im_ground_range
28  end
29  return im_ground_rangerate::R
28 end
```

Referenced In: CorrelateImage(p.C-18)
--

CORRELATEID (Algorithm 346) takes in a single track and the ID of the TARGET in question to attempt to correlate based on ICAO addresses. The algorithm will only check against Mode S, ADS-B, and ADS-R tracks as Mode C tracks do not carry an ICAO address.

This algorithm takes as input *trackB* and *target_id*. This algorithm returns TRUE or FALSE.

Algorithm 346 CorrelateID

```

1 function CorrelateID(trackB::TrackFile(p.E-3), target_id::UInt32)
2   if (TrackExists(p.18)(target_db[target_id].modes_track))
3     if (trackB.modes == target_db[target_id].modes_track.modes)
4       return true
5     end
6   end
7   if (TrackExists(p.18)(target_db[target_id].adsb_track))
8     if (trackB.modes == target_db[target_id].adsb_track.modes)
9       return true
10    end
11  end
12  if (TrackExists(p.18)(target_db[target_id].adsr_track))
13    if (trackB.modes == target_db[target_id].adsr_track.modes)
14      return true
15    end
16  end
17  return false
18 end

```

Referenced In: CorrelateByType(p.C-17)

CORRELATEPOSITION (Algorithm 347) takes two TRACKSUMMARY data structures and determines whether the two tracks correlate with one another with spatial techniques. The range and altitude estimates of each track are compared and will fail correlation if the distance exceeds the *range_gate* and *altitude_gate* to avoid unnecessary computation. The track summaries of Mode C tracks must be inflated with *altitude_inflation* to account for quantization bias. The algorithm calculates a probabilistic score based on how well the two tracks correlate in range, altitude and range rate. The range component of the score is calculated using the Mahalanobis distance for the mean and covariance of the tracked range estimates of the two tracks. The range rate component of the score is calculated using the Mahalanobis distance for the mean and covariance of the tracked range rate estimates of the two tracks. This algorithm can handle when tracks are NAR. If both tracks are NAR, then the scoring threshold is subtracted by the *correlation_alt_threshold*. If either of the tracks are NAR, then the altitude component of the score is set to double the *correlation_threshold*. Otherwise, if both tracks are reporting altitude, then the altitude component of the score is calculated using the Mahalanobis distance for the mean and covariance of the tracked vertical estimates of the two tracks. If ownership is nominally receiving heading (*ownership.heading_state* is valid and not degraded) and the TRACKSUMMARY for both tracks have valid bearing (*valid_rng_az* is TRUE), the algorithm will calculate a probabilistic score based on how well the two tracks correlate in bearing. The bearing component of the score is calculated using the Mahalanobis distance for the mean and covariance of the tracked bearing estimates of the two tracks.

A final correlation *score* is calculated using the sum of the range, altitude, and range rate scores. The bearing score is used in place of the range rate score if it is a higher score and the ownership is nominally receiving heading. If the bearing score is used with tracks that are NAR, the score threshold is increased by the *correlation_az_threshold* to account for the additional uncertainty. If the final *score* is below the *correlation_threshold* scaled by the incoming *unique_scale*, the algorithm returns a Boolean value indicating the two tracks correlate.

This algorithm takes as input *trackA*, *trackB*, and *unique_scale*. This algorithm returns TRUE or FALSE.

Algorithm 347 CorrelatePosition

```

1 function CorrelatePosition(trackA::TrackFile, trackB::TrackFile(p.E-3), unique_scale:::R)
2     const altitude_gate::Z = params().surveillance.correlation.type_selection.position.altitude_gate
3     const range_gate::Z = params().surveillance.correlation.type_selection.position.range_gate
4     const altitude_inflation::R = params().surveillance.correlation.type_selection.position.altitude_inflation
5     const correlation_threshold::R = params().surveillance.correlation.type_selection.correlation_threshold
6     const correlation_alt_threshold::R = params().surveillance.correlation.type_selection.position.correlation_alt_threshold
7     const correlation_az_threshold::R = params().surveillance.correlation.type_selection.position.correlation_az_threshold
8     TS1 = deepcopy(trackA.trk_summary)
9     TS2 = deepcopy(trackB.trk_summary)
10    if (abs(TS1.mu_range[1] - TS2.mu_range[1]) > range_gate)
11        return false
12    elseif (TS1.valid_vert) && (TS2.valid_vert) && (abs(TS1.mu_vert[1] - TS2.mu_vert[1]) > altitude_gate)
13        return false
14    end
15    altitude_score::R = score::R = bearing_score::R = 0.0
16    threshold = correlation_threshold
17    range_score::R = mahal(p.H-1)(TS1.mu_range[1], TS1.Sigma_range[1,1], TS2.mu_range[1], TS2.Sigma_range[1,1])
18    rangerate_score::R = mahal(p.H-1)(TS1.mu_range[2], TS1.Sigma_range[2,2], TS2.mu_range[2], TS2.Sigma_range[2,2])
19    if (!TS1.valid_vert) && (!TS2.valid_vert)
20        threshold = threshold - correlation_alt_threshold
21    elseif (!TS1.valid_vert) || (!TS2.valid_vert)
22        altitude_score = 2*correlation_threshold
23    else
24        if (typeof(trackA) == ModeCTrackFile) || (typeof(trackB) == ModeCTrackFile(p.E-8))
25            TS1.Sigma_vert[1,1] = TS1.Sigma_vert[1,1] + altitude_inflation^2
26        end
27        altitude_score = mahal(p.H-1)(TS1.mu_vert[1], TS1.Sigma_vert[1,1], TS2.mu_vert[1], TS2.Sigma_vert[1,1])
28    end
29    use_bearing = (own.heading_state == OWN_HEADING_NOMINAL) && (TS1.valid_rng_az) && (TS2.valid_rng_az)
30    if (use_bearing)
31        bearing_score = mahal(p.H-1)(WrapToPi(p.H-3)(AngleDifference(p.H-2)(TS1.mu_rng_az[2], TS2.mu_rng_az[2])), TS1.Sigma_rng_az[2,2], 0, TS2.Sigma_rng_az[2,2])
32        if (rangerate_score < bearing_score) && (threshold < correlation_threshold)
33            threshold = threshold + correlation_az_threshold
34        end
35        score = altitude_score + range_score + max(bearing_score, rangerate_score)
36    else
37        score = altitude_score + range_score + rangerate_score
38    end
39    if (score < threshold*unique_scale)
40        return true
41    else
42        return false
43    end
44 end

```

Referenced In: CorrelateByType(p. C-17)

Appendix D Constant Variables

D.1 Geometric Constants

Geometric parameters used to convert data in WGS84 coordinates to ENU coordinates.

Type 1 | GeoUtils

```
1 type  GeoUtils
2   earth_seimajor_axis::R      # Semi-major axis of the Earth
3   earth_seiminor_axis::R      # Semi-minor axis of the Earth
4   earth_flattening::R         # Flattening of the Earth
5   gravity_ratio::R            # Gravity ratio
6   earth_first_eccentricity_sq::R # First eccentricity of the Earth
7   earth_second_eccentricity_sq::R # Second eccentricity of the Earth
8   earth_equatorial_gravity::R   # Gravity of the Earth at the equator
9   earth_gravity_lat45::R       # Gravity of the Earth at 45 degrees latitude
10  somiglianas_constant::R      # Somigliana's constant
11  meters_to_feet::R            # Conversion factor for meters to feet
12  kts_to_mps::R                # Conversion factor for knots to meters per second
13  feet_to_nautical_miles::R    # Conversion factor for feet to nautical miles
14  GeoUtils() = new(
15    6.3781370e6,
16    6.3567523142e6,
17    0.0033528106718309896,
18    0.003449787,
19    6.69437999014e-3,
20    6.73949674228e-3,
21    9.7803253359,
22    9.8061977710758619,
23    1.931853e-3,
24    3.28083989501,
25    0.514444444444,
26    1.6457883369e-4
27  )
28 end
```

D.2 Global Constants

A set of system constants are used throughout the logic. Listed below are the global variables and their values (speeds are in ft/s and accelerations are in ft/s²).

Algorithm 348 constants

```

1 function constants()
2     # Protection Mode Options for TRM Look-Up Tables
3     global const PROTECTION_MODE_Xa      = uint8( 1 )
4     global const PROTECTION_MODE_Xo_CSP03k = uint8( 2 )
5     # Equipage Codes
6     global const EQUIPAGE_ATCRBS   = int( 0 )
7     global const EQUIPAGE_MODES    = int( 1 )
8     global const EQUIPAGE_CASTA   = int( 2 )
9     global const EQUIPAGE_CASRA   = int( 3 )
10    global const EQUIPAGE_CASRESP = int( 4 )
11    global const EQUIPAGE_DAARESP = int( 5 )
12    # Active Collision Avoidance System (CAS) types
13    global const ACTIVE_CAS_TCAS = uint8( 0 )
14    global const ACTIVE_CAS_ACASX = uint8( 1 )
15    # Coordination Message Types
16    global const COORDINATION_NONE = uint32( 0 )
17    global const COORDINATION_TCAS = uint32( 1 )
18    global const COORDINATION_OCM = uint32( 2 )
19    # Surveillance Sources
20    global const SOURCE_MODES      = int( 0 )
21    global const SOURCE_MODEC      = int( 1 )
22    global const SOURCE_1090ES_ADSB = int( 2 )
23    global const SOURCE_NON_COOP   = int( 3 )
24    global const SOURCE_1090ES_ADSR = int( 4 )
25    # Gillham Encode/Decode constants
26    global const GILLHAM_NAR_CODE      = int( 0 )
27    global const GILLHAM_INVALID_VALUE = -9999.0
28    global const GILLHAM_NAR_VALUE     = NaN
29    global const MAX_ENCODED_ALT      = int( 2046 )
30    global const GILLHAM_MAX_BINARY_VALUE = int( 8 )
31    global const GILLHAM_TO_AC        = int( [8 10 12 2 4 6 7 9 11 1 3 5] )
32    global const AC_TO_GILLHAM       = int( [10 4 11 5 12 6 7 1 8 2 9 3] )
33    # RA Message Format (RMF) settings
34    global const RMF_TCAS          = uint8( 0 )
35    global const RMF_ACAS_Xa        = uint8( 1 )
36    global const RMF_ACAS_Xu        = uint8( 2 )
37    global const RMF_RESERVED       = uint8( 3 )
38    # Low-level Descend Inhibits
39    global const LDI_NONE          = uint8( 0 )
40    global const LDI_INCREASE_DESCEND = uint8( 1 )
41    global const LDI_DESCENDS      = uint8( 2 )
42    global const LDI_ALL           = uint8( 3 )
43    # Threat Type Indicator
44    global const TTI_ADDRESS       = uint8( 1 )
45    global const TTI_ALT_RNG_BRG  = uint8( 0 )
46    # TIDA constants
47    global const TIDA_MIN_ALTITUDE = -1000.0
48    global const TIDA_OFFSET        = -1000.0
49    global const TIDA_MAX_CODE      = uint32( 2047 )
50    # Active Validation State
51    global const AV_STATE_NOT_VALIDATED = int( 0 )
52    global const AV_STATE_VALID      = int( 1 )
53    global const AV_STATE_INVALID    = int( 2 )
54    global const AV_STATE_PROVISIONAL = int( 3 )
55    # TCAS Traffic Advisory Codes
56    global const TACODE_CLEAR       = int( 0 )
57    global const TACODE_PA          = int( 1 )
58    global const TACODE_TA_DEGRADED = int( 2 )
59    global const TACODE_TA NOMINAL = int( 3 )
60    global const TACODE_RA          = int( 4 )
61    global const COC = int( 1 )
62    # Force_alarm Settings for Display Logic
63    global const FORCE_ALARM_NONE = int( 0 )
64    global const FORCE_ALARM_ON    = int( 1 )
65    global const FORCE_ALARM_OFF   = int( 2 )

```

```

66     # Target Designations for Intruders
67     global const DESIGNATION_NONE          = uint8( 0 )
68     global const DESIGNATION_DESIGNATED_NO_ALERTS = uint8( 4 )
69     global const DESIGNATION_Xo_CSP03k      = uint8( 5 )
70     global const DESIGNATION_UNDESIGNATE_NO_ALERTS = uint8( 7 )
71     global const DESIGNATION_UNDESIGNATE_PROTECTION_MODE = uint8( 8 )
72     # TA/RA Processing of Intruders
73     global const RA_PROCESSING_NONE        = uint8( 0 )
74     global const RA_PROCESSING_DROPPED_TRACK = uint8( 1 )
75     global const RA_PROCESSING_GLOBAL_TARA   = uint8( 2 )
76     global const RA_PROCESSING_GLOBAL_TA_Only = uint8( 3 )
77     global const RA_PROCESSING_DEGRADED_SURVEILLANCE = uint8( 6 )
78     # Target Designation Indices
79     global const Xo_IDX_NONE              = uint8( 0 )
80     global const Xo_IDX_DNA               = uint8( 1 )
81     global const Xo_IDX_CSP03k           = uint8( 2 )
82     # Target Designation Status
83     global const Xo_STATUS_NONE          = uint8( 0x00 )
84     global const Xo_STATUS_DELAYED_ACTIVE_RA = uint8( 0x11 )
85     global const Xo_STATUS_SUSPENDED_MULTITHREAT = uint8( 0x21 )
86     global const Xo_STATUS_SUSPENDED_INVALID    = uint8( 0x22 )
87     global const Xo_STATUS_SUSPENDED_DROPPED    = uint8( 0x23 )
88     global const Xo_STATUS_SUSPENDED_TIMER     = uint8( 0x24 )
89     global const Xo_STATUS_UNDESIGNATED_UNAVAILABLE = uint8( 0x31 )
90     global const Xo_STATUS_UNDESIGNATED_GEOMETRIC = uint8( 0x32 )
91     global const Xo_STATUS_UNDESIGNATED_DROPPED   = uint8( 0x33 )
92     global const Xo_STATUS_UNDESIGNATED_TIMEOUT   = uint8( 0x34 )
93     global const Xo_STATUS_UNDESIGNATED_NO_BEARING = uint8( 0x35 )
94     # Target Designation Xo mode Availability for any intruders
95     global const Xo_AVAILABILITY_NOT_CONFIGURED = uint8( 0 )
96     global const Xo_AVAILABILITY_UNAVAILABLE_TO_RUN = uint8( 1 )
97     global const Xo_AVAILABILITY_AVAILABLE_TO_RUN = uint8( 2 )
98     global const Xo_AVAILABILITY_ON            = uint8( 3 )
99     # Degraded Surveillance Types Intruder (bit vector)
100    global const DEGRADED_SURVEILLANCE_NONE      = uint16( 0x0000 )
101    global const DEGRADED_SURVEILLANCE_NAR       = uint16( 0x0001 )
102    global const DEGRADED_SURVEILLANCE_ADSB_ONLY = uint16( 0x0002 )
103    global const DEGRADED_SURVEILLANCE_NO_BEARING = uint16( 0x0004 )
104    global const DEGRADED_SURVEILLANCE_REDUCED   = uint16( 0x0008 )
105    global const DEGRADED_SURVEILLANCE_VERT_COAST = uint16( 0x0010 )
106    global const DEGRADED_SURVEILLANCE_HORIZ_COAST = uint16( 0x0020 )
107    global const DEGRADED_SURVEILLANCE_INVALID_UF16UDS30 = uint16( 0x0080 )
108    global const DEGRADED_SURVEILLANCE_ALL        = uint16( 0xffff )
109    # Degraded Surveillance Types Ownship (bit vector)
110    global const DEGRADED_OWN_NONE              = uint16( 0x0000 )
111    global const DEGRADED_OWN_BAROALT_COAST    = uint16( 0x0001 )
112    global const DEGRADED_OWN_WGS84_COAST     = uint16( 0x0002 )
113    global const DEGRADED_OWN_HDG_COAST       = uint16( 0x0004 )
114    global const DEGRADED_OWN_HDG_INVALID     = uint16( 0x0008 )
115    # Ownship Heading States
116    global const OWN_HEADING_INVALID          = 0
117    global const OWN_HEADING_DEGRADED         = 1
118    global const OWN_HEADING NOMINAL          = 2
119    # RA Sense
120    global const RA_SENSE_NONE = uint8( 0 )
121    global const RA_SENSE_DOWN = uint8( 1 )
122    global const RA_SENSE_UP  = uint8( 2 )
123    # CAS Operational Mode Settings
124    global const OPMODE_STANDBY  = uint8( 1 )
125    global const OPMODE_TA      = uint8( 2 )
126    global const OPMODE_RA      = uint8( 3 )
127    # Sensitivity Level (SL) Settings
128    global const SENSITIVITY_LEVEL_AUTOMATIC = 0
129    global const SENSITIVITY_LEVEL_STANDBY   = 1
130    global const SENSITIVITY_LEVEL_TA_MODE   = 2

```

```
131     global const SENSITIVITY_LEVEL_RA_MODE    = 3
132     # Air-to-Air Reply Information
133     global const RI_FIELD_NONE      = int( 0 )
134     global const RI_FIELD_CASTA    = int( 2 )
135     global const RI_FIELD_CASRA    = int( 3 )
136     global const RI_FIELD_CASRA_HORZ = int( 4 )
137     # TCAS Version Indicator (from D0-185B)
138     global const TCAS_VERSION_INDICATOR_DEFAULT = int( 0 )
139     global const TCAS_VERSION_INDICATOR_FUTURE   = int( 1 )
140     global const ZLIMITL = 50.0
141     global const ZLIMITU = 60.0
142     # TCAS Sensitivity Level Boundaries
143     global const ZSL2T03 = 1100.0
144     global const ZSL3T02 = 900.0
145     global const RADARLOST = 10
146     # Mode C
147     global const MODE_C_FAILED_ASSOCIATION = int( -1 )
148     global const MODE_C_QUANT           = uint32( 100 )
149     # Correlation
150     global const CORRELATION_SUCCEEDED = int( 1 )
151     global const CORRELATION_PENDING   = int( 2 )
152     global const CORRELATION_FAILED   = int( 3 )
153     # Altitude boundaries for aircraft that have Non-Altitude Reporting Surveillance
154     global const NARS_THRESHOLD_MIN = ( -1200.0 )
155     global const NARS_THRESHOLD_MAX = ( 126750.0 )
156     # Surveillance Region Mode
157     global const SURVEILLANCE_REGION_NORMAL    = int( 0 )
158     global const SURVEILLANCE_REGION_REDUCED   = int( 1 )
159     global const SURVEILLANCE_REGION_HYBRID    = int( 2 )
160     # Altitude Buffer for TRM Crossing Determination
161     global const CROSSING_ALTITUDE_BUFFER = 100.0
162     # Display Arrow Indicators
163     global const DISPLAY_ARROW_LEVEL    = int( 0 )
164     global const DISPLAY_ARROW_CLIMB    = int( 1 )
165     global const DISPLAY_ARROW_DESCEND = int( -1 )
166     # CAS Coordination Capability/Type
167     global const CAS_ACTIVE_TCAS        = int( 0 )
168     global const CAS_ACTIVE_NON_TCAS   = int( 1 )
169     global const CAS_ACTIVE_NON_TCAS_OCM = int( 2 )
170     global const CASRESP_ACTIVE       = int( 3 )
171     global const CASRESP_PASSV_MODES  = int( 4 )
172     global const CASRESP_PASSV        = int( 5 )
173     # DAA Coordination Capability
174     global const DAA_RCV_NONE        = int( 0 )
175     global const DAA_RCV_ACTIVE      = int( 1 )
176     global const DAA_RCV_PASSV       = int( 2 )
177 end
```

This Page Intentionally Left Blank

Appendix E Data Structure Definitions

E.1 STM Output Data Structures

The algorithms contained within this document rely on a number of data structures to store and pass data within the STM. This appendix provides the definition, description and initial values for data structures used to output data from the STM. The TRMINPUT data structure is used as input by the TRM and, as such, is defined with the other TRM data structures used by the TRM.

Type 2 | StmReport The STM Report is the data structure produced by the STM to be partly consumed by the TRM to perform threat evaluation. The fields of this data structure include an exact input to the TRM, the data to be processed by the transponder, and the data to be sent to the pilot display.

```

1 type StmReport
2   trm_input::TRMInput(p. E-19)           # Input to the TRM
3   transponder::TransponderData(p. E-2)      # Ownship data for the Transponder
4   display::Vector{StmDisplayStruct(p. E-1)} # Intruder data for the display
5   StmReport( trm_input, transponder, display ) = new( trm_input, transponder, display )
6   StmReport() = new( TRMInput(p. E-19)(), TransponderData(p. E-2)(), StmDisplayStruct(p. E-1)[] )
7 end

```

Referenced In: AddADSBTrackToReport(p. 110), AdjustTargetDesignationValidity(p. 143), SetTargetDesignationModeAvailability(p. 145), SetDisplayDataPassive(p. 112), AddModeSTrackToReport(p. 102), AddTracksToReport(p. 98), GenerateStmReport(p. 97), AddModeCTrackToReport(p. 114), SetDisplayDataActive(p. 107), FilterTracksForTRM(p. 100)

Type 3 | StmDisplayStruct This data structure contains the intruder state data that is provided to the display and is part of the STM Report. It includes the vertical rate arrow for the display, indication that the intruder is reporting altitude, the intruder ID, relative altitude, tracked ground range, bearing relative to own airframe, and the intruder Mode S address.

```

1 type StmDisplayStruct
2   arrow::Z          # Display vertical rate arrow
3   alt_reporting::Bool # Indicates that the intruder is altitude reporting
4   bearing_valid::Bool # Indicates that a bearing is available for the intruder
5   id::UInt32         # Intruder ID inside the Target database
6   z_rel::R           # Relative altitude (feet)
7   r_ground::R        # Tracked ground-range (feet)
8   Chi_rel::R          # Bearing relative to own airframe (radians)
9   mode_s::UInt32       # Intruder 24-bit address
10  is_icao::Bool        # Indicates whether intruder address is ICAO compliant
11  StmDisplayStruct( arrow, alt_reporting, bearing_valid, id, z_rel, r_ground, Chi_rel, mode_s, is_icao ) =
12    new( arrow, alt_reporting, bearing_valid, id, z_rel, r_ground, Chi_rel, mode_s, is_icao )
13  StmDisplayStruct() =
14    new( 0, false, false, 0, 0.0, 0.0, 0.0, 0, false )
15 end

```

Referenced In: TRMIIntruderInput(p. E-20), SetDisplayDataPassive(p. 112), GenerateStmReport(p. 97), SetDisplayDataActive(p. 107), StmReport(p. E-1), FilterTracksForTRM(p. 100), CompareIntruderRange(p. 100)

Type 4 | TransponderData This data structure contains the ownship state data that is provided to the transponder. This includes the air to air reply information, ownship sensitivity level, version indicator, and data link subfield bits.

```
1 type TransponderData
2     ri::Z          # Air-to-air reply information
3     sl::Z          # Sensitivity level
4     vi::Z          # Version indicator
5     bit48::Bool    # Data link subfield bit 48 in MB
6     bit69::Bool    # Data link subfield bit 69 in MB
7     bit70::Bool    # Data link subfield bit 70 in MB
8     bit71::Bool    # Data link subfield bit 71 in MB
9     bit72::Bool    # Data link subfield bit 72 in MB
10    TransponderData( ri, sl, vi, bit48, bit69, bit70, bit71, bit72 ) = new( ri, sl, vi, bit48, bit69, bit70,
11                                bit71, bit72 )
11    TransponderData() = new( RI_FIELD_NONE, SENSITIVITY_LEVEL_AUTOMATIC, TCAS_VERSION_INDICATOR_DEFAULT,
12                                false, false, false, false, false )
12 end
```

Referenced In: OwnShipData(p. E-11), StmReport(p. E-1)

E.2 STM Internal Data Structures

The algorithms contained within this document rely on a number of data structures to store and pass data within the STM. This appendix provides the definition, description and initial values for data structures that are internal to the STM.

Type 5 | ActiveValidationHistory This data structure contains two Range Altitude data structures that represent a history of estimates from an active and a passive track. The two data structures are compared against one another during active validation.

```
1 type ActiveValidationHistory
2     active::RangeAltitude(p. E-12)
3     passive::RangeAltitude(p. E-12)
4     ActiveValidationHistory() = new(
5         RangeAltitude(p. E-12)(),
6         RangeAltitude(p. E-12)()
7     )
8 end
```

Referenced In: Target(p. E-14), AdvanceModeSTrackFile(p. 17), RemoveStaleADSBTrack(p. 117), RemoveStaleModeSTrack(p. 116), ActiveValidationUpdate(p. 121)

Type 6 | TrackFile The abstract type, Track File, represents the supertype of all other Track Files.

```
1 abstract TrackFile
```

Referenced In: RemoveDecorrelatedTrackFromTarget(p. C-8), CorrelateByType(p. C-17), CorrelateImage(p. C-18), ADSB-TrackFile(p. E-4), HypotheticalModeCTrackFile(p. E-7), CorrelateIndividualTracks(p. C-14), ExtrapolateTrack(p. C-2), ModeC-TrackFile(p. E-8), AddDecorrelatedTrackToTarget(p. C-8), GetTracksToDecorrelate(p. C-7), VerticalRateArrowUpdate(p. 108), TrackExists(p. 18), DecorrelateTargets(p. C-6), GetTrackID(p. C-8), GetTracksToExtrapolate(p. C-1), IsImageTrack(p. B-8), CorrelatePosition(p. C-21), CorrelateID(p. C-20), TrackMap(p. E-15), GenerateLeadTrackList(p. C-9), ModeSTrackFile(p. E-9)

Type 7 | ADSBTrackFile The ADS-B Track File contains the data elements required to maintain the state of a target aircraft using ADS-B Reports from an ADS-B report generator. The data structure includes two tracks, one for the vertical dimension and one for the horizontal dimension. Each track has their own time of applicability. A Track Summary is included for surveillance source correlation. Additional data fields determined through ADS-B Reports, such as TCAS operational status, reported ADS-B quality metrics, and an indication of a non-standard ICAO address are also stored in the data structure.

```

1 type ADSBTrackFile <: TrackFile(p.E-3)
2   modes::UInt32           # intruder address
3   non_icao::Bool          # intruder address is non-ICAO
4   rebroadcast::Bool        # track file is ADS-R
5   updates_pos::Z           # number of position updates
6   mu_hor::Vector{R}         # horizontal track in relative East-North ([meters, meters / second,
                                meters, meters/second])
7   Sigma_hor::Matrix{R}       # horizontal track covariance matrix
8   odc_hor::Z                # horizontal outlier detection count
9   init_velocity::Bool        # has horizontal velocity been initialized
10  mu_vert::Vector{R}         # vertical track ([feet, feet / second])
11  Sigma_vert::Matrix{R}       # vertical track covariance matrix
12  odc_vert::Z                # vertical outlier detection count
13  updates_vert::R           # number of vertical updates
14  valid_vert::Bool          # vertical track validity
15  quant::UInt32             # intruder vertical quantization (feet)
16  adsb_version::UInt32        # reported ads-b version number
17  nic::UInt32               # reported NIC value
18  nacp::UInt32              # reported NACp value
19  nacv::UInt32              # reported NACV value
20  sil::UInt32                # reported SIL value
21  sda::UInt32               # reported SDA value
22  trk_summary::TrackSummary(p.E-16)    # Track summary data structure used in correlation
23  toa_hor::R                 # ToA of horizontal track (seconds)
24  toa_vert::R                 # ToA of vertical track (seconds)
25  toa_pos_update::R          # ToA of last ADS-B position update (seconds)
26  toa::R                     # ToA of track file (seconds)
27  display_arrow_current::Z      # Current vertical rate arrow on display
28  vert_arrow_history::Vector{Z} # Previous vertical rate arrows sent to display
29  olat_at_hor_toa::R          # ownship latitude at time of horizontal toa (radians)
30  olon_at_hor_toa::R          # ownship longitude at time of horizontal toa (radians)
31  oalt_at_hor_toa::R          # ownship baro. altitude at time of horizontal toa (feet)
32  Xo_at_hor_toa::R            # ownship ECEF position at time of horizontal toa (meters)
33  Yo_at_hor_toa::R            # ownship ECEF position at time of horizontal toa (meters)
34  Zo_at_hor_toa::R            # ownship ECEF position at time of horizontal toa (meters)
35  ADSBTrackFile() = new(
36    0, false, false, 0,
37    zeros(4), zeros(4,4), 0, false,
38    zeros(2), zeros(2,2), 0, 0, false,
39    0, 0, 0, 0, 0, 0,
40    TrackSummary(p.E-16)(),
41    0.0, 0.0, 0.0, 0.0,
42    DISPLAY_ARROW_LEVEL, Array{Z,0},
43    0.0, 0.0, 0.0, 0.0, 0.0, 0.0
44  )
45 end

```

Referenced In: RedefineEstimateInRotatedFrame(p.58), ReceiveStateVectorPositionReport(p.49), InitializeADSBTrackFile(p.51), ExtrapolateADSBTrack(p.C-3), UpdateAdsbQualityHistory(p.65), CorrelateByType(p.C-17), Target(p.E-14), InitializeVerticalTracker(p.42), ExtrapolateTrack(p.C-2), AddDecorrelatedTrackToTarget(p.C-8), AddADSBTrackToDB(p.55), RefineENUHeight(p.59), AdvanceADSBTrackPosition(p.67), PropagateIntruderToToa(p.63), AdvanceVerticalTrack(p.44), GetTrackID(p.C-8), SetIntruderDegradedFlags(p.104), AddADSRTTrackToDB(p.55), AdvanceADSBTrackVelocity(p.68)

Type 8 | CorrelationHistory This data structure contains history of correlated/decorrelated tracks to be able to perform an M of N test. It saves the timing interval when the tracks correlated/decorrelated, and saves the IDs of the two tracks as an array of pairs.

```
1 type CorrelationHistory
2     timing::Vector{R}
3     id_pairs::Vector{Vector{Any}}
4     M::Z
5     N::Z
6     CorrelationHistory(id_type) = new(
7         Array(R,0),
8         Array(id_type,0),
9         1,
10        1
11    )
12 end
```

Referenced In: OwnShipData(p.E-11), CorrelationHistoryCheck(p.C-7), CorrelationHistoryUpdate(p.C-7), Decorrelate-Targets(p.C-6)
--

Type 9 | DesignationState This data structure contains state information associated with target designation. There is one Designation State structure for each Target. Along with specifics about the target designation state, it contains a flag indicating whether the target is currently designated. That flag is used for managing the associated Target. The timer values are in seconds and count up from 0 to the parameterized timer limit for a target to remain designated while designation is invalid due to geometric or data quality constraints. Validity for designation is indicated by the valid[Xo_IDX_DNA] and valid[Xo_IDX_CSPO3k] flags. The validity values output to the ASA processor are based, in part, on these validity flags. Status information intended for output to the ASA processor via the TRM is indicated in the status field.

```

1 type DesignationState
2   mode_s::UInt32          # Mode S address
3   valid::Vector{Bool}      # Valid or invalid for each Xo designation (see Xo_IDX_ constants)
4   active_dna::Bool         # Current Xo Designated No Alerts state (set by TRM)
5   active_protection_mode::UInt8 # Current protection mode (Xa, Xo CSPO-3000, ...)(set by TRM)
6   pending_dna::Bool        # Pending Xo Designated No Alerts state (set by STM)
7   pending_protection_mode::UInt8 # Pending protection mode (Xa, Xo CSPO-3000, ...)(set by STM)
8   timer_dna::Z             # Timer for invalid timeout for Xo Designated No Alerts
9   timer_protection_mode::Z # Timer for invalid timeout for Xo protection mode
10  is_designated::Bool       # In use, don't delete target if all tracks are lost
11  was_dropped::Bool        # No output to TRM for processing and no STM display output
12  designated_mode::UInt8   # Designation setting for output
13  status::UInt8            # Status to be published to ASA processor (see Xo_STATUS_ constants)
14  active_ra::Bool          # This target has an RA active
15  multithreat_ra::Bool     # This target is part of multithreat RA
16  is_airborne::Bool        # Has gone above upper no aurals threshold
17  is_landing::Bool         # Has gone below lower no aurals threshold when airborne
18  was_proximate::Bool      # Whether target has been within 6nmi while designated
19  r_ground_history::Vector{R} # Recent history of distances to target while designated (nm)
20  DesignationState() = new(
21    0,
22    fill(false,2),
23    false, uint8(1),
24    false, uint8(1),
25    0, 0,
26    false,
27    false,
28    uint8(0), uint8(0),
29    false, false,
30    false, false, false,
31    R[]
32  )
33  DesignationState( mode_s::UInt32 ) = new(
34    mode_s,
35    fill(false,2),
36    false, uint8(1),
37    false, uint8(1),
38    0, 0,
39    false,
40    false,
41    uint8(0), uint8(0),
42    false, false,
43    false, false, false,
44    R[]
45  )
46 end

```

Referenced In: SetPendingTargetDesignation(p. 75), IsTargetDesignationActive(p. 141), MergeTargetDesignations(p. C-16), Target(p. E-14), UpdateDNAValidity(p. 131), AutomaticallyUndesignateDNA(p. 131), SetTargetDesignation(p. 125), IsCSPO3000ModeUnavailableToRun(p. 146), SetTargetDesignationModeAvailability(p. 145), UpdateTargetDesignationValidity(p. 127), UpdateDNABasedOnRange(p. 134), UpdateCSPO3000Validity(p. 129), CheckTargetDesignationTimers(p. 135), AutomaticallyUndesignateCSPO3000(p. 130), UpdateDroppedTargetDesignationState(p. 144), UpdateDNABasedOnAltitude(p. 132), SetTargetDesignationInvalid(p. 139), SetIntruderDesignationData(p. 137)

Type 10 | HypotheticalModeCTrackFile

```

1 type HypotheticalModeCTrackFile <: TrackFile(p. E-3)
2   replies::Vector{ModeCReply(p. E-7)}
3   rangerate::R # feet / second
4   range::R # feet
5   altitude::R # feet
6   Chi_rel::R # radians
7   toa::R # seconds
8   age::R # seconds
9   HypotheticalModeCTrackFile() = new( Array(ModeCReply(p. E-7),0), 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 )
10 end

```

Referenced In: HypotheticalTrackTest(p.B-14), RangeSquared(p.B-8)
--

Type 11 | ModeCIntervals

```

1 type ModeCIntervals
2   In1::Vector{ModeCReply(p. E-7)}
3   In2::Vector{ModeCReply(p. E-7)}
4   In3::Vector{ModeCReply(p. E-7)}
5   ModeCIntervals() = new(
6     Array(ModeCReply(p. E-7),0),
7     Array(ModeCReply(p. E-7),0),
8     Array(ModeCReply(p. E-7),0),
9   )
10 end

```

Type 12 | ModeCReply A Mode C Reply contains the measured slant-range, relative bearing, reported altitude code (decoded using Gillham Gray code), altitude confidence bits, an optional externally set track ID, and the time of applicability associated with a single reply from a Mode C transponder.

```

1 type ModeCReply
2   external_ID::UInt32 # Optional external ID set by front-end Mode C tracker
3   coded_alt::Z # Gillham code (feet once decoded)
4   conf::Z # confidence
5   r_slant::R # slant range to target (feet)
6   Chi_rel::R # Relative bearing to target (radians)
7   toa::R # Time of Applicability (seconds)
8   ModeCReply() = new( 0, 0, 0, 0.0, 0.0, 0.0 )
9 end

```

Referenced In: InitializeModeCTrack(p.23), ReceiveModeCReplies(p.B-2), AssociateModeCtoTarget(p.B-5), HypotheticalModeCTrackFile(p.E-7), UpdateModeCTrack(p.24), ModeCIntervals(p.E-7), PromoteModeCTracks(p.B-10), HypotheticalTrackTest(p.B-14), DetermineModeCReplyAssociationScore(p.B-7), FormHypotheticalTracks(p.B-12), MergeModeCReplies(p.B-3), IntervalMaintenance(p.B-16), EstablishModeCTrack(p.B-11), ReceiveModeCReply(p.22)

Type 13 | ModeCTrackFile The Mode C Track File contains the data elements required to maintain the state of a target aircraft using Mode C reply messages. The data structure includes three tracks, one for the vertical dimension and two for the horizontal dimension. Each track has their own time of applicability. A Track Summary is included for surveillance source correlation. Additional data fields, such as an optional externally set track ID, the altitude quantization information, sensitivity level, and if it is an image track, are also stored in the data structure.

```

1 type ModeCTrackFile <: TrackFile(p. E-3)
2   external_ID::UInt32
3   mu_cart::Vector{R}           # Cartesian track in relative E-N ([feet, feet, feet / second, feet /
4     second])
5   Sigma_cart::Matrix{R}        # Cartesian track covariance matrix
6   odc_cart::Z                 # Cartesian track outlier detection count
7   updates_cart::Z             # Cartesian track update count
8   valid_cart::Bool            # Cartesian track validity
9   mu_rng::Vector{R}           # Ground range track ([feet, feet / second, feet / second^2])
10  Sigma_rng::Matrix{R}        # Ground range track covariance matrix
11  odc_rng::Z                 # Ground range track outlier detection count
12  updates_rng::Z             # Ground range track update count
13  valid_rng::Bool            # Ground range track validity
14  mu_vert::Vector{R}          # Vertical track ([feet, feet / second])
15  Sigma_vert::Matrix{R}       # Vertical track covariance matrix
16  odc_vert::Z                # Vertical track outlier detection count
17  updates_vert::Z            # Vertical track update count
18  valid_vert::Bool           # Vertical track validity
19  quant::UInt32              # Vertical track quantization (feet)
20  sensitivity_index::UInt32  # Sensitivity level of intruder
21  is_image::Bool              # Track file is marked as Mode C image track
22  trk_summary::TrackSummary(p. E-16)      # Track summary data structure used in correlation
23  last_update_rng::LastUpdateRng(p. E-17)  # Previous range observation
24  is_rng_coast::Bool          # Flag to indicate range track is coasting
25  alt_not_avail::Bool         # Flag to indicate altitude not available
26  toa_vert::R                 # Vertical track ToA (seconds)
27  toa_cart::R                 # Cartesian track ToA (seconds)
28  toa_rng::R                  # Range track ToA (seconds)
29  toa::R                      # Track file ToA (seconds)
30  display_arrow_current::Z    # Current vertical rate arrow on display
31  vert_arrow_history::Vector{Z} # Previous vertical rate arrows sent to display
32  on_ground::Bool             # Flag to indicate target is on ground
33  vert_history::VertHistory(p. E-12)      # Data structure used in on-ground determination
34  ModeCTrackFile() = new(
35    0,
36    fill(NaN,4), fill(NaN,4,4), 0, 0, false,
37    fill(NaN,3), fill(NaN,3,3), 0, 0, false,
38    fill(NaN,2), fill(NaN,2,2), 0, 0, false,
39    MODE_C_QUANT, SENSITIVITY_LEVEL_AUTOMATIC, false,
40    TrackSummary(p. E-16)(), LastUpdateRng(p. E-17)(), false, false,
41    0.0, 0.0, 0.0, 0.0,
42    DISPLAY_ARROW_LEVEL, Array(Z,0),
43    true, VertHistory(p. E-12)()
44  )
44 end

```

Referenced In:	InitializeModeCTrack(p. 23), EvaluateOnGroundModeC(p. B-17), CorrelateByType(p. C-17), InitializeRangeTracker(p. 35), AssociateModeCtoTarget(p. B-5), Target(p. E-14), AdvanceRangeTrack(p. 37), AddModeCTrackToDB(p. 23), InitializeVerticalTracker(p. 42), CorrelateIndividualTracks(p. C-14), ReinitializeRangeTracker(p. 38), UpdateModeCTrack(p. 24), ExtrapolateTrack(p. C-2), AddDecorrelatedTrackToTarget(p. C-8), SortByRangeSquared(p. B-8), ExtrapolateActiveTrack(p. C-2), DecorrelateTargets(p. C-6), AssociateModeCExternalIDtoTrack(p. 22), DetermineModeCReplyAssociationScore(p. B-7), AdvanceVerticalTrack(p. 44), EvaluateAirToGroundTransitionModeC(p. B-19), RangeSquared(p. B-8), GetTrackID(p. C-8), SetIntruderDegradedFlags(p. 104), IsImageTrack(p. B-8), CorrelatePosition(p. C-21), EvaluateGroundToAirTransitionModeC(p. B-18), EstablishModeCTrack(p. B-11), AdvanceCartesianTrack(p. 31), InitializeCartesianTracker(p. 28), ReceiveModeCReply(p. 22)
-----------------------	--

Type 14 | ModeSTrackFile The Mode S Track File contains the data elements required to maintain the state of a target aircraft using DF0 reply messages elicited through transponder based surveillance. The data structure includes three tracks, one for the vertical dimension and two for the horizontal dimension. Each track has their own time of applicability. A Track Summary is included for surveillance source correlation. Additional data fields determined through a DF0 reply message, such as equipage information and sensitivity level, are also stored in the data structure.

```

1 type ModeSTrackFile <: TrackFile(p.E-3)
2   modes::UInt32
3   ri::UInt32
4   alt_not_avail::Bool
5   mu_cart::Vector{R}           # Cartesian track in relative E-N ([feet, feet, feet / second, feet
6   / second])
7   Sigma_cart::Matrix{R}        # Cartesian track covariance matrix
8   odc_cart::Z                 # Cartesian track outlier detection count
9   updates_cart::Z             # Cartesian track update count
10  valid_cart::Bool            # Cartesian track validity
11  mu_rng::Vector{R}           # Ground range track ([feet, feet / second, feet / second^2])
12  Sigma_rng::Matrix{R}        # Ground range track covariance matrix
13  odc_rng::Z                 # Ground range track outlier detection count
14  updates_rng::Z             # Ground range track update count
15  valid_rng::Bool            # Ground range track validity
16  mu_vert::Vector{R}          # Vertical track ([feet, feet / second])
17  Sigma_vert::Matrix{R}        # Vertical track covariance matrix
18  odc_vert::Z                # Vertical track outlier detection count
19  updates_vert::Z            # Vertical track update count
20  valid_vert::Bool            # Vertical track validity
21  quant::UInt32               # Vertical track quantization (feet)
22  surv_mode::UInt32           # Current surveillance region
23  trk_summary::TrackSummary(p.E-16) # Track summary data structure for correlation
24  last_update_rng::LastUpdateRng(p.E-17) # Previous range observation
25  is_rng_coast::Bool          # Flag to indicate range track coasting
26  toa_vert::R                 # Vertical track ToA (seconds)
27  toa_cart::R                 # Cartesian track ToA (seconds)
28  toa_rng::R                  # Range track ToA (seconds)
29  toa::R                      # Track file ToA (seconds)
30  display_arrow_current::Z    # Currently vertical rate arrow on display
31  vert_arrow_history::Vector{Z} # Previously displayed vertical rate arrows
32  ModeSTrackFile() = new(
33    0, 0, false,
34    fill(NaN,4), fill(NaN,4,4), 0, 0, false,
35    fill(NaN,3), fill(NaN,3,3), 0, 0, false,
36    fill(NaN,2), fill(NaN,2,2), 0, 0, false,
37    0, SURVEILLANCE_REGION_NORMAL,
38    TrackSummary(p.E-16)(), LastUpdateRng(p.E-17)(), false,
39    0.0, 0.0, 0.0, 0.0,
40    DISPLAY_ARROW_LEVEL, Array(Z,0)
41  )
41 end

```

Referenced In: InitializeModeSTrackFile(p.16), CorrelateByType(p.C-17), InitializeRangeTracker(p.35), Target(p.E-14), AdvanceRangeTrack(p.37), AddADSBTrackToReport(p.110), AdvanceModeSTrackFile(p.17), InitializeVerticalTracker(p.42), ReinitializeRangeTracker(p.38), ExtrapolateTrack(p.C-2), ExtrapolateActiveTrack(p.C-2), AdvanceVerticalTrack(p.44), SetIntruderDegradedFlags(p.104), ReceiveDF0(p.15), AddModeSTrackToDB(p.20), AdvanceCartesianTrack(p.31), InitializeCartesianTracker(p.28)

Type 15 | OwnDiscreteData This data structure contains all data elements related to certain ownship parameters that change infrequently or only upon initialization. The discrete values are used to set the sensitivity level of ownship. They include the own Mode S address, the Mode A code, a flag to indicate the system is operational, a manually selected sensitivity level, a flag to indicate if the display is allowed on the ground, a flag to inhibit ownship from issuing climb advisories, a flag to inhibit ownship from issuing increase climb advisories, and a flag to indicate if ownship altitude is coarsely quantized at 100 feet.

```

1 type OwnDiscreteData
2   mode_s:UInt32          # mode S address of ownship
3   modeACode:UInt32        # mode A code for ownship
4   opflg:Bool             # system operational
5   manualSL:UInt32        # manual sensitivity level input from pilot
6   own_ground_display_mode_on:Bool # flag to use display (allow active interrogation) while on ground
7   on_surface:Bool         # flag to indicate that aircraft is operating on surface (vs. taking off / airborne)
8   aoto_on:Bool           # flag to indicate that TAs against ADS-B only intruders (AOT0) are enabled
9   is_coarsely_quant:Bool # indicates if ownship altitude is 100 foot quantized
10  OwnDiscreteData() = new( 0, 0, true, SENSITIVITY_LEVEL_AUTOMATIC, false, false, true, false )
11  OwnDiscreteData(modes, modeACode, opflg, manualSL, own_ground_display_mode_on, on_surface,
12    is_coarsely_quant) = new(modes, modeACode, opflg, manualSL, own_ground_display_mode_on, on_surface,
13    is_coarsely_quant)
12 end
```

Referenced In: OwnShipData(p. E-11)
--

Type 16 | OwnShipData This data structure contains all data elements related to ownship state as derived from several sensor sources. This includes a track of ownship vertical state from barometric altitude measurements, a track of ownship heading, ownship radio altitude, ownship WGS84 position, information related to the advisory modes (TA only mode and TA/RA mode), interrogation mode, current ownship sensitivity level, previous ownship sensitivity level, externally determined sensitivity level, indication if ownship is on the ground, number of invalid radio altitude measurements, the time the report was generated, the threat resolution advisory complement array, the number of display structures, target independent correlation history, target independent decorrelation history, ownship history of heading and barometric altitude, ownship discrete data, and ownship transponder data.

```

1 type OwnShipData
2   modes::UInt32
3   radalt::R          # ownship radar altitude (feet)
4   mu_heading::Array{R} # ownship heading track ([radians, radians / second])
5   Sigma_heading::Matrix{R} # ownship heading track covariance
6   toa_heading::R       # ownship heading track time of applicability (seconds)
7   heading_initialized::Bool # ownship heading initialized
8   heading_state::Z      # ownship heading track state (see constants)
9   heading_ocd::Z        # ownship heading track outlier detection count
10  mu_h::Array{R}         # ownship barometric altitude track ([feet, feet / second])
11  Sigma_h::Matrix{R}    # ownship barometric altitude track covariance
12  toa_h::R              # ownship barometric altitude track time of applicability (seconds)
13  wgs84_lat::R          # ownship WGS84 latitude (degrees)
14  wgs84_lon::R          # ownship WGS84 longitude (degrees)
15  wgs84_vel_ew::R       # ownship WGS84 east-west velocity (knots)
16  wgs84_vel_ns::R       # ownship WGS84 north-south velocity (knots)
17  wgs84_toa::R          # ownship WGS84 time of applicability (seconds)
18  prev_rpt_toa::R       # ownship WGS84 previous report time of applicability (seconds)
19  trm_opmode::Z          # own TRM operating mode
20  sensitivity_level::Z  # own sensitivity level
21  on_ground::Bool        # own aircraft on ground flag
22  invalid_radalt_cycles::Z # number of cycles without valid radar alt. data
23  received_vrcs::Vector{Bool} # threat resolution advisory compl. array
24  corr_history::CorrelationHistory(p.E-5) # history of correlated tracks
25  corr_image_history::CorrelationHistory(p.E-5) # history of correlated tracks
26  decorr_history::CorrelationHistory(p.E-5) # history of decorrelated tracks
27  history::OwnHistory(p.E-12) # history of heading and baroalt observations
28  discrete::OwnDiscreteData(p.E-10) # Data structure of discrete inputs
29  transponder::TransponderData(p.E-2) # Data structure of transponder outputs
30  OwnShipData() = new(
31    0, NaN,
32    zeros(2), zeros(2,2), NaN, false, OWN_HEADING_INVALID, 0,
33    zeros(2), zeros(2,2), NaN,
34    NaN, NaN, NaN, NaN, NaN, NaN,
35    OPMODE_STANDBY,
36    SENSITIVITY_LEVEL_RA_MODE,
37    false, 0,
38    fill(false,4),
39    CorrelationHistory(p.E-5)(Z),
40    CorrelationHistory(p.E-5)(Z),
41    CorrelationHistory(p.E-5)(String),
42    OwnHistory(p.E-12)(),
43    OwnDiscreteData(p.E-10)(),
44    TransponderData(p.E-2)()
45  )
46 end

```

Type 17 | OwnHistory This data structure contains a history of bearing and barometric ownship altitude measurements. The values are used for interpolating past ownship information when receiving DF0 messages from the past.

```

1 type OwnHistory
2   heading::ValueTime(p.E-16)      # own heading observation ([radians], [seconds])
3   baroalt::ValueTime(p.E-16)       # own barometric observation ([radians], [seconds])
4   OwnHistory() = new(
5     ValueTime(p.E-16)(),
6     ValueTime(p.E-16)()
7   )
8 end

```

Referenced In: OwnShipData(p.E-11)

Type 18 | VertHistory This data structure contains a history of the vertical dimensions of both received observations and track estimates. It is used in the Mode C recommended implementation of intruder on-ground determination.

```

1 type VertHistory
2   meas::ValueTime(p.E-16)
3   track::ValueTime(p.E-16)
4   VertHistory() = new(
5     ValueTime(p.E-16)(),
6     ValueTime(p.E-16)()
7   )
8 end

```

Referenced In: ModeCTrackFile(p.E-8)

Type 19 | RangeAltitude This data structure contains three arrays which represent a history of range and altitude estimates. The values contained in these arrays are used for the purpose of active validation.

```

1 type RangeAltitude
2   range::Vector{R}    # range (ft)
3   altitude::Vector{R} # altitude (ft)
4   RangeAltitude() = new(
5     Array(R,0),
6     Array(R,0)
7   )
8 end

```

Referenced In: ActiveValidationHistory(p.E-3)

Type 20 | ReceivedCoordinationData This data structure contains coordination data that has been received from a target aircraft. It includes the Cancel Vertical Resolution Advisory Complement (CVC), Vertical Resolution Advisory Complement (VRC), Vertical Sense Bits (VSB), and the time of applicability (TOA).

```
1 type ReceivedCoordinationData
2     vrc::UInt32          # Vertical RA complement
3     toa::R                # Time of Applicability
4     ReceivedCoordinationData() = new( 0, 0.0 )
5 end
```

Referenced In: Target(p. E-14), RemoveStaleModeSTrack(p. 116), CoordinationTimeoutCheck(p. 89)

Type 21 | Target A Target represents all data related to a single physical aircraft that is under surveillance by ACAS X. The Target data structure contains a single Mode S Track File, ADS-B Track File, and ADS-R Track File as well as multiple Mode C Track Files. It also contains the Received Coordination Data from the associated aircraft, Active Validation History information used to validate an ADS-B Track before it is presented to the TRM, the state of the Active Validation, ADS-B Quality History information, a flag to override the ADS-B Quality Check used for Hybrid Surveillance targets, the target designation state, and the time of initialization.

```

1 type Target
2   modes_track::Union(ModeSTrackFile(p. E-9), Nothing)
3   adsb_track::Union(ADSBTrackFile(p. E-4), Nothing)
4   adsr_track::Union(ADSBTrackFile(p. E-4), Nothing)
5   modec_tracks::Vector{ModeCTrackFile(p. E-8)}
6   coord_data::ReceivedCoordinationData(p. E-13)
7   bad_uf16uds30::Bool
8   av_history::ActiveValidationHistory(p. E-3)
9   av_state::Z
10  adsb_qual_history::Vector{Bool}
11  passive_only_adsb_qual_history::Vector{Bool}
12  adsb_qual_override::Bool
13  designation_state::DesignationState(p. E-6)
14  init_time::R
15  ca_operational::UInt8
16  type_capability::UInt8
17  daa::UInt8
18  Target() = new(
19    nothing,
20    nothing,
21    nothing,
22    Array(ModeCTrackFile(p. E-8), 0),
23    ReceivedCoordinationData(p. E-13)(),
24    false,
25    ActiveValidationHistory(p. E-3)(),
26    AV_STATE_NOT_VALIDATED,
27    Array(Bool, 0),
28    Array(Bool, 0),
29    false,
30    DesignationState(p. E-6)(),
31    -1.0,
32    0,
33    0,
34    0
35  )
36  Target( mode_s::UInt32 ) = new(
37    nothing,
38    nothing,
39    nothing,
40    Array(ModeCTrackFile(p. E-8), 0),
41    ReceivedCoordinationData(p. E-13)(),
42    false,
43    ActiveValidationHistory(p. E-3)(),
44    AV_STATE_NOT_VALIDATED,
45    Array(Bool, 0),
46    Array(Bool, 0),
47    false,
48    DesignationState(p. E-6)( mode_s ),
49    -1.0,
50    0,
51    0,
52    0
53  )
54 end

```

Referenced In: RemoveDecorrelatedTrackFromTarget(p. C-8), UpdateAdsbQualityHistory(p. 65), HasValidModeCTrack(p. 99), AdsbQualityCheck(p. 111), MergeTargetDesignations(p. C-16), AddModeCTrackToDB(p. 23), AdvanceModeSTrackFile(p. 17), ReceiveTargetDesignation(p. 73), AdjustTargetDesignationValidity(p. 143), RemoveStaleADS-BTrack(p. 117), AddDecorrelatedTrackToTarget(p. 8), AddADSBTrackToDB(p. 55), RemoveStaleModeSTrack(p. 116), DecorrelateTargets(p. C-6), RemoveStaleADSRTTrack(p. 118), StmHousekeepingTargetDesignation(p. A-2), GetTrackID(p. C-8), SetIntruderDegradedFlags(p. 104), GetTracksToExtrapolate(p. C-1), RemoveStaleModeCTracks(p. 118), ActiveValidationUpdate(p. 121), AddModeSTrackToDB(p. 20), TargetIsEmpty(p. 119), CoordinationTimeoutCheck(p. 89), ActiveValidationCheck , AddADSRTTrackToDB(p. 55)

Type 22 | Database A Database acts as a wrapper around the Julia Dictionary type (Dict) with the addition of an increment field. The increment is used as a deterministic internal ID for new entries into the dictionary.

```
1 type Database{K,V}
2     increment::K
3     dictionary::Dict{K,V}
4     Database(i)::K = new(i, Dict{K,V}())
5 end
6 Base.delete!(db::Database, k)      = Base.delete!(db.dictionary, k)
7 Base.get(db::Database, k, default) = Base.get(db.dictionary, k, default)
8 Base.getindex(db::Database, k)     = Base.getindex(db.dictionary, k)
9 Base.haskey(db::Database, k)      = Base.haskey(db.dictionary, k)
10 Base.keys(db::Database)          = Base.sort(Base.collect(Base.keys(db.dictionary)))
11 Base.setindex!(db::Database, v, k) = Base.setindex!(db.dictionary, v, k)
```

Referenced In: AddToDB(p. 20), RetrieveWithID(p. 20)

Type 23 | TrackMap A Track Map associates a Track File with an ID in the Target Database. The purpose is to organize which Target the specified Track belongs to during the correlation process.

```
1 type TrackMap
2     track::Union(TrackFile(p. E-3), Nothing)
3     db_id::UInt32
4 end
```

Referenced In: SortByRange(p. C-13), SortByRangeSquared(p. B-8), DecorrelateTargets(p. C-6), PromoteModeCTracks(p. B-10), CorrelateTargets(p. C-12), GenerateLeadTrackList(p. C-9)

Type 24 | TrackSummary This data structure contains three tracks, one for the vertical dimension and two for the horizontal dimension. It also contains validity flags for each track. It's used as a common format for different surveillance sources to be correlated.

```

1 type TrackSummary
2   toa                      # ToA (seconds)
3   mu_vert                  # Vertical track ([feet, feet/second])
4   mu_rng_az                # Polar track ([feet, radians])
5   mu_range                 # Ground range track ([feet, feet/second])
6   Sigma_vert               # Vertical track covariance matrix
7   Sigma_rng_az             # Polar track covariance matrix
8   Sigma_range               # Ground range track covariance matrix
9   valid_rng::Bool           # Ground range track validity
10  valid_vert::Bool          # Vertical track validity
11  valid_rng_az::Bool        # Polar track validity
12  TrackSummary() = new(
13    0,
14    zeros(2),
15    zeros(2),
16    zeros(2),
17    zeros(2,2),
18    zeros(2,2),
19    zeros(2,2),
20    false,
21    false,
22    false
23  )
24 end
```

Referenced In:	ExtrapolateADSBTrack(p.C-3), ADSBTrackFile(p.E-4), ModeCTrackFile(p.E-8), ExtrapolateActive-Track(p.C-2), DetermineModeCReplyAssociationScore(p.B-7), CalculateImageRangeRate(p.C-19), ModeSTrackFile(p.E-9)
-----------------------	--

Type 25 | ValueTime A ValueTime pair stores arrays of specified values with associated times. It is used as a means of storing ownship history.

```

1 type ValueTime
2   value::Vector{R}
3   time::Vector{R}
4   ValueTime() = new(
5     Array(R,0),
6     Array(R,0)
7   )
8 end
```

Referenced In:	ReceiveBaroAltObservation(p.76), OwnHistory(p.E-12), UpdateHistory(p.77), VertHistory(p.E-12)
-----------------------	---

Type 26 | LastUpdateRng

```
1 type LastUpdateRng
2     r_slant::R      # Slant range to target (ft)
3     z_rel::R        # Relative altitude of target (ft)
4     z_baro::R       # Barometric altitude (ft)
5     toa::R          # Time of Applicability (s)
6     measurement_invalid::Bool
7     LastUpdateRng() = new(
8         NaN,
9         NaN,
10        NaN,
11        NaN,
12        false
13    )
14 end
```

Referenced In: ModeCTrackFile _(p. E-8) , ModeSTrackFile _(p. E-9)

E.3 STM Output-TRM Input Data Structures

These data structures contain information generated by the STM and consumed by the TRM. The TRM input data structure is contained within the STM Report data structure, produced by the STM.

Type 27 | TRMInput This data structure is generated as part of the STM Report and gets directly processed by the TRM for threat evaluation. It includes information about ownership, including an estimate of ownership heading, an estimate of ownership radio altitude, and a distribution of ownership vertical state (derived from barometric altitude). It also includes a collection of data structures that represent the tracked state of nearby intruding aircraft.

```

1 type TRMInput
2   own::TRMOwnInput(p.E-19) # Input data for own aircraft
3   intruder::Vector{TRMIIntruderInput(p.E-20)}
4           # Input data for each intruder
5 #
6   TRMInput() = new( TRMOwnInput(p.E-19)(), [] )
7   TRMInput( own::TRMOwnInput(p.E-19), intruder::Vector{TRMIIntruderInput(p.E-20)} ) = new( own, intruder )
8 end

```

Referenced In: VerticalTRMUpdate(p.152), VerticalTRMUpdatePrep(p.1-2), StmReport(p.E-1)
--

Type 28 | TRMOwnInput This data structure is generated as part of the STM Report and represents the state of ownership. It includes radio altitude and heading, identifying information needed for the TRM outputs, operational mode (as sensitivity level), surveillance quality, and availability of Xo modes. The estimated vertical state of ownership is represented by a collection of own vertical beliefs.

```

1 type TRMOwnInput
2   h::R                      # Radar altitude, above ground (ft)
3   psi::R                     # Heading (radians)
4   mode_s::UInt32             # Mode S address
5   mode_a::UInt32             # Mode A ID
6   opmode::UInt8              # Ownship operational mode (See OPMODE_ constants)
7   degraded_own_surveillance::UInt16 # Bit vector indicating quality of surveillance
8                   # (See DEGRADED_OWN_ constants)
9   xo_availability::Vector{UInt8} # Availability of Xo modes (see Xo_AVAILABILITY constants)
10  belief_vert::Vector{OwnVerticalBelief(p.E-21)} # Own ship vertical beliefs (z, dz)
11 #
12  TRMOwnInput( h::R, psi::R, mode_s::UInt32, mode_a::UInt32, opmode::UInt8, degraded_own_surveillance:::
13      UInt16, xo_availability::Vector{UInt8}, belief_vert::Vector{OwnVerticalBelief(p.E-21)} ) =
14      new( h, psi, mode_s, mode_a, opmode, degraded_own_surveillance, xo_availability, belief_vert )
15  TRMOwnInput( h::R, psi::R, mode_s::UInt32, mode_a::UInt32, opmode::UInt8, belief_vert::Vector{
16      OwnVerticalBelief(p.E-21)} ) =
17      new( h, psi, mode_s, mode_a, opmode, DEGRADED_OWN_NONE,
18          fill(Xo_AVAILABILITY_NOT_CONFIGURED,2), belief_vert )
19  TRMOwnInput() =
20      new( 0.0, 0.0, 0, 0, uint8(OPMODE_RA), DEGRADED_OWN_NONE,
21          fill(Xo_AVAILABILITY_NOT_CONFIGURED,2), OwnVerticalBelief(p.E-21)[ ] )
20 end

```

Referenced In: VerticalTRMUpdate(p.152), TRMInput(p.E-19), GenerateTRMOutput(p.291), StateAndCostEstimation(p.161), GetVTRMOwnData(p.153), IntruderPrep(p.165), IsIntruderMaster(p.334)
--

Type 29 | TRMIntruderInput This data structure is generated as part of the STM Report and represents the tracked state of a nearby intruding aircraft. It includes information for identifying the intruder, coordination information, surveillance type and quality, Xo mode information, and the intruder's vertical and horizontal position and velocity as belief states. The STM display output is included for use in forming resolution advisory messages for the ground station.

```

1 type TRMIntruderInput
2   valid::Bool          # Data is valid for processing
3   id::UInt32           # Unique target identifier
4   address::UInt32      # ICAO or non-ICAO address
5   is_icao::Bool         # Is an ICAO address (if not, use AltRngBrg for TID)
6   vrc::UInt32           # VRC received from this intruder
7   equipage::Z           # Intruder equipage (See EQUIPAGE_ constants)
8   active_cas_version::UInt8 # Type of active CAS (see ACTIVE_CAS_ constants)
9   coordination_msg::UInt32 # Coordination message type (see COORDINATION_ constants)
10  source::Z             # Surveillance source (See SOURCE_ constants)
11  degraded_surveillance::UInt16 # Bit vector indicating quality of surveillance
12    # (See DEGRADED_SURVEILLANCE_ constants)
13  is_proximate::Bool     # Whether proximate to own ship
14  designated_mode::UInt8 # Selected designation, may not be active (See DESIGNATION_ constants)
15  protection_mode::UInt8 # Protection mode to be applied (See PROTECTION_MODE_ constants)
16  dna::Bool              # Designated for Xo Designated No Alerts (DNA) processing, or not
17  xo_valid::Vector{Bool}  # Validity for Xo designations (initial settings for output)
18  xo_status::UInt8        # Xo designation status information (See Xo_STATUS_ constants)
19  stm_display::StmDisplayStruct(p.E-1) # Target display information output by the STM
20  belief_vert::Vector{IntruderVerticalBelief(p.E-21)}
21    # Vertical beliefs (z, dz)
22  belief_horiz::Vector{IntruderHorizontalBelief(p.E-21)}
23    # Horizontal beliefs (relative x, y, dx, dy)
24 #
25  TRMIntruderInput( valid::Bool, id::UInt32, address::UInt32, is_icao::Bool, vrc::UInt32, equipage::Z,
26    active_cas_version::UInt8, coordination_msg::UInt32, source::Z,
27    degraded_surveillance::UInt16, is_proximate::Bool,
28    designated_mode::UInt8, protection_mode::UInt8, dna::Bool,
29    xo_valid::Vector{Bool}, xo_status::UInt8,
30    stm_display_struct::StmDisplayStruct(p.E-1),
31    belief_vert::Vector{IntruderVerticalBelief(p.E-21)},
32    belief_horiz::Vector{IntruderHorizontalBelief(p.E-21)} ) =
33    new( valid, id, address, is_icao, vrc, equipage,
34      active_cas_version, coordination_msg, source,
35      degraded_surveillance, is_proximate, designated_mode, protection_mode, dna,
36      xo_valid, xo_status,
37      stm_display_struct, belief_vert, belief_horiz )
38  TRMIntruderInput() =
39    new( false, 0, 0, false, 0, EQUIPAGE_ATCRBS,
40      ACTIVE_CAS_TCAS, COORDINATION_NONE, SOURCE_MODES,
41      DEGRADED_SURVEILLANCE_NONE, false, DESIGNATION_NONE, PROTECTION_MODE_Xa, false,
42      fill( false, 2 ), Xo_STATUS_NONE,
43      StmDisplayStruct(p.E-1)(), IntruderVerticalBelief(p.E-21)[], IntruderHorizontalBelief(p.E-21)[] )
44  TRMIntruderInput( id::UInt32, address::UInt32, degraded_surveillance::UInt16 ) =
45    new( false, id, address, false, 0, EQUIPAGE_ATCRBS,
46      ACTIVE_CAS_TCAS, COORDINATION_NONE, SOURCE_MODES,
47      degraded_surveillance, false, DESIGNATION_NONE, PROTECTION_MODE_Xa, false,
48      fill( false, 2 ), Xo_STATUS_NONE,
49      StmDisplayStruct(p.E-1)(), IntruderVerticalBelief(p.E-21)[], IntruderHorizontalBelief(p.E-21)[] )
50 end

```

Referenced In: IsDesignationInForce(p.311), SetInvalidIntruder(p.324), SetGroundMsgData(p.312), UpdateIntruderInputs(p.211), VerticalTRMUpdate(p.152), TRMInput(p.E-19), GenerateTAOnlyModeOutput(p.292), GenerateTRMOutput(p.291), DetermineTTIAndTID(p.313), VerticalTRMUpdatePrep(p.I-2), DetermineSPI(p.310), AdjustTARAModeIntruderOutput(p.302), StateAndCostEstimation(p.161), DetermineDSI(p.316), GenerateTARAModeOutput(p.299), UpdateIntruderVRC(p.89), AdjustTAOnlyModeIntruderOutput(p.294), SetRABroadcastData(p.317), IntruderPrep(p.165), DroppedIntrudersAdjustment(p.320), UpdateIndivAdjustThreatInfo(p.304), IsIntruderMaster(p.334), SetRAMessageOutput(p.308), GenerateStandbyModeOutput(p.306), SetIntruderBeliefStates(p.103), ProximityEstimation(p.147), AddADSBTrackToReport(p.110), SetTargetDesignation(p.125), AdjustTargetDesignationValidity(p.143), SetCoordination(p.91), SetDisplayDataPassive(p.112), AddModeSTrackToReport(p.102), GenerateStmReport(p.97), AddModeCTrackToReport(p.114), SetIntruderDegradedFlags(p.104), SetDisplayDataActive(p.107), FilterTracksForTRM(p.100), SetIntruderDesignationData(p.137)

Type 30 | IntruderHorizontalBelief This data structure contains the data elements that represent a weighted sample from a four state (x and y position and velocity) distribution in the horizontal dimension.

```

1 type IntruderHorizontalBelief
2   x_rel::R    # E/W component of position relative to own ship (ft)
3   y_rel::R    # N/S component of position relative to own ship (ft)
4   dx_rel::R   # E/W component of velocity relative to own ship (ft/s)
5   dy_rel::R   # N/S component of velocity relative to own ship (ft/s)
6   weight::R   # weight of this sample [0-1]
7 #
8   IntruderHorizontalBelief( x_rel::R, y_rel::R, dx_rel::R, dy_rel::R, w::R ) = new( x_rel, y_rel, dx_rel,
9                                dy_rel, w )
9   IntruderHorizontalBelief() = new( 0.0, 0.0, 0.0, 0.0, 0.0 )
10 end

```

Referenced In: TRMIntruderInput(p. E-20), ConvertHorizontal(p. 169), StateAndCostEstimation(p. 161), StateEstimation(p. 168), SetIntruderBeliefStates(p. 103)
--

Type 31 | IntruderVerticalBelief This data structure contains the data elements of the intruder that represent a weighted sample from a two state (position and velocity) distribution in the vertical dimension. A collection of vertical samples are used to represent the entire distribution.

```

1 type IntruderVerticalBelief
2   z::R        # Barometric altitude (ft)
3   dz::R       # Vertical rate (ft/s)
4   weight::R   # Weight for this sample [0-1]
5 #
6   IntruderVerticalBelief( z::R, dz::R, weight::R ) = new( z, dz, weight )
7   IntruderVerticalBelief() = new( 0.0, 0.0, 0.0 )
8 end

```

Referenced In: TRMIntruderInput(p. E-20), DetermineAltitudeDependentCOCFactor(p. 181), DetermineTauVerticalDistribution(p. 176), CombineVerticalBeliefs(p. 178), StateAndCostEstimation(p. 161), SetBeliefBasedOnlineCostState(p. 179), IntruderPrep(p. 165), StateEstimation(p. 168), TauEstimation(p. 174), SetIntruderBeliefStates(p. 103)
--

Type 32 | OwnVerticalBelief This data structure contains the data elements of the ownership that represent a weighted sample from a two state (position and velocity) distribution in the vertical dimension. A collection of vertical samples are used to represent the entire distribution.

```

1 type OwnVerticalBelief
2   z::R        # Barometric altitude (ft)
3   dz::R       # Vertical rate (ft/s)
4   weight::R   # Weight for this sample [0-1]
5 #
6   OwnVerticalBelief( z::R, dz::R, weight::R ) = new( z, dz, weight )
7   OwnVerticalBelief() = new( 0.0, 0.0, 0.0 )
8 end

```

Referenced In: DetermineTauVerticalDistribution(p. 176), CombineVerticalBeliefs(p. 178), TRMOwnInput(p. E-19), GetVTRMOwnData(p. 153), StateEstimation(p. 168), TauEstimation(p. 174), GenerateStmReport(p. 97)
--

E.4 TRM Output-STM Input Data Structures

These data structures contain information produced by the TRM for the onboard pilot display and coordination. The TRM Report is also provided to the STM housekeeping function. STM housekeeping for target designation uses the relevant fields in the TRM intruder display data structure.

Type 33 | TRMReport This data structure is produced by the TRM at the end of each processing cycle. It contains information for the onboard pilot display, coordination, RA broadcast, RA transmission to the Ground Station, and diagnostics.

```

1 type TRMReport
2   display::TRMDisplayData(p. E-24) # Data for the onboard pilot display
3   coordination::Vector<TRMCoordinationInterrogationData(p. E-25)>
4   # Per intruder data for outgoing coordination messages
5   designation::TRMDesignationData(p. E-25)
6   # Target designation data for ASA system
7   broadcast::TRMRABroadcastData(p. E-27)
8   # Data for RA broadcast
9   ground_msg::TRMGroundMsgData(p. E-27)
10  # Data for RA transmission to Ground Station
11  debug::TRMDebugData(p. E-29) # Purely informational data
12 #
13  TRMReport() = new( TRMDisplayData(p. E-24)(),
14                      TRMCoordinationInterrogationData(p. E-25)[],
15                      TRMDesignationData(p. E-25)(),
16                      TRMRABroadcastData(p. E-27)(),
17                      TRMGroundMsgData(p. E-27)(),
18                      TRMDebugData(p. E-29)() )
19  TRMReport( display_logic::DisplayLogic(p. E-35), dz_min::R, dz_max::R, ddz::R ) =
20    new( TRMDisplayData(p. E-24)( display_logic, TRMIIntruderDisplayData(p. E-24)[ ] ),
21          TRMCoordinationInterrogationData(p. E-25)[],
22          TRMDesignationData(p. E-25)(),
23          TRMRABroadcastData(p. E-27)(),
24          TRMGroundMsgData(p. E-27)(),
25          TRMDebugData(p. E-29)( display_logic.alert, dz_min, dz_max, ddz, TRMIIntruderDebugData(p. E-29)[ ] )
                )
26 end

```

Referenced In: VerticalTRMUpdate(p. 152), GenerateTAOnlyModeOutput(p. 292), GenerateTRMOutput(p. 291), GenerateTARAModeOutput(p. 299), GenerateStandbyModeOutput(p. 306), StmHousekeeping(p. A-1), StmHousekeepingTargetDesignation(p. A-2)

Type 34 | TRMDisplayData This data structure is generated as part of the TRM Report. It contains information needed by the onboard pilot display. The information includes the Label270 codes that represent the advisory, flags to indicate whether an annunciation is indicated and whether aural annunciations are allowed, a flag to indicate whether the advisory reflects own aircraft crossing the path of an intruding aircraft, and the target vertical rate consistent with the advisory, in ft/sec. A collection of display information for each individual intruding aircraft is also included.

```

1 type TRMDisplayData
2   cc::UInt8          # combined control (label270)
3   vc::UInt8          # vertical control (label270)
4   ua::UInt8          # up advisory (label270)
5   da::UInt8          # down advisory (label270)
6   target_rate::R     # vertical rate to target (ft/s)
7   turn_off_aurals::Bool # below low altitude cutoff for aurals
8   crossing::Bool    # crossing alert flag
9   alarm::Bool        # annunciate RA flag
10  intruder::Vector{TRMIIntruderDisplayData(p.E-24)}
11      # Per intruder information for the display
12 #
13  TRMDisplayData() = new( 0, 0, 0, 0, 0.0, false, false, [] )
14  TRMDisplayData( display::DisplayLogic(p.E-35),
15                  intruders::Vector{TRMIIntruderDisplayData(p.E-24)} ) = new(
16                  display.cc, display.vc, display.ua, display.da,
17                  display.target_rate, display.turn_off_aurals, display.crossing,
18                  display.alarm, intruders )
19 end

```

Referenced In: TRMReport(p.E-23)

Type 35 | TRMIIntruderDisplayData This data structure is generated as part of the TRM Report. It contains information for the onboard pilot display about an individual intruding aircraft evaluated by the TRM. The track display score is used for prioritizing intruding aircraft for filtering and display. The track display advisory code is used for determining the color and fill of the intruding aircraft icon.

```

1 type TRMIIntruderDisplayData
2   id::UInt32          # Unique identifier
3   mode_s::UInt32      # Mode S address
4   is_icao::Bool       # Is an ICAO address
5   #
6   tds::R              # Track display score; used for display prioritization
7   code::UInt8         # Track display advisory code (see TACODE_ constants)
8 #
9   TRMIIntruderDisplayData() = new( 0, 0, false, 0.0, TACODE_CLEAR )
10  TRMIIntruderDisplayData( id::UInt32, mode_s::UInt32, is_icao::Bool, tds::R, code::UInt8 ) =
11      new( id, mode_s, is_icao, tds, code )
12 end

```

Referenced In: TRMIIntruderData(p.E-39) , TRMDisplayData(p.E-24) , TRMReport(p.E-23)

Type 36 | TRMCoordinationInterrogationData This data structure is generated as part of the TRM Report. It contains coordination information for output to individual aircraft equipped with TCAS, CAS Responsive, or Detect And Avoid (DAA) Responsive systems.

```

1 type TRMCoordinationInterrogationData
2   id::UInt32           # Unique identifier for intruder
3 #
4   cvc::UInt32          # Cancel Vertical RA Complement
5   vrc::UInt32          # Vertical RA complement
6   vsb::UInt32          # Vertical Sense Bits (parity check)
7 #
8   chc::UInt32          # Cancel Horizontal RA complement
9   hrc::UInt32          # Horizontal RA complement
10  hsb::UInt32          # Horizontal Sense Bits (parity check)
11 #
12  mtb::Bool            # multiple threats for this RA
13  mid::UInt32          # Own Mode S address (sender)
14  taa::UInt32          # Intruder 24-bit aircraft address (intended receiver)
15  coordination_msg::UInt32 # Coordination mechanism (see COORDINATION_ constants)
16 #
17  TRMCoordinationInterrogationData() = new( 0, 0, 0, 0, 0, 0, 0, false, 0, 0, COORDINATION_NONE )
18  TRMCoordinationInterrogationData( id::UInt32, cvc::UInt32, vrc::UInt32, vsb::UInt32, mtb::Bool, mid::UInt32
19    , taa::UInt32, coordination_msg::UInt32 ) = new( id, cvc, vrc, vsb, 0, 0, 0, mtb, mid, taa,
20    coordination_msg )
21  TRMCoordinationInterrogationData( id::UInt32, cvc::UInt32, vrc::UInt32, vsb::UInt32, chc::UInt32, hrc::
22    UInt32, hsb::UInt32, mtb::Bool, mid::UInt32, taa::UInt32, coordination_msg::UInt32 ) = new( id, cvc,
23    vrc, vsb, chc, hrc, hsb, mtb, mid, taa, coordination_msg )
24 end

```

Referenced In:	TRMIIntruderData(p. E-39), TRMReport(p. E-23)
----------------	---

Type 37 | TRMDesignationData This data structure is generated as part of the TRM Report. It contains information on the Xo mode availability. A collection of designation information for each individual intruding aircraft is also included.

```

1 type TRMDesignationData
2   availability::Vector<UInt8> # Xo mode availability (See Xo_AVAILABILITY constants)
3   intruder::Vector<TRMIIntruderDesignationData(p. E-26)>
4     # Per intruder designation information
5 #
6   TRMDesignationData() =
7     new( fill(Xo_AVAILABILITY_NOT_CONFIGURED,2), TRMIIntruderDesignationData(p. E-26)[ ] )
8   TRMDesignationData( xo_availability::Vector<UInt8>, intruders::Vector<TRMIIntruderDesignationData(p. E-26)> )
9     =
10    new( xo_availability, intruders )
11 end

```

Referenced In:	TRMReport(p. E-23)
----------------	--------------------

Type 38 | TRMIntruderDesignationData This data structure is generated as part of the TRM Report. It contains information describing the Xo designation state for the intruding aircraft. The selected designation mode and the logic processing mode are given, as is the current advisory state. The valid field indicates whether the intruder can be designated to a particular Xo mode. The status field provides additional information about transitions and differences between the designated mode and the logic mode.

```

1 type TRMIntruderDesignationData
2   id::UInt32          # Target id of intruder
3   address::UInt32     # 24-bit address of intruder
4   is_icao::Bool       # Is an ICAO address
5   designated_mode::UInt8 # Xo designation, from input (See DESIGNATION_ constants)
6   logic_mode::TRMLogicModeData(p. E-26)
7           # Actual logic mode used for TRM processing
8   active_ra::Bool     # This intruder has an active RA (may be suppressed)
9   suppressed_ra::Bool # This intruder has an active RA for which display is suppressed
10  multithreat::Bool   # There are multiple threats. If active_ra is true, includes this intruder
11  valid::Vector{Bool} # Valid Xo processing modes (See Xo_IDX_ constants for indices)
12  status::UInt8       # Designation status (See Xo_STATUS_ constants)
13 #
14  TRMIntruderDesignationData() =
15    new( 0, 0, false,
16         DESIGNATION_NONE,
17         TRMLogicModeData(p. E-26)(),
18         false, false, false,
19         fill(false,2), Xo_STATUS_NONE )
20  TRMIntruderDesignationData( id::UInt32, address::UInt32, is_icao::Bool,
21    designated_mode::UInt8,
22    processing::UInt8, dna::Bool, protection_mode::UInt8,
23    active_ra::Bool, suppressed_ra::Bool, multithreat::Bool ) =
24    new( id, address, is_icao,
25         designated_mode,
26         TRMLogicModeData(p. E-26)( processing, dna, protection_mode ),
27         active_ra, suppressed_ra, multithreat,
28         fill(false,2), Xo_STATUS_NONE )
29 end

```

Referenced In: TRMDesignationData(p. E-25), SetDroppedIntruder(p. 322), TRMIntruderData(p. E-39), StmHousekeeping-TargetDesignation(p. A-2)
--

Type 39 | TRMLogicModeData This data structure encapsulates information describing the TRM logic processing mode associated with designation. This information reflects the designation and processing settings actually used by the TRM.

```

1 type TRMLogicModeData
2   processing::UInt8      # TRM logic processing mode (See RA_PROCESSING_)
3   dna::Bool              # Whether Designated No Alerts (DNA) is active
4   protection_mode::UInt8 # Active protection mode (See PROTECTION_MODE_)
5 #
6   TRMLogicModeData() =
7     new( RA_PROCESSING_NONE, false, PROTECTION_MODE_Xa )
8   TRMLogicModeData( logic_mode::UInt8, dna::Bool, protection_mode::UInt8 ) =
9     new( logic_mode, dna, protection_mode )
10 end

```

Referenced In: TRMIntruderDesignationData(p. E-26)

Type 40 | TRMGroundMsgData This data structure is generated as part of the TRM Report. It represents resolution advisory information transmitted via the Mode S transponder to the Ground Station.

```

1 type TRMGroundMsgData
2   ra_data::TRMRACoordinationData(p. E-28)
3   # Contains ARA, (LDI), RMF, RAC, RAT, and MTE fields (20 bits)
4   # Setting of RAI is implemented outside ACAS-X
5 #   cnt::Bool      # Not implemented - Continuation bit (1 bit)
6   tti::UInt8       # Threat type indicator for most recent threat (1 bit)
7   #     (see TTI_ in constants)
8   tid::TID(p. E-28) # Threat identifier for most recent threat (24 bits)
9   #     Set to TIDAddress when tti = TTI_ADDRESS
10  #     Set to TIDAltRngBrg when tti = TTI_ALT_RNG_BRG
11  dsi::Bool        # Designation indicator (1 bit)
12  #     True when TID threat designated & designation in force
13  spi::Bool        # RA suppression indicator (1 bit)
14  #     True when MTE=0 & RA is suppressed
15  #     True when MTE=1 & non-TID threat designated & designation in force
16 #
17  TRMGroundMsgData() =
18    new( TRMRACoordinationData(p. E-28)(), TTI_ADDRESS, TIDAddress(p. E-28)(), false, false )
19  TRMGroundMsgData( ra_data::TRMRACoordinationData(p. E-28), tti::UInt8, tid::TID(p. E-28),
20                  dsi::Bool, spi::Bool ) =
21    new( deepcopy( ra_data ), tti, tid, dsi, spi )
22 end

```

Referenced In: SetGroundMsgData(p. 312), GenerateTAOnlyModeOutput(p. 292), GenerateTARAModeOutput(p. 299), SetRAMessageOutput(p. 308), GenerateStandbyModeOutput(p. 306), TRMReport(p. E-23)

Type 41 | TRMRABroadcastData This data structure is generated as part of the TRM Report. It represents resolution advisory information communicated via air-to-air broadcast.

```

1 type TRMRABroadcastData
2   ra_data::TRMRACoordinationData(p. E-28)
3   # Contains ARA, (LDI), RMF, RAC, RAT, and MTE fields (20 bits)
4   spi::Bool        # RA suppression indicator (1 bit)
5   #     True when MTE=0 & RA is suppressed
6   #     True when MTE=1 & any threat designated & designation in force
7   aid::UInt16       # Own Mode A identifier code (13 bits)
8   cac::UInt16       # Own Mode C altitude code (ft) (13 bits)
9 #
10  TRMRABroadcastData() =
11    new( TRMRACoordinationData(p. E-28)(), false, 0, 0 )
12  TRMRABroadcastData( ra_data::TRMRACoordinationData(p. E-28), spi::Bool, aid::UInt16, cac::UInt16 ) =
13    new( deepcopy( ra_data ), spi, aid, cac )
14 end

```

Referenced In: GenerateTAOnlyModeOutput(p. 292), GenerateTARAModeOutput(p. 299), SetRBroadcastData(p. 317), SetRAMessageOutput(p. 308), GenerateStandbyModeOutput(p. 306), TRMReport(p. E-23)

Type 42 | TRMRACoordinationData This data structure contains fields common to TRMRABROADCASTDATA and TRMGROUNDMSGDATA.

```

1 type TRMRACoordinationData
2     avra_single_intent::Bool      # Active Vertical RA (AVRA) (7 bits)
3     avra_crossing::Bool          # Single intent / conflicting intents (1 bit)
4     avra_down::Bool              # Crossing / not crossing (1 bit)
5     avra_strength::UInt8         # Down / Up (1 bit)
6     avra_strength::UInt8         # Strength bits as integer value (4 bits)
7     ahra::UInt8                # Strength bits as integer value (4 bits)
8     ldi::UInt8                 # Not implemented - Active Horizontal RA bits (3 bits)
9     ldi::UInt8                 # Low-level Descend Inhibits, for Xa, as integer value (2 bits)
10    rmf::UInt8                 # RA Message Format as integer value (2 bits)
11    rac::Vector{Bool}           # RA Complements (4 bits)
12    rat::Bool                  # RA terminated (other fields are a repeat of the last message)
13    mte::Bool                  # RA terminated message is not repeated by the TRM
14    mte::Bool                  # There are multiple threats for this RA
14#
15    TRMRACoordinationData() =
16        new( false, false, false, 0, LDI_NONE, RMF_ACAS_Xa, fill(false,4), false, false )
17 end

```

Referenced In: SetGroundMsgData(p. 312), VerticalRAOutputState(p. E-40), TRMGroundMsgData(p. E-27), SetRABroadcastData(p. 317), TRMRABroadcastData(p. E-27), DetermineRACoordinationData(p. 309), SetRAMessageOutput(p. 308)

Type 43 | TID This is an abstract data structure used in TRMGROUNDMSGDATA to reference TIDADDRESS and TIDALTRNGBRG. TID stands for Threat ID.

```
1 abstract TID
```

Referenced In: VerticalRAOutputState(p. E-40), DetermineTTIAndTID(p. 313), TRMGroundMsgData(p. E-27), TIDAltRngBrg(p. E-29), TIDAddress(p. E-28)

Type 44 | TIDAddress This data structure is used in TRMGROUNDMSGDATA when the threat is identified by an ICAO address. It is derived from the abstract type TID. TID stands for Threat ID.

```

1 type TIDAddress <: TID(p. E-28)
2     tid::UInt32      # Threat ID as an address (24 bits)
3#
4     TIDAddress() =
5         new( uint32(0) )
6     TIDAddress( tid::UInt32 ) =
7         new( tid )
8 end

```

Referenced In: SetGroundMsgData(p. 312), VerticalRAOutputState(p. E-40), DetermineTTIAndTID(p. 313), TRMGroundMsgData(p. E-27)

Type 45 | TIDAltRngBrg This data structure is used in TRMGROUNDMSGDATA when the threat is identified by encoded altitude, range, and bearing. It is derived from the abstract type TID. TID stands for Threat ID.

```

1 type TIDAltRngBrg <: TID(p.E-28)
2     tida::UInt16 # Threat ID encoded altitude (ft) (11 bits)
3     tidr::UInt8  # Threat ID encoded range (ft)      ( 7 bits)
4     tidb::UInt8  # Threat ID encoded bearing (deg) ( 6 bits)
5 #
6     TIDAltRngBrg() =
7         new( uint16(0), uint8(0), uint8(0) )
8     TIDAltRngBrg( tida::UInt16, tidr::UInt8, tidb::UInt8 ) =
9         new( tida, tidr, tidb )
10 end
```

Referenced In: DetermineTTIAndTID(p.313)

Type 46 | TRMDebugData This data structure is generated as part of the TRM Report. It provides information about the internal TRM state that can be used for diagnostics, including a collection with information for each individual intruding aircraft.

```

1 type TRMDebugData
2     alert::Bool # TA active (determined by the display from "code"
3     dz_min::R   # intended minimum vertical rate (ft/s)
4     dz_max::R   # intended maximum vertical rate (ft/s)
5     ddz::R     # intended vertical acceleration (ft/s/s)
6     intruder::Vector{TRMIIntruderDebugData(p.E-29)}
7         # Per intruder debug information
8 #
9     TRMDebugData() = new( false, 0.0, 0.0, 0.0, TRMIIntruderDebugData(p.E-29)[ ] )
10    TRMDebugData( alert::Bool, dz_min::R, dz_max::R, ddz::R, intruder::Vector{TRMIIntruderDebugData(p.E-29)} ) =
11        = new( alert, dz_min, dz_max, ddz, intruder )
11 end
```

Referenced In: TRMReport(p.E-23)

Type 47 | TRMIIntruderDebugData This data structure is generated as part of the TRM Report. It provides information about the internal TRM state for an individual intruding aircraft that can be used for diagnostics.

```

1 type TRMIIntruderDebugData
2     id::UInt32      # Unique identifier
3     address::UInt32 # ICAO or non-ICAO address
4     is_icao::Bool   # Is an ICAO address
5     sense::UInt8    # Individual RA sense (see RA_SENSE_ constants)
6 #
7     TRMIIntruderDebugData() = new( 0, 0, false, 0 )
8     TRMIIntruderDebugData( id::UInt32, address::UInt32, is_icao::Bool, sense::UInt8 ) =
9         new( id, address, is_icao, sense )
10 end
```

Referenced In: TRMIIntruderData(p.E-39), TRMDebugData(p.E-29), TRMReport(p.E-23)

E.5 TRM Internal Data Structures

The algorithms contained within this document rely on a number of data structures to store and pass data within the TRM. This appendix provides the definition, description and initial values for data structures that are internal to the TRM. These data structures are used to encapsulate history, state, and variables. The fields in the RDATATABLE data structure are described in the Data Table Format Specification in Appendix G.

Type 48 | TRMState This data structure provides information about internal TRM state. It contains information about own aircraft previous and current states and a collection with information about the processing states for each individual intruding aircraft.

```
1 type TRMState
2   st_own::TRMOwnState(p.E-32)          # Ownship TRM state information
3   st_intruder::Vector<TRMIIntruderState(p.E-33)>
4                                     # Intruder TRM state information
5   params::paramsfile_type
6 #
7   TRMState() = new( TRMOwnState(p.E-32)(), Array( TRMIIntruderState(p.E-33), 0 ) )
8 end
```

Referenced In: VerticalTRMUpdate(p.152), GenerateTRMOutput(p.291), VerticalTRMUpdatePrep(p.1-2), GenerateStandbyModeOutput(p.306)
--

Type 49 | TRMOwnState This data structure is used by the TRM to maintain state information directly associated with previous and current own aircraft state and global advisories. It includes context-specific state structures and information used to produce resolution advisory messages for the Ground Station.

```

1 type TRMOwnState
2   a_prev::GlobalAdvisory(p. E-35)          # Previous global advisory
3   dz_ave_prev::R                           # Previous own vertical rate from vertical beliefs (ft/sec)
4   opmode_prev::R                          # Ownship operational mode on last cycle
5   action_prev::Z                         # Previous global advisory action for display
6   word_prev::Z                           # Previous Label270 word
7   strength_prev::UInt8                   # Previous ARA strength value
8   turn_off_aurals_prev::Bool            # Previous state of low altitude cutoff for aurals
9   crossing_prev::Bool                  # Previous advisory was for a crossing
10  ra_suppressed_prev::Bool             # Previous advisory was suppressed (e.g., DNA)
11  ra_output_prev::VerticalRAOutputState(p. E-40)
12    # Previous TRMGroundMsgData & TRMRABroadcastData
13  st_multithreat_cost_balancing::MultithreatCostBalancingCState(p. E-45)
14    # State variables for MultithreatCostBalancing
15  st_arbitrate::ActionArbitrationGlobalCState(p. E-41)
16    # State variables for ActionArbitration
17  st_alt_inhibit::Vector{AltitudeInhibitCState(p. E-42)}
18    # State variables for AltitudeInhibitCost
19 #
20  TRMOwnState() =
21    new( GlobalAdvisory(p. E-35)(),
22      0.0, OPMODE_STANDBY, COC, 0, uint8(0), true, false, false,
23      VerticalRAOutputState(p. E-40)(),
24      MultithreatCostBalancingCState(p. E-45)(), ActionArbitrationGlobalCState(p. E-41)(),
25      AltitudeInhibitCState(p. E-42)[] )
26 end

```

Referenced In: SetGroundMsgData(p.312), DetermineDisplayData(p.283), MTLODetermination(p.259), VerticalTRMUpdate(p.152), GenerateTAOnlyModeOutput(p.292), DetermineAuralInhibit(p.287), DisplayLogicDetermination(p.280), NoIntrudersAction(p.251), AdjustTARAModeIntruderOutput(p.302), StateAndCostEstimation(p.161), GetVTR-MOwnData(p.153), GenerateTARAModeOutput(p.299), DetermineCrossing(p.281), SetRABroadcastData(p.317), IntruderPrep(p.165), DetermineMultiIntruderAction(p.254), DetermineRACoordinationData(p.309), ActionSelection(p.250), UpdateIndivAdjustThreatInfo(p.304), IndividualCostEstimation(p.248), OnlineCostEstimation(p.214), SetRAMessageOutput(p.308), TRMState(p. E-31)

Type 50 | TRMIntruderState This data structure is used by the TRM to maintain state information directly associated with previous and current intruding aircraft state and individual advisories.

```

1 type TRMIntruderState
2   id::UInt32                      # Unique identifier
3   address::UInt32                  # ICAO or non-ICAO address
4   is_icao::Bool                    # Is ICAO address (if not, use AltRngBrg for TID)
5   protection_mode_prev::UInt8    # Previous protection mode (See PROTECTION_MODE_ constants)
6   protection_mode_curr::UInt8    # Current protection mode (See PROTECTION_MODE_ constants)
7   a_prev::IndividualAdvisory(p.E-36) # Previous individual advisory for this intruder
8   b_prev::AdvisoryBeliefState(p.E-34) # Previous individual advisory beliefs for CombineVerticalBeliefs
9   time_since_ta::Z                # Time the TA has been active (s)
10  sense_prev::Symbol              # Up or down sense of last individual advisory
11  vrc_prev::UInt32                # VRC for transmission to this intruder in the last cycle
12  cvc_prev::UInt32                # CVC for transmission to this intruder in the last cycle
13  equipage_prev::Z               # Previous equipage (See EQUIPAGE_ constants)
14  coordination_msg_prev::UInt32# Previous coordination message type (See COORDINATION_ constants)
15  st_cost_on::OnlineCostState(p.E-37) # State variables for online cost algorithms
16  st_arbitrate::ActionArbitrationCState(p.E-41)
17                                # State variables for ActionArbitration
18  is_threat::Bool                # Has an RA
19  is_identified_threat::Bool     # Has been identified as a threat for setting TTI and TID
20  designated_mode::UInt8         # Selected designation mode (See DESIGNATION_ constants)
21  processing::UInt8              # Processing type (See RA_PROCESSING_ constants)
22  no_alerts::Bool                # Whether advisories should be suppressed in output (processing modifier)
23  is_dna_coordination::Bool     # Special TA/RA processing because this intruder is
24                                # designated to Xo DNA mode and VRCs will be output
25  is_desonly_output::Bool        # Output only designation data for this intruder
26  status::Symbol                 # Processing status (:New, :InUse, :Dropped, :NotValid,
27                                #:DesignationStateTimeout, :MarkedForDeletion)
28 #
29 TRMIntruderState( id::UInt32,
30                   address::UInt32, is_icao::Bool, protection_mode::UInt8 ) =
31   new( id, address, is_icao, protection_mode, protection_mode,
32         IndividualAdvisory(p.E-36)(), AdvisoryBeliefState(p.E-34)(),
33         0, 0, :None, uint32(0), uint32(0), EQUIPAGE_ATCRBS, COORDINATION_NONE,
34         OnlineCostState(p.E-37)(), ActionArbitrationCState(p.E-41)(),
35         false, false,
36         DESIGNATION_NONE, RA_PROCESSING_NONE, false, false, false, :New )
37 end

```

Referenced In: IsDesignationInForce(p.311), SetInvalidIntruder(p.324), SetGroundMsgData(p.312), UpdateIntruderInputs(p.211), VerticalTRMUpdate(p.152), TrackThreatAssessment(p.273), ActionArbitration(p.264), GenerateTAOnlyModeOutput(p.292), GenerateTRMOutput(p.291), VerticalTRMUpdatePrep(p.1-2), DetermineSPI(p.310), SetDroppedIntruder(p.322), AdjustTARAModeIntruderOutput(p.302), StateAndCostEstimation(p.161), DetermineDSI(p.316), GenerateTARAModeOutput(p.299), TRMIntruderStateUpdate(p.344), ArbitrateMatchingSenses(p.267), AdjustTAOnlyModeIntruderOutput(p.294), SetRBroadcastData(p.317), IntruderPrep(p.165), DetermineIntruderAlert(p.275), DetermineMultiIntruderAction(p.254), DroppedIntrudersAdjustment(p.320), ArbitrateConflictingSenses(p.265), ActionSelection(p.250), UpdateIndivAdjustThreatInfo(p.304), OnlineCostEstimation(p.214), SetRAMessageOutput(p.308), GenerateStandbyModeOutput(p.306), UpdateIndivAdjustCounts(p.305), TRMState(p.E-31), ConvertAdvisory(p.296)

Type 51 | AdvisoryBeliefState This data structure is used to store and pass information about the vertical beliefs and the previous advisory state for a single intruding aircraft. The vertical belief values are expressed as weights and are generated in COMBINEVERTICALBELIEFS.

```

1 type AdvisoryBeliefState
2   need_init::Bool          # Uninitialized advisory state
3   w_vert_prev::Vector{R}    # Previous vertical sample weights
4 #
5   AdvisoryBeliefState() = new( true, R[1.0] )
6 end

```

Referenced In: CombineVerticalBeliefs(p. 178), TRMIIntruderStateUpdate(p. 344), TRMIIntruderState(p. E-33), StateEstimation(p. 168)
--

Type 52 | CombinedVerticalBelief This data structure is used for storing and passing vertical state samples used for estimating the offline cost in OFFLINECOSTESTIMATION. The samples are produced by COMBINEVERTICALBELIEFS.

```

1 type CombinedVerticalBelief
2   z_rel::R      # Vertical separation (ft)
3   dz_own::R    # Vertical rate of ownship (ft/s)
4   dz_int::R    # Vertical rate of intruder (ft/s)
5   weight::R    # Sample weight (0.0-1.0)
6 #
7   CombinedVerticalBelief() = new( 0.0, 0.0, 0.0, 0.0 )
8   CombinedVerticalBelief( z_rel::R, dz_own::R, dz_int::R, weight::R ) =
9     new( z_rel, dz_own, dz_int, weight )
10 end

```

Referenced In: CombineVerticalBeliefs(p. 178), StateAndCostEstimation(p. 161), OfflineStatesScaleFactor(p. 186), OfflineCostEstimation(p. 184), StateEstimation(p. 168), GetEquivClassTableMaxCutValues(p. 186)
--



Type 53 | DisplayLogic This data structure is used within the TRM to store and pass display data associated with the global resolution advisory.

```

1 type DisplayLogic
2   crossing::Bool
3   cc::UInt8          # combined control (label270)
4   vc::UInt8          # vertical control (label270)
5   ua::UInt8          # up advisory (label270)
6   da::UInt8          # down advisory (label270)
7   target_rate::R     # vertical rate to target (ft/s)
8   alarm::Bool         # annunciate RA flag
9   turn_off_aurals::Bool # below low altitude cutoff for aurals
10  alert::Bool         # TA active
11  strength::UInt8    # ARA strength bits setting representing display output
12  down::Bool          # ARA down/up bit setting representing display output
13 #
14  DisplayLogic( crossing::Bool, cc::UInt8, vc::UInt8, ua::UInt8, da::UInt8, target_rate::R, alarm::Bool,
                  turn_off_aurals::Bool, alert::Bool, strength::UInt8, down::Bool ) =
15    new( crossing, cc, vc, ua, da, target_rate, alarm, turn_off_aurals, alert, strength, down )
16  DisplayLogic() = new( false, uint8(0), uint8(0), uint8(0), uint8(0), 0.0, false, false, uint8(0),
                         false )
17 end

```

Referenced In: VerticalTRMUpdate(p. 152), GenerateTAOnlyModeOutput(p. 292), GenerateTRMOutput(p. 291), DisplayLogicDetermination(p. 280), GenerateTARAModeOutput(p. 299), AdjustTAOnlyModeIntruderOutput(p. 294), DeterminerRACoordinationData(p. 309), TRMDisplayData(p. E-24), SetRAMessageOutput(p. 308), GenerateStandbyModeOutput(p. 306), TRMReport(p. E-23)

Type 54 | GlobalAdvisory This data structure is used within the TRM to store information that defines the global resolution advisory.

```

1 type GlobalAdvisory
2   action::Z           # global advisory action
3   dz_min::R           # global advisory minimum vertical rate (ft/s)
4   dz_max::R           # global advisory maximum vertical rate (ft/s)
5   ddz::R              # global advisory acceleration (ft/s/s)
6   multithreat::Bool   # multiple threats for global advisory
7 #
8   GlobalAdvisory() = new( COC, -Inf, Inf, 0.0, false )
9 end

```

Referenced In: OnlineUncoordinatedActionCostEstimation(p. 193), CoordinationSelection(p. 269), GenerateTAOnlyModeOutput(p. 292), DetermineSenses(p. 260), TRMOwnState(p. E-32), OnlineUncoordinatedCostEstimation(p. 191), PersistMTLO(p. 341), OnlineCoordinatedActionCostEstimation(p. 216), GenerateStandbyModeOutput(p. 306)

Type 55 | IndividualAdvisory This data structure is used within the TRM to store information that defines a resolution advisory for an individual intruding aircraft.

```
1 type IndividualAdvisory
2   action::Z      # individual advisory action
3   dz_min::R     # individual advisory minimum vertical rate (ft/s)
4   dz_max::R     # individual advisory maximum vertical rate (ft/s)
5   ddz::R        # individual advisory acceleration (ft/s/s)
6   ra_prev::Bool # previous individual advisory was an RA
7   sense::Symbol # up or down sense of the advisory for this intruder
8   vrc::UInt32    # Vertical RA Complement for transmission
9   cvc::UInt32    # Cancel Vertical RA Complement for transmission
10 #
11 IndividualAdvisory() = new( COC, -Inf, Inf, 0.0, false, :None, 0, 0 )
12 end
```

Referenced In:	OnlineUncoordinatedActionCostEstimation(p. 193), CoordinationSelection(p. 269), TRMIintruderState-Update(p. 344), OnlineUncoordinatedCostEstimation(p. 191), TRMIintruderState(p. E-33), OnlineCoordinatedActionCostEs-timation(p. 216)
-----------------------	---

Type 56 | OnlineCostState This data structure is used within the TRM to store and pass context-specific state information. It contains all of the context-specific state information for the algorithms that apply different online costs in ONLINEUNCOORDINATEDCOSTESTIMATION and ONLINECOSTESTIMATION. See the TRM Context-Specific State Data Structures in Appendix E.6 for descriptions of each context-specific state data structure.

```

1 type OnlineCostState
2   alt_inhibit::Union(AltitudeInhibitCState(p. E-42),Nothing) # Read-only
3   advisory_restart::AdvisoryRestartCState(p. E-42)
4   initialization::InitializationCState(p. E-44)
5   restrict_coc::RestrictCOCCState(p. E-45)
6   max_reversal::MaxReversalCState(p. E-44)
7   own_response_est::ResponseEstimationCState(p. E-45)
8   int_response_est::ResponseEstimationCState(p. E-45)
9   compatibility_cost::CompatibilityCState(p. E-43)
10  sa01_heuristic::SA01HeuristicCState(p. E-46)
11  bad_transition::BadTransitionCState(p. E-42)
12  crossing_no_alert::CrossingNoAlertCState(p. E-44)
13  prevent_early_coc::PreventEarlyCOCCState(p. E-45)
14  coord_ra_deferral::CoordinatedRADeferralCState(p. E-43)
15  altitude_dependent_coc::AltitudeDependentCOCCState(p. E-42)
16  low_alt_parallel_ra_deferral::LowAltitudeParallelRADeferralCState(p. E-44)
17  ta_low_alt_parallel_ra_deferral::LowAltitudeParallelRADeferralCState(p. E-44)
18  safe_crossing_ra_deferral::SafeCrossingRADeferralCState(p. E-46)
19  critical_interval_protection::CriticalIntervalProtectionCState(p. E-43)
20  time_based_non_compliance::TimeBasedNonComplianceCState(p. E-46)
21  coord_delay::CoordinationDelayCState(p. E-43)
22 #
23 OnlineCostState() = new(
24   nothing,
25   AdvisoryRestartCState(p. E-42)(),
26   InitializationCState(p. E-44)(),
27   RestrictCOCCState(p. E-45)(),
28   MaxReversalCState(p. E-44)(),
29   ResponseEstimationCState(p. E-45)(),
30   ResponseEstimationCState(p. E-45)(),
31   CompatibilityCState(p. E-43)(),
32   SA01HeuristicCState(p. E-46)(),
33   BadTransitionCState(p. E-42)(),
34   CrossingNoAlertCState(p. E-44)(),
35   PreventEarlyCOCCState(p. E-45)(),
36   CoordinatedRADeferralCState(p. E-43)(),
37   AltitudeDependentCOCCState(p. E-42)(),
38   LowAltitudeParallelRADeferralCState(p. E-44)(),
39   LowAltitudeParallelRADeferralCState(p. E-44)( true ),
40   SafeCrossingRADeferralCState(p. E-46)(),
41   CriticalIntervalProtectionCState(p. E-43)(),
42   TimeBasedNonComplianceCState(p. E-46)(),
43   CoordinationDelayCState(p. E-43)( false )
44 )
45 end

```

Referenced In: OnlineUncoordinatedActionCostEstimation(p. 193), TRMIintruderStateUpdate(p. 344), OnlineUncoordinatedCostEstimation(p. 191), TRMIintruderState(p. E-33), SetBeliefBasedOnlineCostState(p. 179), IntruderPrep(p. 165), OnlineCoordinatedActionCostEstimation(p. 216), StateEstimation(p. 168)

Type 57 | RDDataTable This data structure is used within the TRM to store the entry distribution tables used for tau estimation and the offline cost tables used for offline cost estimation. See the Data Table Format Specification in Appendix G.

```

1 type RDDataTable
2   cut_names::Vector<String> # The names associated with the cut dimensions
3   cut_counts::Vector<Z> # The number of cut points for each dimension
4   cuts::Vector<R> # All the cut points for all the dimensions
5   index::Vector<UInt32> # Offsets into data, may contain all 0s
6   data::Vector<Float16> # Contents of the block dictionary data field
7 #
8   RDDataTable() =
9     new( Array<String,0>, Array<Z,0>, Array<R,0>, Array<UInt32,0>, Array<Float16,0> )
10  RDDataTable( cut_names::Vector<String>, cut_counts::Vector<Z>, cuts::Vector<R>, index::Vector<UInt32>,
11    data::Vector<Float16> ) =
12      new( cut_names, cut_counts, cuts, index, data )
12 end

```

Referenced In: GetTableMaxCutValue(p. 187), DetermineTauVerticalDistribution(p. 176), InterpolateBlocks(p. 177), GetEntryDistributionTables(p. 175), LookupEntryDistribution(p. 176), TauEstimation(p. 174), GetEquivClassTableMaxCut-Values(p. 186), GetOfflineCostTauBeliefs(p. 185), LookupOfflineCost(p. 188)
--

Type 58 | TauBelief This data structure is used within the TRM to store and pass tau belief information. It contains the data elements that represent a weighted sample from a distribution of tau values.

```

1 type TauBelief
2   tau::R # Time to closest point of approach (sec)
3   weight::R # Weight for this sample [0-1]
4 #
5   TauBelief( tau::R, weight::R ) = new( tau, weight )
6   TauBelief() = new( 0.0, 0.0 )
7 end

```

Referenced In: DetermineAltitudeDependentCOCFactor(p. 181), StateAndCostEstimation(p. 161), OnlineUncoordinatedCostEstimation(p. 191), SetBeliefBasedOnlineCostState(p. 179), ExpectedTau(p. 327), OfflineCostEstimation(p. 184), StateEstimation(p. 168), TauEstimation(p. 174), OnlineCostEstimation(p. 214), GetOfflineCostTauBeliefs(p. 185), LookupOfflineCost(p. 188)
--

Type 59 | TRMIntruderData This data structure is used within the TRM, during output processing, for storing and passing information about the individual intruder.

```

1 type TRMIIntruderData
2     id::UInt32      # Unique id for this intruder
3     sense::Symbol   # Up or down sense of individual RA
4     coordination::TRMCordinationInterrogationData(p. E-25)
5         # Coordination data for this intruder
6     display::TRMIIntruderDisplayData(p. E-24)
7         # Display data for this intruder
8     designation::TRMIIntruderDesignationData(p. E-26)
9         # Xo designation data for this intruder
10    debug::TRMIIntruderDebugData(p. E-29)
11        # Debug data for this intruder
12 #
13 TRMIIntruderData() = new( 0, :None,
14                             TRMCordinationInterrogationData(p. E-25)(),
15                             TRMIIntruderDisplayData(p. E-24)(),
16                             TRMIIntruderDesignationData(p. E-26)(),
17                             TRMIIntruderDebugData(p. E-29)() )
18 TRMIIntruderData( id::UInt32, sense::Symbol, mode_s_own::UInt32, mode_s_int::UInt32,
19                     cvc::UInt32, vrc::UInt32, vsb::UInt32, multithreat::Bool,
20                     coordination_msg::UInt32, tds::R, code::UInt8,
21                     designated_mode::UInt8, processing::UInt8, dna::Bool, protection_mode::UInt8,
22                     active_ra::Bool, address_int::UInt32, is_icao::Bool ) =
23     new( id, sense,
24           TRMCordinationInterrogationData(p. E-25)( id, cvc, vrc, vsb, multithreat, mode_s_own, mode_s_int,
25             coordination_msg ),
26           TRMIIntruderDisplayData(p. E-24)( id, mode_s_int, is_icao, tds, code ),
27           TRMIIntruderDesignationData(p. E-26)( id, mode_s_int, is_icao, designated_mode, processing, dna,
28             protection_mode, active_ra, false, multithreat ),
29           TRMIIntruderDebugData(p. E-29)( id, address_int, is_icao,
30             uint8(EncodeVRC(p. 272)(sense, EQUIPAGE_CASRA)) ) )
31 end

```

Referenced In: SetInvalidIntruder(p. 324), VerticalTRMUpdate(p. 152), GenerateTAOnlyModeOutput(p. 292), GenerateTR-MOutput(p. 291), VerticalTRMUpdatePrep(p. 1-2), SetDroppedIntruder(p. 322), AdjustTARAModeIntruderOutput(p. 302), GenerateTARAModeOutput(p. 299), AdjustTAOnlyModeIntruderOutput(p. 294), DroppedIntrudersAdjustment(p. 320), UpdateIndivAdjustThreatInfo(p. 304), ConvertAdvisory(p. 296)

Type 60 | TRMIndivAdjustState This data structure is used within GENERATETARAMODEOUTPUT for storing local state and passing information to helper algorithms for update.

```

1 type TRMIndivAdjustState
2   ra_count::Z          # Individual RAs used to determine global RA
3   ra_change_count::Z   # RA conversions for RAs included in ra_count
4   ra_prev_count::Z    # Individual RAs that transitioned to COC
5   force_silent_count::Z # RAs in ra_prev_count excluded from global RA determination
6   upsense::Bool         # Presence of an individual up-sense RA
7   downsense::Bool       # Presence of an individual down-sense RA
8   idx_threat_new::Z    # Index of previously unreported threat (used for TID)
9   idx_threat_upd::Z    # Index of an unprioritized threat (used for TID)
10  idx_threat_last::Z   # Index of threat reported using TID on the previous cycle
11  idx_threat_designated::Z # Index of threat designated to an Xo mode (used for TID)
12 #
13  TRMIndivAdjustState( num_intruders::Z ) =
14    new( num_intruders, 0, 0, 0, false, false, -1, -1, -1, -1 )
15 end

```

Referenced In: [AdjustTARAModeIntruderOutput](#)(p. 302), [GenerateTARAModeOutput](#)(p. 299), [UpdateIndivAdjustThreatInfo](#)(p. 304), [UpdateIndivAdjustCounts](#)(p. 305)

Type 61 | VerticalRAOutputState This data structure is used within the TRM to store information associated with generation of TRMGROUNDMMSGDATA and TRMRABROADCASTDATA messages.

```
1 type VerticalRAOutputState
2     ra_data::TRMRACoordinationData(p.E-28) # Setting of TRMRACoordinationData(p.E-28)
3     tgtid_tid::UInt32                      # Target id associated with the TID value
4     tti::UInt8                            # TTI setting in TRMGroundMsgData
5     tid::TID                             # TID setting in TRMGroundMsgData
6     dsi::Bool                           # DSI setting in TRMGroundMsgData
7     gnd_msg_spi::Bool                   # SPI setting in TRMGroundMsgData
8     broadcast_spi::Bool                 # SPI setting in TRMRABroadcastData
9 #
10    VerticalRAOutputState() =
11        new( TRMRACoordinationData(p.E-28)(),
12              uint32(0), TTI_ADDRESS, TIDAddress(p.E-28)(), false, false,
13              false )
14 end
```

Referenced In: TRMOwnState(p.E-32)

E.6 TRM Context-Specific State Data Structures

While some state variables are used throughout the TRM, others are specific to the algorithms in which they are used. To maintain separation of the algorithms, separate context-specific state data structures are defined for each algorithm. This appendix defines these context-specific data structures. The name of each data structure reflects the algorithm that uses it. 'CState' in the data structure name stands for Cost State. For example, ACTIONARBITRATIONCSTATE (Type 62) is the cost state data structure used within ACTIONARBITRATION (Algorithm 230). ADVISORYRESTARTCSTATE (Type 64) is the cost state data structure used within ADVISORYRESTARTCOST (Algorithm 170).

Most of the context-specific data structures are associated with an individual intruder and are used when determining online cost. They are stored in the ONLINECOSTSTATE data structure.

Other context-specific data structures are used to maintain state for specific algorithms associated with the multi-threat logic. They are stored in TRMOWNSTATE or TRMINTRUDERSTATE.

Type 62 | ActionArbitrationCState

```

1 type ActionArbitrationCState
2   action::Z          # advisory action
3   dz_min::R          # advisory minimum vertical rate (ft/s)
4   dz_max::R          # advisory maximum vertical rate (ft/s)
5   ddz::R             # advisory acceleration (ft/s/s)
6   was_worst_case::Bool # had worst case action on previous cycle
7 #
8   ActionArbitrationCState() = new( COC, -Inf, Inf, 0.0, false )
9 end
```

Referenced In: TRMINtruderStateUpdate(p. 344), TRMINtruderState(p. E-33)

Type 63 | ActionArbitrationGlobalCState

```

1 type ActionArbitrationGlobalCState
2   mtlo_prohibited_prev::Bool    # MTLO was prohibited on the previous cycle
3   worst_case_cost_prev::R       # Value of worst case cost on previous cycle
4   worst_case_action_prev::Z     # Worst case action selected on previous cycle
5 #
6   ActionArbitrationGlobalCState() = new( false, -Inf, COC )
7 end
```

Referenced In: ActionArbitration(p. 264), GenerateTAOnlyModeOutput(p. 292), NoIntrudersAction(p. 251), TRMOwnState(p. E-32), DetermineMultiIntruderAction(p. 254), ArbitrateConflictingSenses(p. 265), ActionSelection(p. 250)

Type 64 | AdvisoryRestartCState

```

1 type AdvisoryRestartCState
2   alerted::Bool    # RA generated on previous cycle
3   term::Bool        # RA ended on a previous cycle; timer is active
4   t_term::R          # Time since the RA ended (s)
5 #
6   AdvisoryRestartCState() = new( false, false, 0.0 )
7 end

```

Referenced In: UpdateAdvisoryRestartCState(p. 195), AdvisoryRestartCost(p. 194), OnlineCostState(p. E-37)

Type 65 | AltitudeDependentCOCCState

```

1 type AltitudeDependentCOCCState
2   scaled_cost_coc_ra::R      # Cost to be applied to force RAs
3   scaled_cost_coc_ta::R      # Cost to be applied to force TAs
4 #
5   AltitudeDependentCOCCState() = new( 0.0, 0.0 )
6 end

```

Referenced In: OnlineCostState(p. E-37)

Type 66 | AltitudeInhibitCState

```

1 type AltitudeInhibitCState
2   inhibited::Vector{Bool}    # Whether each altitude inhibit is in effect
3   initialized::Bool          # Whether the AltitudeInhibitCost has been initialized
4   ldi::UInt8                 # Low-level Descend Inhibits
5 #
6   AltitudeInhibitCState( params::paramsfile_type, mode_idx::Z ) = new( ones( Bool, length( params.modes[
      mode_idx].cost_estimation.online.altitude_inhibit.B_init ) ), false, LDI_ALL )
7 end

```

Referenced In: VerticalTRMUpdatePrep(p. I-2), TRMOwnState(p. E-32), UpdateAltitudeInhibitCState(p. 154), AltitudeInhibitCost(p. 197), OnlineCostState(p. E-37)

Type 67 | BadTransitionCState

```

1 type BadTransitionCState
2   dz_min_prev::R            # Min vertical rate for RA on previous cycle (ft/s)
3   dz_max_prev::R            # Max vertical rate for RA on previous cycle (ft/s)
4   sense_own_prev::Symbol    # Up/down RA sense on previous cycle
5   ra_is_maintain_prev::Bool # Maintain RA on previous cycle
6 #
7   BadTransitionCState() = new( -Inf, Inf, :None, false )
8 end

```

Referenced In: UpdateBadTransitionCState(p. 199), BadTransitionCost(p. 198), OnlineCostState(p. E-37), BadMaintainTransitionCost(p. 200)

Type 68 | CompatibilityCState

```

1 type CompatibilityCState
2   t_since_first_vrc::Z          # Time since the first VRC was received (s)
3   c_slave_init::R               # Cost of initiating an RA
4   c_slave_init_crossing::R     # Cost of initiating an RA when there is a crossing
5   c_slave_sub::R                # Cost for a subsequent RA
6   c_slave_sub_nocrossing::R    # Cost for a subsequent RA when there is no crossing
7   c_slave_sub_no_response::R   # Cost for any RA against a non-responsive intruder
8 #
9   CompatibilityCState() = new( 0, 0.0, 0.0, 0.0, 0.0, 0.0 )
10 end

```

Referenced In: [OnlineCostState\(p. E-37\)](#), [UpdateCompatibilityCState\(p. 224\)](#), [CompatibilityCost\(p. 223\)](#)

Type 69 | CoordinationDelayCState

```

1 type CoordinationDelayCState
2   t_count::Z                  # CoordinationDelay counter (s)
3   is_count_enabled::Bool      # Whether the CoordinationDelay counter should be used
4 #
5   CoordinationDelayCState( is_count_enabled::Bool ) = new( 0, is_count_enabled )
6 end

```

Referenced In: [UpdateCoordinationDelayCState\(p. 227\)](#), [OnlineCostState\(p. E-37\)](#), [IntruderPrep\(p. 165\)](#), [CoordinationDelayCost\(p. 227\)](#)

DO-385

Type 70 | CoordinatedRADeferralCState

```

1 type CoordinatedRADeferralCState
2   deferral_cost::R            # Cost to defer an RA
3 #
4   CoordinatedRADeferralCState() = new( 0.0 )
5 end

```

Referenced In: [OnlineCostState\(p. E-37\)](#), [UpdateCoordinatedRADeferralCState\(p. 226\)](#), [CoordinatedRADeferral-Cost\(p. 225\)](#)

Type 71 | CriticalIntervalProtectionCState

```

1 type CriticalIntervalProtectionCState
2   angle::R                   # Estimated angle to intruder (0-pi) (|rad|)
3   range::R                    # Estimated ground range to intruder (|ft|)
4   speed::R                   # Estimated relative ground speed of intruder (|ft/s|)
5   force_alert_prev::Bool     # Value of force_alert for last cycle
6   force_alert::Bool          # Force an alert
7   force_climbdescend::Bool   # Force a Climb or Descend
8   force_inc_climbdescend::Bool # Force an Increase Climb or Descend
9 #
10  CriticalIntervalProtectionCState() = new( NaN, NaN, NaN, false, false, false, false )
11 end

```

Referenced In: [CriticalIntervalRequiresVerticalDivergence\(p. 205\)](#), [CriticalIntervalProtectionCost\(p. 201\)](#), [OnlineCost-State\(p. E-37\)](#), [UpdateCriticalIntervalProtectionCState\(p. 203\)](#), [IsOutsideCriticalInterval\(p. 204\)](#)

Type 72 | CrossingNoAlertCState

```

1 type CrossingNoAlertCState
2   is_crossing::Bool          # Crossing is anticipated
3   is_crossing_prev::Bool      # Value of is_crossing on last cycle
4   vrc_int_prev::UInt32        # Received VRC on the previous cycle
5   is_crossing_caused_by_geometry::Bool # Crossing is due to geometry
6 #
7   CrossingNoAlertCState() = new( false, false, 0, false )
8 end

```

Referenced In: CrossingNoAlertCost(p. 228), OnlineCostState(p. E-37), UpdateCrossingNoAlertCState(p. 229)

Type 73 | InitializationCState

```

1 type InitializationCState
2   t_count::Z                 # Initialization counter (s)
3 #
4   InitializationCState() = new( 0 )
5 end

```

Referenced In: InitializationCost(p. 205), OnlineCostState(p. E-37)

Type 74 | LowAltitudeParallelRADeferralCState

```

1 type LowAltitudeParallelRADeferralCState
2   speed_rel::R                # Estimated relative ground speed of intruder aircraft (|ft/s|)
3   range::R                     # Estimated ground range to intruder aircraft (|ft|)
4   range_prev::R                # Estimated ground range to intruder from previous cycle (|ft|)
5   c_alert_deferral::R          # Cost to be applied to defer an RA
6   c_non_climb_descend::R       # Cost to be applied to any RA that is not Climb or Descend
7   for_ta::Bool                 # Costs are for TA, not RA
8 #
9   LowAltitudeParallelRADeferralCState() =
10    new( NaN, NaN, NaN, 0.0, 0.0, false )
11   LowAltitudeParallelRADeferralCState( for_ta::Bool ) =
12    new( NaN, NaN, NaN, 0.0, 0.0, for_ta )
13 end

```

Referenced In: LowAltitudeParallelRADeferralCost(p. 229), OnlineCostState(p. E-37), DetermineLowAltitudeCost(p. 232), UpdateLowAltitudeParallelRADeferralCState(p. 231)

Type 75 | MaxReversalCState

```

1 type MaxReversalCState
2   num_reversals::Z            # Count of reversals
3   crossed_thres_time::R       # Time since the number of reversals reached the limit (s)
4   sense_own_prev::Symbol      # Vertical RA up/down sense for ownship
5   sense_int::Symbol           # Vertical RA up/down sense based on intruder VRC
6 #
7   MaxReversalCState() = new( 0, 0, :None, :None )
8 end

```

Referenced In: UpdateMaxReversalCState(p. 236), MaxReversalCost(p. 235), OnlineCostState(p. E-37)

Type 76 | MultithreatCostBalancingCState

```

1 type MultithreatCostBalancingCState
2   costs_allow_mtlo_prev::Bool      # MTLO allowed based on costs on previous cycle
3   above_abs_threshold_prev::Bool    # Whether to apply hysteresis for cost threshold
4 #
5   MultithreatCostBalancingCState() = new( true, false )
6 end

```

Referenced In: GenerateTAOnlyModeOutput(p. 292), MultithreatCostBalancing(p. 263), NoIntrudersAction(p. 251), TR-MOwnState(p. E-32), ActionSelection(p. 250)

Type 77 | PreventEarlyCOCCState

```

1 type PreventEarlyCOCCState
2   range::R                      # Estimated ground range to intruder (|ft|)
3   range_prev::R                  # Estimated ground range to intruder from last cycle (|ft|)
4   t_consec_range_divergence::Z   # Time since ground range to intruder began diverging (s)
5   is_early_coc::Bool             # Issuing Clear of Conflict would be considered early
6 #
7   PreventEarlyCOCCState() = new( NaN, NaN, 0, false )
8 end

```

Referenced In: UpdatePreventEarlyCOCCState(p. 207), PreventEarlyCOCCCost(p. 206), OnlineCostState(p. E-37)

Type 78 | ResponseEstimationCState

```

1 type ResponseEstimationCState
2   t_same_sense::Z                # Amount of time at the same up/down sense (s)
3   sense_prev::Symbol             # The previous RA sense
4   dz_after_wait::R              # Observed vertical rate (ft/s)
5   is_responding_prev::Bool       # Deemed to be responsive on last cycle
6   dz_response_initial::R        # Expected initial vertical rate when responsive (ft/s)
7   dz_response_max::R            # Maximum observed vertical rate when responsive (ft/s)
8   t_dz_response_max::Z          # Value of t_same_sense when dz_response_max was observed (s)
9 #
10  ResponseEstimationCState() = new( 0, :None, NaN, true, NaN, NaN, 0 )
11 end

```

Referenced In: OwnResponseEstimation(p. 219), IntruderResponseEstimation(p. 218), DetermineResponse(p. 220), OnlineCostState(p. E-37)

Type 79 | RestrictCOCCState

```

1 type RestrictCOCCState
2   coc_prev::Bool                 # Clear of conflict on previous cycle
3   sense_own_prev::Symbol          # Up/down sense of previous global RA
4 #
5   RestrictCOCCState() = new( true, :None )
6 end

```

Referenced In: RestrictCOCDueToReversal(p. 238), UpdateRestrictCOCDueToReversalCState(p. 238), OnlineCostState(p. E-37)

Type 80 | SA01HeuristicCState

```

1 type SA01HeuristicCState
2   range::R           # Estimated ground range to intruder (|ft|)
3   force_reversal::Bool # Force a reversal
4 #
5   SA01HeuristicCState() = new( NaN, false )
6 end

```

Referenced In: UpdateSA01HeuristicCState(p. 240), OnlineCostState(p. E-37), SA01Heuristic(p. 239)

Type 81 | SafeCrossingRADeferralCState

```

1 type SafeCrossingRADeferralCState
2   c_deferral::R      # Cost to be applied to defer an RA
3 #
4   SafeCrossingRADeferralCState() = new( 0.0 )
5 end

```

Referenced In: UpdateSafeCrossingRADeferralCState(p. 209), SafeCrossingRADeferralCost(p. 208), OnlineCostState(p. E-37)

Type 82 | TimeBasedNonComplianceCState

```

1 type TimeBasedNonComplianceCState
2   current_ra_max_tau::R      # Maximum tau observed during the current advisory (s)
3   current_ra_min_tau_time::Z # Amount of time spent at or below the minimum tau (s)
4   current_ra_dz_initial_delta::R # Vertical rate difference required to be compliant when
5                                #   the advisory is issued (ft/s)
6   dz_min_prev2::R            # Minimum desired rate from advisory two cycles previous (ft/s)
7   dz_max_prev2::R            # Maximum desired rate from advisory two cycles previous (ft/s)
8   current_ra_cost::R         # Cost applied to the current advisory
9 #
10  TimeBasedNonComplianceCState() = new( 0.0, 0, 0.0, -Inf, Inf, 0.0 )
11 end

```

Referenced In: DetermineOwnNonComplianceFactor(p. 246), OnlineCostState(p. E-37), TimeBasedNonComplianceCost(p. 242), UpdateTimeBasedNonComplianceCState(p. 244)

This Page Intentionally Left Blank

Appendix F Parameter File Specification

F.1 ACAS X Parameter Data Item File (PDIF) Checksums

The ACAS X design is highly dependent on the Parameter Data Item Files (PDIFs), which are used in the ACAS X equipment, to resolve conflicts in real time. The values of all the data elements for Xa and Xo are found in the "Parameter file" PDIF. Those data element values include the names of the other PDIFs used in the ACAS X system. The data elements in the "Parameter file" PDIF and their values are described in the other two sections in this Appendix. The table below lists the PDIFs used in the ACAS X equipment and their MD5 checksums. The PDIF "Horizontal entry table for active surveillance only" is for Xo only. All other PDIFs are for both Xa and Xo.

Table F-1. Parameter Data Item File (PDIF) List

PDIF	File Name	MD5 Checksum
Minimum blocks index table	DO-385_q_31ec_minBlocks.dat	789d4dacbf1d7734005e95b8143d223
Equivalence class cost table	DO-385_q_31ec_allSplits.dat	2264d6a9e4612c4224dad854e6a8d85b
Horizontal entry table	DO-385_entry_horiz.dat	a1532cc1cddc56e753b374bce46d573f
Horizontal entry table for active surveillance only	DO-385_entry_horiz_active_cspo3k.dat	2dab51409fad3e868e4ca0b4e3eba0ec
Vertical entry table	DO-385_entry_vert.dat	a1532cc1cddc56e753b374bce46d573f
Parameter file	DO-385_parameter_file.txt	6290ebc44e3b69b868d43633bda8caeb

F.2 STM Parameters

A full specification of all the parameters used in the STM can be found in the "Parameter File" PDIF, *DO-385_parameter_file.txt*. The parameter file contains a hierarchy of structures encoded in the JavaScript Object Notation (JSON) [6]. The JSON parameter hierarchy is organized as follows:

- **coordination:** Information for coordination processing
 - **parity_table:** table used to encode the vertical sense bits of the coordination message
value: [0, 14, 7, 9, 11, 5, 12, 2, 13, 3, 10, 4, 6, 8, 1, 15]
 - **timeout:** maximum time allowed between coordination messages before timing out (s)
value: 6.0
 - **vert_intent:** map of codes in the coordination message into the vertical intent
value: [1, 2, 0]
- **display:** Information for the display logic
 - **arrow_M:** number of successful vertical rate comparisons necessary to transition the displayed rate
value: 3
 - **arrow_N:** number of vertical rate comparisons
value: 4
 - **level_arrow_threshold:** vertical rate threshold to indicate intruder is level (ft/s)
value: 10.0
 - **nars_threshold:** own ship altitude below which non-altitude reporting intruders are considered valid (ft)
value: 15500.0
 - **proximity_altitude_threshold:** relative altitude within which the intruder is considered proximate to own ship (ft)
value: 1200.0
 - **proximity_range_threshold:** relative ground range within which the intruder is considered proximate to own ship (ft)
value: 36456.69
- **surveillance:** Information about the parameters used in the STM
 - **av_threshold:** Active validation parameters
 - **M:** Number of successful comparisons to consider an ADS-B track validated
value: 2
 - **N:** Total number of comparisons
value: 3
 - **a:** Altitude validation threshold (ft)
value: 100.0
 - **init_M:** Initial M value to mark active validation state as provisional
value: 1
 - **init_N:** Initial N value to mark active validation state as provisional
value: 1
 - **r:** Range validation threshold (ft)
value: 1115.4855643
 - **correlation:** Correlation processing parameters

- **M:** Number of successful comparisons to consider correlating tracks
value: 1
- **N:** Total number of comparisons
value: 1
- **image_range_gate:** Maximum multipath range window for efficient iteration of targets (ft)
value: 20001
- **range_gate:** Maximum range window for efficient iteration (ft)
value: 5001
- **type_selection:** Parameters for different methods of correlation
 - **image:** Parameters for correlating an image track
 - altitude_gate:** Maximum altitude window for efficient iteration (ft)
value: 1001
 - altitude_inflation:** Altitude inflation to account for observation noise (ft)
value: 28.9
 - correlation_alt_threshold:** Score threshold reduction value for NAR tracks
value: 5.9
 - correlation_az_threshold:** Score threshold reduction value for bearingless tracks
value: 2
 - correlation_threshold:** Score threshold for image correlation based
value: 6
 - rangerate_gate:** Maximum range rate window for efficient iteration (ft/s)
value: 201
 - variance_scale:** Scaling factor applied to range rate uncertainty
value: 2
 - **position:** Parameters for correlating by position
 - altitude_gate:** Maximum altitude window for efficient iteration (ft)
value: 5001
 - altitude_inflation:** Altitude inflation to account for observation noise (ft)
value: 28.9
 - correlation_alt_threshold:** Score threshold reduction value for NAR tracks
value: 4
 - correlation_az_threshold:** Score threshold reduction value for bearingless tracks
value: 1
 - correlation_threshold:** Score threshold for position correlation based
value: 6
 - range_gate:** Maximum range window for efficient iteration (ft)
value: 10001
- **unique_multiplier:** Scaling applied to correlation thresholds based on track uniqueness
value: 0.5
- **unique_time:** Time threshold used to mark Target as unique (s)
value: 10
- **decorrelation:** Decorrelation parameters
 - **M:** Number of successful comparisons to consider decorrelating tracks
value: 1

- **N:** Total number of comparisons
value: 1
- **max_coast_time:** Maximum coast time before a track is decorrelated (s)
value: 1
- **unique_multiplier:** Scaling applied to decorrelation thresholds based on track uniqueness
value: 2
- **unique_time:** Time threshold used to mark Target as unique (s)
value: 10
- **horizontal:** Horizontal tracking parameters for active surveillance
 - **cartfilter:** Cartesian filter tracking parameters
 - **Q:** Process noise variance (ft^2/s^4)
value: 512.0
 - **R:** 2×2 observation noise covariance matrix (diagonal terms: ft^2 , rad^2)
value: [2500.0 0.0 0.0 0.030461742]
 - **U:** 2×2 initial velocity covariance (diagonal terms: ft^2/s^2 , ft^2/s^2)
value: [62500.0 0.0 0.0 62500.0]
 - **altitude_threshold:** Threshold for invalid slant range measurement (ft)
value: 5000.0
 - **detections_to_recover:** Number of detections needed to recover from an invalid state
value: 3
 - **gamma:** Covariance inflation factor
value: 1000
 - **kappa:** Initial covariance inflation applied to guarantee positive semidefinitene matrix
value: 0.1
 - **max_outlier_detections:** Number of outlier detections before track reinitialization in the normal (1 Hz) surveillance region
value: 4
 - **max_outlier_detections_reduced:** Number of outlier detections before track reinitialization in the reduced (0.2 Hz) surveillance region
value: 1
 - **outlier_threshold:** Mahalanobis distance threshold for outlier detection
value: 25.0
 - **rangefilter:** Range filter tracking parameters
 - **Q:** Process noise variance (ft^2/s^4)
value: 64.0
 - **R:** Observation noise variance (ft^2)
value: 2500.0
 - **U:** 2×2 initial velocity and acceleration covariance (diagonal terms: ft^2/s^2 , ft^2/s^4)
value: [160000.0 0.0 0.0 512.0]
 - **altitude_threshold:** Threshold for invalid slant range measurement (ft)
value: 5000.0
 - **dt_too_small_for_velocity:** Minimum time threshold for seeding range filter velocity by differencing two measurements (s)

-
- value: 0.3*
- **epsilon:** Threshold for equivalence to 0.0 (m)
value: 1.0e-9
 - **max_outlier_detections:** Number of outlier detections before track reinitialization in the normal (1 Hz) surveillance region
value: 2
 - **max_outlier_detections_reduced:** Number of outlier detections before track reinitialization in the reduced (0.2 Hz) surveillance region
value: 1
 - **max_zero_gr_count:** Maximum number of allowable ground range samples of zero
value: 3
 - **outlier_threshold:** Mahalanobis distance threshold for outlier detection
value: 49.0
 - **sqrt_r2_threshold:** Threshold used to check if the squared range is too close to zero (ft)
value: 10.0
 - **horizontal_adsb:** Horizontal tracking parameters for passive surveillance
 - **Q:** Process noise variance (m^2)
value: 48.0
 - **cov_inflation_factor:** Posterior state estimate covariance inflation factor
value: 1.5
 - **init_nacp:** Initial NACp value assigned to track until receipt of Mode Status Report
value: 5
 - **init_nacv:** Initial NACv value assigned to track until receipt of Mode Status Report
value: 1
 - **init_sigma_pos:** Position standard deviation used during initialization (m)
value: 378.26797385620915
 - **init_sil:** Initial SIL value assigned to track until receipt of Mode Status Report
value: 0
 - **kappa:** Stability factor added during initialization to ensure matrix is positive semidefinite
value: 1.0e-8
 - **max_outlier_detections:** Number of outlier detections before track reinitialization
value: 2
 - **outlier_threshold:** Threshold for outlier detection
value: 16.0
 - **update_sigma_pos:** Position standard deviation used during update (m)
value: 75.65359477124183
 - **vel_init_lat_thresh:** Limit of absolute latitude value below which a track can be initialized on a single position and velocity report (deg)
value: 85.0
 - **intruder_vertical:** Intruder vertical tracking parameters
 - **L:** Number of partitions
value: 20

- **Q:** Process noise variance (ft^2/s^4)
value: 5.0
- **R_100ft:** Observation noise variance for 100 ft quantization (ft^2)
value: 22.09
- **R_25ft:** Observation noise variance for 25 ft quantization (ft^2)
value: 22.09
- **asarp_max_alt:** Altitude bands used to apply altitude bias (ft)
value: [2300, 5000, 10000, 20000, 41000, 99999]
- **asarp_sigma:** Uncertainty due to altitude bias (ft)
value: [49.5, 53.7, 60.8, 65.2, 65.2, 65.2]
- **detections_to_recover:** Number of detections needed to recover from an invalid state
value: 3
- **max_outlier_detections:** Number of outlier detections before track reinitialization in the normal (1 Hz) surveillance region
value: 2
- **max_outlier_detections_reduced:** Number of outlier detections before track reinitialization in the reduced (0.2 Hz) surveillance region
value: 1
- **outlier_threshold:** Mahalanobis distance threshold for outlier detection
value: 49.0
- **outlier_window:** Static outlier window applied during second received update (ft/s)
value: 356
- **stability_exponent:** Exponent used in stability factor
value: -38
- **var_dzint:** Initial altitude rate variance (ft^2/s^2)
value: 125.0
- **var_zint:** Initial altitude variance (ft^2)
value: 73.0
- **max_duplicate_coasts:** Maximum number of coasts before a Mode C track file is deleted when there are multiple Mode C track files
value: 4
- **max_heading_coasts:** Maximum number of heading coasts before a track file is deleted
value: 5
- **max_intruders:** Maximum number of intruders sent to the TRM and display
value: 30
- **max_normal_coasts:** Maximum number of coasts under Normal surveillance before a track file is deleted
value: 6
- **max_reduced_coasts:** Maximum number of coasts under Reduced surveillance before a track file is deleted
value: 10
- **min_extrap_toa_step:** Minimum duration of time since previous track update required to extrapolate track to a TOA (s)
value: 0.01

- **min_obs_toa_step:** Minimum duration of time since previous track update required to process an observation (s)
value: 0.01
- **mode_c:** Mode C parameters
 - **association:** Mode C association parameters
 - **altitude_gate:** Maximum altitude window for efficient iteration (ft)
value: 1001
 - **altitude_inflation:** Altitude inflation to account for observation noise (ft)
value: 100
 - **association_alt_threshold:** Score threshold reduction value for NAR tracks
value: 1
 - **association_az_threshold:** Score threshold reduction value for bearingless tracks
value: 1
 - **association_threshold:** Score threshold for associating targets based on scoring
value: 6
 - **bearing_inflation:** Bearing inflation to account for observation noise (rad)
value: 0.1745
 - **bearingless_penalty:** Score penalty value for using bearingless associations
value: 2
 - **coast_penalty_multiplier:** Coast time multiplier applied as penalty to association score
value: 0.3
 - **image_penalty:** Score penalty given to Mode C image tracks
value: 5
 - **image_threshold:** Score threshold reduction value used for Image tracks
value: 3
 - **min_alt_score:** Minimum association score assigned due to altitude errors (ft)
value: 1
 - **nar_cpa_range:** Maximum range at which bearing dimension is ignored for NAR tracks (ft)
value: 15001
 - **range_gate:** Maximum range window for efficient iteration (ft)
value: 10001
 - **range_inflation:** Range inflation to account for observation noise (ft)
value: 50
- **initialize:** Mode C initialization parameters
 - **Cbits:** Bit mask for the C bit of the Gillham altitude code
value: 7
 - **DABbits:** Bit mask for DAB bits of the Gillham altitude code
value: 2040
 - **initialize_alt_threshold:** Score threshold reduction value for NAR tracks
value: 1.3
 - **initialize_az_threshold:** Score threshold reduction value for bearingless tracks
value: 1.3
 - **initialize_threshold:** Score threshold for initialization
value: 4

- **range_gate**: Maximum range window for efficient iteration (ft)
value: 6001
- **rangerate_gate**: Maximum range rate window for efficient iteration (ft/s)
value: 2026
- **weight_Cbit**: Weighting factor to assess altitude code agreement in the C bit
value: 0.2
- **weight_DAB**: Weighting factor to assess altitude code agreement in the DAB bits
value: 1.0
- **merging**: Mode C merging parameters
 - **altitude_sigma**: Uncertainty in the altitude measurement used in the scoring process (ft)
value: 100
 - **chi_sigma**: Uncertainty in the bearing measurement used in the scoring process (rad)
value: 0.1745
 - **merging_alt_threshold**: Score threshold reduction value for NAR tracks
value: 1
 - **merging_az_threshold**: Score threshold reduction value for bearingless tracks
value: 1
 - **merging_duplicate_selector**: Factor in determining merged reply selection
value: 3
 - **merging_threshold**: Threshold for merging based on scoring
value: 3
- **range_bias**: Range uncertainty due to beam to beam interrogation timing (ft)
value: 100
- **range_gate**: Maximum range window for efficient iteration (ft)
value: 501
- **range_sigma**: Uncertainty in the range measurement used in the scoring process (ft)
value: 50
- **on_ground**: Mode C on-ground determination parameters
 - **alt_evaluate_in_air**: Above this intruder height above ground, the transition from on-ground to in-air is considered (ft)
value: 250.0
 - **alt_evaluate_on_ground**: The entire intruder vertical track history must be below this value in order to consider transitioning from in-air to on-ground (ft)
value: 360.0
 - **max_alt_on_ground**: Above this intruder height above the ground, the intruder is automatically considered in the air (ft)
value: 400.0
 - **max_height_above_min_on_ground**: The difference between maximum and minimum intruder altitude in the measurement history must be above this value in order to consider transitioning to in-air. The difference must be below (or equal to) this value to consider transitioning to on-ground (ft)
value: 100.0

- **max_rad_alt:** Maximum own radio altitude that is considered credible for ground level calculation (ft)
value: 1700.0
- **min_alt_in_air:** Below this intruder height above the ground, the intruder is automatically considered on the ground (ft)
value: 150.0
- **min_num_after_pos_transition:** Number of updates after a positive transition to consider the positive transition as having persisted
value: 3
- **rad_alt_hysteresis:** Hysteresis margin applied to maximum own radio altitude considered credible for ground level calculation (ft)
value: 50.0
- **vert_history_window:** Length of time over which the vertical measurement and track histories are stored (s)
value: 20
- **promote:** Mode C promotion parameters
 - **altitude_sigma:** Uncertainty in the altitude measurement used in the scoring process (ft)
value: 40.9
 - **chi_sigma:** Uncertainty in the bearing measurement used in the scoring process (rad)
value: 0.1745
 - **promote_alt_threshold:** Score threshold reduction value for NAR tracks
value: 1.3
 - **promote_az_threshold:** Score threshold reduction value for bearingless tracks
value: 1.3
 - **promote_limit:** Age limit to initiate hypothetical track deletion (s)
value: 3
 - **promote_threshold:** Score threshold for promotion based on scoring
value: 4
 - **range_gate:** Maximum range window for efficient iteration (ft)
value: 12001
 - **range_sigma:** Uncertainty in the range measurement used in the scoring process (ft)
value: 50
- **own_history_window:** Time window in seconds to indicate how long to save ownship history (s)
value: 10
- **own_wgs84_timeout:** Time duration after which the ownship WGS84 information is no longer considered valid (s)
value: 3.0
- **ownship_heading:** Ownship heading tracking parameters
 - **Q:** Process noise variance (rad^2/s^4)
value: 4.759647184167321e-6
 - **R:** Observation noise variance (rad^2)
value: 0.00030461741978670857

- **max_outlier_detections:** Number of outlier detections before track reinitialization
value: 4
- **outlier_threshold:** Mahalanobis distance threshold for outlier detection
value: 1000.0
- **var_dhint:** Initial heading rate variance (rad^2/s^2)
value: 0.0010153913992890287
- **ownship_vertical:** Ownship vertical tracking parameters
 - **Q:** Process noise variance (ft^2/s^4)
value: 15.5236
 - **R:** Observation noise variance (ft^2)
value: 22.09
- **max_invalid_radalt_cycles:** Maximum number of invalid radio altitude observations
value: 2
- **outlier_threshold:** Mahalanobis distance threshold for outlier detection / credibility determination
value: 100.0
- **var_dzint:** Initial altitude rate variance (ft^2/s^2)
value: 2500.0
- **var_zint:** Initial altitude variance (ft^2)
value: 0.0833333333
- **psd_stability_factor:** Stability factor to ensure matrices are positive semidefinite during Upper Cholesky Decomposition
value: 1.0e-8
- **report_generation:** Report generation parameters
 - **min_adsb_quality:** Minimum ADS-B Data Quality Parameters
 - **M:** Number of successful ADS-B data quality check comparisons needed to consider an ADS-B track validated
value: 2
 - **N:** Total number of comparisons
value: 3
 - **adsb_version:** Minimum allowable ADS-B version
value: 2
 - **nacp:** Minimum NACp value
value: 7
 - **nacv:** Minimum NACv value
value: 1
 - **nic:** Minimum NIC value
value: 6
 - **passive_only:** Quality parameters which apply to passive-only targets
 - **sda:** Minimum SDA value
value: 1
 - **sil:** Minimum SIL value
value: 3
- **target_designation:** Xo target designation parameters

- **N_targets_limit:** maximum number of targets that can be designated at one time
value: 1
- **S_standard_protection_mode:** tables and parameters to be used by TRM when target is not designated for Xo processing. Registered values are listed in Global Constants. 1 for "PROTECTION_MODE_Xa".
value: 1
- **T_invalid_limit:** expiration time for invalid timer (s)
value: 30
- **td_cspo3000:** parameters specific to Xo CSPO-3000 designation
 - **H_threshold_hi:** altitude above which invalid timer is active for target designated Xo CSPO-3000 (ft)
value: 14500.0
 - **H_threshold_lo:** altitude above which Xo CSPO-3000 is not valid for designation (ft)
value: 14000.0
- **td_dna:** parameters specific to Xo Designated No Alerts (DNA)
 - **D_rng_threshold_hi:** ground range beyond which the target designated DNA can be automatically undesignated (ft)
value: 36456.84
 - **D_rng_threshold_lo:** ground range beyond which DNA is not valid for designation (ft)
value: 36456.69
 - **M:** Number of comparisons to decide Target track is diverging
value: 4
 - **N:** Total number of ground ranges for comparison
value: 5

F.3 TRM Parameters

A full specification of all the parameters used in the threat resolution module can be found in the "Parameter File" PDIF, *DO-385_parameter_file.txt*. The parameter file contains a hierarchy of structures encoded in the JavaScript Object Notation (JSON) [6]. Several special symbols mentioned in the Julia code, like *Inf* (positive infinity), *-Inf* (negative infinity), and *NaN* (not a number), are encoded in JSON using strings. In particular, *Inf* is represented by the string “*_Inf_*”, *-Inf* by “*-_Inf_*”, and *NaN* by “*_NaN_*”.

The JSON parameter hierarchy is organized as follows. The abbreviation "COC" is used in the definitions for "clear of conflict"; which is also treated as a synonym for "no advisory". The abbreviation "MTLO" is used in the definitions for "multi-threat level-off".

- **actions:** Information about the resolution advisories represented in the cost table. When a subfield description does not provide a value, the value is provided in table(s) following the last subfield.
 - **accelerations:** vector of accelerations that are compliant with each resolution advisory. Actual acceleration is dependent upon equipment capabilities and pilot response to the advisory. (ft/s^2). Values are listed in table(s) below.
 - **corrective_rate:** magnitude of target vertical rate for corrective advisories (ft/s)
value: 25.0
 - **initial_acceleration:** acceleration associated with the first advisory against a given intruder (ft/s^2)
value: 8.0435
 - **max_rates:** vector of maximum vertical rates that are compliant with each resolution advisory. The maximum vertical rate for a maintain rate advisory is represented as NaN. (ft/s). Values are listed in table(s) below.
 - **min_rates:** vector of minimum vertical rates that are compliant with each resolution advisory. The minimum vertical rate for a maintain rate advisory is represented as NaN. (ft/s). Values are listed in table(s) below.
 - **names:** vector of the possible resolution advisories. The first advisory is always COC. Values are listed in table(s) below.
 - **num_actions:** total number of advisory actions
value: 11
 - **strengthen_rate:** magnitude of target vertical rate for strengthening advisories (ft/s)
value: 41.6667
 - **subsequent_acceleration:** acceleration associated with all corrective advisories following the first advisory against a given intruder (ft/s^2)
value: 10.7247

Table F-2. actions

no.	names	min_rates	max_rates	accelerations
1	CO	-Inf	Inf	0
2	DNC	-Inf	0	8.0435
3	DND	0	Inf	8.0435
4	MAINTAIN	NaN	NaN	8.0435
5	DES1500	-Inf	-25	8.0435
6	CL1500	25	Inf	8.0435
7	SDES1500	-Inf	-25	10.7247
8	SCL1500	25	Inf	10.7247
9	SDES2500	-Inf	-41.6667	10.7247
10	SCL2500	41.6667	Inf	10.7247
11	MTLO	-4.1667	4.1667	8.0435

- **coordination:** Information for the coordination selection
 - **parity_table:** table used to encode the vertical sense bits of the coordination message
value: [0, 14, 7, 9, 11, 5, 12, 2, 13, 3, 10, 4, 6, 8, 1, 15]
 - **display:** Information for the display logic
 - **crossthri:** upper threshold for vertical separation during crossing, at which crossing bit is first set (ft)
value: 100.0
 - **crossthlo:** lower threshold for vertical separation during crossing, above which crossing bit is maintained once set (ft)
value: -100.0
 - **designatedscore:** track display score: base score for designated intruder when there is no traffic advisory or resolution advisory
value: 300.0
 - **hdescendthr:** label270rules.altinhibit: altitude at and below which default annunciation is overridden (ft)
value: 1000.0
 - **hiscore:** track display score: score for intruder-specific resolution advisory
value: 1200.0
 - **hnoauralhi:** upper altitude threshold for aural inhibits (ft)
value: 600.0
 - **hnoaurallo:** lower altitude threshold for aural inhibits (ft)
value: 400.0
 - **label270rules:** Rules for setting the Label 270 values. A single rule is obtained using the same index into all vectors. A value of NaN means any value will match. The rules are evaluated in order, with the first matching index giving the values of cc, vc, ua, and da that will be used. When a subfield description does not provide a value, the value is provided in table(s) following the last subfield.
 - **action:** vector of current advisory action values. Names associated with the values are given in actions.names. Values are listed in table(s) below.
 - **altinhibit:** vector indicating whether hdescendthr is taken into account. Values are listed in table(s) below.

- **cc**: vector of combined control values (output). Values are listed in table(s) below.
- **crossing**: vector of crossing flags. Values are listed in table(s) below.
- **da**: vector of down advisory values (output). Values are listed in table(s) below.
- **forcealarm**: vector of force alarm values for explicit control of alarm flag, and therefore annunciation, status (e.g., force Level-Off Level-Off annunciation after Monitor Vertical Speed for Do Not Climb advisory). Values are listed in table(s) below.
- **hidz**: vector of upper vertical rate limit values (ft/s). Values are listed in table(s) below.
- **lodz**: vector of lower vertical rate limit values (ft/s). Values are listed in table(s) below.
- **prevaction**: vector of previous advisory action values. Names associated with the values are given in actions.names. Values are listed in table(s) below.
- **prevstrength**: vector of previous Vertical Active RA strength values. Values are listed in table(s) below.
- **prevword**: vector of previous word values (Label 270). A word value is given by $cc*1000+vc*100+ua*10+da$. Values are listed in table(s) below.
- **strength**: vector of Vertical Active RA strength values for output in RA ground messages (output). Values are listed in table(s) below.
- **ua**: vector of up advisory values (output). Values are listed in table(s) below.
- **vc**: vector of vertical control values (output). Values are listed in table(s) below.

Table F-3. display,label270rules (1 of 4)

no.	action	prevaction	prevword	lodz	hidz	crossing	altinhibit	cc	vc	ua	da	forcealarm	prevstrength	strength
1	1	1	1	Nan	Nan	Nan	Nan	0	0	0	0	Nan	0	0
2	2	1	Nan	Nan	Nan	Nan	Nan	1	0	0	0	Nan	0	0
3	3	2	5	Nan	Nan	Nan	Nan	1	6	0	2	0	Nan	10
4	4	2	7	Nan	Nan	Nan	Nan	1	6	0	0	2	0	Nan
5	5	2	9	Nan	Nan	Nan	Nan	1	6	0	0	2	0	Nan
6	6	2	1	Nan	-Inf	Nan	8.333333333	Nan	6	0	0	2	0	Nan
7	7	2	4	5401	Nan	Nan	Nan	6	0	0	2	0	0	Nan
8	8	2	4	5201	Nan	Nan	Nan	6	0	0	2	0	0	Nan
9	9	2	Nan	5002	Nan	Nan	Nan	5	0	0	2	0	0	2
10	10	2	Nan	5002	Nan	Nan	Nan	5	0	0	2	0	0	3
11	11	2	Nan	5002	Nan	Nan	Nan	5	0	0	2	0	0	3
12	12	2	Nan	6002	-Inf	12.5	Nan	6	0	0	2	0	0	10
13	13	2	Nan	6002	-Inf	12.5	Nan	6	0	0	2	0	0	1
14	14	2	Nan	6002	Nan	Nan	Nan	5	0	0	2	1	0	3
15	15	2	3	Nan	-Inf	8.333333333	Nan	6	0	0	2	0	0	Nan
16	16	2	4	Nan	-Inf	8.333333333	Nan	6	0	0	2	0	0	Nan
17	17	2	6	Nan	-Inf	8.333333333	Nan	6	0	0	2	0	0	Nan
18	18	2	8	Nan	-Inf	8.333333333	Nan	6	0	0	2	0	0	Nan
19	19	2	10	Nan	-Inf	8.333333333	Nan	6	0	0	2	0	0	Nan
20	20	2	Nan	-8.333333333	8.333333333	Nan	Nan	6	0	0	2	0	0	Nan
21	21	2	1	Nan	Nan	Nan	Nan	5	0	0	2	0	0	Nan
22	22	2	5	Nan	Nan	Nan	Nan	5	0	0	2	0	0	Nan
23	23	2	7	Nan	Nan	Nan	Nan	5	0	0	2	0	0	Nan
24	24	2	9	Nan	Nan	Nan	Nan	5	0	0	2	0	0	Nan
25	25	2	11	Nan	-Inf	8.333333333	Nan	6	0	0	2	0	0	Nan
26	26	2	11	Nan	Nan	Nan	Nan	5	0	0	2	0	0	Nan
27	27	2	Nan	Nan	Nan	Nan	Nan	5	0	0	2	0	0	Nan
28	28	3	1	Nan	-8.333333333	Inf	Nan	6	0	2	0	0	0	Nan
29	29	3	4	4410	Nan	Nan	Nan	6	0	2	0	0	0	Nan
30	30	3	4	4210	Nan	Nan	Nan	6	0	2	0	0	0	Nan
31	31	3	Nan	4020	Nan	Nan	Nan	4	0	2	0	0	0	2
32	32	3	Nan	4020	Nan	Nan	Nan	4	0	2	0	0	0	3

Table F-4. display,label270rules (2 of 4)

no.	action	prevaction	prevword	lodz	hidz	crossing	altinhibit	cc	vc	ua	da	forcealarm	prestrength	strength
33	3	NaN	4020	NaN	NaN	NaN	NaN	4	0	2	0	0	9	3
34	3	NaN	6020	-12.5	Inf	NaN	NaN	6	0	2	0	0	NaN	1
35	3	NaN	6020	NaN	NaN	NaN	NaN	4	0	2	0	1	NaN	3
36	3	2	NaN	-8.3333333333	Inf	NaN	NaN	6	0	2	0	0	NaN	11
37	3	4	NaN	-8.3333333333	Inf	NaN	NaN	6	0	2	0	0	NaN	11
38	3	5	NaN	-8.3333333333	Inf	NaN	NaN	6	0	2	0	0	NaN	11
39	3	7	NaN	-8.3333333333	Inf	NaN	NaN	6	0	2	0	0	NaN	11
40	3	9	NaN	-8.3333333333	Inf	NaN	NaN	6	0	2	0	0	NaN	11
41	3	NaN	NaN	-8.3333333333	8.3333333333	NaN	NaN	6	0	2	0	0	NaN	1
42	3	1	NaN	NaN	NaN	NaN	NaN	4	0	2	0	0	NaN	3
43	3	6	NaN	NaN	NaN	NaN	NaN	4	0	2	0	0	NaN	2
44	3	8	NaN	NaN	NaN	NaN	NaN	4	0	2	0	0	NaN	2
45	3	10	NaN	NaN	NaN	NaN	NaN	4	0	2	0	0	NaN	2
46	3	11	NaN	-8.3333333333	Inf	NaN	NaN	6	0	2	0	0	NaN	1
47	3	11	NaN	NaN	NaN	NaN	NaN	4	0	2	0	0	NaN	3
48	3	NaN	NaN	NaN	NaN	NaN	NaN	4	0	2	0	0	NaN	9
49	4	2	NaN	0	Inf	NaN	NaN	4	2	1	0	0	NaN	8
50	4	5	NaN	0	Inf	NaN	NaN	4	2	1	0	0	NaN	8
51	4	7	NaN	0	Inf	NaN	NaN	4	2	1	0	0	NaN	8
52	4	9	NaN	0	Inf	NaN	NaN	4	2	1	0	0	NaN	8
53	4	3	NaN	-Inf	0	NaN	NaN	5	2	0	1	0	NaN	8
54	4	6	NaN	-Inf	0	NaN	NaN	5	2	0	1	0	NaN	8
55	4	8	NaN	-Inf	0	NaN	NaN	5	2	0	1	0	NaN	8
56	4	10	NaN	-Inf	0	NaN	NaN	5	2	0	1	0	NaN	8
57	4	4	4210	NaN	NaN	NaN	NaN	4	4	1	0	0	NaN	7
58	4	NaN	4410	NaN	NaN	NaN	NaN	4	4	1	0	0	NaN	7
59	4	4	5201	NaN	NaN	NaN	NaN	5	4	0	1	0	NaN	7
60	4	NaN	5401	NaN	NaN	NaN	NaN	5	4	0	1	0	NaN	7
61	4	NaN	0	Inf	NaN	NaN	NaN	4	4	1	0	0	NaN	7
62	4	NaN	NaN	NaN	NaN	NaN	NaN	5	4	0	1	0	NaN	7
63	5	3	NaN	NaN	NaN	NaN	NaN	5	2	0	1	0	NaN	5
64	5	6	NaN	NaN	NaN	NaN	NaN	5	2	0	1	0	NaN	5

Table F-5. display,label270rules (3 of 4)

no.	action	prevaction	prevword	lodz	hidz	crossing	altinhibit	cc	vc	ua	da	forcealarm	prevstrength	strength
65	5	8	Nan	Nan	Nan	Nan	Nan	5	2	0	1	0	Nan	5
66	5	10	Nan	Nan	Nan	Nan	Nan	5	2	0	1	0	Nan	5
67	5	NaN	4410	Nan	Nan	Nan	Nan	5	2	0	1	0	Nan	5
68	5	4	4210	Nan	Nan	Nan	Nan	5	2	0	1	0	Nan	5
69	5	NaN	5101	Nan	Nan	1	Nan	5	1	0	1	2	Nan	4
70	5	NaN	5201	Nan	Nan	1	Nan	5	1	0	1	2	Nan	4
71	5	NaN	Nan	Nan	Nan	1	Nan	5	1	0	1	1	Nan	4
72	5	7	Nan	Nan	Nan	Nan	Nan	5	0	0	1	2	Nan	4
73	5	NaN	Nan	Nan	Nan	Nan	Nan	5	0	0	1	0	Nan	5
74	6	2	Nan	Nan	Nan	Nan	Nan	4	2	1	0	0	Nan	5
75	6	5	Nan	Nan	Nan	Nan	Nan	4	2	1	0	0	Nan	5
76	6	7	Nan	Nan	Nan	Nan	Nan	4	2	1	0	0	Nan	5
77	6	9	Nan	Nan	Nan	Nan	Nan	4	2	1	0	0	Nan	5
78	6	NaN	5401	Nan	Nan	Nan	Nan	4	2	1	0	0	Nan	5
79	6	4	5201	Nan	Nan	Nan	Nan	4	2	1	0	0	Nan	5
80	6	NaN	4110	Nan	Nan	1	Nan	4	1	1	0	2	Nan	4
81	6	NaN	4210	Nan	Nan	1	Nan	4	1	1	0	2	Nan	4
82	6	NaN	Nan	Nan	Nan	1	Nan	4	1	1	0	1	Nan	4
83	6	8	Nan	Nan	Nan	Nan	Nan	4	0	1	0	2	Nan	4
84	6	NaN	Nan	Nan	Nan	Nan	Nan	4	0	1	0	0	Nan	4
85	7	3	Nan	Nan	Nan	Nan	Nan	5	2	0	1	0	Nan	5
86	7	6	Nan	Nan	Nan	Nan	Nan	5	2	0	1	0	Nan	5
87	7	8	Nan	Nan	Nan	Nan	Nan	5	2	0	1	0	Nan	5
88	7	10	Nan	Nan	Nan	Nan	Nan	5	2	0	1	0	Nan	5
89	7	NaN	4410	Nan	Nan	Nan	Nan	5	2	0	1	0	Nan	5
90	7	4	4210	Nan	Nan	Nan	Nan	5	2	0	1	0	Nan	5
91	7	NaN	5101	Nan	1	Nan	5	1	0	1	2	Nan	4	
92	7	NaN	5201	Nan	Nan	1	Nan	5	1	0	1	2	Nan	4
93	7	NaN	Nan	Nan	Nan	1	Nan	5	1	0	1	1	Nan	4
94	7	5	Nan	Nan	Nan	Nan	Nan	5	0	0	1	2	Nan	4
95	7	NaN	Nan	Nan	Nan	Nan	Nan	5	0	0	1	0	Nan	4
96	8	2	Nan	Nan	Nan	Nan	Nan	4	2	1	0	0	Nan	5

Table F-6. display.label270rules (4 of 4)

no.	action	prevaction	prevword	lodz	hdz	crossing	altinhibit	cc	vc	ua	da	forcealarm	prestrength	strength
97	8	5	NaN	NaN	NaN	NaN	NaN	4	2	1	0	0	NaN	5
98	8	7	NaN	NaN	NaN	NaN	NaN	4	2	1	0	0	NaN	5
99	8	9	NaN	NaN	NaN	NaN	NaN	4	2	1	0	0	NaN	5
100	8	NaN	5401	NaN	NaN	NaN	NaN	4	2	1	0	0	NaN	5
101	8	4	5201	NaN	NaN	NaN	NaN	4	2	1	0	0	NaN	5
102	8	NaN	4110	NaN	NaN	1	NaN	4	1	1	0	2	NaN	4
103	8	NaN	4210	NaN	NaN	1	NaN	4	1	1	0	2	NaN	4
104	8	NaN	NaN	NaN	NaN	1	NaN	4	1	1	0	1	NaN	4
105	8	6	NaN	NaN	NaN	NaN	NaN	4	0	1	0	2	NaN	4
106	8	NaN	NaN	NaN	NaN	NaN	NaN	4	0	1	0	0	NaN	4
107	9	NaN	NaN	NaN	NaN	NaN	NaN	5	3	0	1	0	NaN	6
108	10	NaN	NaN	NaN	NaN	NaN	NaN	4	3	1	0	0	NaN	6
109	11	NaN	4022	NaN	NaN	NaN	NaN	4	0	2	2	0	NaN	15
110	11	NaN	5022	NaN	NaN	NaN	NaN	5	0	2	2	0	NaN	15
111	11	NaN	6422	NaN	NaN	NaN	NaN	6	4	2	2	0	NaN	14
112	11	NaN	-Inf	-8.3333333333	NaN	NaN	NaN	4	0	2	2	0	NaN	15
113	11	NaN	8.3333333333	Inf	NaN	NaN	NaN	5	0	2	2	0	NaN	15
114	11	NaN	NaN	NaN	NaN	6	4	2	2	0	0	NaN	14	

Xo value: [1.0 5.0 -5.0 11.0 15.0 -15.0 15.0 -15.0 23.0 -23.0 28.0 2.0 6.0 -7.0
 6.0 16.0 -17.0 16.0 -17.0 24.0 -17.0 -7.0 -2.0 7.0 -6.0 -6.0 17.0 -16.0
 17.0 -24.0 7.0 3.0 8.0 -8.0 12.0 18.0 -18.0 18.0 -18.0 25.0 -25.0 29.0 4.0 0.0 0.0
 0.0 19.0 0.0 0.0 0.0 0.0 0.0 0.0 -4.0 0.0 0.0 0.0 0.0 -19.0 0.0 0.0 0.0 0.0 0.0 0.0
 9.0 -10.0 13.0 0.0 -20.0 22.0 -20.0 0.0 -26.0 30.0 0.0 10.0 -9.0 -13.0 20.0 0.0
 20.0 -22.0 26.0 0.0 -30.0 0.0 0.0 0.0 14.0 21.0 0.0 21.0 0.0 27.0 0.0 0.0 0.0 0.0
 0.0 -14.0 0.0 -21.0 0.0 -21.0 0.0 -27.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
 31.0]

equiv_class_table: name of the equivalence class tables file

Xa value: DO-385_q_31ec_allSplits.dat

Xo value: DO-385_q_31ec_allSplits.dat

minblocks_index: indexes into the minblocks table, 1-based. 0 indicates no mapping

Xa value: [1.0 7.0 13.0 19.0 27.0 34.0 41.0 48.0 55.0 61.0 67.0 2.0 8.0 14.0 20.0
 28.0 35.0 42.0 49.0 56.0 62.0 68.0 3.0 9.0 15.0 21.0 29.0 36.0 43.0 50.0 57.0
 63.0 69.0 4.0 10.0 16.0 22.0 30.0 37.0 44.0 51.0 58.0 64.0 70.0 5.0 0.0 0.0 0.0
 31.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 11.0
 17.0 23.0 0.0 39.0 45.0 52.0 0.0 65.0 71.0 0.0 12.0 18.0 24.0 32.0 0.0 46.0 53.0
 59.0 0.0 72.0 0.0 0.0 0.0 25.0 33.0 0.0 47.0 0.0 60.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
 40.0 0.0 54.0 0.0 66.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 73.0]

Xo value: [1.0 7.0 13.0 19.0 27.0 34.0 41.0 48.0 55.0 61.0 67.0 2.0 8.0 14.0 20.0
 28.0 35.0 42.0 49.0 56.0 62.0 68.0 3.0 9.0 15.0 21.0 29.0 36.0 43.0 50.0 57.0
 63.0 69.0 4.0 10.0 16.0 22.0 30.0 37.0 44.0 51.0 58.0 64.0 70.0 5.0 0.0 0.0 0.0
 31.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 11.0
 17.0 23.0 0.0 39.0 45.0 52.0 0.0 65.0 71.0 0.0 12.0 18.0 24.0 32.0 0.0 46.0 53.0
 59.0 0.0 72.0 0.0 0.0 0.0 25.0 33.0 0.0 47.0 0.0 60.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
 40.0 0.0 54.0 0.0 66.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 73.0]

minblocks_table: name of the minblocks table file

Xa value: DO-385_q_31ec_minBlocks.dat

Xo value: DO-385_q_31ec_minBlocks.dat

- **online:** Online cost estimation parameters

- **SA01:** SA01Heuristic cost parameters

C_coc: cost for COC when the SA01 heuristic determines a reversal is needed

Xa value: 201.0

Xo value: 201.0

C_non_reversal: cost for non-reversal advisories when the SA01 heuristic determines a reversal is needed

Xa value: 201.0

Xo value: 201.0

D_range_threshold: horizontal separation below which a reversal cannot be considered if it would induce an altitude crossing (ft)

Xa value: 3000.0

Xo value: 3000.0

H_rel_threshold: relative vertical separation above which a reversal cannot be considered if it would induce an altitude crossing (ft)

Xa value: 400.0

Xo value: 400.0

R_acceleration: assumed acceleration of aircraft used in the linear approximation (ft/s²)

Xa value: 10.7247

Xo value: 10.7247

R_min: the minimum rate that both aircraft need to have for the SA01 heuristic to be activated (ft/s)

Xa value: 20.0

Xo value: 20.0

R_standard: vertical speed of aircraft allowed when reversing the advisory in the linear approximation (ft/s)

Xa value: 25.0

Xo value: 25.0

R_strengthen: vertical speed of aircraft allowed when maintaining the original advisory in the linear approximation (ft/s)

Xa value: 41.6667

Xo value: 41.6667

T_delay: time delay in response used for the linear approximation (s)

Xa value: 3.0

Xo value: 3.0

T_max: maximum time until closest approach that the SA01 heuristic should consider a reversal (s)

Xa value: 35.0

Xo value: 35.0

T_min: minimum time until closest approach that the SA01 heuristic should consider a reversal (s)

Xa value: 5.0

Xo value: 5.0

- **advisory_restart:** AdvisoryRestartCost parameters

C_advisory: cost for restarting alert after having terminated for fewer than N_restart

Xa value: 0.05

Xo value: 0.05

T_limit: minimum number of cycles for which an alert must be terminated (s)

Xa value: 10

Xo value: 10

- **altitude_dependent_coc:** AltitudeDependentCOCCost parameters

C_coc: cost for issuing a clear of conflict or preventive advisory

Xa value: 1.0

Xo value: 1.0

H_rel_highalt_rolloff: ramp down interval for H_rel_highalt_threshold (ft)

Xa value: 75.0

Xo value: 75.0

H_rel_highalt_threshold: minimum proximate separation for applying cost at high altitudes (ft)

Xa value: 900.0

Xo value: 900.0

H_rel_rolloff_proximate: ramp down interval for H_rel_threshold_proximate (ft)

Xa value: 250.0

Xo value: 250.0

H_rel_rolloff_vrrf: ramp up interval for H_rel_threshold_vrrf (ft)

Xa value: 150.0

Xo value: 150.0

H_rel_threshold_proximate: minimum proximate separation for applying cost (ft)

Xa value: 600.0

Xo value: 600.0

H_rel_threshold_vrrf_hi: separation for altitudes above H_threshold_hi at which to begin applying vertical rate reduction factor (ft)

Xa value: 750.0

Xo value: 750.0

H_rel_threshold_vrrf_lo: separation for altitudes at or below H_threshold_hi at which to begin applying vertical rate reduction factor (ft)

Xa value: 300.0

Xo value: 300.0

H_threshold_hi: altitude of own aircraft at or above which the full cost is applied, below this altitude the cost is adjusted based on the closeness of own aircraft to this altitude relative to H_threshold_lo (ft)

Xa value: 20000.0

Xo value: 20000.0

H_threshold_lo: altitude of own aircraft above which cost can be applied (ft)

Xa value: 5000.0

Xo value: 5000.0

R_own_rolloff_vrrf: ramp down interval for R_own_threshold_vrrf (ft/s)

Xa value: 8.3333333333

Xo value: 8.3333333333

R_own_threshold_vrrf: own aircraft altitude at which to apply vertical rate reduction factor (ft/s)

Xa value: 8.3333333333

Xo value: 8.3333333333

R_rolloff_lolo: ramp down interval for R_threshold_lolo (ft/s)

Xa value: 8.3333333333

Xo value: 8.3333333333

R_threshold_lolo: vertical rate threshold for further scaling the cost when a preventive would cause a level-off (ft/s)

Xa value: 12.5

Xo value: 12.5

T_proximate: time value defining 'proximate' for determining proximate loss of separation (s)

Xa value: 7.5

Xo value: 7.5

T_rolloff: ramp down interval for T_threshold (s)

Xa value: 7.0

Xo value: 7.0

T_threshold: tau value below which the full cost can be applied (s)

Xa value: 24.7

Xo value: 24.7

X_maintain_factor: factor applied to the clear of conflict penalty to apply to maintains

Xa value: 0.4

Xo value: 0.4

X_proximate_to_actual_factor: factor applied to the proximate separation thresholds for use in the actual separation checks

Xa value: 1.98

Xo value: 1.98

X_vrrf: vertical rate reduction scale factor

Xa value: 0.25

Xo value: 0.25

- **altitude_inhibit:** Online cost parameters for the altitude inhibit rules; a negative cost for one rule is used as an exclusion from another rule. When a subfield description does not provide a value, the value is provided in table(s) following the last subfield.

B_init: initial setting of the altitude inhibit flag for each altitude inhibit rule. Values are listed in table(s) below.

C_inhibit: vector of costs for each altitude inhibit rule. Values are listed in table(s) below.

H_hi: vector of upper hysteresis bound thresholds for each altitude inhibit rule; altitude at which the resolution advisory ceases to be inhibited (ft). Values are listed in table(s) below.

H_lo: vector of lower hysteresis bound thresholds for each altitude inhibit rule; altitude at which the resolution advisory first becomes inhibited (ft). Values are listed in table(s) below.

R_dz_hi: vector of upper vertical rates for each altitude inhibit rule; with dz_lo, specifies resolution advisories to be inhibited (ft/s). Values are listed in table(s) below.

R_dz_hi_prev: vector of previous upper vertical rates for each altitude inhibit rule; with prev_dz_lo, specifies previously issued resolution advisories (ft/s). Values are listed in table(s) below.

R_dz_lo: vector of lower vertical rates for each altitude inhibit rule; with dz_hi, specifies resolution advisories to be inhibited (ft/s). Values are listed in table(s) below.

R_dz_lo_prev: vector of previous lower vertical rates for each altitude inhibit rule; with prev_dz_hi, specifies previously issued resolution advisories (ft/s). Values are listed in table(s) below.

Table F-7. modes[n].cost_estimation.online.altitude_inhibit - Xa

no.	H_lo	H_hi	R_dz_lo	R_dz_hi	R_dz_lo_prev	R_dz_hi_prev	B_init	C_inhibit
1	1450	1650	-Inf	-40.0000333333	-Inf	-23.3333333333	1	100
2	1450	1650	-Inf	-40.0000333333	-Inf	-40.0000333333	1	-100
3	1000	1200	-Inf	-23.3333333333	-Inf	Inf	1	100
4	1000	1200	-1.6666666667	Inf	-Inf	-23.3333333333	1	100
5	900	1100	-Inf	Inf	-Inf	Inf	0	100

Table F-8. modes[n].cost_estimation.online.altitude_inhibit - Xo

no.	H_lo	H_hi	R_dz_lo	R_dz_hi	R_dz_lo_prev	R_dz_hi_prev	B_init	C_inhibit
1	1450	1650	-Inf	-40.0000333333	-Inf	-23.3333333333	1	100
2	1450	1650	-Inf	-40.0000333333	-Inf	-40.0000333333	1	-100
3	1000	1200	-Inf	-23.3333333333	-Inf	Inf	1	100
4	1000	1200	-1.6666666667	Inf	-Inf	-23.3333333333	1	100
5	900	1100	-Inf	Inf	-Inf	Inf	0	100

- **bad_transition:** BadTransitionCost parameters
 - C_bad_maintain_initiation:** huge cost to prevent initiation of maintain when ownship vertical rate is too low
 Xa value: $1.0e10$
 Xo value: $1.0e10$
 - C_bad_transition:** cost for advisories that would result in illegal state transitions
 Xa value: 10.0
 Xo value: 10.0
- **compatibility:** CompatibilityCost parameters
 - C_master:** cost received for issuing action incompatible with master aircraft
 Xa value: $1.0e10$
 Xo value: $1.0e10$
 - C_slave_init:** cost received for issuing action incompatible with slave aircraft for first advisory
 Xa value: $1.0e10$
 Xo value: $1.0e10$
 - C_slave_init_crossing:** cost received for issuing action incompatible with slave aircraft
 Xa value: 0.1
 Xo value: 0.1
 - C_slave_sub:** cost received for issuing action incompatible with slave aircraft after an initial advisory
 Xa value: 0.6
 Xo value: 0.6
 - C_slave_sub_multithreat:** cost received for issuing action incompatible with

slave aircraft after an initial advisory in a multi-threat scenario

Xa value: 0.65

Xo value: 0.65

C_slave_sub_no_response: cost received for issuing action incompatible with non-responsive slave aircraft

Xa value: 0.0

Xo value: 0.0

C_slave_sub_noncrossing: cost received for issuing action incompatible with slave aircraft after an initial advisory with no crossing

Xa value: 0.05

Xo value: 0.05

H_noncrossing_thres: vertical separation distance above which the C_slave_-sub_noncrossing cost can be applied (ft)

Xa value: 100.0

Xo value: 100.0

T_noncrossing_thres: maximum time that VRC can be active and have C_-slave_sub_noncrossing cost applied (s)

Xa value: 5.0

Xo value: 5.0

T_reversal_thres: time allowed for a master aircraft to reverse an advisory of a slave if the slave issued the advisory first (s)

Xa value: 2.0

Xo value: 2.0

compat_dna: structure containing parameter values used when slave intruder is designated to Xo DNA mode and is coordinating with ownship

compat_dna.C_slave_init: increased cost received for issuing action incompatible with slave aircraft for first advisory

Xa value: 1.0

Xo value: 1.0

compat_dna.C_slave_init_crossing: increased cost received for issuing action incompatible with slave aircraft

Xa value: 1.0

Xo value: 1.0

compat_dna.C_slave_sub: increased cost received for issuing action incompatible with slave aircraft after an initial advisory

Xa value: 1.0

Xo value: 1.0

compat_dna.C_slave_sub_no_response: increased cost received for issuing action incompatible with non-responsive slave aircraft

Xa value: 1.0

Xo value: 1.0

compat_dna.C_slave_sub_noncrossing: increased cost received for issuing action incompatible with slave aircraft after an initial advisory with no crossing

Xa value: 1.0

Xo value: 1.0

- **coord_delay:** CoordinationDelayCost parameters

T_init: minimum number of time steps to wait for a VRC from an intruder equipped with TCAS (s)

Xa value: 4.0

Xo value: 4.0

- **coord_ra_deferral:** CoordinatedRADeferralCost parameters

C_deferral: cost applied to resolution advisories to force deferral based on coordination information

Xa value: 0.15

Xo value: 0.15

H_rolloff_proximate: ramp up interval for H_threshold_proximate (ft)

Xa value: 0.0

Xo value: 0.0

H_threshold_proximate: threshold for loss of separation (ft)

Xa value: 500.0

Xo value: 500.0

T_proximate: time value defining 'proximate' for determining proximate loss of separation (s)

Xa value: 10.0

Xo value: 10.0

- **critical_interval_protection:** CriticalIntervalProtectionCost parameters

C_force_alert: cost applied to force a resolution advisory

Xa value: 0.5

Xo value: 0.5

D_range_ground_expanded: ground range at or above which cost is not applicable (ft)

Xa value: 3000.0

Xo value: 3000.0

D_range_ground_required: ground range at or above which cost is not applicable (ft)

Xa value: 1500.0

Xo value: 1500.0

H_rel_required: required vertical separation (ft)

Xa value: 800.0

Xo value: 800.0

R_int_vert_threshold: intruder aircraft vertical rate below which cost is not applicable (ft/s)

Xa value: 15.0

Xo value: 15.0

R_own_vert_threshold: own aircraft vertical rate below which cost is not applicable (ft/s)

Xa value: 12.5

Xo value: 12.5

R_speed_ground_force_diverge_threshold: ground speed above which the force diverge is not applicable (ft/s)

Xa value: 300.0

Xo value: 300.0

R_speed_ground_threshold: ground speed above which cost is not applicable (ft/s)

Xa value: 150.0

Xo value: 150.0

T_proximate: time value defining 'proximate' for determining proximate loss of separation (s)

Xa value: 30.0

Xo value: 30.0

T_proximate_climbdescend: time value defining 'proximate' for determining proximate loss of separation for corrective climb and descend advisories (s)

Xa value: 20.0

Xo value: 20.0

T_proximate_inc_climbdescend: time value defining 'proximate' for determining proximate loss of separation for strengthening climb and descend advisories (s)

Xa value: 10.0

Xo value: 10.0

- **crossing_no_alert:** CrossingNoAlertCost parameters

C_coc: cost applied to force an advisory when ownship would cross altitudes with another equipped aircraft

Xa value: 5.0

Xo value: 5.0

T_threshold: time threshold for tau below which COC will be penalized when aircraft altitudes would cross (s)

Xa value: 30.0

Xo value: 30.0

- **force_alert:** Online cost parameters applied to penalize a clear of conflict

C_coc_threshold_lower: low altitude cost threshold which COC needs to attain before an advisory is terminated

Xa value: 1.75

Xo value: 1.75

C_coc_threshold_upper: cost threshold which COC needs to attain before an advisory is terminated

Xa value: 1.25

Xo value: 1.25

C_force_alert: cost applied to COC or corrective advisories when forcing an alert

Xa value: 1.0

Xo value: 1.0

H_threshold_lower: low altitude threshold for using C_coc_threshold_lower (ft)

Xa value: 1650.0

Xo value: 1650.0

H_threshold_upper: altitude threshold for using C_coc_threshold_upper (ft)

Xa value: 2350.0

Xo value: 2350.0

- **initialization:** InitializationCost parameters

T_init: minimum number of time steps before a resolution advisory can be issued (s)

Xa value: 3.0

Xo value: 3.0

- **low_alt_parallel_ra_deferral:** LowAltitudeParallelRADeferralCost parameters

C_deferral: cost applied to resolution advisories to force deferral during low altitude parallel encounters

Xa value: 0.5

Xo value: 0.5

C_deferral_dnx: cost applied to non-climb/descend advisories as altitude decreases so long as tau is large enough

Xa value: 0.1

Xo value: 0.1

C_deferral_low_alt: cost applied to resolution advisories in equipped encounters at low altitude so long as tau and range are large enough

Xa value: 2.0

Xo value: 2.0

C_deferral_very_low_alt: cost applied to resolution advisories in equipped encounters at very low altitude so long as tau and range are large enough

Xa value: 2.0

Xo value: 2.0

D_range_delta_min: minimum change in range required to estimate tau based solely on range states (ft)

Xa value: 100.0

Xo value: 100.0

D_range_expansion: amount to expand range threshold as a factor of speed (ft)

Xa value: 3000.0

Xo value: 3000.0

D_range_ground_rolloff: ramp up interval for D_range_ground_threshold (ft)

Xa value: 500.0

Xo value: 500.0

D_range_ground_threshold: ground range, in combination with range_expansion, at and above which full cost applies (ft)

Xa value: 1500.0

Xo value: 1500.0

H_low_alt: altitude threshold at or below which the C_deferral_low_alt penalty can be applied (ft)

Xa value: 5000.0

Xo value: 5000.0

H_rolloff: ramp down interval for H_threshold (ft)

Xa value: 3000.0

Xo value: 3000.0

H_rolloff_dnx: ramp down interval for H_threshold_dnx (ft)

Xa value: 2000.0

Xo value: 2000.0

H_threshold: altitude threshold at or below which full cost applies (ft)

Xa value: 2000.0
Xo value: 2000.0

H_threshold_dnx: altitude threshold at or below which the C_deferral_dnx penalty can be fully applied (ft)

Xa value: 8000.0
Xo value: 8000.0

H_very_low_alt: altitude threshold at or below which the C_deferral_very_low_alt penalty can be applied (ft)

Xa value: 2350.0
Xo value: 2350.0

R_speed_ground_rolloff: ramp down interval for R_speed_ground_threshold (ft/s)

Xa value: 50.0
Xo value: 50.0

R_speed_ground_threshold: ground speed at and below which cost applies (ft/s)

Xa value: 125.0
Xo value: 125.0

T_threshold_dnx: tau threshold at or above which the C_deferral_dnx penalty can be applied (s)

Xa value: 15.0
Xo value: 15.0

lowalt_equip: LowAltitudeParallelRADeferralCost parameters specific to equipped intruders used for selecting resolution advisories

lowalt_equip.D_range_low_alt: horizontal range threshold at or above which the C_deferral_low_alt penalty can be applied (ft)

Xa value: 2500.0
Xo value: 2500.0

lowalt_equip.D_range_very_low_alt: horizontal range threshold at or above which the C_deferral_very_low_alt penalty can be applied (ft)

Xa value: 2000.0
Xo value: 2000.0

lowalt_equip.T_low_alt: tau threshold at or above which the C_deferral_low_alt penalty can be applied (s)

Xa value: 21.0
Xo value: 24.0

lowalt_equip.T_very_low_alt: tau threshold at or above which the C_deferral_very_low_alt penalty can be applied (s)

Xa value: 18.0
Xo value: 20.0

lowalt_unequip: LowAltitudeParallelRADeferralCost parameters specific to unequipped intruders used for selecting resolution advisories

lowalt_unequip.D_range_low_alt: horizontal range threshold at or above which the C_deferral_low_alt penalty can be applied (ft)

Xa value: 4000.0
Xo value: 4000.0

lowalt_unequip.D_range_very_low_alt: horizontal range threshold at or above which the C_deferral_very_low_alt penalty can be applied (ft)

Xa value: 2000.0

Xo value: 2000.0

lowalt_unequip.T_low_alt: tau threshold at or above which the C_deferral_low_alt penalty can be applied (s)

Xa value: 22.5

Xo value: 24.0

lowalt_unequip.T_very_low_alt: tau threshold at or above which the C_deferral_very_low_alt penalty can be applied (s)

Xa value: 18.5

Xo value: 20.0

- **max_reversal:** MaxReversalCost parameters

C_coc: cost for issuing COC after the maximum reversal limit has been reached

Xa value: 400.0

Xo value: 400.0

C_reversal: cost for issuing a reversal after the maximum limit threshold has been reached

Xa value: 401.0

Xo value: 401.0

N_limit: maximum number of reversals threshold

Xa value: 1

Xo value: 1

T_coc_threshold: minimum number of time steps after the maximum reversal limit has been reached to penalize COC (s)

Xa value: 10.0

Xo value: 10.0

- **prevent_early_coc:** PreventEarlyCOCCost parameters

C_coc: cost for issuing a premature clear of conflict advisory

Xa value: 0.5

Xo value: 0.5

D_range_min: horizontal range below which clear of conflict is penalized unless rel_mid vertical separation has been achieved (ft)

Xa value: 1500.0

Xo value: 1500.0

H_rel_hi_threshold: relative vertical separation above which clear of conflict is not penalized so long as the aircraft are not converging vertically (ft)

Xa value: 2400.0

Xo value: 2400.0

H_rel_lo_threshold: relative vertical separation above which clear of conflict is not penalized so long as the aircraft are diverging vertically, one horizontal divergence test is passed, and the minimum horizontal range has been achieved (ft)

Xa value: 1200.0

Xo value: 1200.0

H_rel_mid_threshold: relative vertical separation above which clear of conflict is not penalized so long as the aircraft are not converging vertically and one

horizontal divergence test is passed (ft)

Xa value: 1800.0

Xo value: 1800.0

R_rel_vert_threshold: relative vertical rate above which a determination of converging/diverging can be made; below is considered neither (ft/s)

Xa value: 15.0

Xo value: 15.0

T_min_horizontal_divergence: time that divergence in range must persist to allow clear of conflict to be issued unless rel_lo vertical separation has been achieved (s)

Xa value: 3

Xo value: 3

T_threshold: time threshold for tau above which clear of conflict will not be penalized if the rel_lo vertical separation has been achieved (s)

Xa value: 37.0

Xo value: 37.0

- **prevent_early_weakening:** PreventEarlyWeakeningCost parameters

C_weakening: cost for issuing a weakening advisory

Xa value: 0.25

Xo value: 0.25

- **response_estimation:** Parameters for estimating whether an aircraft is responding to an advisory

H_own_wait_int_threshold: altitude threshold above which T_wait_int is used for the intruder wait time (ft)

Xa value: 5000.0

Xo value: 5000.0

R_acceleration: acceleration assumed (ft/s²)

Xa value: 6.4348

Xo value: 6.4348

R_buffer_int: vertical rate buffer for intruder aircraft (ft/s)

Xa value: 8.3333333333

Xo value: 8.3333333333

R_buffer_own: vertical rate buffer for own aircraft (ft/s)

Xa value: 8.3333333333

Xo value: 8.3333333333

R_threshold_acceleration_hi: magnitude of acceleration below which the aircraft is judged to be leveling off, and thus non-responding (ft/s²)

Xa value: 2.0

Xo value: 2.0

R_threshold_acceleration_lo: magnitude of acceleration above which the aircraft is judged to be non-responding (ft/s²)

Xa value: 1.75

Xo value: 1.75

T_wait_int: time to wait before the intruder's rate is stored (s)

Xa value: 9

Xo value: 9

T_wait_int_low_alt: time to wait before the intruder's rate is stored when own altitude is at or below H_own_wait_int_threshold (s)

Xa value: 8

Xo value: 8

T_wait_own: time to wait before the own aircraft's rate is stored (s)

Xa value: 8

Xo value: 8

resp_dna: structure containing parameter values used when slave intruder is designated to Xo DNA mode and is coordinating with ownship

resp_dna.T_wait_int: increased time to wait before the intruder rate is stored (s). Used when slave intruder is designated to Xo DNA mode and is coordinating with ownship

Xa value: 10000

Xo value: 10000

- **restrict_coc_due_to_reversal:** RestrictCOCDueToReversal parameters

C_restrict_coc: cost for issuing an inappropriate clear of conflict advisory when the intruder aircraft is master and is forcing a reversal

Xa value: 500.0

Xo value: 500.0

- **safe_crossing_ra_deferral:** SafeCrossingRADeferralCost parameters

C_crossing_deferral: cost applied to any alert when the safe crossing deferral heuristic determines that a deferral is required

Xa value: 10.0

Xo value: 10.0

H_cpa_separation_threshold: projected relative altitude threshold at closest point of approach between two aircraft above which deferral can be applied (ft)

Xa value: 350.0

Xo value: 350.0

H_min_threshold: relative altitude threshold between two aircraft above which deferral can be applied (ft)

Xa value: 300.0

Xo value: 300.0

H_own_dz_cpa_vertical_threshold: increase in the relative altitude threshold at closest point of approach due to ownship vertical rate (ft)

Xa value: 250.0

Xo value: 250.0

H_own_dz_vertical_threshold: increase in the relative altitude threshold due to ownship vertical rate (ft)

Xa value: 300.0

Xo value: 300.0

R_int_vertical_threshold: intruder vertical rate above which the deferral can be applied (ft/s)

Xa value: 30.0

Xo value: 30.0

R_own_rolloff: ramp up interval for the R_own_threshold after which the full increased thresholds are used (ft/s)

Xa value: 15.0

Xo value: 15.0

R_own_threshold: ownship vertical rate above which the vertical thresholds begin to increase (ft/s)

Xa value: 15.0

Xo value: 15.0

- **sandwich_prevention:** SandwichPreventionCost parameters

C_coc_threshold: cost of COC above which RAs are considered likely and the sandwich cost may be applied

Xa value: 0.05

Xo value: 0.05

C_sandwich_equip: cost applied to actions associated with equipped intruder to prevent own aircraft from being sandwiched in a multithreat situation

Xa value: 0.2

Xo value: 0.2

C_sandwich_unequip: cost applied to actions associated with intruder to prevent own aircraft from being sandwiched in a multithreat situation

Xa value: 0.56

Xo value: 0.56

- **time_based_non_compliance:** TimeBasedNonComplianceCost parameters

C_coc: cost applied to clear of conflict to ensure that force a resolution advisory transition does not lead to premature clear of conflict

Xa value: 10.0

Xo value: 10.0

C_current_ra: cost applied to current action force a resolution advisory transition

Xa value: 0.25

Xo value: 0.25

C_ra_epsilon: value of current_ra_cost which is effectively equal to 0.0

Xa value: 1.0e-13

Xo value: 1.0e-13

H_rel_rolloff: ramp down interval for H_rel_threshold (ft)

Xa value: 125.0

Xo value: 125.0

H_rel_rolloff_cross: ramp down interval for H_rel_threshold_cross (ft)

Xa value: 250.0

Xo value: 250.0

H_rel_threshold: projected vertical separation at or below which the cost can be fully applied (ft)

Xa value: 75.0

Xo value: 75.0

H_rel_threshold_cross: projected vertical separation at or below which the cost can be fully applied if crossing is expected (ft)

Xa value: 150.0

Xo value: 150.0

R_vert_rel_compare_min: minimum vertical rate difference that can be used when computing vertical rate compliance factor (ft/s)

Xa value: 15.0

Xo value: 15.0

T_rolloff_apply: ramp down interval for T_threshold_apply (s)

Xa value: 4.0

Xo value: 4.0

T_rolloff_tau_delta: ramp up interval for T_threshold_tau_delta (s)

Xa value: 5.0

Xo value: 5.0

T_threshold_apply: tau value at or below which the cost can be fully applied (s)

Xa value: 7.0

Xo value: 7.0

T_threshold_min_tau: tau threshold below which the minimum tau counter is incremented (s)

Xa value: 7.5

Xo value: 7.5

T_threshold_tau_delta: tau difference at which the cost begins to apply (s)

Xa value: 5.0

Xo value: 5.0

X_vert_rel_factor_max: maximum factor that can be applied based on non-compliance with target vertical rate

Xa value: 3.0

Xo value: 3.0

- **unequipped_mtlo:** AllowUnequippedMTLO parameters

H_threshold: altitude threshold for a multithreat level-off check (ft)

Xa value: 250.0

Xo value: 250.0

R_threshold: vertical rate threshold for a multithreat level-off check (ft/s)

Xa value: 4.1667

Xo value: 4.1667

- **protection_mode_code:** registered number for this protection mode; for example, 1 for "PROTECTION_MODE_Xa" or 2 for "PROTECTION_MODE_Xo_CSPO3k". Registered values are listed in Global Constants.

Xa value: 1

Xo value: 2

- **state_estimation:** Information about the parameters used in the state estimation subsystem

- **tau:** Tau estimation parameters

- **D_range_diverge_thres:** ground range below which the vertical entry table can be used (ft)

Xa value: 3000.0

Xo value: 3000.0

- **D_range_thres:** horizontal range below which an encounter is considered a slow closure encounter (ft)

Xa value: 4500.0

Xo value: 4500.0

- **H_alt_rel_thres:** vertical separation above which an encounter is considered a slow closure encounter (ft)

Xa value: 1000.0
Xo value: 1000.0

- **R_converge_thres_vert:** relative vertical rate below which the encounter is not converging (ft/s)

Xa value: 4.0
Xo value: 4.0

- **R_speed_thres:** relative horizontal speed below which an encounter is considered a slow closure encounter (ft/s)

Xa value: 300.0
Xo value: 300.0

- **T_thres_horiz:** tau threshold for determining if encounter is slow closure (s)

Xa value: 9.0
Xo value: 9.0

- **W_thres_horiz:** sum of horizontal weights for taus before T_thres_horiz

Xa value: 0.5
Xo value: 0.5

- **W_thres_vert:** sum of vertical weights for taus before T_thres_horiz

Xa value: 0.75
Xo value: 0.75

- **X_phi_converge_thres:** relative bearing below which the vertical entry table can be used (rad)

Xa value: 6.2831853072
Xo value: 6.2831853072

- **entry_dist:** Entry distribution parameters

T_tau_values: vector of time steps in the entry tables (s)

Xa value: [0.0, 1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0, 11.0, 12.0, 13.0, 14.0, 15.0, 16.0, 17.0, 18.0, 19.0, 20.0, 21.0, 22.0, 23.0, 24.0, 25.0, 26.0, 27.0, 28.0, 29.0, 30.0, 31.0, 32.0, 33.0, 34.0, 35.0, 36.0, 37.0, 38.0, 39.0, 40.0]
Xo value: [0.0, 1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0, 11.0, 12.0, 13.0, 14.0, 15.0, 16.0, 17.0, 18.0, 19.0, 20.0, 21.0, 22.0, 23.0, 24.0, 25.0, 26.0, 27.0, 28.0, 29.0, 30.0, 31.0, 32.0, 33.0, 34.0, 35.0, 36.0, 37.0, 38.0, 39.0, 40.0]

horizontal_active_table: name of the horizontal entry table for active surveillance only

Xa value: DO-385_entry_horiz.dat
Xo value: DO-385_entry_horiz_active_cspo3k.dat

horizontal_table: name of the horizontal entry table

Xa value: DO-385_entry_horiz.dat
Xo value: DO-385_entry_horiz.dat

vertical_table: name of the vertical entry table

Xa value: DO-385_entry_vert.dat
Xo value: DO-385_entry_vert.dat

- **horiz_worst_case:** HorizontalWorstCase parameters

D_near_mean_range_threshold_hi: range between the two aircraft above which the maximum worst case near mean factor can be applied (ft)

Xa value: 30000.0
Xo value: 30000.0

D_near_mean_range_threshold_lo: range between the two aircraft below which no factor related to worst case near mean can be applied (ft)

Xa value: 6000.0

Xo value: 6000.0

H_phi_spread_threshold: altitude threshold at or below which the worst case scale factor may be reduced due to angle spread of belief states (ft)

Xa value: 12500.0

Xo value: 12500.0

X_near_mean_angle_threshold_hi: angle between the mean belief state and the worst case above which no factor related to worst case near mean can be applied (rad)

Xa value: 0.436332312998582

Xo value: 0.436332312998582

X_near_mean_angle_threshold_lo: angle between the mean belief state and the worst case below which the maximum worst case near mean factor can be applied (rad)

Xa value: 0.261799387799149

Xo value: 0.261799387799149

X_near_mean_factor_max: maximum factor than can be applied to worst case weight due to the angle between the mean and the worst case

Xa value: 10.0

Xo value: 10.0

X_phi_spread_factor_hi: largest scale factor applied to worst case weight due to angle spread of belief states

Xa value: 4.0

Xo value: 4.0

X_phi_spread_factor_lo: smallest scale factor applied to worst case weight due to angle spread of belief states

Xa value: 0.5

Xo value: 0.5

X_phi_spread_threshold_hi: relative bearing spread above which the scale factor will be X_factor_lo due to angle spread of belief states (rad)

Xa value: 1.5707963267948966

Xo value: 1.5707963267948966

X_phi_spread_threshold_lo: relative bearing spread below which the scale factor will not be reduced due to angle spread of belief states (rad)

Xa value: 0.7853981633974483

Xo value: 0.7853981633974483

- **threshold_factor:** threshold factor for determining whether to include a weight in the tau distribution

Xa value: 1.0e-6

Xo value: 1.0e-6

- **track_threat:** Information for configuring traffic advisories

- **maintain_ta_after_ra:** minimum traffic advisory alert after resolution advisory (s)

Xa value: 8

Xo value: 8

- **ta_altitude_dependent_coc:** thresholds for adjusting offline cost based on altitude using AltitudeDependentCOCCost
 - **C_coc:** cost for issuing a clear of conflict or preventive advisory
 Xa value: 1.0
 Xo value: 1.0
 - **H_rel_threshold_proximate:** minimum proximate separation for applying cost (ft)
 Xa value: 800.0
 Xo value: 800.0
 - **T_threshold:** tau value below which the full cost can be applied (s)
 Xa value: 39.0
 Xo value: 39.0
 - **ta_altitude_threshold:** vector of altitude thresholds used for activating traffic advisories (ft)
 Xa value: [0.0, 1000.0, 2350.0, 5000.0, 8000.0, 10000.0, 20000.0, 42000.0, 42100.0]
 Xo value: [0.0, 1000.0, 2350.0, 5000.0, 8000.0, 10000.0, 20000.0, 42000.0, 42100.0]
 - **ta_cost_threshold_off:** cost threshold for deactivating traffic advisories
 Xa value: 1.0e-6
 Xo value: 1.0e-6
 - **ta_cost_threshold_on:** vector of altitude-dependent cost thresholds for activating traffic advisories
 Xa value: [0.34935, 0.47158, 0.24602, 0.13204, 0.05585, 0.09348, 0.07459, 0.07459, 0.02822]
 Xo value: [0.34935, 0.47158, 0.24602, 0.13204, 0.05585, 0.09348, 0.07459, 0.07459, 0.02822]
 - **ta_differential_threshold:** vector of altitude-dependent thresholds determining closeness of COC and non-COC costs
 Xa value: [1.64031, 0.8183, 0.7507, 0.29203, 0.30517, 0.23322, 0.2155, 0.2155, 0.41451]
 Xo value: [1.64031, 0.8183, 0.7507, 0.29203, 0.30517, 0.23322, 0.2155, 0.2155, 0.41451]
 - **ta_low_alt_parallel_ra_deferral:** LowAltitudeParallelRADeferralCost parameters for determining traffic advisories
 - **ta_lowalt_equip:** LowAltitudeParallelRADeferralCost parameters specific to equipped intruders, used for determining traffic advisories
 - **ta_lowalt_equip.D_range_low_alt:** horizontal range threshold at or above which the C_deferral_low_alt penalty can be applied (ft)
 Xa value: 5000.0
 Xo value: 5000.0
 - **ta_lowalt_equip.D_range_very_low_alt:** horizontal range threshold at or above which the C_deferral_very_low_alt penalty can be applied (ft)
 Xa value: 4000.0
 Xo value: 4000.0
 - **ta_lowalt_equip.T_low_alt:** tau threshold at or above which the C_deferral_low_alt penalty can be applied (s)
 Xa value: 28.0
 Xo value: 28.0

- **ta_lowalt_equip.T_very_low_alt:** tau threshold at or above which the C_deferral_very_low_alt penalty can be applied (s)
Xa value: 28.0
Xo value: 28.0
- **ta_lowalt_unequip:** LowAltitudeParallelRADeferralCost parameters specific to unequipped intruders, used for determining traffic advisories
- **ta_lowalt_unequip.D_range_low_alt:** horizontal range threshold at or above which the C_deferral_low_alt penalty can be applied (ft)
Xa value: 4000.0
Xo value: 4000.0
- **ta_lowalt_unequip.D_range_very_low_alt:** horizontal range threshold at or above which the C_deferral_very_low_alt penalty can be applied (ft)
Xa value: 2000.0
Xo value: 2000.0
- **ta_lowalt_unequip.T_low_alt:** tau threshold at or above which the C_deferral_low_alt penalty can be applied (s)
Xa value: 31.5
Xo value: 31.5
- **ta_lowalt_unequip.T_very_low_alt:** tau threshold at or above which the C_deferral_very_low_alt penalty can be applied (s)
Xa value: 28.0
Xo value: 28.0
- **ta_min_alert_time:** minimum traffic advisory active duration (s)
Xa value: 8
Xo value: 8
- **target_designation:** Xo target designation parameters
 - **S_standard_protection_mode:** tables and parameters to be used by TRM when target is not designated for Xo processing. Registered values are listed in Global Constants. 1 for "PROTECTION_MODE_Xa".
value: 1
- **threat_resolution:** threat resolution parameters independent of protection mode
 - **C_differential_threshold:** differential by which one cost is considered better than another
value: 1.0e-6
 - **C_restrict:** very high cost used to prevent selection of an action; applied to COC to force an alert and to MTLO prevent an individual MTLO
value: 500.0
 - **H_radalt_limit:** altitude at or below which radar altimeter value can be used for setting height (ft)
value: 2350.0
 - **action_arbitration:** ActionArbitration parameters
 - **C_differential_worst_case_threshold:** cost difference hysteresis used by Arbitrate-ConflictingSenses
value: 2.0
 - **balance_costs:** MultithreatCostBalancing parameters

- **C_absolute_threshold_hi**: cost above which MTLO is not allowed, regardless of whether costs balance
value: 3.5
- **C_absolute_threshold_lo**: hysteresis on cost above which MTLO is not allowed, regardless of whether costs balance
value: 0.9
- **C_differential_threshold_hi**: difference in costs above which the costs for any two intruders are considered too unbalanced to allow an MTLO
value: 2.0
- **C_differential_threshold_lo**: difference in costs below which the costs for any two intruders are no longer considered too unbalanced to allow an MTLO
value: 0.15

All references made in the text to data contained in this file use dot notation to access the data. The names of the fields are preceded by `params()` and followed by an appropriate 1-based index, if needed. For example,

`params().modes[i].protection_mode_code`

refers to the *protection_mode_code* for the *i*th *modes* data structure in the file.

Similarly,

`params().modes[mode_int].cost_estimation.online.crossing_no_alert.C_coc`

refers to the value of the CROSSINGNOALERTCOST online cost applied to the clear of conflict, or no advisory, action. The *mode_int* index value is derived from the value of the *protection_mode* field in the TRMINTRUDERINPUT data structure.

This Page Intentionally Left Blank

Appendix G Data Table Format Specification

The entry distribution tables and offline cost tables use a file format designed for storing block dictionaries. Given a particular state, these block dictionaries allow efficient lookup of the entry distributions and the offline costs for actions. This appendix provides the format specification.

The term "entry distribution tables" refers to the horizontal entry distribution table and vertical entry distribution table. There is a single entry distribution table per file.

The term "offline cost tables" refers to the equivalence class tables and MinBlocks table. There is a single MinBlocks table per file.

There are multiple equivalence class tables. As described in this document, all of the equivalence class tables are concatenated into a single file and loaded into a single variable. Each individual equivalence class table follows the format specification provided below.

There are five basic types used in the binary representation of the block dictionary:

- **uint8**: unsigned 8 bit integer.
- **uint32**: unsigned 32 bit integer.
- **half**: IEEE 754-2008 binary16 format half-precision floating point number.
- **double**: IEEE 754-2008 binary64 format double-precision floating point number.
- **varchar**: an array of (8 bit) bytes, beginning with a uint8 that specifies the number of single-byte ASCII characters that follow.

The table contains the following fields:

- **magic_number** (uint32). This field serves as validation and contains the number 1634 for files processed as described in this document.
- **file_type** (varchar). Depending on whether the block size is fixed or variable, this field should contain the string **fixedblockdictionary** or **varblockdictionary**, respectively. The entry distribution tables and offline cost tables use fixed block dictionaries.
- **auxiliary_data_size** (uint32). This field specifies the size of the auxiliary data in bytes. The auxiliary data contains information about the grid edges, as described later.
- **auxiliary_data**. This field contains the auxiliary data, described later.
- **index_type** (varchar). Data type for **index** field. The entry distribution tables and offline cost tables are **uint32**.
- **data_type** (varchar). Data type for contents of the **data** field. The entry distribution tables and offline cost tables are **half**.
- **count_included** (uint8). Indicates whether the count is included. The value in this field is undefined for fixed block dictionaries.
- **maximum_block_elements** (uint8). Specifies the maximum number of block elements. For fixed block dictionaries, this is the number of elements in each block.
- **index_element_count** (uint32). This field contains the number of index elements.
- **data_element_count** (uint32). This field contains the number of data elements.
- **index**. This field is omitted if **index_element_count** is 0. Otherwise, it contains the index. The index provides data element (not necessarily byte) offsets, which are 0-based, into the data segment. The entry distribution tables have index elements, the offline cost tables do not.

- **data**. This field contains the data. If the count is included, then the first element of each block contains the count of the number of elements in the block.

The auxiliary data specifies the grid names and has the following fields:

- **dimension_count** (uint32). This specifies the number of dimensions. The equivalence class tables described in this document have four dimensions (tau, relative vertical separation, own vertical rate, and intruder vertical rate). The MinBlocks table described in this document has four dimensions (split, relative vertical separation, own vertical rate, and intruder vertical rate). The vertical entry distribution table described in this document has three dimensions (relative vertical separation, own vertical rate, and intruder vertical rate). The horizontal entry distribution tables described in this document have three dimensions (ground range, ground speed, and relative bearing).
- **cut_counts** (uint32 array). This field contains the number of cuts for each dimension.
- **names** (varchar array). This field contains the names for each dimension and can be used for validation purposes.
- **cuts** (double array). This field contains all the cuts for all the dimensions. The cuts for each dimension are specified in increasing order.

The entry distribution tables and MinBlocks table are loaded into RDATATABLE (p.E-38) data structures. The equivalence class tables are loaded into an array of RDATATABLE data structures, one for each table. If the returned structure is **s**, the cut counts can be accessed using dot notation, which would be *s.cut_counts*.



Appendix H Math Utilities

Utility functions used in the ACAS X algorithms.

The MAHAL algorithm computes the Mahalanobis distance. This calculation uses the covariance of one or two distributions to provide a normalized measure of distance. It is used extensively in association and correlation algorithms of the STM where distributions are related to one another.

Algorithm 349 mahal

```

1 function mahal(muA::Union(Vector{R},R,Z), SigmaA::Union(Matrix{R},R,Z), muB::Union(Vector{R},R,Z), SigmaB::Union(Matrix{R},R,Z))
2   if (SigmaB != 0)
3     SigmaA .+= SigmaB
4   end
5   dmu = muA
6   if (muB != 0)
7     dmu = muA .- muB
8   end
9   if (ndims(SigmaA) == 2)
10    D = dmu' * inv(SigmaA) * dmu
11    D = D[1]
12  else
13    D = (dmu * dmu) / SigmaA
14  end
15  return sqrt(D)::R
16 end
```

Referenced In: CorrelateImage_(p. C-18), PromoteModeCTracks_(p. B-10), HypotheticalTrackTest_(p. B-14), DetermineMode-CReplyAssociationScore_(p. B-7), MergeModeCReplies_(p. B-3), CorrelatePosition_(p. C-21), IsOutlier_(p. 32)

The UCHOL algorithm computes the Upper Cholesky Decomposition which is an efficient and stable method for calculating a matrix square root. It is used within SIGMAPONTSAMPLE to draw discrete samples from a continuous distribution. Sometimes numerical errors will cause the diagonal terms of the input matrix to evaluate to very small negative values (defined by the *psd_stability_factor* threshold). This function includes a check that will make such values very small positive numbers, thus guaranteeing the input matrix is positive semidefinite.

Algorithm 350 uchol

```

1 function uchol(A::Matrix{R})
2   const psd_stability_factor::R = params().surveillance.psd_stability_factor
3   n = size(A,1)
4   AA = copy(A)
5   U = zeros(size(A))
6   for j in n:-1:1
7     if AA[j,j] < psd_stability_factor
8       AA[j,j] = psd_stability_factor
9     end
10    U[j,j] = sqrt(AA[j,j])
11    d = 1 / U[j,j]
12    if (j > 1)
13      for k in 1:j
14        U[j,k] = d * AA[j,k]
15      end
16      for k in 1:j
17        for i in 1:k
18          AA[k,i] -= U[j,k] * U[j,i]
19        end
20      end
21    end
22  end
23  return (U')::Matrix{R}
24 end

```

Referenced In: SigmaPointSample(p.28)

BLOCK_DIAG (Algorithm 351) combines two input matrices ($m \times n$ matrix A and $p \times q$ matrix B) diagonally into an output matrix of size $(M+P) \times (N+Q)$. All other elements of the output matrix are zeros.

Algorithm 351 block_diag

```

1 function block_diag(A::Matrix{R}, B::Matrix{R})
2   return [A zeros(size(A)[1], size(B)[2]); zeros(size(B)[1], size(A)[2]) B]
3 end

```

Referenced In: RedefineEstimateInRotatedFrame(p.58), InitializeRangeTracker(p.35), UpdateCartesianTracker(p.34), CombineAndSample(p.105), PredictRangeTracker(p.39), InitializeCartesianTracker(p.28), UpdateRangeTracker(p.41)

ANGLEDIFFERENCE (Algorithm 352) takes as input two angles (A and B). This algorithm returns the angle subtending the inputs.

Algorithm 352 AngleDifference

```

1 function AngleDifference(A::R, B::R)
2   return mod(B-A + pi,2pi) - pi
3 end

```

Referenced In: ReceiveHeadingObservation(p.81), HeadingAtToa(p.85), CorrelateImage(p.C-18), PromoteMod-eCTracks(p.B-10), HypotheticalTrackTest(p.B-14), DetermineModeCReplyAssociationScore(p.B-7), UpdateHeading-Tracker(p.84), MergeModeCReplies(p.B-3), CorrelatePosition(p.C-21), AdvanceCartesianTrack(p.31)

NORMALIZE (Algorithm 353) generates a vector of values that sum to 1.0 given a vector of positive values. This algorithm is used to produce weights for belief structures.

Algorithm 353 Normalize

```

1 function Normalize( x::Vector{R} )
2   x_norm::Vector{R} = x / sum( x )
3   return (x_norm::Vector{R})
4 end
```

Referenced In: CombineVerticalBeliefs(p. 178), StateEstimation(p. 168)

WRAPTO2PI (Algorithm 354) maps an angle into the interval 0 to 2π . The input and output angles are both in radians.

Algorithm 354 WrapTo2Pi

```

1 function WrapTo2Pi( phi::R )
2   phi_out::R = WrapToPi(p. H-3)( phi )
3   if ( phi_out < 0.0)
4     phi_out = phi_out + 2pi
5   end
6   phi_out = max( 0.0, phi_out )
7   phi_out = min( 2pi, phi_out )
8   return phi_out::R
9 end
```

Referenced In: HorizontalWorstCase(p. 170)

WRAPTOPI (Algorithm 355) maps an angle into the interval $-\pi$ to $+\pi$. The input and output angles are both in radians.

Algorithm 355 WrapToPi

```

1 function WrapToPi(phi::R)
2   phi_out::R = phi - (2pi * floor( (phi + pi) / (2pi) ))
3   return phi_out::R
4 end
```

Referenced In: WrapTo2Pi(p. H-3), EncodeTIDBearing(p. 315), StateEstimation(p. 168), ReceiveHeadingObservation(p. 81), HeadingAtToa(p. 85), CorrelateImage(p. C-18), SetDisplayDataPassive(p. 112), PromoteModeCTracks(p. B-10), HypotheticalTrackTest(p. B-14), DetermineModeCReplyAssociationScore(p. B-7), UpdateHeadingTracker(p. 84), SetDisplayDataActive(p. 107), MergeModeCReplies(p. B-3), CorrelatePosition(p. C-21), AdvanceCartesianTrack(p. 31)

This page intentionally left blank.

Appendix I Data Management Utilities

VERTICALTRMUPDATEPREP (Algorithm 356) is included here for informational purposes only. It is called from VERTICALTRMUPDATE and is used to format and prepare the TRM inputs for TRM processing.

The formatting and preparation required by any particular ACAS X implementation is dependent on the details of that system. The algorithm shown here performs a minimal amount of preparation, other than formatting, and assumes close to perfect data quality. It also makes use of Julia language capabilities for conciseness. To have the algorithm fit on a single page in this document, warning and error code has been hidden. For example, code that tests whether *mode_int* is a valid index is not shown. Subsequent processing in the TRM assumes that the data inputs are good.

The manner in which data quality requirements are handled for the ACAS X system are determined by the implementer. Many potential issues with data quality are overlooked in this document.

This algorithm takes as input *st_trm* and *input*. This algorithm returns *st_int*, *input_int_invalid*, *input_int_valid*, *num_intruders*, *z_int_ave*, *dz_int_ave*, *mode_int*, *code_int*, *sense_int*, *cost_ta*, and *output_int*.

Algorithm 356 VerticalTRMUpdatePrep

```

1 function VerticalTRMUpdatePrep( st_trm::TRMState(p.E-31), input::TRMInput(p.E-19) )
2   const N_actions::Z = params().actions.num_actions
3   input_int_invalid::Vector{TRMIIntruderInput(p.E-20)} = []
4   input_int_valid::Vector{TRMIIntruderInput(p.E-20)} = []
5   mode_idx::Z = 1
6   input_dict_int::Dict{Z,TRMIIntruderInput} = Dict{Z,TRMIIntruderInput(p.E-20)}()
7   for intruder::TRMIIntruderInput(p.E-20) in input.intruder
8     input_dict_int[intruder.id] = intruder
9     if (OPMODE_RA != input.own.opmode) &&
10       (OPMODE_TA != input.own.opmode)
11       push!( input_int_invalid, intruder )
12     elseif !intruder.valid
13       push!( input_int_invalid, intruder )
14     else
15       push!( input_int_valid, intruder )
16     end
17   end
18   num_intruders::Z = length( input_int_valid )
19   TRM_st_int::Dict{UInt32,TRMIIntruderState} = Dict{UInt32,TRMIIntruderState(p.E-33)}()
20   for st_int_iter::TRMIIntruderState(p.E-33) in st_trm.st_intruder
21     if !haskey( input_dict_int, st_int_iter.id )
22       st_int_iter.status = :Dropped
23     else
24       TRM_st_int[st_int_iter.id] = st_int_iter
25       st_int_iter.status = :InUse
26     end
27   end
28   for intruder in input_int_invalid
29     if !haskey( TRM_st_int, intruder.id )
30       mode_idx = GetProtectionModeIndex(p.328)( intruder.protection_mode )
31       st_int_iter = TRMIIntruderState(p.E-33)( intruder.id, intruder.address, intruder.is_icao, intruder.
32         protection_mode )
33       push!( st_trm.st_intruder, st_int_iter )
34       TRM_st_int[st_int_iter.id] = st_int_iter
35     end
36   end
37   cost_ta::Matrix{R} = zeros( R, num_intruders, N_actions )
38   z_int_ave::Vector{R} = zeros( R, num_intruders )
39   dz_int_ave::Vector{R} = zeros( R, num_intruders )
40   mode_int::Vector{Z} = zeros( Z, num_intruders )
41   code_int::Vector{UInt8} = zeros( UInt8, num_intruders )
42   sense_int::Vector{Symbol} = fill( :None, num_intruders )
43   st_int::Vector{TRMIIntruderState} = Array( TRMIIntruderState(p.E-33), num_intruders )
44   output_int::Vector{TRMIIntruderData} = Array( TRMIIntruderData(p.E-39), num_intruders )
45   if (0 == length( st_trm.st_own.st_alt_inhibit ))
46     const N_modes::Z = length( params().modes )
47     st_trm.st_own.st_alt_inhibit = Array( AltitudeInhibitCState(p.E-42), N_modes )
48     for mode_idx in 1:N_modes
49       st_trm.st_own.st_alt_inhibit[mode_idx] = AltitudeInhibitCState(p.E-42)( params(), mode_idx )
50     end
51   end
52   for j in 1:length( input_int_valid )
53     intruder = input_int_valid[j]
54     id::UInt32 = intruder.id
55     mode_idx = GetProtectionModeIndex(p.328)( intruder.protection_mode )
56     if haskey( TRM_st_int, id )
57       st_int[j] = TRM_st_int[id]
58     else
59       st_int[j] = TRMIIntruderState(p.E-33)( id, intruder.address, intruder.is_icao, intruder.
60         protection_mode )
61       push!( st_trm.st_intruder, st_int[j] )
62       TRM_st_int[id] = st_int[j]
63     end
64   return (st_int::Vector{TRMIIntruderState(p.E-33)}, input_int_invalid::Vector{TRMIIntruderInput(p.E-20)},
65           input_int_valid::Vector{TRMIIntruderInput(p.E-20)}, num_intruders::Z,
66           z_int_ave::Vector{R}, dz_int_ave::Vector{R}, mode_int::Vector{Z}, code_int::Vector{UInt8}),

```

```
66           sense_int::Vector<Symbol>, cost_ta::Matrix<R>, output_int::Vector<TRMIntruderData(p.E-39)>)
67 end
```

Referenced In: VerticalTRMUpdate(p. 152)

This page intentionally left blank.

Appendix J Julia Standard Library Functions

Several built-in functions in Julia are used in the ACAS X algorithms. Those functions are listed in the tables below. Definitions contain text taken verbatim from the Julia help function.

Julia data types are listed in Table 1-2. Julia keywords are listed in Table 1-3. The version of Julia used in this document is 0.3.10.

Note: To create a shallow copy of an array, a call to copy() is required due to Julia's intended behavior to assign by reference. To make a deep copy of an object, a call to deepcopy() is required.

By convention, "!" at the end of a built-in Julia function name indicates that function modifies its arguments. For example, the "delete!" function, modifies the collection passed to it as the first argument ("A") to delete the key passed to it as the second argument ("k"). If the Julia built-in function name has no "!" at the end of its name, its arguments are unmodified by the call.

Table J-1. Julia Functions Used in the ACAS X Algorithms

Function	Interface	Description
abs	abs(x)	Returns the absolute value of the argument
all	all(A,dims)	Test whether all elements of a boolean collection are TRUE.
any	any(A,dims)	Test whether any elements of a boolean collection are TRUE.
append!	append!(A,x)	Add the elements of "x" to the end of a collection "A".
atan2	atan2(y,x)	Compute the inverse tangent of "y/x", using the signs of both "x" and "y" to determine the quadrant of the return value
bool	bool(x) or bool(A)	Convert a number or numeric array to boolean If the value is 0, bool returns FALSE. Otherwise, it returns TRUE
convert	convert(T,x)	Try to convert "x" to the given type
copy	copy(x)	Create a shallow copy of "x": the outer structure is copied, but not all internal values
cos	cos(x)	Compute cosine of "x", where "x" is in radians
deepcopy	deepcopy(x)	Create a deep copy of "x": everything is copied recursively, resulting in a fully independent object
deg2rad	deg2rad(x)	Convert "x" from degrees to radians
delete!	delete!(A,k)	Delete the mapping for the given key "k" in collection "A"
deleteat!	deleteat!(A,i)	Remove the item(s) at the given index (indices), and return the modified collection "A" Subsequent items are shifted to fill the resulting gap
diagm	diagm(v)	Construct a diagonal matrix from a vector
error	error(v1,v2,...)	Output error message "v1v2..." and exit
exp	exp(x)	Compute the exponential of x (e^x)
eye	eye(n)	Return an n-by-n identity matrix
falses	falses(dims)	Create a "BitArray" with the specified dimensions and all values set to FALSE
fill	fill(v,dims)	Create an array with the specified dimensions and all values set to "v" The array type is determined by the type of "v"
findfirst	findfirst(A,v)	Return the index of the first element equal to "v" in "A"
floor	floor(x)	Returns the nearest integral value of the same type as "x" not greater than "x"
get	get(A,k,default)	Return the value stored in collection "A" for the given key "k" Return the given default value if no mapping for the key is present
getindex	getindex(A, key...)	Retrieve the value(s) stored at the given key or index within collection "A"
haskey	haskey(A,key)	Determine whether collection "A" has a mapping for a given key
hypot	hypot(x,y)	Compute the $\sqrt{x^2 + y^2}$ avoiding overflow and underflow
indmin	indmin(A)	Returns the index of the minimum element in collection "A"
int	int(x) or int(A)	Convert a number or array to the platform-dependent integer type

Table J-2. Julia Functions Used in the ACAS X Algorithms (cont.)

Function	Interface	Description
inv	inv(M)	Matrix inverse of "M"
isdefined	isdefined(:var)	Test whether global variable "var" is defined
isempty	isempty(A)	Determine whether collection "A" is empty (has no elements)
isnan	isnan(x)	Test whether a floating point number is not a number (NaN)
		Returns a Bool
keys	keys(A)	Return an iterator over all keys in collection "A"
length	length(A)	For ordered, indexable collections, the maximum index "i" for which "getindex(A,i)" is valid
		For unordered collections, the number of elements in "A"
max	max(x,y,...)	Return the maximum of the arguments. Operates elementwise over arrays
maximum	maximum(A)	Returns the largest element in collection "A"
	maximum(A,dims)	Compute the maximum value of array "A" over the given dimensions
min	min(x,y,...)	Return the minimum of the arguments. Operates elementwise over arrays
minimum	minimum(A)	Returns the smallest element in collection "A"
	minimum(A,dims)	Compute the minimum value of array "A" over the given dimensions
mod	mod(x, m)	Modulus of "x" after division by "m", returning in the range [0,m)
oftype	oftype(x,y)	Convert "y" to the type of "x".
ones	ones(T,dims)	Create an array of all ones of specified type "T"
pop!	pop!(A)	Remove the last item in collection "A"
push!	push!(A,x)	Insert items at the end of collection "A"
rad2deg	rad2deg(x)	Convert "x" from radians to degrees
reshape	reshape(A,dims)	Create an array with the same data as array "A", but with dimensions given by "dims"
resize!	resize!(A,n)	Resize A to contain n elements. If n is smaller than the current length of A, the first n elements will be retained. If n is larger, the new elements are not guaranteed to be initialized.
		Return a copy of "v" reversed from 1 to length(v)
round	round(x)	Returns the nearest integral value of the same type as "x" to "x"
setindex!	setindex!(A,value,key...)	Store the given value at the given key or index within collection "A"
sign	sign(x)	Return "+1" if "x" is positive, "0" if "x == 0", and "-1" if "x" is negative
sin	sin(x)	Compute sine of "x", where "x" is in radians
size	size(A)	Returns a tuple containing the dimensions of "A"
size	size(A,n)	Returns the size of "A" in dimension "n"
sort	sort(v)	Returns a sorted copy of "v" leaving "v" itself unmodified
sortperm	sortperm(v,lt=comparison)	Return a permutation vector of indices of "v" that puts it in sorted order The "lt=" keyword specifies a custom "less than" function "comparison" The resulting permutation will be the lexicographically first one that puts the input array into sorted order i.e. indices of equal elements appear in ascending order
		Remove items in the specified index range from collection "A"
splice!	splice!(A, range)	Subsequent items are shifted down to fill the resulting gap
sqrt	sqrt(x)	Returns the square root of "x"
string	string(xs...)	Create a string from any values
sum	sum(A)	Returns the sum of all elements in collection "A"
typemin	typemin(T)	The lowest value representable by the given (real) numeric type "T"
typeof	typeof(x)	Get the concrete type of "x"
uint8	uint8(x) or uint8(A)	Convert a number or array to "Uint8" (8-bit unsigned integer) data type
uint16	uint16(x) or uint16(A)	Convert a number or array to "Uint16" (16-bit unsigned integer) data type
uint32	uint32(x) or uint32(A)	Convert a number or array to "Uint32" (32-bit unsigned integer) data type
unshift!	unshift!(A,x)	Insert item "x" at the beginning of collection "A"
vcat	vcat(A...)	Concatenate the list of collections "A" along dimension 1
zeros	zeros(T,dims)	Create an array of type "T" with the specified dimensions and all values set to zero

The *params()* notation used for accessing parameter values is not a function. It is short-hand for (*params_file* :: *paramsfile_type*), where *params_file* is the variable storing the parameters and *paramsfile_type* is its data type.

The TOVEC utility algorithm (Algorithm 357), shown below, provides a simpler interface for the Julia reshape function when the desired output is a 1-dimensional vector.

Algorithm 357 ToVec

```
1 function ToVec( M )
2     return reshape( M, length( M ) )
3 end
```

Referenced In: DetermineMinimumCostAction(p. 252), VerticalTRMUpdate(p. 152), IndividualSelectionCostEstimation(p. 255), StateAndCostEstimation(p. 161)
--

This page intentionally left blank.

Appendix K Design Notes

K.1 Overview

This appendix provides guidance on several low-level design considerations which are out of scope of the high-level algorithmic specifications found elsewhere in this document. The recommendations found here are aimed at ACAS X software developers and will point to design concerns and choices which are best addressed early in the development process.

K.2 Interfacing with ACAS X

K.2.1 General

The ACAS X algorithms are designed to run in a continuous cycle consisting of a period of input processing followed by end-of-cycle report generation. In practice, it is recommended that the implementation cycle consist of two processes:

- A process dedicated to ongoing buffering of sensor readings and transponder events (collecting input data)
- The ACAS X process, consisting of:
 - Pre-report processing of buffered input messages, i.e. the "Receive" STM entry points.
Note, immediate asynchronous processing (i.e. without buffering) of coordination messages with the RECEIVEUEF16UDS30 entry point is recommended for best alerting performance, further discussed below.
 - STM and TRM report generation on a fixed timestep, nominally on a 1 Hz schedule

There are several constraints placed on the relative ordering of entry point calls, described in Section 2.2.3.12 ("Interfaces With Other Systems") in ACAS Xa/Xo MOPS Volume I [16]. As a summary:

- Coordination messages of a tracked intruder have highest priority. ACAS X must process coordination messages before messages of any other type. If these messages are not being processed asynchronously, it is recommended to use a priority queue mechanism to prioritize these messages over others.
- All ownership message processing must be called in a timely order, i.e. with monotonically increasing Time of Applicability (ToA) values
- Similarly, for any given intruder, all input intruder message processing must be called in a timely order with monotonically increasing ToA values for those input message types that include a "Mode S ID" uniquely identifying the intruder.

The ACAS X TRM algorithms are based on a 1 Hz processing cycle. An appropriate time-overrun scheme should be considered to handle the case when ACAS X cycles are taking longer than 1 Hz, such as in cases of extremely high traffic.

K.2.2 Input Message Entry Points

The following section lays out the entry points to the ACAS X system and considerations in interacting with them. Each call handles a specific sensor reading or transponder message, allowing the STM to combine those individual data points into a coherent, per-aircraft estimation. Entry points are categorized below by the type of traffic they support and their expected rate of use, which can be useful in system design and while debugging test cases.

Mandatory Ownship State Messages:

- RECEIVEHEADINGOBSERVATION: Nominally 1 Hz
- RECEIVERADALTOSERVATION: Nominally 1 Hz
- RECEIVEBAROALTOSERVATION: Nominally 1 Hz
- RECEIVEDISCRETES: Nominally 1 Hz or asynchronously when values change

Entry points for ADS-B inputs:

- RECEIVESTATEVECTORPOSITIONREPORT and RECEIVESTATEVECTORVELOCITYREPORT: Expected message reception rate of 2 Hz per target per report type. For any given ADS-B target, reports are expected to have at least 10 ms of timing separation. If no such timing delay is present, input any position report prior to any velocity report with the same time of applicability.
- RECEIVEMODESTATUSREPORT: Expected message reception rate of 1 Hz per target
- RECEIVEWGS84OSERVATION: Nominally 1Hz. A minimum of 1/3 Hz is required to enable ADS-B functionality. An ownship WGS84 message must be received before intruder ADS-B messages will be processed.

Entry points for DF0 (Mode S) inputs:

- RECEIVEDF0: Expected message reception rate of up to 1 Hz per target. Rate may be lower due to Hybrid Surveillance behavior. See Section 1.3.6.1 ("Hybrid Surveillance Overview") in ACAS Xa/Xo MOPS Volume I [16].

Entry points for Mode C (ATCRBS) inputs:

This ADD provides example implementations for Mode C traffic processing and supports two methods of ModeC reply processing:

- RECEIVEMODECREPLY: Asynchronous. Single Mode C reply processing. Assumes that, through additional front-end processing, the input *replies* are pre-filtered against extreme range and pre-associated to a unique track id (*reply.external_ID*)
or
- RECEIVEMODECREPLIES: Nominally called at 1 Hz. Batch Mode C reply processing. Does not assume any reply pre-merging or pre-association.

Entry points for coordination inputs:

- RECEIVEUF16UDS30: Expected message reception rate of 1 Hz per target while coordinating. To be processed with the highest priority if a buffer-and-process methodology is chosen. The preferred alternate method is immediate asynchronous processing. This entry point is unique in that it may be asynchronously updated until the time of the UPDATEINTRUDERINPUTS call in STATEANDCOSTESTIMATION and still inform the final TRM report. This should be taken into account when designing the entry point processing architecture.
- RECEIVECAPABILITYREPORT: Asynchronous. Future ADS-B version 3 report type which describes an intruder's CAS equipage and coordination capabilities. Required to support coordination with passive CAS equipages. If no Capability Report is input, active CAS equipage defaults to TCAS.

Entry points for target designation inputs (e.g. Xo applications such as DNA or CSPO-3000):

- RECEIVETARGETDESIGNATION: Received asynchronously from the flight deck when the values change
- STMHOUSEKEEPING: To be called immediately after TRM report generation
- RECEIVECAPABILITYREPORT: See "Entry points for coordination inputs" above. Provides information about whether active CAS equipage is ACAS X or TCAS. Needed for coordination with CAS-equipped intruders designated to DNA.

K.2.3 Report Generation

Once per cycle, after the period of input message processing, the implementation generates STM and TRM Reports. This process is performed in the following sequence:

1. Call GENERATESTMREPORT to generate an output STMREPORT. The STMREPORT primarily summarizes the tracked state and display information for the ownship and all intruders
2. Input the TRMINPUT substructure of the STMREPORT, along with stored TRM state information, into a call to VERTICALTRMUPDATE. This function will return the TRMREPORT structure which contains important advisory, coordination, and display information
3. Input the TRMREPORT into a call to STMHOUSEKEEPING to update the STM state with new designation information.

K.2.3.1 Module Memory Management

The STM uses global data structures to maintain internal state memory between cycles. If implementing the ADD as specified, the implementation is expected to maintain instances of the following at a level globally visible to the STM:

- target_db::DATABASE{Uint32,TARGET}
- own::OWNSHIPDATA
- hyp_track_db::DATABASE{Uint32,HYPOTHETICALMODECTRACKFILE}
- modecIntervals::MODECINTERVALS

The TRM differs from the STM in memory management. The TRM's state information is not stored globally but is instead kept in a data structure that is passed as an argument to VERTICALTRMUPDATE, provided as the *st_trm*::TRMSTATE parameter. This data structure is actively updated within the VERTICALTRMUPDATE function, and so a pass-by-reference for this parameter is expected if implementing the ADD as specified. The implementation is responsible for persisting the state of this data structure between cycles.

A data structure to hold static parameters, of type *paramsfile_type*, must be prepared and accessible to the implementation. This data structure is to be made globally visible to both the STM and TRM through a *params()* function handle. It is recommended to populate this data structure once at startup by parsing the provided parameters file or to compile the parameters directly into the system's binary.

The implementation will need a reset function for reinitializing the previously mentioned state variables to their initial state. The reset function should be called at system startup, system restart, and between test encounters when testing. The reset method should reinitialize the state objects as defined by their default constructors in their respective data structure definitions. The values in the

Appendix K

K-4

parameter data structure are constant and do not need to be reset during a system restart or between test encounters.



K.3 Specific Implementation Details

K.3.1 Julia Pseudocode

All of the algorithms in this ADD are executable software written in the Julia programming language. These algorithms completely specify the operation of the ACAS X STM and TRM and were used to produce the ACAS X test suite supplement to the ACAS Xa/Xo MOPS. The algorithms also satisfy the requirements in the ACAS Xa/Xo MOPS Volume I [16].

This section highlights aspects of the Julia programming language requiring attention during implementation in other programming languages. Julia data types are listed in Table 1-2. Julia keywords are listed in Table 1-3. Standard Julia functions, including math functions, are described in Appendix J.

K.3.1.1 Julia Math and Array Notation

Julia has built-in math and array operators which are used in the algorithms throughout this document. Several of the math operators are used for manipulating vectors and matrices. In other programming languages, it will be necessary to implement some or all of these operations as functions. The following table describes Julia operators used in this ADD.

Table K-1. Julia Operators

Operator	Description
A'	Matrix transpose of A
A .+ B	Elementwise addition of arrays A and B; B may be the same size as A in all dimensions or only one dimension
A .- B	Elementwise subtraction of matrices A and B; B may be the same size as A in all dimensions or only one dimension
A + B	Elementwise addition of arrays A and B; A and B must have the same dimensions
A - B	Elementwise subtraction of arrays A and B; A and B must have the same dimensions
A * B	Elementwise multiplication of arrays A and B; A and B must have the same dimensions
A / B	Elementwise division of arrays A and B; A and B must have the same dimensions
x ^ y	x^y
&	Bitwise AND
	Bitwise OR
\$	Exclusive OR
»	Right bit shift operator
«	Left bit shift operator
x += y	Equivalent to $x = x + y$
x -= y	Equivalent to $x = x - y$
x *= y	Equivalent to $x = x * y$
x /= y	Equivalent to $x = x / y$
[]	Construct an uninitialized array of type "Any"
T[]	Construct an uninitialized array of type T
[x y]	Construct a one-dimensional array (row vector) containing x and y
[x, y]	Construct a one-dimensional array (column vector) containing x and y
[A, B]	Construct a one-dimensional array (column vector) by concatenating the contents of column vector B to the contents of column vector A
[x1 x2 ; y1 y2]	Construct a two-dimensional array (matrix) with x1 and x2 in row 1 and y1 and y2 in row 2
A[1,:]	Returns all of the elements in row 1 of array A
A[:,1]	Returns all of the elements in column 1 of array A
A[1,2]	Returns the array element $A_{1,2}$
A[end]	Returns the last element in array A

K.3.1.2 Automatic Data Type Conversions

Algorithms in this ADD define the data types of their input arguments. However, the data types of local variables are not always defined. Julia assigns data types to variables at the time that a line of code is executed. The symbol ":" following a variable name is a data type assertion that enforces the variable having the indicated type at that point. If a variable is not forced to have a particular type, Julia will choose a type for the variable when a value is assigned to it. In that case, the data type of the variable can change within an algorithm. When possible, Julia performs automatic type

conversions when assigning a value of one type to a variable with a different type. An example of an automatic type conversion that is not possible is a conversion from NaN to an integer value.

Implementers should be aware of the potential for implicit data type conversions in the ADD and be careful to declare variable types appropriately.

K.3.1.3 Handling Symbols

This ADD makes use of a Julia language primitive called a "symbol", seen as an identifier preceded by a single colon. For example, see ":Up" in the VRCToSENSE algorithm. This type is similar to an enumeration with the exception that the possible values do not need to be predefined. If the implementation language does not explicitly support symbols, it is recommended to instead use enumerations or constants.

K.3.2 Single versus Multi-Threaded Execution

The ACAS X algorithms were developed and tested as a single-threaded application, and the algorithms in this ADD do not describe any concurrency protection. However the ACAS X algorithms design is intended to support adaptation to run asynchronously and concurrently, especially for coordination processing. If pursuing this design, make sure to introduce protective code to prevent concurrent access to all globally shared state data and intruder coordination data.

K.3.3 32-Bit versus 64-Bit Applications

The ACAS X implementation will not require more than 2 gigabytes of RAM (excepting any exceptionally large memory allocations introduced by the implementer) and so both 32-bit and 64-bit applications are suitable. Note that testing and evaluation of the algorithms in this ADD used 64-bit integer and floating point values. See Appendix K.3.4 ("Floating Point Precision") below for information regarding 32-bit vs 64-bit data types.

K.3.4 Floating Point Precision

This ADD represents all real values with the "R" symbol and uses a 64-bit floating primitive to store these values. If 32-bit floats are preferred over 64-bit floats in an effort to improve processing efficiency or reduce memory usage, it is recommended to closely monitor the function-level results as some algorithms are particularly sensitive to precision loss.

The ACAS X algorithms have not been tested for robustness over the full range of possible floating point inputs. The implementer is responsible for assuring that all algorithms operate as intended for all such inputs. The following is recommended:

- Adjust algorithms to maximize numerical stability. Some algorithms may improperly introduce a loss of precision through repeated mathematical operations/transformations on the input data, especially with extremely small and large inputs. To best preserve the dynamic range of output, an implementation might reorder an algorithm's order of operations or use intermediate variables with higher bit precision than is specified in the ADD
- Introduce checks to handle otherwise unexpected exceptional floating point values, such as NaN and Infinity

Where floating point equivalence ("==") is tested in the ADD algorithms, the comparison should not be changed. For example, in INTERPOLATEONEDIMENSION the comparison (*cuts[i] == point*) is intended to apply only when the two values are exactly equal.

K.3.5 Subsystem Modularity

This ADD maintains the STM and TRM as two separately encapsulated modules, with the following notable exceptions:

- The UPDATEINTRUDERVRC algorithm allows the TRM to directly access the STM target database, creating a dependency between the two modules
- A single parameters structure is shared between the two modules

These inter-dependencies should be taken into account in early system design.

K.3.6 Running Julia as a Development Tool

The Julia code included in this document completely describes the behavior of the prescribed and suggested ACAS X algorithms. All necessary parameters, including data structure sizes, are either directly specified or can be derived from specified information through a static examination of the provided code.

In place of static analysis, use of the Julia interpreter to examine running instances of the ACAS X algorithms may be a significant aid in quickly determining the types and sizes of various data structures, especially those passed as function arguments or stored as internal state. This dynamic analysis accelerates and simplifies the task of developing low-level implementations of these algorithms in the language of choice.

For example, the Julia algorithms often utilize dynamically sized containers with no pre-defined dimensions. While a static analysis will reveal the ultimate size of these containers, running the algorithms in the Julia interpreter and inspecting the data states during execution (utilizing triggered breakpoints, targeted print statements, or other forms of inspection) provides an alternate and possibly more efficient means to establish these dimensions.

One specific example is the *Sigma_vert* field within the MODESTRACKFILE. It is defined as matrix with real-valued elements with no additional information on its dimensions. Running the Julia algorithms with nominal input and printing out the state of *Sigma_vert* periodically demonstrates the matrix to be of 2x2 size. The ACAS Xa Test Suite provides a near-comprehensive exercise of all ACAS X branches and may be utilized in demonstrating array sizing with a high confidence of covering all dynamic structure states.

The following Julia commands are useful to inspect runtime behavior:

Table K-2. Example Julia Commands

Command	Output	Description
println("Sigma_vert", trk.Sigma_vert)	Sigma_vert[73.0 0.0 0.0 125.0]	Print matrix
println("Size = ", size(trk.Sigma_vert))	Size = (2,2)	Print matrix dimensions
dump(trk)	ADSBTrackFile modes: UInt32 999 non_icao: Bool false rebroadcast: Bool false updates_pos: Int64 2 ...	Prints the contents of a structure (and any substructures) with field labels
@bp	*breakpoint at this line*	Provides an interactive terminal at this location (requires the external Julia "Debug" package)
@bp size(trk.Sigma_vert) != (2,2)	*breakpoint at this line if a condition is met*	Triggers a breakpoint at this location if ever the Sigma_vert matrix is not 2x2 at this line (requires external Julia "Debug" package)

K.3.7 Static Allocation

This ADD uses dynamically sized containers for the data storage and the passing of arguments. For a given implementation, dynamic memory allocation may be impossible or may be undesirable for a safety-critical embedded target, with tightly controlled memory management being preferred. In such cases, it is recommended to statically preallocate a container with a maximum storage size and to mark the number of populated indices with an accompanying counter variable.

The table below points out several prominent dynamically sized containers and suggests a recommended minimum capacity for a statically sized replacement. In practice, the system should scale these capacities to take effective advantage of all memory that is made available to the ACAS X process.

Note for all such statically sized replacements, protective code must be added to handle cases where storage demands exceed what is available, usually necessitating the system to discard data or to return from the function without further action. It is recommended to intelligently shed data as to minimally affect system performance in this regard. For example:

- Prefer to maintain and report intruders which are closer to ownship and especially those in-

volved in active advisories, coordination, or target designation

- Prefer to persist track data elements which are closest or most recent

Table K-3. (Minimum) Recommended Container Allocations

ADD Container	Suggested Size	Description	Sizing Rationale
target_db	120	Number of intruders independently tracked in the STM	An arbitrary limit chosen based on observed flight recordings
StmReport.display	30	Number of intruder records passed to the hardware display by the STM	An arbitrary limit chosen based on observed flight recordings
TRMInput.intruder	30	Number of intruder records passed to and processed by the TRM	An arbitrary limit chosen based on observed flight recordings
TRMReport.coordination TRMDesignationData.intruder TRMDisplayData.intruder TRMDebugData.intruder	61	Number of intruder records included in the TRMReport	A precise limit of twice the TRMInput.intruder array length, doubled to support the maximum number of dropped intruders independent of designation, plus 1 for the limit on the number of designated intruders
Target.modec_tracks	60	Number of Mode C tracks contained in an intruder	An arbitrary limit chosen based on observed flight recordings
Target.adsb_qual_history	3	Size of ADS-B Quality History	A precise limit tied to the min_adsb_quality.n parameter
hyp_track_db	480	Number of stored hypothetical Mode C tracks	An arbitrary limit chosen based on observed flight recordings
hyp_track_db.replies	6	Number of Mode C replies contained in a Hypothetical Mode C track	A precise limit calculated as the theoretical maximum

Table K-4. (Minimum) Recommended Container Allocations (Continued)

ADD Container	Suggested Size	Description	Sizing Rationale
ReceiveModeCReplies <i>replies</i> argument	600	Maximum number of replies handled per cycle by the ReceiveModeCReplies entry point. The maximum sizes of derivative arrays such as merged_replies, unassociated_replies, and associated_tracks should also be set to this value	An arbitrary limit informed by vendor hardware capabilities
TRMOwnInput.belief_vert	1	Number of vertical belief states maintained for Ownship	An arbitrary limit chosen by convention
TRMIIntruderInput.belief_vert	5	Number of vertical belief states maintained per intruder	An arbitrary limit chosen by convention
TRMIIntruderInput.belief_horiz	9	Number of horizontal belief states maintained per intruder	An arbitrary limit chosen by convention
OfflineCostEstimation <i>b_tau_ec_int</i> variable	41	Maximum number of tau estimation bins	A precise limit derived from table properties

K.3.8 Target IDs

The STM generates and assigns a numeric identifier to each Target. This Target identifier is propagated through the STM and TRM output data structures as the *id* field. The *id* can be used to associate the outputs from the STM and TRM with each other. It is recommended that implementers ensure identifiers are generated such that any given value of *id* is not reused until the new Target can reliably be disambiguated from any prior Target with the same id.

K.4 Error Checking

This ADD aims to provide an abstract, high-level representation of the code and leaves the issue of protective code and error handling to the implementer. Depending on the implementation language and architecture, unprotected operations may lead to immediate runtime errors, undefined belief states leading to later errors, or generally degraded performance.

It is advised that the implementers independently apply best coding practices and achieve any necessary DO-178C objectives [14] in this regard to ensure reliable and robust runtime behavior.

The lists provided below are included as examples and are not comprehensive lists; implementers are responsible for ensuring the robust and reliable operation of their implementation.

K.4.1 Divide-by-Zero Errors

The section points out areas of the ADD which are susceptible to divide-by-zero errors. To address these susceptibilities, protective checks may be added. A protective check may consist of replacing zero denominators with a small epsilon number, returning from the enclosing function with a nominal value, or avoiding a block of code which relies on that denominator.

Critical areas to review and to consider for protection include the following:

ADVANCEADSBTRACKPOSITION:

When implemented correctly, the denominator dt values enumerated below are prevented from approaching zero by a check against the *min_obs_toa_step* parameter value in the RECEIVESTAT-EVECTORPOSITIONREPORT calling algorithm. If an additional error check is desired in this context, the implementer may choose to bypass the velocity seeding and track prediction step.

- $\text{trk.mu_hor[2]} = (\text{x_rel}-\text{trk.mu_hor[1]})/\text{dt}$
- $\text{trk.mu_hor[4]} = (\text{y_rel}-\text{trk.mu_hor[3]})/\text{dt}$

ADVANCERANGETRACK:

When implemented correctly, the denominator dt value below is prevented from approaching zero by the preceding check against the *dt_too_small_for_velocity* parameter value.

- $\text{trk.mu_rng[2]} = (\text{GroundRange}(\text{r_slant}, \text{z_rel}) - \text{trk.mu_rng[1]}) / \text{dt}$

BAROALTATTOA:

When implemented correctly, the denominator value below is prevented from approaching zero by a check against the *min_obs_toa_step* parameter value within the RECEIVEBAROALTOBSERVATION algorithm (which adds entries to the *own.history.baroalt* structure). If an additional error check is desired in this context, the implementer may choose to bypass this altitude interpolation step, returning the value of a single altitude measurement retrieved from the *own.history.baroalt* structure.

- $\text{rate::R} = (\text{b2} - \text{b1}) / (\text{time2} - \text{time1})$

CALCULATEIMAGERANGERATE:

No explicit protection in the suggested algorithms prevents the denominators below from approaching zero. If an error check is desired in this context, the implementer may choose to bypass image

correlation for the track in the event of zero-valued denominators.

- $\text{im_rangerate} = (\text{real_im_rangerate} + ((\sqrt{(\text{tm1} * \text{tm1} + \text{tm3})}) * (\text{own_alt_rate} + \text{real_alt_rate})) + \text{tm4}) / (2 * \text{im_range} - \text{real_im_range})) / 2$
- $\text{im_rangerate} = ((\sqrt{(\text{im_range2} + \text{tm3})}) * (\text{own_alt_rate} + \text{real_alt_rate})) + \text{tm4}) / \text{im_range}$

CONVERTECEFTOWGS84:

Only in geographically improbable locations can the denominators below approach zero. These cases are partly protected against by outlier detection within the RECEIVESTATEVECTORPOSITIONREPORT and RECEIVESTATEVECTORVELOCITYREPORT algorithms. If an additional error check is desired in this context, the implementer may choose to abandon the calling receive algorithm in the case of a zero-valued denominator.

- $C = (e1_{sq} * e1_{sq}) * F * (r_ecef * r_ecef) / (G * G * G)$
- $P = F / (3 * ((S + (1/S) + 1) * \bar{C}^2) * (G * G))$
- $\text{inner_square_root} = ((a * a) / 2) * (1 + (1/Q)) - (P * (1 - e1_{sq}) * (Za * Za)) / (Q * (1 + Q)) - P * (r_ecef * r_ecef) / 2$
- $Z_0 = ((b * b) * Za) / (a * V)$
- $h = U * (1 - (b * b) / (a * V)) * ecef / (G * G * G)$

CONVERTORTHOMETRICTOGEOEDETICHEIGHT:

Only in geographically improbable locations can the denominators below approach zero. These cases are partly protected against by outlier detection within the RECEIVESTATEVECTORPOSITIONREPORT and RECEIVESTATEVECTORVELOCITYREPORT algorithms. If an additional error check is desired in this context, the implementer may choose to abandon the calling receive algorithm in the case of a zero-valued denominator.

- $\text{geo_h::R} = R_{\phi} * h_o / ((R_{\phi} * \gamma_s_{\phi} / \gamma_s_{45}) - h_o)$

HEADINGATTOA:

Under most circumstances the denominator value below is prevented from approaching zero by a check against the *min_obs_toa_step* parameter value within the RECEIVEHEADINGOBSERVATION algorithm (which adds entries to the *own.history.heading* structure). If an additional error check is desired in this context, the implementer may choose to bypass the heading interpolation step, returning the value of a single altitude measurement retrieved from the *own.history.heading* structure when two exists with similar time values.

- $\text{rate::R} = \text{AngleDifference}(h1, h2) / (\text{time2} - \text{time1})$

HYPOTHETICALTRACKTEST:

No explicit protection in the suggested algorithms prevents the denominators below from approaching zero, however the replies referenced are provided in subsequent calls to RECEIVEMODECREPLIES. An implementation which ensures adequate variation in the *toa* values of the Mode

C replies provided to subsequent calls of RECEIVEMODECREPLIES will avoid a potential divide-by-zero error. If an additional error check is desired in this context, the implementer may choose to return from the algorithm immediately if the *toa* values are equal.

- rangerate::R = (replyA.r_slant - replyC.r_slant)/(replyA.toa - replyC.toa)
- trk.rangerate = (replyA.r_slant - replyC.r_slant)/(replyA.toa - replyC.toa)

LINEARTRANSFORM:

The linear transform between Cartesian and polar coordinates is undefined when range is zero. In this circumstance the implementation can assume an epsilon range value or apply other numerical techniques to complete the transformation.

- $H = [\mu_{xy}[1]/\mu_{ra}[1] \mu_{xy}[2]/\mu_{ra}[1]; \mu_{xy}[2]/\mu_{ra}[1]^2 - \mu_{xy}[1]/\mu_{ra}[1]^2]$

MAHAL:

Under correct usage the Mahalanobis distance utility function should not be called with two zero-valued variance or covariance parameters. In such a circumstance, the implementer may choose to return a normalized distance measure between the two mean inputs.

- $D = (dmu^*dmu)/\Sigma A$

NORMALIZE:

This algorithm is used only in the TRM and only for normalizing weights. If the STM is operating correctly, there will always be at least one weight value in the vector and the sum of the weight values will never equal 0. If an error check is desired in this context, the values in *x* should be modified only when there are at least two values in *x* AND the sum of the values in *x* is greater than 0. If the algorithm is to be used more generally, it is recommended the implementer assess the need for additional error checking in that context.

- $x_norm::VectorR = x / \text{sum}(x)$

K.4.2 Invalid Input

Certain functions have undefined behavior for a subset of possible inputs. It is recommended that the implementation include checks to prevent against invalid inputs to such functions.

The first such function is *atan2*. Julia returns 0.0 for *atan2(0,0)*. It is recommended implementers use a version of *atan2* that produces the same result.

The second such function is *sqrt*. Cases are pointed out below where the *sqrt* function may have a negative input. The following steps can be taken to protect these areas of the code:

- Swap in a custom version of the function which explicitly returns a nominal value for invalid input cases
- Check the inputs before the expression is called, returning from the function with a nominal value
- Check the inputs before the expression is called, skipping the block of code in question
- In the case of *sqrt*, take the absolute value of the inputs

Critical areas to review and to consider for protection include the following:

CONVERTECEFToWGS84:

- $S = (1 + C + \sqrt{C*C+2*C})*(1.0/3.0)$
- $Q = \sqrt{1+2*(e1_sq*e1_sq)*P}$
- $V = \sqrt{(r_ecef-e1_sq*r_0)^2 + (1-e1_sq)*(Za^*Za)}$

MAHAL:

- return $\sqrt{D[1]}$

