

: A G'E-Review &\$\$) Cover Sheet

\$+/% /%&

Review ID:		Sä^Ä~ &^K		DO-178 Level:	
Review Type:		ACM Project:		Rework Effort (hours):	
Produced:		ACM Subproject:		Closure Effort (hours):	

Ü^çã, : Date Time	Meeting Duration:	Moderator Closure →
Ü^çã, Ä[&ä]:	# Ü^çã, Participants:	APPROVED By Liu Mingyang at 11:58 am, Jul 06, 2014
Ö[] ^!^ &^Ä[[{ K	Date Complete:	Audit: Stamp Here
Telephone Participant Code:	Review Status: (result of review)	

Work Product Type(s): Supporting A aterial(s) / Comments:

--	--

Ü:[ä~ &^!Ä[]][^!^K	Uç^!•ä @Öjã ä ^K
# of Ü:[ä~ &^!Ätechnical Öefects:	fÄUç^!•ä @Öjã Ätechnical Öefects:
# of Ü:[ä~ &^!Äon-technical Öefects:	# of Uç^!•ä @Öjã Äon-technical Öefects:
# of Ü:[ä~ &^!Ärocess Öefect•:	# of Uç^!•ä @Öjã Ärocess Öefects:

Work Products Under Review

Reuse Scope:

Problem Report	File Name	File Version	Review Size	Size Units	Approved Version

Participants

Name	Function (discipline)/ Responsibility	Review Time (hours)	Role in review	Attend	Will Close	Signature check complete
						REVIEWED By Chen Yongbing at 1:31 pm, Jul 02, 2014
						REVIEWED By Liu Mingyang at 11:56 am, Jul 06, 2014
						REVIEWED By Zou Xing at 11:55 am, Jul 06, 2014
						REVIEWED By Zhou, Qiong at 4:08 pm, Jul 03, 2014

Assignee's signature (stamp) confirms that a review was performed and any action Items and markups were incorporated or dispositioned.

Participant's signature (stamp) confirms participation in the review. A lack of signature (stamp) indicates nonparticipation.

Moderator's signature (stamp) indicates record is complete.

Uç^!•ä @Ü^çã, ^!CÄä) æ' ^!Äçä] Dä ääæ• ÄçäÜ"]] |ä!ÄUç^!•ä @Ä æÄ[] ä~ &^!ÄÄ

Üæ^C Ä[&äCÄä) æ' ^!Äçä] Dä ääæ• Ä çäÄ] ää) Ä ÄæÜæ^C Ä[&äÄ

Coversheet Continued

[illegible]

**Component Test Procedure (CTP)
Checklist**

(CTP_CHECKLIST_WORD.doc 10/24/07)

ACM Project: _____

ACM Sub-Project: _____

SCR Number: _____

Affected Area: _____

Overview: CTPs are generated to verify an individual software element or group of elements properly implement requirements the software element(s) trace to. Use this checklist to inspect test cases and associated test procedures, drivers, and stubs against requirements the software element(s) implement. The CTP(s) are verified to conform to standards, and fully test requirements with appropriate structural coverage. The associate tracing data and test coverage analysis/disposition data (if any) is also verified.

Misc Info Reference: FMS Test Process C71-5780-043, Section 5.

Yes No N/A Administrative

1. Do the CTPs elements follow the standard naming conventions?

CTP_<A/C>_<FAREA>_<FUNC-NAME>.TDF file – CTP Test Definition File

CTP_< A/C >_<FAREA>_< FUNC-NAME >.ZIP file – miscellaneous test related files

CTP_< A/C >_<FAREA>_< FUNC-NAME>.TRT file – CTP Trace file(Core only)

CTP elements configured in the CM tool:

2. Is *.TDF file – CTP Test Definition File present?
3. Is *.ZIP file present?
4. Is *.TRT file – CTP Trace file present (Core only)?

Review Packet information details:

5. Is SCR Number and a copy of the SCR (Sec state) present?
6. Is TDF, TRT(If present), ZIP files with correct generation information present?
7. Support files (SRD, SDD, and Checklist) with Generation information.
8. Does the review packet contain a difference listing of the old test to the new test and are the differences limited to the changes specified in this SCR?
9. Is the version of the material under review and supporting material correct for the SCR(s)?
10. Has the material/version been identified on the cover sheet of the review packet (may reference SCR)?
11. Have all SCR fields (e.g. Analysis/Solution) been filled out properly?

Component Test Procedure (Ctp) Checklist

Yes No N/A TDF (CTP Test Definition File)

Does the TDF header include the following fields:

12. Does the TDF header include the following fields:
 - Filename
 - Title
 - Author
 - Creation Date
 - Modification History
 - Source
 - Description of TDF
13. Is the SCR number and description updated for this SCR?
14. Does the TDF header include a unique ANCHOR name for this CTP?
15. Is the list of SRD/SDD element references (and their generation numbers) updated and correct? (including formatting of this information)

Yes No N/A ZIP File (CTP Related Miscellaneous Files)

16. Does the ZIP file contain the updated necessary test files ?
 - *.BAT
 - *.CUL
 - ~~*.DRV~~ (*_D.ADA)
 - *.VER (~~*.RST~~)
 - *.RPT
 - Optional files: STB, DSP, and INC (if necessary).
 - Has the *.CUL file been updated to show the correct span of source code procedures/functions that are being tested by this CTP?

Component Test Procedure (Ctp) Checklist

Yes	No	N/A	TRT File (Core only)
-----	----	-----	----------------------

17. Does the TRT header include the following fields:

- Filename
- Title
- Author
- Creation Date
- Modification History
- Is the modification history with date, author, SCR number, and description updated??

18. Has the traceability matrix been updated/verified (trace to the correct requirements)??

Yes	No	N/A	Test Case Design
-----	----	-----	------------------

19. Are the test case ID numbers present in sequential order?

20. Does the test script have test case descriptions which describe the objectives, intent, and operation for each test case?

21. Are all the allocated requirements tested?

22. If anchor is found to be a bad trace or vague/ambiguous, has it been disposed with a reference SCR.

23. Does the test case description section of each test case identify the specific requirements (SRD anchors) that are being tested?

24. Does the test case description section of each test case identify the specific requirements (SRD anchors) that are supporting requirements?

25. To ensure robust testing, are all test cases inputs set with at least 2 different values?

26. To ensure robust testing, are boundary conditions and tolerances tested where ever applicable?

Component Test Procedure (Ctp) Checklist

Yes	No	N/A	Test Case Design con't
			27. Coverage Levels – Has every point of entry and exit in the program been invoked at least once?
			28. Coverage Levels – Has every decision in the program taken on all possible outcomes at least once?
			29. Coverage Levels – Has every condition in a decision in the program taken on all possible outcomes at least once?
			30. Coverage Levels – Has every condition in a decision been shown to independently affect that decision's outcome? A condition is shown to independently affect a decision's outcome by varying just that condition while holding fixed all other possible conditions.
			31. Data Coupling – Are there test cases which exercise “data coupling” between software modules (i.e., the dependence of a software component on data not exclusively under the control of that software component)?
			32. Data Coupling – Are there test cases which exercise “control coupling” between software modules (i.e., the manner or degree by which one software component influences the execution of another software component)?
			33. Error Guessing - Do areas in the software known to have complex algorithms have a sufficient number of test cases to ensure they are working as expected?
			34. Error Guessing - Do areas in the software associated with complex requirements have a sufficient number of test cases to ensure they are working as expected?
			35. Outputs - Are all test case outputs measured for at least two different values?
			36. Outputs - Have variables with expected output values been initialized to other values before input to the test process (e.g., If a variable is expected to have an output result of TRUE, is the input state of this variable set to FALSE before executing the test case?)
			37. Coverage Analysis - Are the entire test paths covered as per the structural coverage requirements mandated for Flight Management Systems? If not, are such structural coverage deficiencies dispositioned? If not determined to be a tool problem, then the disposition must reference to an SCR.
			38. Coverage Analysis - For uncovered requirements, is there another test that provides the coverage?
			39. Has the Test name and Anchor required if one exists, been identified? If not, has an SCR been written and the SCR number referenced?
			40. Coverage Analysis – Have all the failures been analyzed and disposed appropriately in the DSP quoting a correct SCR number documenting the reason for the failures.

Component Test Procedure (Ctp) Checklist

Yes	No	N/A	Polymorphism Related Issues (C++)
			41. Has the code under test been examined for the existence of dynamic dispatch (can be determined by virtual functions in the code or a virtual table in the assembly code)?
			42. Does each test case appearing in the set of test cases associated with a class appear in the set of test cases associated with each of its subclasses?
			43. If dynamic dispatch is involved in the execution of a function, is the method separately tested in the context of every concrete class in which it appears, irrespective of whether it is defined by the class or inherited by it? An exception is made for simple get and set methods that only assign a value to, or return the value of an attribute or association. Such methods need only be tested once, in the context of the defining class.
			44. Are errors dispositioned to an SCR or has the test been updated?
Yes	No	N/A	Other
			45. Are all defects identified by the previous questions?

N N/A Justification Box

Change Category: PROBLEM SCR No.: P 15655.04
SCR Status: SEC SCR Status Date: 1-JUL-2014
Originator: Patrick Caulfield Date Originated: 13-AUG-2013
Affected Area: PERF Customer No.:
Assignee: Chen, YongBing Priority: 3
Verification Assignee: Zou, Xing
Found in Configuration: CERT2_LD_5 Hardcopy Attachment: None
Target Configuration: BP3_TST_REV_X03

Planned Impact: Test
Found During: CUSTOMER ACTIVITY
TcSE Assignee:
TcSE Verifier:
Uses TsCE: No
Task:

SCR Copied To: < None Entered >

SCR Copied From: < None Entered >

SCR Reissued To: < None Entered >

SCR Reissued From: < None Entered >

Title: Step Location Toggling with RTA While On Ground

Description:

In BP2 load 3, Perf SCR 11939 made corrections to the step stability logic that applies when an RTA is active. This logic works to keep the step position stable, since RTA speed changes tend to make the step move, which impacts the RTA solution, causing feedback. A scenario has been found in which the fix made for that SCR is failing, however. The scenario:
ENG RATING: TRENT 1000-A FMS OPS: HNP2D-CL11-506G Model: 787-8
On the ground at KBFI. ORIGIN: KBFI DEST: KSFO PERF: Fuel 75, ZFW 300, RES 5, FL330, CI 100. RTE: KBFI.13R.LACKR1.ORTIN, Dir To ALDER, J1 to RBL, J126 to SACE, Dir to UPEND, Dir to NORMM, KSFO.RNV10L. Close any discons on LEGS page. WINDS entered on LACKR: FL410 086/80, FL280 086/100, 11000 086/60, 0500 086/15. RTA PROGRESS page: RTA FIX = RBL, MAX SPD = .900, T/O Time = 1825 (15 minute ahead of current time), ETA at RBL prior to Entering RTA = 1924.6; RTA at RBL = 1923.0. Observation: On VNAV RTA CRZ page, Step to FL370 is toggling between NONE and at a time displayed 2R VSD is displaying the toggling step climb. This was observed first by Boeing (Jay Koszola and Mark Wisted) and reproduced by me using build 2099 (BP2 RL5). The step is toggling between being taken 30nm or so after top of climb, to not being taken at all. Also, UNABLE RTA is toggling, since when you don't take the step the RTA cannot be met.

Analysis/Solution:

27-Dec-2013-Akhil J-HTSB- This scr needs to be retarget to load 2 (BP3_TST_REV_X02) as problem scr is target to load 2.

=====

< Solution field continued > SCR No. 15655.04 Page 2 of 2

4/28/14 (K. Do): Retargeted Test split to BP3 Load 3 since parent and other splits were moved to Load 3.

<2-Jul-2014><HTSC-E803143-Chen Yongbing> Updated CTP_B787_PERF_CRZINITE (TDF:6,ZIP:8,TRT:3)for B787 BP3 Load3 on Build SBC2415_93C and executed in simics mode.

TDF:-

1. Updated SRD generation:

FMF_PERF_PREDSD_CRZ_PHASE.SRD; 22 ->

FMF_PERF_PREDSD_CRZ_PHASE_SRD.DOCX; 23.

2. Updated breakpoints as per SCR 15655.03.

3. Updated TCs 1,6 for newly added PERF_SRD_B_00413 as per SCR 15655.01.

ZIP:- Modified STUB,GPR,BAT,DSP,VER and RPT file.

TRT:- Added anchor PERF_SRD_B_00413 as per SCR 15655.01.

Elements Affected:

Doc.	Element	Generation
TRACE	CTP_B787_PERF_CRZINITE.TRT	3
TST	CTP_B787_PERF_CRZINITE.TDF	6
TST	CTP_B787_PERF_CRZINITE.ZIP	8

ASSIGNEE: Chen, YongBing

Date: 2-JUL-2014

VERIFIER:

Date:

CCB COORDINATOR:

Date:

Closure Category: Fixed/Added Duplicate SCR No.: 00000.00

Project Status: Done

Addendum/Trgt_Date:

Visual Review Info:

Cert/Sys Concern: 0 - CC1/S1 None/Level 1

Cust Notification: 0 - CN1 None

Expected Inservice: 0 - I1 Not expected to occur in-service

Flight Deck Effect: 0 - FD1 None

Non Customer Input: 0 - P1 None

Workload Wrkaround: 0 - W1 No Workaround Necessary

Must Fix: 0 - MF1 Use Score

Score/Meeting:

Score Comment:

Closed in Config.: BP3_TST_REV_X03

Left File: D:\B787_Download\CTP_B787_PERF_CRZINITE\OLD\CTP_B787_PERF_CRZINITE.TDF Right File: D:\B787_Download\CTP_B787_PERF_CRZINITE\NEW\CTP_B787_PERF_CRZINITE.TDF

	40			2. Updated breakpoints as per SCR 15655.03.
	41			3. Updated TCs 1,6 for newly added PERF_SRD_B_00413
	42			as per SCR 15655.01.
35	43			
36	44	SRD and SDD DETAILS	: FMF_PERF_CRZ_PHASE.SDD ; 12	
37			FMF_PERF_PREDS_CRZ_PHASE.SRD ; 22	
	45		FMF_PERF_PREDS_CRZ_PHASE.SRD.DOCX ; 23	
38	46			
39	47	TRACE DETAILS	:	
40	48		ANCHOR : PERF_TEST_00014	
41	49			
42	50	SOURCE	: SDD: FMCS_19_21027005	
43	51		SRD: FMCS_19_20006025, FMCS_19_20006026, FMCS_19_20006027, FMCS_19_20006028,	
44	52		FMCS_19_20006029, FMCS_19_20006097, FMCS_19_20006098, FMCS_19_20006099,	
45			FMCS_19_20006102, FMCS_19_20006100	
	53		FMCS_19_20006102, FMCS_19_20006100, PERF_SRD_B_00413	
46	54	*****		
47	55	INITIALIZATIONS:		
48	56			
49	57	FP_DEF_TOL = 0.0001		
50	58			
51	59	SUT_VARS		
52	60			
53	61	boolean'()		
54	62	Test_Firstpass		
55	63	Test_Steptype		
56	64	Test_LGB_Search		
57	65	AC_Position_Types.SC		
58	66	AC_Position_Types.At_Alt		
59	67	CFP_PERF_STEP_IFTYPES.Nostep		
60	68	CFP_PERF_STEP_IFTYPES.specstep		
61	69	CFP_PERF_STEP_IFTYPES.Opt		
62	70	CFP_PERF_STEP_IFTYPES.PEopt		
63	71	CFP_PERF_STEP_IFTYPES.PastSpecStep		
64	72	FMCS_Base_Types.climb		
65	73	FMCS_Base_Types.Descent		
66	74	Perf_Preds_Lfdata.Fltphase		
67	75	Perf_Preds_Lfdata.NavPtr		
68	76	Perf_Preds_Lfdata.PrevNavPtr		
69	77	Step_Ptr		
70	78	Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Valid		
71	79	Perf_Integrators_Lfdata.TermBuf.TermArray().Active		

Left File: D:\B787_Download\CTP_B787_PERF_CRZINITE\OLD\CTP_B787_PERF_CRZINITE.TDF Right File: D:\B787_Download\CTP_B787_PERF_CRZINITE\NEW\CTP_B787_PERF_CRZINITE.TDF

72	80	Perf_Integrators_Lfdata.TermBuf.TermArray().Value
73	81	Perf_Integrators_Lfdata.IntProgBuf.Xprog
74	82	Idx_Profile_Ildata.Ialtprofptrec().Dtd.Data
75	83	Idx_Profile_Ildata.Ialtprofptrec().Dtd.Valid
76	84	Perf_LGB_Lfdata.LGB().Fpln_Data.SpAlt1Val
77	85	Perf_LGB_Lfdata.LGB().Fpln_Data.SpAlt1Pos
78	86	Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Stepptr
79	87	Loc_Clb_Step_Exec
80	88	Perf_Crz_Pkg.Step_Size.Valid
81	89	Perf_Crz_Pkg.Step_Size.Data
82	90	Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Fixdistodest
83	91	Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Steptype
84	92	Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Opt_Stepalt
85	93	Next_Step_High
86	94	Next_Step_Low
87	95	Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Data
88	96	Perf_Crzalt_Lfdata.Crzalt
89	97	Start_Sut
90	98	load_ge_config
91	99	
92	100	Perf_Preds_Lfdata.VTPlogic.Firstpass
93	101	Perf_Crzalt_Lfdata.LastCrzAlt.Valid
94	102	Perf_Preds_Lfdata.Vgbptr
95	103	CLB2L
96	104	Perf_RTA_Lfdata.Pred_Pastrta
97	105	Perf_WTS_Lfdata.Always_Compute_Max_Speed
98	106	Perf_Integrators_Lfdata.IntProgBuf.Hprog
99	107	opt
100		
	108	Perf_Crz_Pkg.Opt_Step_PND_Ptr
	109	Perf_WTS_Lfdata.Previous_Pass_Info_Rec.Step().Distance.Data
	110	Perf_WTS_Lfdata.Previous_Pass_Info_Rec.Step().Distance.Valid
101	111	END_SUT_VARS
102	112	
103	113	DEFAULTS
104	114	
105	115	Start_Sut := 1
106	116	load_ge_config := boolean'(true)
107	117	
108	118	END_DEFAULTS
109	119	
110	120	MACRO end_test

Left File: D:\B787_Download\CTP_B787_PERF_CRZINITE\OLD\CTP_B787_PERF_CRZINITE.TDF Right File: D:\B787_Download\CTP_B787_PERF_CRZINITE\NEW\CTP_B787_PERF_CI

```

111 121 # break end_dummy
112 122 # continue
113 123 # delete
114 124 ENDMACRO
115 125
116 126 -- NOTES:
117 127 -- The Crz_InitStepTerms procedure shall implement the listed requirements as specified in the SRD.
118 128 -- FMCS_19_21027005(FMCS_19_20006025, FMCS_19_20006026, FMCS_19_20006027, FMCS_19_20006028,
119 129 -- FMCS_19_20006029, FMCS_19_20006097, FMCS_19_20006098, FMCS_19_20006099,
120 130 -- FMCS_19_20006102)
121 131 -- add FMCS_19_20006100 from perf cycle 2#
122 132 *****
123 133 TESTID: 1
124 134 This verify When If the step type is SpecStep, the StepAlt termination value is set to the Specified step
125 135 altitude.If the aircraft is in climb, and a step was moved to the cruise phase from the climb phase the StepAlt
126 136 value is set to the climb specified step altitude, and the StepDist is activated with a distance value indicating
127 137 the step should begin immediately.
128 138
129 139 REQUIREMENTS UNDER EVALUATION: FMCS_19_20006025, FMCS_19_20006026, FMCS_19_20006027, FMCS_19_20006028, FMCS_19_20006029
130 139 FMCS_19_20006097, FMCS_19_20006098, FMCS_19_20006099, FMCS_19_20006102
140 140 FMCS_19_20006097, FMCS_19_20006098, FMCS_19_20006099, FMCS_19_20006102, PERF_SRD_B_00413
131 141 SUPPORTING REQUIREMENTS : FMCS_19_21027005
132 142
133 143
134 144 --INPUTS:
135 145 -- SETLANGMODE = ADA
136 146 Test_Firstpass := boolean'(false)
137 147 Test_Steptype := CFP_PERF_STEP_IFTYPES.specstep
138 148 Test_LGB_Search := 0
139 149 -- Test_ICAO_Low := 6000.0
140 150 -- Test_ICAO_High := 10000.0
141 151
142 152 -----
143 153 -- set variable for enter SUT
144 154 Perf_Preds_Lfdata.VTPlogic.Firstpass := boolean'(true)
145 155 Perf_Crzalt_Lfdata.LastCrzAlt.Valid := boolean'(false)
146 156 Perf_Preds_Lfdata.Vgbptr := CLB2L
147 157
148 158 Perf_RTA_Lfdata.Pred_Pastrta := boolean'(true)
149 159 Perf_WTS_Lfdata.Always_Compute_Max_Speed := boolean'(false)
150 160
151 161 Perf_Integrators_Lfdata.IntProgBuf.Hprog := 27004.0

```

Left File: D:\B787_Download\CTP_B787_PERF_CRZINITE\OLD\CTP_B787_PERF_CRZINITE.TDF Right File: D:\B787_Download\CTP_B787_PERF_CRZINITE\NEW\CTP_B787_PERF_CRZINITE.TDF

152	162	Perf_Crzalt_Lfdata.Crzalt	:= 27000.0
153	163	-----	
154	164		
155	165	-- Initialize the variable	
156	166	Perf_Preds_Lfdata.Fltphase	:= FMCS_Base_Types.climb
157	167	Perf_Preds_Lfdata.NavPtr	:= 1
158	168	Perf_Preds_Lfdata.PrevNavPtr	:= 2
159	169	--	
160	170	# break Perf_Su_Spd_Utils_Pkg.Su_Frmtgtspdrec	
161	171	# continue	
162	172	# return	
163	173		
164	174	-- enter SUT	
165	175	# break Crz_InitStepTerms	
166	176	# continue	
167	177	Perf_Integrators_Lfdata.IntProgBuf.Xprog	:= 1000.0
168	178	Perf_Integrators_Lfdata.TermBuf.TermArray(11).Active	:= boolean'(true)
169	179	Perf_Integrators_Lfdata.TermBuf.TermArray(11).Value	:= 1.0
170		Idx_Profile_Lfdata.Ialtprofptree(0)(1).Dtd.Data	:= 100.0
171		Idx_Profile_Lfdata.Ialtprofptree(0)(1).Dtd.Valid	:= boolean'(true)
	180	Perf_Crz_Pkg.Opt_Step_PND_Ptr	:= 1
	181	Perf_WTS_Lfdata.Previous_Pass_Info_Rec.Step(1).Distance.Data	:= 100.0
	182	Perf_WTS_Lfdata.Previous_Pass_Info_Rec.Step(1).Distance.Valid	:= boolean'(true)
172	183	Perf_LGB_Lfdata.LGB(2).Fpln_Data.SpAlt1Val	:= boolean'(false)
173	184	Perf_LGB_Lfdata.LGB(2).Fpln_Data.SpAlt1Pos	:= AC_Position_Types.At_Alt
174	185	Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Stepptr	:= 99
175	186		
176	187	Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Valid	:= boolean'(true)
177	188	Perf_Integrators_Lfdata.TermBuf.TermArray(6).Active	:= boolean'(false)
178	189	Perf_Integrators_Lfdata.TermBuf.TermArray(6).Value	:= 1.0
179	190		
180		# break PERF_CRZ_INITSTEPTERMS_SEP.ADA:193	
	191	# break PERF_CRZ_INITSTEPTERMS_SEP.ADA:198	
181	192	# continue	
182	193	Step_Ptr := 99	
183	194		
184		# break PERF_CRZ_INITSTEPTERMS_SEP.ADA:245	
	195	# break PERF_CRZ_INITSTEPTERMS_SEP.ADA:250	
185	196	# continue	
186	197	Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Stepptr	= 99
187	198	Perf_Integrators_Lfdata.TermBuf.TermArray(6).Active	= boolean'(false)
188	199	Perf_Integrators_Lfdata.TermBuf.TermArray(6).Value	= 1.0

Left File: D:\B787_Download\CTP_B787_PERF_CRZINITE\OLD\CTP_B787_PERF_CRZINITE.TDF Right File: D:\B787_Download\CTP_B787_PERF_CRZINITE\NEW\CTP_B787_PERF_CRZINITE.TDF

189	200	
190		# break perf_crz_predexec_sep.ada:657
	201	# break perf_crz_predexec_sep.ada:659
191	202	# continue
192	203	# return
193	204	# return
194	205	
195	206	!end_test()
196	207	--OUTPUTS:
197	208	Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Valid = boolean'(false)
198	209	Perf_Integrators_Lfdata.TermBuf.TermArray(11).Value = 100.0
199	210	-----
200	211	TESTID: 2
201	212	This verify When If the step type is SpecStep, the StepAlt termination value is set to the Specified step
202	213	altitude.If the aircraft is in climb, and a step was moved to the cruise phase from the climb phase the StepAlt
203	214	value is set to the climb specified step altitude, and the StepDist is activated with a distance value indicating
204	215	the step should begin immediately.
205	216	
206	217	REQUIREMENTS UNDER EVALUATION: FMCS_19_20006025, FMCS_19_20006026, FMCS_19_20006027, FMCS_19_20006028, FMCS_19_20006029
207	218	FMCS_19_20006097, FMCS_19_20006098, FMCS_19_20006099, FMCS_19_20006102, FMCS_19_20006100
208	219	SUPPORTING REQUIREMENTS : FMCS_19_21027005
209	220	
210	221	
211	222	--INPUTS:
212	223	-- SETLANGMODE = ADA
213	224	Test_Firstpass := boolean'(true)
214	225	Test_Steptype := CFP_PERF_STEP_IFTYPES.specstep
215	226	Test_LGB_Search := 0
216	227	-- Test_ICAO_Low := 6000.0
217	228	-- Test_ICAO_High := 10000.0
218	229	
219	230	-----
220	231	-- set variable for enter SUT
221	232	Perf_Preds_Lfdata.VTPlogic.Firstpass := boolean'(true)
222	233	Perf_Crztalt_Lfdata.LastCrzAlt.Valid := boolean'(false)
223	234	Perf_Preds_Lfdata.Vgbptr := CLB2L
224	235	
225	236	Perf_RTA_Lfdata.Pred_Pastrta := boolean'(true)
226	237	Perf_WTS_Lfdata.Always_Compute_Max_Speed := boolean'(false)
227	238	
228	239	Perf_Integrators_Lfdata.IntProgBuf.Hprog := 27004.0
229	240	Perf_Crztalt_Lfdata.Crztalt := 27000.0

Left File: D:\B787_Download\CTP_B787_PERF_CRZINITE\OLD\CTP_B787_PERF_CRZINITE.TDF Right File: D:\B787_Download\CTP_B787_PERF_CRZINITE\NEW\CTP_B787_PERF_CRZINITE.TDF

230	241	-----
231	242	
232	243	-- Initialize the variable
233	244	Perf_Preds_Lfdata.Fltphase := FMCS_Base_Types.climb
234	245	Perf_Preds_Lfdata.NavPtr := 1
235	246	Perf_Preds_Lfdata.PrevNavPtr := 2
236	247	
237	248	# break Perf_Su_Spd_Utils_Pkg.Su_Frmtgtspdrec
238	249	# continue
239	250	# return
240	251	
241	252	-- enter SUT
242	253	# break Crz_InitStepTerms
243	254	# continue
244	255	Perf_LGB_Lfdata.LGB(2).Fpln_Data.SpAlt1Val := boolean'(true)
245	256	Perf_LGB_Lfdata.LGB(2).Fpln_Data.SpAlt1Pos := AC_Position_Types.SC
246	257	Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Stepptr := 0
247	258	Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Valid := boolean'(true)
248	259	Perf_Integrators_Lfdata.TermBuf.TermArray(6).Active := boolean'(false)
249	260	Perf_Integrators_Lfdata.TermBuf.TermArray(6).Value := 1.0
250	261	Perf_Integrators_Lfdata.IntProgBuf.Xprog := 1000.0
251	262	
252		# break PERF_CRZ_INITSTEPTERMS_SEP.ADA:193
	263	# break PERF_CRZ_INITSTEPTERMS_SEP.ADA:198
253	264	# continue
254	265	Step_Ptr := 99
255	266	Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Steptype := CFP_PERF_STEP_IFTYPES.PastSpecStep
256	267	
257		# break PERF_CRZ_INITSTEPTERMS_SEP.ADA:245
	268	# break PERF_CRZ_INITSTEPTERMS_SEP.ADA:250
258	269	# continue
259	270	Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Stepptr = 2
260	271	Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Steptype = CFP_PERF_STEP_IFTYPES.specstep
261	272	Perf_Integrators_Lfdata.TermBuf.TermArray(6).Active = boolean'(true)
262	273	Perf_Integrators_Lfdata.TermBuf.TermArray(6).Value = 1000.0
263	274	
264		# break perf_crz_predexec_sep.ada:657
	275	# break perf_crz_predexec_sep.ada:659
265	276	# continue
266	277	# return
267	278	# return
268	279	

Left File: D:\B787_Download\CTP_B787_PERF_CRZINITE\OLD\CTP_B787_PERF_CRZINITE.TDF Right File: D:\B787_Download\CTP_B787_PERF_CRZINITE\NEW\CTP_B787_PERF_CRZINITE.TDF

```

269 280 !end_test()
270 281 --OUTPUTS:
271 282 Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Valid = boolean'(false)
272 283
273 284 -----
274 285 TESTID: 3
275 286 This verify When If the step type is SpecStep, the StepAlt termination value is set to the Specified step
276 287 altitude.If the aircraft is in climb, and a step was moved to the cruise phase from the climb phase the StepAlt
277 288 value is set to the climb specified step altitude, and the StepDist is activated with a distance value indicating
278 289 the step should begin immediately.
279 290
280 291 REQUIREMENTS UNDER EVALUATION: FMCS_19_20006025, FMCS_19_20006026, FMCS_19_20006027, FMCS_19_20006028, FMCS_19_20006029
281 292 FMCS_19_20006097, FMCS_19_20006098, FMCS_19_20006099, FMCS_19_20006102
282 293
283 294 SUPPORTING REQUIREMENTS : FMCS_19_21027005
284 295
285 296
286 297 --INPUTS:
287 298 -- SETLANGMODE = ADA
288 299 Test_Firstpass := boolean'(true)
289 300 Test_Steptype := CFP_PERF_STEP_IFTYPES.specstep
290 301 Test_LGB_Search := 0
291 302 -- Test_ICAO_Low := 6000.0
292 303 -- Test_ICAO_High := 10000.0
293 304
294 305 -----
295 306 -- set variable for enter SUT
296 307 Perf_Preds_Lfdata.VTPlogic.Firstpass := boolean'(true)
297 308 Perf_Crztalt_Lfdata.LastCrzAlt.Valid := boolean'(false)
298 309 Perf_Preds_Lfdata.Vgbptr := CLB2L
299 310
300 311 Perf_RTA_Lfdata.Pred_Pastrta := boolean'(true)
301 312 Perf_WTS_Lfdata.Always_Compute_Max_Speed := boolean'(false)
302 313
303 314 Perf_Integrators_Lfdata.IntProgBuf.Hprog := 27004.0
304 315 Perf_Crztalt_Lfdata.Crztalt := 27000.0
305 316 -----
306 317
307 318 -- Initialize the variable
308 319 perf_Preds_Lfdata.Fltphase := FMCS_Base_Types.climb
309 320 Perf_Preds_Lfdata.NavPtr := 1
310 321 Perf_Preds_Lfdata.PrevNavPtr := 2

```

Left File: D:\B787_Download\CTP_B787_PERF_CRZINITE\OLD\CTP_B787_PERF_CRZINITE.TDF Right File: D:\B787_Download\CTP_B787_PERF_CRZINITE\NEW\CTP_B787_PERF_CRZINITE.TDF

311	322		
312	323	# break Perf_Su_Spd_Utils_Pkg.Su_Frmtgtspdrec	
313	324	# continue	
314	325	# return	
315	326		
316	327	-- enter SUT	
317	328	# break Crz_InitStepTerms	
318	329	# continue	
319	330	#define Loc_Clb_Step_Exec	:= boolean'(false)
320	331	Perf_LGB_Lfdata.LGB(2).Fpln_Data.SpAlt1Val	:= boolean'(false)
321	332	Perf_LGB_Lfdata.LGB(2).Fpln_Data.SpAlt1Pos	:= AC_Position_Types.SC
322	333	Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Stepptr	:= 0
323	334	Perf_Integrators_Lfdata.TermBuf.TermArray(6).Active	:= boolean'(false)
324	335	Perf_Integrators_Lfdata.TermBuf.TermArray(6).Value	:= 1.0
325	336	Perf_Integrators_Lfdata.IntProgBuf.Xprog	:= 1000.0
326	337	Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Valid	:= boolean'(true)
327	338		
328		# break PERF_CRZ_INITSTEPTERMS_SEP.ADA:193	
	339	# break PERF_CRZ_INITSTEPTERMS_SEP.ADA:198	
329	340	# continue	
330	341	Step_Ptr := 99	
331	342		
332		# break PERF_CRZ_INITSTEPTERMS_SEP.ADA:245	
	343	# break PERF_CRZ_INITSTEPTERMS_SEP.ADA:250	
333	344	# continue	
334	345	Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Stepptr	= 0
335	346	Perf_Integrators_Lfdata.TermBuf.TermArray(6).Active	= boolean'(false)
336	347	Perf_Integrators_Lfdata.TermBuf.TermArray(6).Value	= 1.0
337	348		
338		# break perf_crz_predexec_sep.ada:657	
	349	# break perf_crz_predexec_sep.ada:659	
339	350	# continue	
340	351	# return	
341	352	# return	
342	353		
343	354	!end_test()	
344	355	--OUTPUTS:	
345	356	Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Valid	= boolean'(false)
346	357		
347	358	-----	
348	359	TESTID: 4	
349	360	This verify When If the step type is SpecStep, the StepAlt termination value is set to the Specified step	

Left File: D:\B787_Download\CTP_B787_PERF_CRZINITE\OLD\CTP_B787_PERF_CRZINITE.TDF Right File: D:\B787_Download\CTP_B787_PERF_CRZINITE\NEW\CTP_B787_PERF_CRZINITE.TDF

```

350 361 altitude.If the aircraft is not in climb, and the StepDist is invalid.
351 362
352 363 REQUIREMENTS UNDER EVALUATION: FMCS_19_20006025, FMCS_19_20006026, FMCS_19_20006027, FMCS_19_20006028, FMCS_19_20006029
353 364 FMCS_19_20006097, FMCS_19_20006098, FMCS_19_20006099, FMCS_19_20006102
354 365
355 366 SUPPORTING REQUIREMENTS : FMCS_19_21027005
356 367
357 368
358 369 --INPUTS:
359 370 -- SETLANGMODE = ADA
360 371 Test_Firstpass := boolean'(false)
361 372 Test_Steptype := CFP_PERF_STEP_IFTYPES.specstep
362 373 Test_LGB_Search := 0
363 374 -- Test_ICAO_Low := 6000.0
364 375 -- Test_ICAO_High := 10000.0
365 376 -----
366 377 -- set variable for enter SUT
367 378 Perf_Preds_Lfdata.VTPlogic.Firstpass := boolean'(true)
368 379 Perf_Crzalt_Lfdata.LastCrzAlt.Valid := boolean'(false)
369 380 Perf_Preds_Lfdata.Vgbptr := CLB2L
370 381
371 382 Perf_RTA_Lfdata.Pred_Pastrta := boolean'(true)
372 383 Perf_WTS_Lfdata.Always_Compute_Max_Speed := boolean'(false)
373 384
374 385 Perf_Integrators_Lfdata.IntProgBuf.Hprog := 27004.0
375 386 Perf_Crzalt_Lfdata.Crzalt := 27000.0
376 387 -----
377 388
378 389 -- Initialize the variable
379 390 Perf_Preds_Lfdata.Fltphase := FMCS_Base_Types.Descent
380 391 Perf_Preds_Lfdata.NavPtr := 1
381 392 Perf_Preds_Lfdata.PrevNavPtr := 1
382 393
383 394 # break Perf_Su_Spd_Utils_Pkg.Su_Frmtgtspdrec
384 395 # continue
385 396 # return
386 397
387 398 -- enter SUT
388 399 # break Crz_InitStepTerms
389 400 # continue
390 401 #define Loc_Clb_Step_Exec := boolean'(false)
391 402 Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Valid := boolean'(true)

```

Left File: D:\B787_Download\CTP_B787_PERF_CRZINITE\OLD\CTP_B787_PERF_CRZINITE.TDF Right File: D:\B787_Download\CTP_B787_PERF_CRZINITE\NEW\CTP_B787_PERF_CRZINITE.TDF

392	403	
393		# break perf_crz_predexec_sep.ada:657
	404	# break perf_crz_predexec_sep.ada:659
394	405	# continue
395	406	# return
396	407	# return
397	408	
398	409	!end_test()
399	410	--OUTPUTS:
400	411	Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Valid = boolean'(false)
401	412	
402	413	-----
403	414	TESTID: 5
404	415	This verify When If the Step type is PastSpecStep, the StepAlt termination value is set to the current specified step
405	416	altitude, and the StepDist termination is activated with a distance value indicating the step should begin immediately
406	417	unless the step is moved due to maximum altitude restrictions, in which case the StepDist termination is set to the
407	418	estimated step point distance. If the aircraft is in climb, and no previous step point exists, the StepAlt termination
408	419	value is set to the current specified step altitude, and the StepDist termination is activated with a distance value
409	420	indicating the step should begin immediately.
410	421	
411	422	REQUIREMENTS UNDER EVALUATION: FMCS_19_20006025, FMCS_19_20006026, FMCS_19_20006027, FMCS_19_20006028, FMCS_19_20006029
412	423	FMCS_19_20006097, FMCS_19_20006098, FMCS_19_20006099, FMCS_19_20006102, FMCS_19_20006100
413	424	
414	425	SUPPORTING REQUIREMENTS : FMCS_19_21027005
415	426	
416	427	
417	428	--INPUTS:
418	429	-- SETLANGMODE = ADA
419	430	Test_Firstpass := boolean'(false)
420	431	Test_Steptype := CFP_PERF_STEP_IFTYPES.PastSpecStep
421	432	Test_LGB_Search := 0
422	433	-- Test_ICAO_Low := 6000.0
423	434	-- Test_ICAO_High := 10000.0
424	435	-----
425	436	-- set variable for enter SUT
426	437	Perf_Preds_Lfdata.VTPlogic.Firstpass := boolean'(true)
427	438	Perf_Crzalt_Lfdata.LastCrzAlt.Valid := boolean'(false)
428	439	Perf_Preds_Lfdata.Vgbptr := CLB2L
429	440	
430	441	Perf_RTA_Lfdata.Pred_Pastrta := boolean'(true)
431	442	Perf_WTS_Lfdata.Always_Compute_Max_Speed := boolean'(false)
432	443	

Left File: D:\B787_Download\CTP_B787_PERF_CRZINITE\OLD\CTP_B787_PERF_CRZINITE.TDF Right File: D:\B787_Download\CTP_B787_PERF_CRZINITE\NEW\CTP_B787_PERF_CRZINITE.TDF

433	444	Perf_Integrators_Lfdata.IntProgBuf.Hprog	:= 27004.0
434	445	Perf_Crzalt_Lfdata.Crzalt	:= 27000.0
435	446	-----	
436	447		
437	448	-- Initialize the variable	
438	449	Perf_Preds_Lfdata.Fltphase	:= FMCS_Base_Types.climb
439	450	Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Stepptr	:= 0
440	451	Perf_Preds_Lfdata.Navptr	:= 1
441	452	Perf_Preds_Lfdata.PrevNavptr	:= 0
442	453		
443	454	# break Perf_Su_Spd_Utils_Pkg.Su_Frmtgtspdrec	
444	455	# continue	
445	456	# return	
446	457		
447	458	-- enter SUT	
448	459	# break Crz_InitStepTerms	
449	460	# continue	
450	461	Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Valid	:= boolean'(true)
451	462	Perf_Integrators_Lfdata.TermBuf.TermArray(6).Active	:= boolean'(false)
452	463	Perf_Integrators_Lfdata.TermBuf.TermArray(6).Value	:= 0.0
453	464	Perf_Integrators_Lfdata.IntProgBuf.Xprog	:= 1000.0
454	465		
455		# break PERF_CRZ_INITSTEPTERMS_SEP.ADA:193	
	466	# break PERF_CRZ_INITSTEPTERMS_SEP.ADA:198	
456	467	# continue	
457	468	Step_Ptr := 99	
458	469		
459		# break PERF_CRZ_INITSTEPTERMS_SEP.ADA:261	
	470	# break PERF_CRZ_INITSTEPTERMS_SEP.ADA:266	
460	471	# continue	
461	472	Perf_Integrators_Lfdata.TermBuf.TermArray(6).Active	= boolean'(true)
462	473	Perf_Integrators_Lfdata.TermBuf.TermArray(6).Value	= 1000.0
463	474		
464		# break perf_crz_predexec_sep.ada:657	
	475	# break perf_crz_predexec_sep.ada:659	
465	476	# continue	
466	477	# return	
467	478	# return	
468	479		
469	480	!end_test()	
470	481	--OUTPUTS:	
471	482	Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Stepptr	= 1

Left File: D:\B787_Download\CTP_B787_PERF_CRZINITE\OLD\CTP_B787_PERF_CRZINITE.TDF Right File: D:\B787_Download\CTP_B787_PERF_CRZINITE\NEW\CTP_B787_PERF_CRZINITE.TDF

```

472 483 Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Valid = boolean'(false)
473 484 -----
474 485 TESTID: 6
475 486 This verify When If the Step type is PEOpt, a check is made to see if the aircraft is past
476 487 the current step distance to destination.
477 488
478 489 REQUIREMENTS UNDER EVALUATION: FMCS_19_20006025, FMCS_19_20006026, FMCS_19_20006027, FMCS_19_20006028, FMCS_19_20006029
479 489 FMCS_19_20006097, FMCS_19_20006098, FMCS_19_20006099, FMCS_19_20006102
480 490 FMCS_19_20006097, FMCS_19_20006098, FMCS_19_20006099, FMCS_19_20006102, PERF_SRD_B_00413
481 491
482 492 SUPPORTING REQUIREMENTS : FMCS_19_21027005
483 493
484 494
485 495 --INPUTS:
486 496 -- SETLANGMODE = ADA
487 497 Test_Firstpass := boolean'(false)
488 498 Test_Steptype := CFP_PERF_STEP_IFTYPES.PEOpt
489 499 Test_LGB_Search := 0
490 500 -- Test_ICAO_Low := 6000.0
491 501 -- Test_ICAO_High := 10000.0
492 502 -----
493 503 -- set variable for enter SUT
494 504 Perf_Preds_Lfdata.VTPlogic.Firstpass := boolean'(true)
495 505 Perf_Crzalt_Lfdata.LastCrzAlt.Valid := boolean'(false)
496 506 Perf_Preds_Lfdata.Vgbptr := CLB2L
497 507
498 508 Perf_RTA_Lfdata.Pred_Pastrta := boolean'(true)
499 509 Perf_WTS_Lfdata.Always_Compute_Max_Speed := boolean'(false)
500 510
501 511 Perf_Integrators_Lfdata.IntProgBuf.Hprog := 27004.0
502 512 Perf_Crzalt_Lfdata.Crzalt := 27000.0
503 513 -----
504 514
505 515 -- Initialize the variable
506 516 # break Perf_Su_Spd_Utils_Pkg.Su_Frmtgtspdrec
507 517 # continue
508 518 # return
509 519
510 520 -- enter SUT
511 521 # break Crz_InitStepTerms
512 522 # continue
513 523

```

Left File: D:\B787_Download\CTP_B787_PERF_CRZINITE\OLD\CTP_B787_PERF_CRZINITE.TDF Right File: D:\B787_Download\CTP_B787_PERF_CRZINITE\NEW\CTP_B787_PERF_CRZINITE.TDF

513	524	Perf_Integrators_Lfdata.TermBuf.TermArray(6).Active	:= boolean'(false)
514	525	Perf_Integrators_Lfdata.TermBuf.TermArray(6).Value	:= 9999.9
515	526	Perf_Integrators_Lfdata.IntProgBuf.Xprog	:= 10000.0
516		Idx_Profile_Lfdata.Ialtprofptree(0)(1).Dtd.Valid	:= boolean'(true)
517		Idx_Profile_Lfdata.Ialtprofptree(0)(1).Dtd.Data	:= 1000.0
	527	Perf_Crz_Pkg.Opt_Step_PND_Ptr	:= 1
	528	Perf_WTS_Lfdata.Previous_Pass_Info_Rec.Step(1).Distance.Data	:= 1000.0
	529	Perf_WTS_Lfdata.Previous_Pass_Info_Rec.Step(1).Distance.Valid	:= boolean'(true)
518	530	Perf_Integrators_Lfdata.TermBuf.TermArray(11).Value	:= 1.0
519	531		
520	532	Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Valid	:= boolean'(true)
521	533	Perf_Integrators_Lfdata.TermBuf.TermArray(11).Active	:= boolean'(false)
522	534		
523		# break perf_crz_predexec_sep.ada:657	
	535	# break perf_crz_predexec_sep.ada:659	
524	536	# continue	
525	537	# return	
526	538	# return	
527	539		
528	540	!end_test()	
529	541	--OUTPUTS:	
530	542	Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Valid	= boolean'(false)
531	543	Perf_Integrators_Lfdata.TermBuf.TermArray(11).Active	= boolean'(true)
532	544	Perf_Integrators_Lfdata.TermBuf.TermArray(11).Value	= 1000.0
533	545		
534	546	-----	
535	547	TESTID: 7	
536	548	This verify When If the step type is Opt, it is determined whether: both a new step altitude	
537	549	and gross weight need to be computed.	
538	550		
539	551	REQUIREMENTS UNDER EVALUATION: FMCS_19_20006025, FMCS_19_20006026, FMCS_19_20006027, FMCS_19_20006028, FMCS_19_20006029	
540	552	FMCS_19_20006097, FMCS_19_20006098, FMCS_19_20006099, FMCS_19_20006102	
541	553		
542	554	SUPPORTING REQUIREMENTS : FMCS_19_21027005	
543	555		
544	556		
545	557	--INPUTS:	
546	558	-- SETLANGMODE = ADA	
547	559	Test_Firstpass	:= boolean'(false)
548	560	Test_Steptype	:= CFP_PERF_STEP_IFTYPES.Opt
549	561	Test_LGB_Search	:= 0
550	562	-- Test_ICAO_Low	:= 6000.0

Left File: D:\B787_Download\CTP_B787_PERF_CRZINITE\OLD\CTP_B787_PERF_CRZINITE.TDF Right File: D:\B787_Download\CTP_B787_PERF_CRZINITE\NEW\CTP_B787_PERF_CRZINITE.TDF

551	563	-- Test_ICAO_High	:= 10000.0
552	564	-----	
553	565	-- set variable for enter SUT	
554	566	Perf_Preds_Lfdata.VTPlogic.Firstpass	:= boolean'(true)
555	567	Perf_Crzalt_Lfdata.LastCrzAlt.Valid	:= boolean'(false)
556	568	Perf_Preds_Lfdata.Vgbptr	:= CLB2L
557	569		
558	570	Perf_RTA_Lfdata.Pred_Pastrta	:= boolean'(true)
559	571	Perf_WTS_Lfdata.Always_Compute_Max_Speed	:= boolean'(false)
560	572		
561	573	Perf_Integrators_Lfdata.IntProgBuf.Hprog	:= 907.0
562	574	Perf_Crzalt_Lfdata.Crzalt	:= 900.0
563	575	-----	
564	576		
565	577	-- Initialize the variable	
566	578	-- Perf_Crzalt_Lfdata.Crzalt	:= 900.0
567	579		
568	580	# break Perf_Su_Spd_Utils_Pkg.Su_Frmtgtspdrec	
569	581	# continue	
570	582	# return	
571	583		
572	584	-- enter SUT	
573	585	# break Crz_InitStepTerms	
574	586	# continue	
575	587	Perf_Crz_Pkg.Step_Size.Valid	:= boolean'(true)
576	588	Perf_Crz_Pkg.Step_Size.Data	:= 100.0
577	589	Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Stepptr	:= 99
578	590	Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Valid	:= boolean'(true)
579	591	Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Opt_Stepalt	:= 0.0
580	592	Perf_Integrators_Lfdata.TermBuf.TermArray(11).Active	:= boolean'(false)
581	593		
582	594	-- Next_Step_High := 0.0	
583	595	-- Next_Step_Low := 0.0	
584	596		
585		# break PERF_CRZ_INITSTEPTEMS_SEP.ADA:325	
	597	# break PERF_CRZ_INITSTEPTEMS_SEP.ADA:330	
586	598	# continue	
587	599	-- Next_Step_High = 1000.0	
588	600	-- Next_Step_Low = 900.0	
589	601		
590		# break perf_crz_predexec_sep.ada:657	
	602	# break perf_crz_predexec_sep.ada:659	

Left File: D:\B787_Download\CTP_B787_PERF_CRZINITE\OLD\CTP_B787_PERF_CRZINITE.TDF Right File: D:\B787_Download\CTP_B787_PERF_CRZINITE\NEW\CTP_B787_PERF_CI

```

591 603 # continue
592 604 # return
593 605 # return
594 606
595 607 !end_test()
596 608 --OUTPUTS:
597 609 Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Stepptr      = 0
598 610 Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Valid = boolean'(false)
599 611 Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Opt_Stepalt   = 1000.0
600 612 Perf_Integrators_Lfdata.TermBuf.TermArray(11).Active           = boolean'(true)
601 613
602 614 -----
603 615 TESTID: 8
604 616 This verify When If the step type is Opt, it is determined whether: both a new step altitude
605 617 and gross weight need to be computed.
606 618
607 619 REQUIREMENTS UNDER EVALUATION: FMCS_19_20006025, FMCS_19_20006026, FMCS_19_20006027, FMCS_19_20006028, FMCS_19_20006029,
608 620                               FMCS_19_20006097, FMCS_19_20006098, FMCS_19_20006099, FMCS_19_20006102
609 621
610 622 SUPPORTING REQUIREMENTS :      FMCS_19_21027005
611 623
612 624
613 625 --INPUTS:
614 626 -- SETLANGMODE = ADA
615 627 Test_Firstpass                                     := boolean'(false)
616 628 Test_Steptype                                       := CFP_PERF_STEP_IFTYPES.Opt
617 629 Test_LGB_Search                                     := 0
618 630 -- Test_ICAO_Low                                   := 6000.0
619 631 -- Test_ICAO_High                                   := 60000.0
620 632 -----
621 633 -- set variable for enter SUT
622 634 Perf_Preds_Lfdata.VTPlogic.Firstpass               := boolean'(true)
623 635 Perf_Crzalt_Lfdata.LastCrzAlt.Valid                 := boolean'(false)
624 636 Perf_Preds_Lfdata.Vgbptr                            := CLB2L
625 637
626 638 Perf_RTA_Lfdata.Pred_Pastrta                        := boolean'(true)
627 639 Perf_WTS_Lfdata.Always_Compute_Max_Speed           := boolean'(false)
628 640
629 641 Perf_Integrators_Lfdata.IntProgBuf.Hprog            := 80006.0
630 642 Perf_Crzalt_Lfdata.Crzalt                          := 80000.0
631 643 -----
632 644

```

Left File: D:\B787_Download\CTP_B787_PERF_CRZINITE\OLD\CTP_B787_PERF_CRZINITE.TDF Right File: D:\B787_Download\CTP_B787_PERF_CRZINITE\NEW\CTP_B787_PERF_CRZINITE.TDF

633	645	-- Initialize the variable	
634	646	# break Perf_Su_Spd_Utils_Pkg.Su_Frmtgtspdrec	
635	647	# continue	
636	648	# return	
637	649		
638	650	-- enter SUT	
639	651	# break Crz_InitStepTerms	
640	652	# continue	
641	653	Perf_Crz_Pkg.Step_Size.Valid	:= boolean'(false)
642	654	Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Stepptr	:= 99
643	655	Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Valid	:= boolean'(true)
644	656	Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Opt_Stepalt	:= 0.0
645	657	-- Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Steptype	:= CFP_PERF_STEP_IFTYPES.specstep
646	658		
647		# break perf_crz_predexec_sep.ada:657	
	659	# break perf_crz_predexec_sep.ada:659	
648	660	# continue	
649	661	# return	
650	662	# return	
651	663		
652	664	!end_test()	
653	665	--OUTPUTS:	
654	666	Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Stepptr	= 0
655	667	Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Valid	= boolean'(false)
656	668	Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Opt_Stepalt	= 81000.0
657	669	Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Steptype	= CFP_PERF_STEP_IFTYPES.Nostep
658	670		
659	671	-----	
660	672	TESTID: 9	
661	673	This verify When If the Step type is PastSpecStep, the StepAlt termination value is set to	
662	674	the current specified step altitude.	
663	675		
664	676	REQUIREMENTS UNDER EVALUATION: FMCS_19_20006025, FMCS_19_20006026, FMCS_19_20006027, FMCS_19_20006028, FMCS_19_20006029	
665	677	FMCS_19_20006097, FMCS_19_20006098, FMCS_19_20006099, FMCS_19_20006102	
666	678		
667	679	SUPPORTING REQUIREMENTS : FMCS_19_21027005	
668	680		
669	681		
670	682	--INPUTS:	
671	683	-- SETLANGMODE = ADA	
672	684	Test_Firstpass	:= boolean'(false)
673	685	Test_Steptype	:= CFP_PERF_STEP_IFTYPES.PastSpecStep

Left File: D:\B787_Download\CTP_B787_PERF_CRZINITE\OLD\CTP_B787_PERF_CRZINITE.TDF Right File: D:\B787_Download\CTP_B787_PERF_CRZINITE\NEW\CTP_B787_PERF_CRZINITE.TDF

674	686	Test_LGB_Search	:= 0
675	687	-- Test_ICAO_Low	:= 6000.0
676	688	-- Test_ICAO_High	:= 10000.0
677	689	-----	
678	690	-- set variable for enter SUT	
679	691		
680	692	Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Stepptr	:= 0
681	693	Perf_Preds_Lfdata.Navptr	:= 1
682	694	Perf_Preds_Lfdata.Fltphase	:= FMCS_Base_Types.climb
683	695		
684	696	# break Perf_Su_Spd_Utils_Pkg.Su_Frmtgtspdrec	
685	697	# continue	
686	698	# return	
687	699		
688	700	-- enter SUT	
689	701	# break Crz_InitStepTerms	
690	702	# continue	
691	703	Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Valid	:= boolean'(true)
692	704	Perf_Preds_Lfdata.PrevNavPtr	:= 0
693	705	Perf_Integrators_Lfdata.TermBuf.TermArray(6).Active	:= boolean'(false)
694	706	Perf_Integrators_Lfdata.TermBuf.TermArray(6).Value	:= 0.0
695	707	Perf_Integrators_Lfdata.IntProgBuf.Xprog	:= 10000.0
696	708		
697		# break PERF_CRZ_INITSTEPTERMS_SEP.ADA:193	
	709	# break PERF_CRZ_INITSTEPTERMS_SEP.ADA:198	
698	710	# continue	
699	711	#define Loc_Clb_Step_Exec := boolean'(true)	
700	712	Step_Ptr := 99	
701	713		
702		# break PERF_CRZ_INITSTEPTERMS_SEP.ADA:261	
	714	# break PERF_CRZ_INITSTEPTERMS_SEP.ADA:266	
703	715	# continue	
704	716	Perf_Integrators_Lfdata.TermBuf.TermArray(6).Active	= boolean'(true)
705	717	Perf_Integrators_Lfdata.TermBuf.TermArray(6).Value	= 10000.0
706	718		
707		# break perf_crz_predexec_sep.ada:657	
	719	# break perf_crz_predexec_sep.ada:659	
708	720	# continue	
709	721	# return	
710	722	# return	
711	723		
712	724	!end_test()	

Left File: D:\B787_Download\CTP_B787_PERF_CRZINITE\OLD\CTP_B787_PERF_CRZINITE.TDF Right File: D:\B787_Download\CTP_B787_PERF_CRZINITE\NEW\CTP_B787_PERF_CRZINITE.TDF

```

713 725 --OUTPUTS:
714 726 Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Stepptr          = 1
715 727 Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Valid = boolean'(false)
716 728
717 729 -----
718 730 TESTID: 10
719 731 This verify When If the Step type is PastSpecStep, the StepAlt termination value is set to
720 732 the current specified step altitude.
721 733
722 734 REQUIREMENTS UNDER EVALUATION: FMCS_19_20006025, FMCS_19_20006026, FMCS_19_20006027, FMCS_19_20006028, FMCS_19_20006029
723 735 FMCS_19_20006097, FMCS_19_20006098, FMCS_19_20006099, FMCS_19_20006102
724 736
725 737 SUPPORTING REQUIREMENTS : FMCS_19_21027005
726 738
727 739
728 740 --INPUTS:
729 741 -- SETLANGMODE = ADA
730 742 Test_Firstpass := boolean'(false)
731 743 Test_Steptype := CFP_PERF_STEP_IFTYPES.PastSpecStep
732 744 Test_LGB_Search := 0
733 745 -- Test_ICAO_Low := 6000.0
734 746 -- Test_ICAO_High := 10000.0
735 747 -----
736 748 -- set variable for enter SUT
737 749
738 750 Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Stepptr := 0
739 751 Perf_Preds_Lfdata.Navptr := 1
740 752 Perf_Preds_Lfdata.Fltphase := FMCS_Base_Types.Descent
741 753
742 754 # break Perf_Su_Spd_Utils_Pkg.Su_Frmtgtspdrec
743 755 # continue
744 756 # return
745 757
746 758 -- enter SUT
747 759 # break Crz_InitStepTerms
748 760 # continue
749 761 Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Valid := boolean'(true)
750 762 Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Data := 500.0
751 763 Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Fixdistodest := 1000.0
752 764 Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Valid := boolean'(true)
753 765 Perf_Integrators_Lfdata.TermBuf.TermArray(6).Active := boolean'(false)
754 766 Perf_Integrators_Lfdata.TermBuf.TermArray(6).Value := 0.0

```

Left File: D:\B787_Download\CTP_B787_PERF_CRZINITE\OLD\CTP_B787_PERF_CRZINITE.TDF Right File: D:\B787_Download\CTP_B787_PERF_CRZINITE\NEW\CTP_B787_PERF_CRZINITE.TDF

755	767	
756		# break perf_crz_predexec_sep.ada:657
	768	# break perf_crz_predexec_sep.ada:659
757	769	# continue
758	770	# return
759	771	# return
760	772	
761	773	!end_test()
762	774	--OUTPUTS:
763	775	Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Stepptr = 1
764	776	Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Valid = boolean'(false)
765	777	Perf_Integrators_Lfdata.TermBuf.TermArray(6).Active = boolean'(true)
766	778	Perf_Integrators_Lfdata.TermBuf.TermArray(6).Value = 500.0
767	779	
768	780	-----
769	781	TESTID: 11
770	782	This verify When If the Step type is PastSpecStep, the StepAlt termination value is set to
771	783	the current specified step altitude.
772	784	
773	785	REQUIREMENTS UNDER EVALUATION: FMCS_19_20006025, FMCS_19_20006026, FMCS_19_20006027, FMCS_19_20006028, FMCS_19_20006029
774	786	FMCS_19_20006097, FMCS_19_20006098, FMCS_19_20006099, FMCS_19_20006102
775	787	
776	788	SUPPORTING REQUIREMENTS : FMCS_19_21027005
777	789	
778	790	
779	791	--INPUTS:
780	792	-- SETLANGMODE = ADA
781	793	Test_Firstpass := boolean'(false)
782	794	Test_Steptype := CFP_PERF_STEP_IFTYPES.PastSpecStep
783	795	Test_LGB_Search := 0
784	796	-- Test_ICAO_Low := 6000.0
785	797	-- Test_ICAO_High := 10000.0
786	798	-----
787	799	-- set variable for enter SUT
788	800	Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Stepptr := 0
789	801	Perf_Preds_Lfdata.Navptr := 1
790	802	Perf_Preds_Lfdata.Fltphase := FMCS_Base_Types.Descent
791	803	
792	804	# break Perf_Su_Spd_Utils_Pkg.Su_Frmtgtspdrec
793	805	# continue
794	806	# return
795	807	

Left File: D:\B787_Download\CTP_B787_PERF_CRZINITE\OLD\CTP_B787_PERF_CRZINITE.TDF Right File: D:\B787_Download\CTP_B787_PERF_CRZINITE\NEW\CTP_B787_PERF_CRZINITE.TDF

796	808	-- enter SUT	
797	809	# break Crz_InitStepTerms	
798	810	# continue	
799	811	Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Valid := boolean'(false)	
800	812	Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Data := 500.0	
801	813	Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Fixdistodest := 1000.0	
802	814	Perf_Integrators_Lfdata.TermBuf.TermArray(6).Active := boolean'(false)	
803	815	Perf_Integrators_Lfdata.TermBuf.TermArray(6).Value := 0.0	
804	816	Perf_Integrators_Lfdata.IntProgBuf.Xprog := 999.0	
805	817		
806		# break perf_crz_predexec_sep.ada:657	
	818	# break perf_crz_predexec_sep.ada:659	
807	819	# continue	
808	820	# return	
809	821	# return	
810	822		
811	823	!end_test()	
812	824	--OUTPUTS:	
813	825	Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Stepptr = 1	
814	826	Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Valid = boolean'(false)	
815	827	Perf_Integrators_Lfdata.TermBuf.TermArray(6).Active = boolean'(true)	
816	828	Perf_Integrators_Lfdata.TermBuf.TermArray(6).Value = 999.0	
817	829	-----	
818	830	TESTID: 12	
819	831	This verify When If the Step type is PastSpecStep, the StepAlt termination value is set to	
820	832	the current specified step altitude.	
821	833		
822	834	Robust test follows:	
823	835	Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Data >	
824	836	Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Fixdistodest.	
825	837	Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Valid is true.	
826	838		
827	839	REQUIREMENTS UNDER EVALUATION: FMCS_19_20006025, FMCS_19_20006026, FMCS_19_20006027, FMCS_19_20006028, FMCS_19_20006029	
828	840	FMCS_19_20006097, FMCS_19_20006098, FMCS_19_20006099, FMCS_19_20006102, FMCS_19_20006100	
829	841		
830	842	SUPPORTING REQUIREMENTS : FMCS_19_21027005	
831	843		
832	844		
833	845	--INPUTS:	
834	846	-- SETLANGMODE = ADA	
835	847	Test_Firstpass := boolean'(false)	
836	848	Test_Steptype := CFP_PERF_STEP_IFTYPES.PastSpecStep	

Left File: D:\B787_Download\CTP_B787_PERF_CRZINITE\OLD\CTP_B787_PERF_CRZINITE.TDF Right File: D:\B787_Download\CTP_B787_PERF_CRZINITE\NEW\CTP_B787_PERF_CRZINITE.TDF

837	849	Test_LGB_Search	:= 0
838	850	-- Test_ICAO_Low	:= 6000.0
839	851	-- Test_ICAO_High	:= 10000.0
840	852	-----	
841	853	-- set variable for enter SUT	
842	854	Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Stepptr	:= 0
843	855	Perf_Preds_Lfdata.Navptr	:= 1
844	856	Perf_Preds_Lfdata.Fltphase	:= FMCS_Base_Types.Descent
845	857	# break Perf_Su_Spd_Utils_Pkg.Su_Frmtgtspdrec	
846	858	# continue	
847	859	# return	
848	860		
849	861	-- enter SUT	
850	862	# break Crz_InitStepTerms	
851	863	# continue	
852	864	Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Valid	:= boolean'(true)
853	865	Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Data	:= 1000.0
854	866	Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Fixdistodest	:= 500.0
855	867	Perf_Integrators_Lfdata.TermBuf.TermArray(6).Active	:= boolean'(false)
856	868	Perf_Integrators_Lfdata.TermBuf.TermArray(6).Value	:= 0.0
857	869	Perf_Integrators_Lfdata.IntProgBuf.Xprog	:= 999.0
858	870		
859		# break perf_crz_predexec_sep.ada:657	
	871	# break perf_crz_predexec_sep.ada:659	
860	872	# continue	
861	873	# return	
862	874	# return	
863	875		
864	876	!end_test()	
865	877	--OUTPUTS:	
866	878	Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Stepptr	= 1
867	879	Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Valid	= boolean'(false)
868	880	Perf_Integrators_Lfdata.TermBuf.TermArray(6).Active	= boolean'(true)
869	881	Perf_Integrators_Lfdata.TermBuf.TermArray(6).Value	= 999.0
870	882	-----	

Mode: All Lines

Left base folder: D:\B787_Download\CTP_B787_PERF_CRZINITE\OLD

Right base folder: D:\B787_Download\CTP_B787_PERF_CRZINITE\NEW

File: CTP_B787_PERF_CRZINITE.ZIP\CTP_B787_PERF_CRZINITE.adb

```

1      1 with CTP_B787_PERF_CRZINITE_Drv;
2      2 use CTP_B787_PERF_CRZINITE_Drv;
3      3
4      4 with Scoe_Amio_Enable_Itf;
5      5 with Scoe_Iolib_Api;
6      6 use Scoe_Iolib_Api;
7      7
8      8 with Gnat.IO; use Gnat.IO;
9      9
10     10 procedure CTP_B787_PERF_CRZINITE is
11     11
12     12 begin
13     13
14     14     If Scoe_Amio_Enable_Itf.Scoe_Amio_Enable /= Scoe_Iolib_Api.Scoe_Status_Ok then
15     15         Gnat.IO.put_line("$$$$");
16     16     End if;
17     17
18     18     Gnat.IO.Put_line ("Entry point for PDB!");
19     19
20     20     loop
21     21         CTP_B787_PERF_CRZINITE_D;
22     22     end loop;
23     23
24     24 end CTP_B787_PERF_CRZINITE;

```

File: CTP_B787_PERF_CRZINITE.ZIP\CTP_B787_PERF_CRZINITE.bat

```

1      1 @echo off
2      2 REM
3      3 REM /*+
4      4 REM *****
5      5 REM *
6      6 REM *          HONEYWELL PROPRIETARY, CONFIDENTIAL, AND/OR TRADE SECRET
7      7 REM *          Copyright (c) 2007 Honeywell International, Inc.
8      8 REM *          Unpublished Work -- All Rights Reserved
9      9 REM *
10     10 REM *    NAME:    CTP_B787_PERF_CRZINITE.BAT
11     11 REM *
12     12 REM *    PURPOSE:  This is a B787 CTP batch that is used to build and execute a B787 CTP test.

```

File: CTP_B787_PERF_CRZINITE.ZIP\CTP_B787_PERF_CRZINITE.bat (continued)

```
13 13 REM *
14 14 REM *   COMMAND LINE:   CTP_B787_PERF_CRZINITE.BAT
15 15 REM *
16 16 REM *   INPUT:   None
17 17 REM *
18 18 REM *   OUTPUT:
19 19 REM *       The B787 CTP test will be built and executed with no human interaction.
20 20 REM *       All results files will be stored in appropriate directories.
21 21 REM *
22 22 REM *   INSTRUCTIONS:
23 23 REM *       There are three items that you need to change in this B787 CTP batch template file.
24 24 REM *       They are:
25 25 REM *
26 26 REM *       1) Make a copy of the B787 CTP batch template file and rename it for the real CTP.
27 27 REM *       Substitute the "TEMPLATE" in this file name with the name of the real CTP.
28 28 REM *
29 29 REM *       For example: COPY CTP_B787_TEMPLATE.BAT CTP_B787_PERF_CRZINITE.BAT /V
30 30 REM *
31 31 REM *       2) Edit the new B787 CTP batch file and just before the ":step_1" line, replace the
32 32 REM *       string to change "CTP_B787_XXXX" to use the name of the real CTP.
33 33 REM *
34 34 REM *       For example: Change from: SET CTP_INPUT_NAME=CTP_B787_XXXX
35 35 REM *       Change to:   SET CTP_INPUT_NAME=CTP_B787_PERF_CRZINITE
36 36 REM *
37 37 REM *       Set the string INPUT_LANG to C or A depending on the language being used for CTP
38 38 REM *
39 39 REM *       For example: If the CTP is in ADA and C or only ADA
40 40 REM *       Change from: SET INPUT_LANG=C/A
41 41 REM *       Change to:   SET INPUT_LANG=A
42 42 REM *
43 43 REM *
44 44 REM *       If the CTP is in ADA and C++
45 45 REM *       Change from: SET INPUT_LANG=C/A
46 46 REM *       Change to:   SET INPUT_LANG=C
47 47 REM *
48 48 REM *       3) In Step 3 of this new B787 CTP batch file, follow the instructions there
49 49 REM *       for all C source code files that are a part of this test.
50 50 REM *
51 51 REM *       4) In Step 4 of this new B787 CTP batch file, follow the instructions there
52 52 REM *       for all CPP source code files that are a part of this test.
53 53 REM *
54 54 REM *       5) In Step 5 of this new B787 CTP batch file, follow the instructions there
```

File: CTP_B787_PERF_CRZINITE.ZIP\CTP_B787_PERF_CRZINITE.bat (continued)

```
55 55 REM *           for all ADA source code files that are a part of this test.
56 56 REM *
57 57 REM *   NOTES:
58 58 REM *           Do not make any other changes to this B787 CTP batch file other than those
59 59 REM *           instructions shown above.
60 60 REM *
61 61 REM *   HISTORY:
62 62 REM *           Feb. 2007  Rev. 1.0  Jayalakshmi Kadiwal  Author
63 63 REM *
64 64 REM *****
65 65 REM -*/
66 66 REM
67 67 ECHO.
68 68 ECHO CTP_B787_PERF_CRZINITE.BAT, Version Number 1.0
69 69 ECHO Copyright (c) 2007 Honeywell International, Inc. All rights reserved.
70 70 ECHO Batch file to build and execute a B787 CTP test.
71 71 ECHO.
72 72 REM
73 73 ECHO.
74 74 REM
75 75 SET CTP_INPUT_NAME=CTP_B787_PERF_CRZINITE
76 76 SET INPUT_LANG=C/A
77 77 REM
78 78 SET CTP_START_DATE=%DATE%
79 79 SET CTP_START_TIME=%TIME%
80 80 REM
81 81 :step_1
82 82 REM
83 83 ECHO.
84 84 ECHO *****
85 85 ECHO * CTP_B787_PERF_CRZINITE.BAT-I-Step 1 of 11 *
86 86 ECHO * Check the tool installations for Tornado and GPS. *
87 87 ECHO *****
88 88 ECHO.
89 89 REM
90 90 %chk_init% %CTP_INPUT_NAME%
91 91 if %temp_err% EQU %BATCH_ERROR% goto exit_fail
92 92 :step_2
93 93 REM
94 94 ECHO.
95 95 ECHO *****
96 96 ECHO * CTP_B787_PERF_CRZINITE.BAT-I-Step 2 of 11 *
```

File: CTP_B787_PERF_CRZINITE.ZIP\CTP_B787_PERF_CRZINITE.bat (continued)

```
97      97 ECHO * Generate Makefile in Test_Partition *
98      98 ECHO * and Netboot directories *
99      99 ECHO *****
100     100 ECHO.
101     101 REM
102     102 %make_file% %CTP_INPUT_NAME% %INPUT_LANG%
103     103 REM
104     104 :step_3
105     105 REM
106     106 ECHO.
107     107 ECHO *****
108     108 ECHO * CTP_B787_PERF_CRZINITE.BAT-I-Step 3 of 11 *
109     109 ECHO * Compile all C files (if any). *
110     110 ECHO *****
111     111 ECHO.
112     112 REM
113     113 REM *****
114     114 REM * If there are no C files for this CTP test, then comment out the "make_c" line below. *
115     115 REM * *
116     116 REM * If there is only one C file for this CTP test, then edit the "make_c" line below to *
117     117 REM *     replace the "YYYY" with the B787 CTP GPR file name and *
118     118 REM *     replace the "ZZZZ" with the name of the C source code file to be compiled. *
119     119 REM * *
120     120 REM * If there is more than one C file for this CTP test, then make copies of the "make_c" line, *
121     121 REM *     below and edit each line as follows: *
122     122 REM *     replace the "YYYY" with the B787 CTP GPR file name and *
123     123 REM *     replace the "ZZZZ" with the name of the C source code file to be compiled. *
124     124 REM * *
125     125 REM * Notes: *
126     126 REM * 1) If the GPR file does not follow the B787 CTP file name conventions, you will need to *
127     127 REM *     rename the GPR file to match B787 CTP file name conventions with "GPR" as the file type. *
128     128 REM * 2) If the GPR file contains hardcoded pathname references, these will have to be edited to *
129     129 REM *     follow the guidelines contained in the README file for this B787 CTP batch file. *
130     130 REM * 3) Include the line "if EXIST..." and "md..." only if the CTP requires creation of "obj_c" *
131     131 REM *     folder. Else comment these 2 lines using "REM" command *
132     132 REM *****
133     133 REM
134     134 REM
135     135 if EXIST %BUILD_BASE%\CFG\%CTP_INPUT_NAME%\obj_c goto sub_step_3
136     136 md %BUILD_BASE%\CFG\%CTP_INPUT_NAME%\obj_c
137     137 goto sub_step_3
138     138
```

File: CTP_B787_PERF_CRZINITE.ZIP\CTP_B787_PERF_CRZINITE.bat (continued)

```
139 139 :sub_step_3
140 140 REM %make_c% CTP_B787_PERF_CRZINITE_c.gpr CTP_B787_PERF_CRZINITE_drv_c.C
141 141 REM
142 142 :step_4
143 143 REM
144 144 ECHO.
145 145 ECHO *****
146 146 ECHO * CTP_B787_PERF_CRZINITE.BAT-I-Step 4 of 11 *
147 147 ECHO * Compile all CPP files (if any). *
148 148 ECHO *****
149 149 ECHO.
150 150 REM
151 151 REM *****
152 152 REM * If there are no CPP files for this CTP test, then comment out the "make_cpp" line below. *
153 153 REM * *
154 154 REM * If there is only one CPP file for this CTP test, then edit the "make_cpp" line below to *
155 155 REM * replace the "YYYY" with the B787 CTP GPR file name and *
156 156 REM * replace the "ZZZZ" with the name of the CPP source code file to be compiled. *
157 157 REM * *
158 158 REM * If there is more than one CPP file for this CTP test, then make copies of the "make_cpp" *
159 159 REM * line below and edit each line as follows: *
160 160 REM * replace the "YYYY" with the B787 CTP GPR file name and *
161 161 REM * replace the "ZZZZ" with the name of the CPP source code file to be compiled. *
162 162 REM * *
163 163 REM * Notes: *
164 164 REM * 1) If the GPR file does not follow the B787 CTP file name conventions, you will need to *
165 165 REM * rename the GPR file to match B787 CTP file name conventions with "GPR" as the file type. *
166 166 REM * 2) If the GPR file contains hardcoded pathname references, these will have to be edited to *
167 167 REM * follow the guidelines contained in the README file for this B787 CTP batch file. *
168 168 REM *****
169 169 REM %make_cpp% CTP_B787_YYYY_cpp.gpr CTP_B787_ZZZZ.CC
170 170 REM
171 171 :step_5
172 172 REM
173 173 ECHO.
174 174 ECHO *****
175 175 ECHO * CTP_B787_PERF_CRZINITE.BAT-I-Step 5 of 11 *
176 176 ECHO * Compile all ADA files. *
177 177 ECHO * Link the test executable. *
178 178 ECHO *****
179 179 ECHO.
180 180 REM
```

File: CTP_B787_PERF_CRZINITE.ZIP\CTP_B787_PERF_CRZINITE.bat (continued)

```

181 181 REM *****
182 182 REM * Notes: *
183 183 REM * 1) Replace the "YYYY" with the B787 CTP GPR file name *
184 184 REM * 2) If the GPR file does not follow the B787 CTP file name conventions, you will need to *
185 185 REM * rename the GPR file to match B787 CTP file name conventions with "GPR" as the file type. *
186 186 REM * 3) If the GPR file contains hardcoded pathname references, these will have to be edited to *
187 187 REM * follow the guidelines contained in the README file for this B787 CTP batch file. *
188 188 REM *****
189 189 REM
190 190 REM %make_ada% CTP_B787_PERF_CRZINITE_ada.gpr
191 191 %make_stub% stubs.gpr PERF_LGB_PKG.ADA
192 192 %make_ada% CTP_B787_PERF_CRZINITE_ada.gpr stub
193 193 REM
194 194 :step_6
195 195 REM
196 196 ECHO.
197 197 ECHO *****
198 198 ECHO * CTP_B787_PERF_CRZINITE.BAT-I-Step 6 of 11 *
199 199 ECHO * Generate the "boot.txt" file. *
200 200 ECHO *****
201 201 ECHO.
202 202 REM
203 203 REM
204 204 REM *****
205 205 REM * Notes *
206 206 REM * Remove "REM" to set FMS_NAV_DB=HEF0509007.SM /set FMS_NAV_DB=HEF0509008.SM for respective Databases *
207 207 REM * By default it is as set FMS_NAV_DB=NO_DB_CHANGE. *
208 208 REM *****
209 209 REM
210 210 set FMS_NAV_DB=HEF0509031.SM
210 210 set FMS_NAV_DB=BEF1404500.SM
211 211 REM set FMS_NAV_DB=HEF0509008.SM
212 212 REM
213 213 REM
214 214 %copy_DB% %FMS_NAV_DB%
215 215 REM
216 216 %gen_boot% %CTP_INPUT_NAME%
217 217 if %temp_err% EQU %BATCH_ERROR% goto exit_fail
218 218 REM
219 219 :step_7
220 220 REM
221 221 ECHO.

```

File: CTP_B787_PERF_CRZINITE.ZIP\CTP_B787_PERF_CRZINITE.bat (continued)

```
222 222 ECHO *****
223 223 ECHO * CTP_B787_PERF_CRZINITE.BAT-I-Step 7 of 11 *
224 224 ECHO * Invoke the ftp server, simics and target server. *
225 225 ECHO *****
226 226 ECHO.
227 227 REM
228 228 %run_tools%
229 229 if %temp_err% EQU %BATCH_ERROR% goto exit_fail
230 230 REM
231 231 :step_8
232 232 REM
233 233 ECHO.
234 234 ECHO *****
235 235 ECHO * CTP_B787_PERF_CRZINITE.BAT-I-Step 8 of 11 *
236 236 ECHO * Run the Test Generation System (TGS) tool. *
237 237 ECHO * CTPs with multiple TDF should comment out this step-8 *
238 238 ECHO *****
239 239 ECHO.
240 240 REM
241 241 %runtgs% %CTP_INPUT_NAME%
242 242 if %temp_err% EQU %BATCH_ERROR% goto exit_fail
243 243 REM
244 244 REM Copy the TGS results files to the B787 "save" directory.
245 245 REM
246 246 if exist %BUILD_BASE%\CFG\%CTP_INPUT_NAME%\%CTP_INPUT_NAME%.RES COPY %BUILD_BASE%\CFG\%CTP_INPUT_NAME%\%CTP_INPUT_NAME%.RES
    » %CTP_BATCH_DIR%\TGS_Results\ . /v
247 247 if exist %BUILD_BASE%\CFG\%CTP_INPUT_NAME%\%CTP_INPUT_NAME%.VER COPY %BUILD_BASE%\CFG\%CTP_INPUT_NAME%\%CTP_INPUT_NAME%.VER
    » %CTP_BATCH_DIR%\TGS_Results\ . /v
248 248 REM
249 249
250 250
251 251 :step_9
252 252 REM
253 253 ECHO.
254 254 ECHO *****
255 255 ECHO * CTP_B787_PERF_CRZINITE.BAT-I-Step 9 of 11 *
256 256 ECHO * Run the Test Generation System (TGS) tool. *
257 257 ECHO * ONLY CTPs with multiple TDFS should include this step *
258 258 ECHO * If there is more than 1 TDF, copy the %runtgs_multiple% *
259 259 ECHO * line below and replace "DDDD" with the TDF name *
260 260 ECHO *****
261 261 ECHO.
```

File: CTP_B787_PERF_CRZINITE.ZIP\CTP_B787_PERF_CRZINITE.bat (continued)

```
262 262 REM
263 263 REM %runtgs_multiple% %CTP_INPUT_NAME% CTP_B787_DDDDn
264 264
265 265 :step_10
266 266 REM
267 267 ECHO.
268 268 ECHO *****
269 269 ECHO * CTP_B787_PERF_CRZINITE.BAT-I-Step 10 of 11 *
270 270 ECHO * Run the Test Coverage Analyzer (TCA) tool. *
271 271 ECHO * CTPs with multiple TDF should comment out this step-10 *
272 272 ECHO *****
273 273 ECHO.
274 274 REM
275 275 %runtca% %CTP_INPUT_NAME%
276 276 if %temp_err% EQU %BATCH_ERROR% goto exit_fail
277 277 REM
278 278 REM Copy the TCA results files to the B787 "save" directory.
279 279 if exist %BUILD_BASE%\Output\Netboot\wrSbc750gx_scoe\%PARTITION_NAME%.PTH COPY
    » %BUILD_BASE%\Output\Netboot\wrSbc750gx_scoe\%PARTITION_NAME%.PTH %CTP_BATCH_DIR%\TCA_Results\%CTP_INPUT_NAME%.PTH /v
280 280 if exist %BUILD_BASE%\Output\Netboot\wrSbc750gx_scoe\%PARTITION_NAME%.XIN COPY
    » %BUILD_BASE%\Output\Netboot\wrSbc750gx_scoe\%PARTITION_NAME%.XIN %CTP_BATCH_DIR%\TCA_Results\%CTP_INPUT_NAME%.XIN /v
281 281 if exist %BUILD_BASE%\Output\Netboot\wrSbc750gx_scoe\%CTP_INPUT_NAME%.RPT COPY
    » %BUILD_BASE%\Output\Netboot\wrSbc750gx_scoe\%CTP_INPUT_NAME%.RPT %CTP_BATCH_DIR%\TCA_Results\ . /v
282 282 REM
283 283 REM
284 284 :step_11
285 285 REM
286 286 ECHO.
287 287 ECHO *****
288 288 ECHO * CTP_B787_PERF_CRZINITE.BAT-I-Step 11 of 11 *
289 289 ECHO * Exit. *
290 290 ECHO *****
291 291 ECHO.
292 292 REM
293 293 :exit_success
294 294 Set temp_err=0
295 295 ECHO CTP_B787_PERF_CRZINITE.BAT-S-NORMAL, Normal termination.
296 296 ECHO.
297 297 goto exit
298 298 REM
299 299 :exit_fail
300 300 Set temp_err=1
```


File: CTP_B787_PERF_CRZINITE.ZIP\CTP_B787_PERF_CRZINITE.bat (continued)

301	301	ECHO CTP_B787_PERF_CRZINITE.BAT-F-ABNORMAL, Abnormal termination.
302	302	ECHO.
303	303	:exit
304	304	ECHO CTP start date/time = %CTP_START_DATE% %CTP_START_TIME%
305	305	ECHO CTP end date/time = %DATE% %TIME%
306	306	ECHO .

File: CTP_B787_PERF_CRZINITE.ZIP\CTP_B787_PERF_CRZINITE.CUL

1	1	##
2	2	## CUL FILE
3	3	##
4	4	## CTP_B787_PERF_CRZINITE.CUL
5	5	##
6	6	PERF_CRZ_INITSTEPTEMS_SEP.ADA.PERF_CRZ_PKG.CRZ_INITSTEPTEMS
7	7	PERF_CRZ_INITSTEPTEMS_SEP.ADA.PERF_CRZ_PKG.CRZ_INITSTEPTEMS.LOC_CLB_STEP

File: CTP_B787_PERF_CRZINITE.ZIP\CTP_B787_PERF_CRZINITE.DSP

	1	#####
	2	## DSP Generator Tool Version 1.0
	3	#####
1	4	##
2		## DSP File
	5	## CTP_B787_PERF_CRZINITE.DSP
	6	##
	7	## NOTE:
	8	## A. "Any" SCR that is mentioned in this DSP file must contain the prefix "SCR_disposed#: "
	9	## B. Template of this DSP file is created by tool and it should not be modified/deleted.
	10	## C. If any information is not applicable then mark the corresponding field as N/A instead of deleting it.
	11	## D. If more than one SCR has to be used for one issue, make separate entry. SCRs should not be captured
	12	## in the same line using comma or any other separators.
3	13	##
4		## CTP_B787_PERF_CRZINITE.DSP
5	14	##
6	15	
7		1. REASON FOR FAILURES OF TEST CASES
	16	-----
	17	1. REASON_FOR_FAILURES_OF_TEST_CASE(S):
	18	## The below mentioned group of lines need to be repeated for each Test case ID, which is having test failures in it.
	19	-----
8	20	
9		N/A
10		

File: CTP_B787_PERF_CRZINITE.ZIP\CTP_B787_PERF_CRZINITE.DSP (continued)

11		
	21	Test_case_Id: N/A
	22	#_of_Failures: N/A
	23	Failed_Requirements: N/A
	24	SCR_disposed#: N/A
	25	SCR_PROJECT: N/A
	26	SCR_SUB_PROJECT: N/A
	27	Disposition: N/A
12	28	
13		2. REASON FOR NOT GETTING 100% COVERAGE
	29	-----
	30	2. COVERAGE_PROBLEM(S):
	31	## Standard excuse and SCR related details need to be mentioned for each and every sub unit separately.
	32	-----
	33	Compilation_Unit_Name: PERF_CRZ_INITSTEPТЕРMS_SEP.ADA.PERF_CRZ_PKG.CRZ_INITSTEPТЕРMS
	34	Uncovered_Code:
14	35	
15		----- N/A
16		
17		-----
	36	
	37	TCH(Test_Coverage_Hole)_Excuse: N/A
	38	N/A
	39	SCR_disposed#: N/A
	40	SCR_PROJECT: N/A
	41	SCR_SUB_PROJECT: N/A
18	42	
19		3. ANY OTHER ISSUES
	43	-----
	44	3. ANY_OTHER_ISSUE(S):
	45	## A. Every entry in Any_Other_Issue should be followed by a SCR_number, its corresponding CM 21 project and subproject.
	46	## B. If SCR is not applicable then mention N/A.
	47	## C. If more than one SCR has to be used for one issue, make separate entry. SCRs should not be captured
	48	## in the same line using comma or any other separators.
	49	-----
20	50	
21		<a> FMCS_19_20006100, FMCS_19_20006025, FMCS_19_20006027, FMCS_19_20006028 and FMCS_19_20006102 are partly tested here,
22		they were also tested in CTP_B787_PERF_CRZPRCSTEP.
23		 FMCS_19_20006026 is partly tested here, it was also tested in CTP_B787_PERF_CRZPRCSTEP and CTP_B787_PERF_CRZPRCTERM.
24		<c> FMCS_19_20006096 was tested in SLTP_B787_04_01_03.
25		<d> FMCS_19_20006097, FMCS_19_20006098, FMCS_19_20006099 and FMCS_19_20006029 are partly tested here,
26		they were also tested in CTP_B787_PERF_CRZPRDEXE and CTP_B787_PERF_CRZPRCSTEP.

File: CTP_B787_PERF_CRZINITE.ZIP\CTP_B787_PERF_CRZINITE.DSP (continued)

51	<1> FMCS_19_20006100, FMCS_19_20006025, FMCS_19_20006027, FMCS_19_20006028 and FMCS_19_20006102 are partly tested here,
52	they were also tested in CTP_B787_PERF_CRZPRCSTEP.
53	SCR_disposed#: N/A
54	SCR_PROJECT: N/A
55	SCR_SUB_PROJECT: N/A
56	
57	<2> FMCS_19_20006026 is partly tested here, it was also tested in CTP_B787_PERF_CRZPRCSTEP and CTP_B787_PERF_CRZPRCTERM.
58	SCR_disposed#: N/A
59	SCR_PROJECT: N/A
60	SCR_SUB_PROJECT: N/A
61	
62	<3> FMCS_19_20006096 was tested in SLTP_B787_04_01_03.
63	SCR_disposed#: N/A
64	SCR_PROJECT: N/A
65	SCR_SUB_PROJECT: N/A
66	
67	<4> FMCS_19_20006097, FMCS_19_20006098, FMCS_19_20006099 and FMCS_19_20006029 are partly tested here,
68	they were also tested in CTP_B787_PERF_CRZPRDEXE and CTP_B787_PERF_CRZPRCSTEP.
69	SCR_disposed#: N/A
70	SCR_PROJECT: N/A
71	SCR_SUB_PROJECT: N/A
27	72
28	
73	-----
74	4. SPECIAL_EXECUTION_INSTRUCTION(S):
75	## Capture all additional information and/or supporting file(s) required for this CTP execution.
76	## For example:
77	## (i) "nav_db23.o" is required for execution.
78	## (ii) "apex_traps.o"/gen=xx and "common file"/gen=xx are required for execution.
79	## Database_Details:
80	## 1. <Enter the database name>
81	-----
82	
83	N/A
84	
85	Database_Details:
86	1. BEF1404500.SM
87	
88	***** End of Report *****

File: CTP_B787_PERF_CRZINITE.ZIP\CTP_B787_PERF_CRZINITE.RPT

1	1	#####
---	---	-------

File: CTP_B787_PERF_CRZINITE.ZIP\CTP_B787_PERF_CRZINITE.RPT (continued)

2	2	#	#
3	3	#	Test Coverage Analyzer
4	4	#	Short Summary Coverage Report
5	5	#	#
6	6	#####	#####
7	7		
8		Mon Mar 25 09:56:03 China Standard Time 2013	
	8	Wed Jul 02 13:02:59 China Standard Time 2014	
9	9		
10	10	Test Coverage Analyzer (TCA) V6.7 CLASS A ps4082880-115	
11		Win32 Host: WinNT 5.1 Build 2600 UserID: E527970 Node: CH71DT56F653X (Intel PentPro Model 23 Step 10)	
12		Current Dir: C:\B787\Builds\ACMBLD_070_SBC\OUTPUT\NETBOOT\wrSbc750gx_scoe	
	11	Win32 Host: WinNT 6.1 Build 7601 UserID: E803143 Node: CH71DT517T0W1 (Intel PentPro Model 58 Step 9)	
	12	Current Dir: C:\B787\Builds\SBC2415_93C\Output\Netboot\wrSbc750gx_scoe	
13	13		
14	14	-----	
15		TCA invoked Mon Mar 25 09:55:46 China Standard Time 2013 with command line:	
	15	TCA invoked Wed Jul 02 13:02:40 China Standard Time 2014 with command line:	
16	16	tca.exe -TABS -s -X -V -v configRecord.xml -p CTP_B787_PERF_CRZINITE.pth ...	
17	17	-x CTP_B787_PERF_CRZINITE.xin -r CTP_B787_PERF_CRZINITE.RPT -c ...	
18		C:\B787\Builds\ACMBLD_070_SBC\CFG\CTP_B787_PERF_CRZINITE\CTP_B787_PERF_CRZINITE.CUL ...	
	18	C:\B787\Builds\SBC2415_93C\CFG\CTP_B787_PERF_CRZINITE\CTP_B787_PERF_CRZINITE.CUL ...	
19	19	-type 3 --ignore=I,C,A,K	
20	20	-----	
21	21	Expanded command line:	
22	22	tca.exe -TABS -s -X -V -v configRecord.xml -p CTP_B787_PERF_CRZINITE.pth ...	
23	23	-x CTP_B787_PERF_CRZINITE.xin -r CTP_B787_PERF_CRZINITE.RPT -c ...	
24		C:\B787\Builds\ACMBLD_070_SBC\CFG\CTP_B787_PERF_CRZINITE\CTP_B787_PERF_CRZINITE.CUL ...	
	24	C:\B787\Builds\SBC2415_93C\CFG\CTP_B787_PERF_CRZINITE\CTP_B787_PERF_CRZINITE.CUL ...	
25	25	-type 3 --ignore=I,C,A,K	
26	26	-----	
27	27		
28	28		
29	29	Test Coverage Type: 3	
30	30		
31	31	Report File Name : CTP_B787_PERF_CRZINITE.RPT	
32	32		
33	33	Paths file(s) :	
34	34		
35		(P01) CTP_B787_PERF_CRZINITE.pth Mon Mar 25 09:42:35 2013	
	35	(P01) CTP_B787_PERF_CRZINITE.pth Wed Jul 02 12:50:20 2014	
36	36	ELFPOPP, Version v1.5, ps4090055-106	

File: CTP_B787_PERF_CRZINITE.ZIP\CTP_B787_PERF_CRZINITE.RPT (continued)

77		C:\B787\Builds\ACMBLD_070_SBC\OUTPUT\NETBOOT\wrSbc750gx_scoe
	77	C:\B787\Builds\SBC2415_93C\Output\Netboot\wrSbc750gx_scoe
78	78	
79	79	Paths file(s) :
80	80	
81		(P01) CTP_B787_PERF_CRZINITE.pth Mon Mar 25 09:42:35 2013
	81	(P01) CTP_B787_PERF_CRZINITE.pth Wed Jul 02 12:50:20 2014
82	82	ELFPOPP, Version v1.5, ps4090055-106
83	83	
84	84	XInfo file(s) Test Date Test Platform:
85	85	
86	86	(P01) CTP_B787_PERF_CRZINITE.pth
87		(X01) CTP_B787_PERF_CRZINITE.xin Mon Mar 25 09:46:20 2013 Simics PowerPC TCA XInfo, Platform V3.00.09
	87	(X01) CTP_B787_PERF_CRZINITE.xin Wed Jul 02 12:53:59 2014 Simics PowerPC TCA XInfo, Platform V3.00.09
88	88	
89	89	Source file(s) :
90	90	
91		C:\B787\Builds\ACMBLD_070_SBC\SRC\FM\perf_crz_initstepterms_sep.ada
	91	C:\B787\Builds\SBC2415_93C\SRC\fm\perf_crz_initstepterms_sep.ada
92	92	
93	93	Total Coverage statistics :
94	94	
95	95	TYPE 3, 100.0%
96	96	
97	97	
98	98	*****
99	99	Source Report Legend Key
100	100	(Legend Key may be suppressed by -k option)
101	101	
102	102	Coverage messages preceding source code lines are annotated with
103	103	object code block tags of the form [x-y BLOCKTYPE]. For example,
104	104	[263-17 JMPT] is a block tag for the 17th block of the 263rd unit
105	105	in the pathsfile and is a jump true block.
106	106	This block tag annotation is intended to be used as a reference to
107	107	the object code level block report (.tcb) generated with the -B option.
108	108	Each object code block is labeled with a unique block tag.
109	109	
110	110	Each line of source code may be prefixed by one of the following
111	111	indicators:
112	112	. = source line completely or partially executed
113	113	* = source line shown ONLY to clarify previous source lines and
114	114	is NOT actually part of the uncovered source TCA is reporting on

File: CTP_B787_PERF_CRZINITE.ZIP\CTP_B787_PERF_CRZINITE.RPT (continued)

115	115	Note that no prefix indicates source line was not executed
116	116	
117	117	
118	118	*****
119	119	
120	120	Compilation Unit / Source file :
121	121	
122	122	PERF_CRZ_INITSTEPTEMS_SEP.ADA.PERF_CRZ_PKG.CRZ_INITSTEPTEMS.LOC_CL -
123	123	B_STEP
124		C:\B787\Builds\ACMBLD_070_SBC\src\fm\perf_crz_initstepterms_sep.ada
	124	C:\B787\Builds\SBC2415_93C\src\fm\perf_crz_initstepterms_sep.ada
125	125	
126	126	Coverage statistics :
127	127	
128	128	TYPE 3, 100.0%
129	129	
130	130	Executed Total
131	131	Decision Paths 8 8
132	132	Condition Paths n/a n/a
133	133	Statements 16 16
134	134	Blocks 11 11
135	135	
136	136	
137	137	
138	138	*****
139	139	
140	140	Compilation Unit / Source file :
141	141	
142	142	PERF_CRZ_INITSTEPTEMS_SEP.ADA.PERF_CRZ_PKG.CRZ_INITSTEPTEMS
143		C:\B787\Builds\ACMBLD_070_SBC\src\fm\perf_crz_initstepterms_sep.ada
	143	C:\B787\Builds\SBC2415_93C\src\fm\perf_crz_initstepterms_sep.ada
144	144	
145	145	Coverage statistics :
146	146	
147	147	TYPE 3, 100.0%
148	148	
149	149	Executed Total
150	150	Decision Paths 16 16
151	151	Condition Paths n/a n/a
152	152	Statements 32 32
153	153	Blocks 32 32
154	154	

File: CTP_B787_PERF_CRZINITE.ZIP\CTP_B787_PERF_CRZINITE.RPT (continued)

155	155	
156	156	
157	157	***** End of Report *****

File: CTP_B787_PERF_CRZINITE.ZIP\CTP_B787_PERF_CRZINITE.VER

1	1	
2	2	
3	3	RESULTS FILE
4	4	
5	5	*****
6	6	Test Results Summary
7	7	
8	8	Percentage of Comparisons Passed : 100.0000%
9	9	
10	10	Total Number of Comparisons Failed : 0
11	11	Total Number of Unknown Comparisons : 0
12	12	Total Number of Comparisons Passed : 46
13	13	Total Number of Comparisons : 46
14	14	Total Number of Test Cases Included : 12
15	15	
16	16	Test Complete
17	17	
18	18	
19	19	
20	20	*****
21	21	
22	22	
23		Test Start Time: 03/25/2013 Mon 9:45:38"
	23	Test Start Time: 07/02/2014 Wed 12:53:16"
24	24	FILE : CTP_B787_PERF_CRZINITE.TDF
25	25	SOURCE CONFIGURATION : ISS (Instruction Set Simulator)
26	26	DESCRIPTION : B787 Crz_InitStepTerms initializes the step climb terminations.
27	27	MODIFICATION HISTORY :
28	28	DATE SCR # AUTHOR DESCRIPTION
29	29	=====
30	30	11-May-2006 1134.00 Henson Zhao Initial Development for B787 cycle 1 phase 1 Build
		» ML134.
31	31	25-Aug-2006 1134.00 Alex Xie Execution for B787 cycle 1 phase 1 Build SBC127.
32	32	1.Format changed from HDB to GDB.
33	33	2.Updated SUT_VARS.
34	34	17-Sep-2007 4845.00 He Wang Update for B787 Load 4.5 Build SBC425.
35	35	1. Updated SRD/SDD generations:

File: CTP_B787_PERF_CRZINITE.ZIP\CTP_B787_PERF_CRZINITE.VER (continued)

36	36				FMF_PERF_CRZ_PHASE.SDD; 6 --> 12
37	37				FMF_PERF_PREDS_CRZ_PHASE.SRD; 13 --> 20
38	38				2. Removed FMF_PERF_PREDS_PHASES.SRD;7
39	39				3. Removed SRD FMCS_19_3067 and FMCS_19_20006096.
40	40				4. Added some SUTs and removed extra SUTs.
41	41				5. Updated all breakpoints as per code changing.
42	42				6. Added TC 12 for robust test.
43	43	24-Jun-2008	6880.00	Xinghua Liu	Updated for B787 Load 7.0 Build SBC617_8F2.
44	44				1. Updated SRD generation:
45	45				FMF_PERF_PREDS_CRZ_PHASE.SRD ; 20 -->22
46	46				2. Modified all TC for remove stub.
47	47	May-24-2010	13550.00	Sumei Li	Updated for B787 RFS Build SBC922_811B2
48	48				1. Updated the break points.
49	49	18-Mar-2013	15875.00	Lu Shubo	Update for B787 BP2 LD5 on Build ACMBLD_070_SBC.
50	50				1. Updated breakpoints in TC 1~12.
	51	2-Jul-2014	15655.04	Chen Yongbing	Update for B787 BP3 LD3 on Build SBC2415_93C.
	52				1. Updated SRD generation:
	53				FMF_PERF_PREDS_CRZ_PHASE.SRD; 22 ->
	54				FMF_PERF_PREDS_CRZ_PHASE_SRD.DOCX; 23.
	55				2. Updated breakpoints as per SCR 15655.03.
	56				3. Updated TCs 1,6 for newly added PERF_SRD_B_00413
	57				as per SCR 15655.01.
51	58	SRD and SDD DETAILS : FMF_PERF_CRZ_PHASE.SDD ; 12			
52		FMF_PERF_PREDS_CRZ_PHASE.SRD ; 22			
	59	FMF_PERF_PREDS_CRZ_PHASE_SRD.DOCX ; 23			
53	60	TRACE DETAILS :			
54	61	ANCHOR	:	PERF_TEST_00014	
55	62	SOURCE	:	SDD: FMCS_19_21027005	
56	63			SRD: FMCS_19_20006025, FMCS_19_20006026, FMCS_19_20006027, FMCS_19_20006028,	
57	64			FMCS_19_20006029, FMCS_19_20006097, FMCS_19_20006098, FMCS_19_20006099,	
58		FMCS_19_20006102, FMCS_19_20006100			
	65	FMCS_19_20006102, FMCS_19_20006100, PERF_SRD_B_00413			
59	66				
60	67				
61	68	CONSTANT			VALUE
62	69	-----			-----
63	70	FP_DEF_TOL			0.0001
64	71				
65	72				
66	73	DEFAULTS			VALUE
67	74	-----			-----
68	75	Start_Sut			1

File: CTP_B787_PERF_CRZINITE.ZIP\CTP_B787_PERF_CRZINITE.VER (continued)

69	76	load_ge_config	boolean'(true)
70	77		
71	78		
72	79	CONSTANT	VALUE
73	80	-----	-----
74	81	DBG_TIMEOUT	300
75	82		
76	83		
77	84	TESTID: 1	
78	85	This verify When If the step type is SpecStep, the StepAlt termination value is set to the Specified step	
79	86	altitude.If the aircraft is in climb, and a step was moved to the cruise phase from the climb phase the StepAlt	
80	87	value is set to the climb specified step altitude, and the StepDist is activated with a distance value indicating	
81	88	the step should begin immediately.	
82	89		
83	90	REQUIREMENTS UNDER EVALUATION: FMCS_19_20006025, FMCS_19_20006026, FMCS_19_20006027, FMCS_19_20006028, FMCS_19_20006029	
84		FMCS_19_20006097, FMCS_19_20006098, FMCS_19_20006099, FMCS_19_20006102	
	91	FMCS_19_20006097, FMCS_19_20006098, FMCS_19_20006099, FMCS_19_20006102, PERF_SRD_B_00413	
85	92	SUPPORTING REQUIREMENTS : FMCS_19_21027005	
86	93		
87	94		
88	95	INPUT	VALUE
89	96	-----	-----
90	97	Test_Firstpass	boolean'(false)
91	98	Test_Steptype	CFP_PERF_STEP_IFTYPES.specstep
92	99	Test_LGB_Search	0
93	100	Perf_Preds_Lfdata.VTPlogic.Firstpass	boolean'(true)
94	101	Perf_Crztalt_Lfdata.LastCrzAlt.Valid	boolean'(false)
95	102	Perf_Preds_Lfdata.Vgbptr	CLB2L
96	103	Perf_RTA_Lfdata.Pred_Pastrta	boolean'(true)
97	104	Perf_WTS_Lfdata.Always_Compute_Max_Speed	boolean'(false)
98	105	Perf_Integrators_Lfdata.IntProgBuf.Hprog	27004.0
99	106	Perf_Crztalt_Lfdata.Crztalt	27000.0
100	107	Perf_Preds_Lfdata.Fltphase	FMCS_Base_Types.climb
101	108	Perf_Preds_Lfdata.NavPtr	1
102	109	Perf_Preds_Lfdata.PrevNavPtr	2
103	110	Perf_Integrators_Lfdata.IntProgBuf.Xprog	1000.0
104	111	Perf_Integrators_Lfdata.TermBuf.TermArray(11).Active	boolean'(true)
105	112	Perf_Integrators_Lfdata.TermBuf.TermArray(11).Value	1.0
106		Idx_Profile_Ildata.Ialtprofptree(0)(1).Dtd.Data	100.0
107		Idx_Profile_Ildata.Ialtprofptree(0)(1).Dtd.Valid	boolean'(true)
	113	Perf_Crz_Pkg.Opt_Step_PND_Ptr	1
	114	Perf_WTS_Lfdata.Previous_Pass_Info_Rec.Step(1).Distance.Data	100.0

File: CTP_B787_PERF_CRZINITE.ZIP\CTP_B787_PERF_CRZINITE.VER (continued)

115	Perf_WTS_Lfdata.Previous_Pass_Info_Rec.Step(1).Distance.Valid			boolean'(true)
108	116 Perf_LGB_Lfdata.LGB(2).Fpln_Data.SpAlt1Val			boolean'(false)
109	117 Perf_LGB_Lfdata.LGB(2).Fpln_Data.SpAlt1Pos			AC_Position_Types.At_Alt
110	118 Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Stepptr			99
111	119 Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Valid			boolean'(true)
112	120 Perf_Integrators_Lfdata.TermBuf.TermArray(6).Active			boolean'(false)
113	121 Perf_Integrators_Lfdata.TermBuf.TermArray(6).Value			1.0
114	122 Step_Ptr			99
115	123			
116	124			
117	125 OUTPUT	EXPECTED	TOLERANCE	ACTUAL
	» P/F			
118	126 -----			
	» ----			
119	127 Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Stepptr	99	(N/A)	99
	» P			
120	128 Perf_Integrators_Lfdata.TermBuf.TermArray(6).Active	boolean'(false)	(N/A)	false
	» P			
121	129 Perf_Integrators_Lfdata.TermBuf.TermArray(6).Value	1.0	0.0001	1.0
	» P			
122	130 Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Valid	boolean'(false)	(N/A)	false
	» P			
123	131 Perf_Integrators_Lfdata.TermBuf.TermArray(11).Value	100.0	0.0001	100.0
	» P			
124	132			
125	133			
126	134 =====> All 5 Comparisons Passed <=====			
127	135			
128	136			
129	137 TESTID: 2			
130	138 This verify When If the step type is SpecStep, the StepAlt termination value is set to the Specified step			
131	139 altitude.If the aircraft is in climb, and a step was moved to the cruise phase from the climb phase the StepAlt			
132	140 value is set to the climb specified step altitude, and the StepDist is activated with a distance value indicating			
133	141 the step should begin immediately.			
134	142			
135	143 REQUIREMENTS UNDER EVALUATION: FMCS_19_20006025, FMCS_19_20006026, FMCS_19_20006027, FMCS_19_20006028, FMCS_19_20006029			
136	144 FMCS_19_20006097, FMCS_19_20006098, FMCS_19_20006099, FMCS_19_20006102, FMCS_19_20006100			
137	145 SUPPORTING REQUIREMENTS : FMCS_19_21027005			
138	146			
139	147			
140	148 INPUT			VALUE
141	149 -----			

File: CTP_B787_PERF_CRZINITE.ZIP\CTP_B787_PERF_CRZINITE.VER (continued)

142	150	Test_Firstpass			boolean'(true)
143	151	Test_Steptype			CFP_PERF_STEP_IFTYPES.specstep
144	152	Test_LGB_Search			0
145	153	Perf_Preds_Lfdata.VTPlogic.Firstpass			boolean'(true)
146	154	Perf_Crztalt_Lfdata.LastCrzAlt.Valid			boolean'(false)
147	155	Perf_Preds_Lfdata.Vgbptr			CLB2L
148	156	Perf_RTA_Lfdata.Pred_Pastrta			boolean'(true)
149	157	Perf_WTS_Lfdata.Always_Compute_Max_Speed			boolean'(false)
150	158	Perf_Integrators_Lfdata.IntProgBuf.Hprog			27004.0
151	159	Perf_Crztalt_Lfdata.Crztalt			27000.0
152	160	Perf_Preds_Lfdata.Fltphase			FMCS_Base_Types.climb
153	161	Perf_Preds_Lfdata.NavPtr			1
154	162	Perf_Preds_Lfdata.PrevNavPtr			2
155	163	Perf_LGB_Lfdata.LGB(2).Fpln_Data.SpAlt1Val			boolean'(true)
156	164	Perf_LGB_Lfdata.LGB(2).Fpln_Data.SpAlt1Pos			AC_Position_Types.SC
157	165	Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Stepptr			0
158	166	Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Valid			boolean'(true)
159	167	Perf_Integrators_Lfdata.TermBuf.TermArray(6).Active			boolean'(false)
160	168	Perf_Integrators_Lfdata.TermBuf.TermArray(6).Value			1.0
161	169	Perf_Integrators_Lfdata.IntProgBuf.Xprog			1000.0
162	170	Step_Ptr			99
163	171	Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Steptype			CFP_PERF_STEP_IFTYPES.PastSpecStep
164	172				
165	173				
166	174	OUTPUT	EXPECTED	TOLERANCE	ACTUAL
		» P/F			
167	175	-----	-----	-----	-----
		» ----			
168	176	Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Stepptr	2	(N/A)	2
		» P			
169	177	Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Steptype			
170	178	CFP_PERF_STEP_IFTYPES.specstep		(N/A)	specstep
		» P			
171	179	Perf_Integrators_Lfdata.TermBuf.TermArray(6).Active	boolean'(true)	(N/A)	true
		» P			
172	180	Perf_Integrators_Lfdata.TermBuf.TermArray(6).Value	1000.0	0.0001	1000.0
		» P			
173	181	Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Valid	boolean'(false)	(N/A)	false
		» P			
174	182				
175	183				
176	184	====> All 5 Comparisons Passed <====			

File: CTP_B787_PERF_CRZINITE.ZIP\CTP_B787_PERF_CRZINITE.VER (continued)

177	185				
178	186				
179	187	TESTID: 3			
180	188	This verify When If the step type is SpecStep, the StepAlt termination value is set to the Specified step			
181	189	altitude.If the aircraft is in climb, and a step was moved to the cruise phase from the climb phase the StepAlt			
182	190	value is set to the climb specified step altitude, and the StepDist is activated with a distance value indicating			
183	191	the step should begin immediately.			
184	192				
185	193	REQUIREMENTS UNDER EVALUATION: FMCS_19_20006025, FMCS_19_20006026, FMCS_19_20006027, FMCS_19_20006028, FMCS_19_20006029			
186	194	FMCS_19_20006097, FMCS_19_20006098, FMCS_19_20006099, FMCS_19_20006102			
187	195				
188	196	SUPPORTING REQUIREMENTS : FMCS_19_21027005			
189	197				
190	198				
191	199	INPUT		VALUE	
192	200	-----		-----	
193	201	Test_Firstpass		boolean'(true)	
194	202	Test_Steptype		CFP_PERF_STEP_IFTYPES.specstep	
195	203	Test_LGB_Search		0	
196	204	Perf_Preds_Lfdata.VTPlogic.Firstpass		boolean'(true)	
197	205	Perf_Crzalt_Lfdata.LastCrzAlt.Valid		boolean'(false)	
198	206	Perf_Preds_Lfdata.Vgbptr		CLB2L	
199	207	Perf_RTA_Lfdata.Pred_Pastrta		boolean'(true)	
200	208	Perf_WTS_Lfdata.Always_Compute_Max_Speed		boolean'(false)	
201	209	Perf_Integrators_Lfdata.IntProgBuf.Hprog		27004.0	
202	210	Perf_Crzalt_Lfdata.Crzalt		27000.0	
203	211	perf_Preds_Lfdata.Fltphase		FMCS_Base_Types.climb	
204	212	Perf_Preds_Lfdata.NavPtr		1	
205	213	Perf_Preds_Lfdata.PrevNavPtr		2	
206	214	false			
207	215	Perf_LGB_Lfdata.LGB(2).Fpln_Data.SpAlt1Val		boolean'(false)	
208	216	Perf_LGB_Lfdata.LGB(2).Fpln_Data.SpAlt1Pos		AC_Position_Types.SC	
209	217	Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Stepptr		0	
210	218	Perf_Integrators_Lfdata.TermBuf.TermArray(6).Active		boolean'(false)	
211	219	Perf_Integrators_Lfdata.TermBuf.TermArray(6).Value		1.0	
212	220	Perf_Integrators_Lfdata.IntProgBuf.Xprog		1000.0	
213	221	Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Valid		boolean'(true)	
214	222	Step_Ptr		99	
215	223				
216	224				
217	225	OUTPUT	EXPECTED	TOLERANCE	ACTUAL
		» P/F			

File: CTP_B787_PERF_CRZINITE.ZIP\CTP_B787_PERF_CRZINITE.VER (continued)

218	226	-----	-----	-----	-----
		» ----			
219	227	Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Stepptr	0	(N/A)	0
		» P			
220	228	Perf_Integrators_Lfdata.TermBuf.TermArray(6).Active	boolean'(false)	(N/A)	false
		» P			
221	229	Perf_Integrators_Lfdata.TermBuf.TermArray(6).Value	1.0	0.0001	1.0
		» P			
222	230	Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Valid	boolean'(false)	(N/A)	false
		» P			
223	231				
224	232				
225	233	====> All 4 Comparisons Passed <====			
226	234				
227	235				
228	236	TESTID: 4			
229	237	This verify When If the step type is SpecStep, the StepAlt termination value is set to the Specified step			
230	238	altitude.If the aircraft is not in climb, and the StepDist is invalid.			
231	239				
232	240	REQUIREMENTS UNDER EVALUATION: FMCS_19_20006025, FMCS_19_20006026, FMCS_19_20006027, FMCS_19_20006028, FMCS_19_20006029			
233	241	FMCS_19_20006097, FMCS_19_20006098, FMCS_19_20006099, FMCS_19_20006102			
234	242				
235	243	SUPPORTING REQUIREMENTS : FMCS_19_21027005			
236	244				
237	245				
238	246	INPUT		VALUE	
239	247	-----	-----	-----	-----
240	248	Test_Firstpass		boolean'(false)	
241	249	Test_Steptype		CFP_PERF_STEP_IFTYPES.specstep	
242	250	Test_LGB_Search		0	
243	251	Perf_Preds_Lfdata.VTPlogic.Firstpass		boolean'(true)	
244	252	Perf_Crzalt_Lfdata.LastCrzAlt.Valid		boolean'(false)	
245	253	Perf_Preds_Lfdata.Vgbptr		CLB2L	
246	254	Perf_RTA_Lfdata.Pred_Pastrta		boolean'(true)	
247	255	Perf_WTS_Lfdata.Always_Compute_Max_Speed		boolean'(false)	
248	256	Perf_Integrators_Lfdata.IntProgBuf.Hprog		27004.0	
249	257	Perf_Crzalt_Lfdata.Crzalt		27000.0	
250	258	Perf_Preds_Lfdata.Fltphase		FMCS_Base_Types.Descent	
251	259	Perf_Preds_Lfdata.NavPtr		1	
252	260	Perf_Preds_Lfdata.PrevNavPtr		1	
253	261	false			
254	262	Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Valid		boolean'(true)	

File: CTP_B787_PERF_CRZINITE.ZIP\CTP_B787_PERF_CRZINITE.VER (continued)

255	263				
256	264				
257	265	OUTPUT	EXPECTED	TOLERANCE	ACTUAL
		» P/F			
258	266	-----	-----	-----	-----
		» ----			
259	267	Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Valid	boolean'(false)	(N/A)	false
		» P			
260	268				
261	269				
262	270	====> All 1 Comparisons Passed <====			
263	271				
264	272				
265	273	TESTID: 5			
266	274	This verify When If the Step type is PastSpecStep, the StepAlt termination value is set to the current specified step			
267	275	altitude, and the StepDist termination is activated with a distance value indicating the step should begin immediately			
268	276	unless the step is moved due to maximum altitude restrictions, in which case the StepDist termination is set to the			
269	277	estimated step point distance. If the aircraft is in climb, and no previous step point exists, the StepAlt termination			
270	278	value is set to the current specified step altitude, and the StepDist termination is activated with a distance value			
271	279	indicating the step should begin immediately.			
272	280				
273	281	REQUIREMENTS UNDER EVALUATION: FMCS_19_20006025, FMCS_19_20006026, FMCS_19_20006027, FMCS_19_20006028, FMCS_19_20006029			
274	282	FMCS_19_20006097, FMCS_19_20006098, FMCS_19_20006099, FMCS_19_20006102, FMCS_19_20006100			
275	283				
276	284	SUPPORTING REQUIREMENTS : FMCS_19_21027005			
277	285				
278	286				
279	287	INPUT		VALUE	
280	288	-----	-----	-----	-----
281	289	Test_Firstpass		boolean'(false)	
282	290	Test_Steptype		CFP_PERF_STEP_IFTYPES.PastSpecStep	
283	291	Test_LGB_Search		0	
284	292	Perf_Preds_Lfdata.VTPlogic.Firstpass		boolean'(true)	
285	293	Perf_Crzalt_Lfdata.LastCrzAlt.Valid		boolean'(false)	
286	294	Perf_Preds_Lfdata.Vgbptr		CLB2L	
287	295	Perf_RTA_Lfdata.Pred_Pastrta		boolean'(true)	
288	296	Perf_WTS_Lfdata.Always_Compute_Max_Speed		boolean'(false)	
289	297	Perf_Integrators_Lfdata.IntProgBuf.Hprog		27004.0	
290	298	Perf_Crzalt_Lfdata.Crzalt		27000.0	
291	299	Perf_Preds_Lfdata.Fltphase		FMCS_Base_Types.climb	
292	300	Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Stepptr		0	
293	301	Perf_Preds_Lfdata.Navptr		1	

File: CTP_B787_PERF_CRZINITE.ZIP\CTP_B787_PERF_CRZINITE.VER (continued)

294	302	Perf_Preds_Lfdata.PrevNavptr			0
295	303	Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Valid		boolean'(true)	
296	304	Perf_Integrators_Lfdata.TermBuf.TermArray(6).Active		boolean'(false)	
297	305	Perf_Integrators_Lfdata.TermBuf.TermArray(6).Value		0.0	
298	306	Perf_Integrators_Lfdata.IntProgBuf.Xprog		1000.0	
299	307	Step_Ptr		99	
300	308				
301	309				
302	310	OUTPUT	EXPECTED	TOLERANCE	ACTUAL
		» P/F			
303	311	-----	-----	-----	-----
		» ----			
304	312	Perf_Integrators_Lfdata.TermBuf.TermArray(6).Active	boolean'(true)	(N/A)	true
		» P			
305	313	Perf_Integrators_Lfdata.TermBuf.TermArray(6).Value	1000.0	0.0001	1000.0
		» P			
306	314	Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Stepptr	1	(N/A)	1
		» P			
307	315	Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Valid	boolean'(false)	(N/A)	false
		» P			
308	316				
309	317				
310	318	====> All 4 Comparisons Passed <====			
311	319				
312	320				
313	321	TESTID: 6			
314	322	This verify When If the Step type is PEopt, a check is made to see if the aircraft is past			
315	323	the current step distance to destination.			
316	324				
317	325	REQUIREMENTS UNDER EVALUATION: FMCS_19_20006025, FMCS_19_20006026, FMCS_19_20006027, FMCS_19_20006028, FMCS_19_20006029			
318		FMCS_19_20006097, FMCS_19_20006098, FMCS_19_20006099, FMCS_19_20006102			
	326	FMCS_19_20006097, FMCS_19_20006098, FMCS_19_20006099, FMCS_19_20006102, PERF_SRD_B_00413			
319	327				
320	328	SUPPORTING REQUIREMENTS : FMCS_19_21027005			
321	329				
322	330				
323	331	INPUT		VALUE	
324	332	-----	-----	-----	-----
325	333	Test_Firstpass		boolean'(false)	
326	334	Test_Steptype		CFP_PERF_STEP_IFTYPES.PEopt	
327	335	Test_LGB_Search		0	
328	336	Perf_Preds_Lfdata.VTPlogic.Firstpass		boolean'(true)	

File: CTP_B787_PERF_CRZINITE.ZIP\CTP_B787_PERF_CRZINITE.VER (continued)

329	337	Perf_Crzalt_Lfdata.LastCrzAlt.Valid			boolean'(false)
330	338	Perf_Preds_Lfdata.Vgbptr			CLB2L
331	339	Perf_RTA_Lfdata.Pred_Pastrta			boolean'(true)
332	340	Perf_WTS_Lfdata.Always_Compute_Max_Speed			boolean'(false)
333	341	Perf_Integrators_Lfdata.IntProgBuf.Hprog			27004.0
334	342	Perf_Crzalt_Lfdata.Crzalt			27000.0
335	343	Perf_Integrators_Lfdata.TermBuf.TermArray(6).Active			boolean'(false)
336	344	Perf_Integrators_Lfdata.TermBuf.TermArray(6).Value			9999.9
337	345	Perf_Integrators_Lfdata.IntProgBuf.Xprog			10000.0
338		Idx_Profile_Ildata.Ialtprofptree(0)(1).Dtd.Valid			boolean'(true)
339		Idx_Profile_Ildata.Ialtprofptree(0)(1).Dtd.Data			1000.0
	346	Perf_Crz_Pkg.Opt_Step_PND_Ptr			1
	347	Perf_WTS_Lfdata.Previous_Pass_Info_Rec.Step(1).Distance.Data			1000.0
	348	Perf_WTS_Lfdata.Previous_Pass_Info_Rec.Step(1).Distance.Valid			boolean'(true)
340	349	Perf_Integrators_Lfdata.TermBuf.TermArray(11).Value			1.0
341	350	Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Valid			boolean'(true)
342	351	Perf_Integrators_Lfdata.TermBuf.TermArray(11).Active			boolean'(false)
343	352				
344	353				
345	354	OUTPUT	EXPECTED	TOLERANCE	ACTUAL
		» P/F			
346	355	-----	-----	-----	-----
		» ----			
347	356	Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Valid	boolean'(false)	(N/A)	false
		» P			
348	357	Perf_Integrators_Lfdata.TermBuf.TermArray(11).Active	boolean'(true)	(N/A)	true
		» P			
349	358	Perf_Integrators_Lfdata.TermBuf.TermArray(11).Value	1000.0	0.0001	1000.0
		» P			
350	359				
351	360				
352	361	====> All 3 Comparisons Passed <====			
353	362				
354	363				
355	364	TESTID: 7			
356	365	This verify When If the step type is Opt, it is determined whether: both a new step altitude			
357	366	and gross weight need to be computed.			
358	367				
359	368	REQUIREMENTS UNDER EVALUATION: FMCS_19_20006025, FMCS_19_20006026, FMCS_19_20006027, FMCS_19_20006028, FMCS_19_20006029			
360	369	FMCS_19_20006097, FMCS_19_20006098, FMCS_19_20006099, FMCS_19_20006102			
361	370				
362	371	SUPPORTING REQUIREMENTS : FMCS_19_21027005			

File: CTP_B787_PERF_CRZINITE.ZIP\CTP_B787_PERF_CRZINITE.VER (continued)

399	408						
400	409	REQUIREMENTS UNDER EVALUATION: FMCS_19_20006025, FMCS_19_20006026, FMCS_19_20006027, FMCS_19_20006028, FMCS_19_20006029,					
401	410	FMCS_19_20006097, FMCS_19_20006098, FMCS_19_20006099, FMCS_19_20006102					
402	411						
403	412	SUPPORTING REQUIREMENTS :		FMCS_19_21027005			
404	413						
405	414						
406	415	INPUT			VALUE		
407	416	-----					
408	417	Test_Firstpass			boolean'(false)		
409	418	Test_Steptype			CFP_PERF_STEP_IFTYPES.Opt		
410	419	Test_LGB_Search			0		
411	420	Perf_Preds_Lfdata.VTPlogic.Firstpass			boolean'(true)		
412	421	Perf_Crzalt_Lfdata.LastCrzAlt.Valid			boolean'(false)		
413	422	Perf_Preds_Lfdata.Vgbptr			CLB2L		
414	423	Perf_RTA_Lfdata.Pred_Pastrta			boolean'(true)		
415	424	Perf_WTS_Lfdata.Always_Compute_Max_Speed			boolean'(false)		
416	425	Perf_Integrators_Lfdata.IntProgBuf.Hprog			80006.0		
417	426	Perf_Crzalt_Lfdata.Crzalt			80000.0		
418	427	Perf_Crz_Pkg.Step_Size.Valid			boolean'(false)		
419	428	Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Stepptr			99		
420	429	Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Valid			boolean'(true)		
421	430	Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Opt_Stepalt			0.0		
422	431						
423	432						
424	433	OUTPUT		EXPECTED	TOLERANCE	ACTUAL	
		» P/F					
425	434	-----					
		» ----					
426	435	Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Stepptr			0	(N/A)	0
		» P					
427	436	Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Valid			boolean'(false)	(N/A)	false
		» P					
428	437	Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Opt_Stepalt			81000.0	0.0001	81000.0
		» P					
429	438	Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Steptype					
430	439	CFP_PERF_STEP_IFTYPES.Nostep				(N/A)	nostep
		» P					
431	440						
432	441						
433	442	====> All 4 Comparisons Passed <====					
434	443						

File: CTP_B787_PERF_CRZINITE.ZIP\CTP_B787_PERF_CRZINITE.VER (continued)

435	444				
436	445	TESTID: 9			
437	446	This verify When If the Step type is PastSpecStep, the StepAlt termination value is set to			
438	447	the current specified step altitude.			
439	448				
440	449	REQUIREMENTS UNDER EVALUATION: FMCS_19_20006025, FMCS_19_20006026, FMCS_19_20006027, FMCS_19_20006028, FMCS_19_20006029			
441	450	FMCS_19_20006097, FMCS_19_20006098, FMCS_19_20006099, FMCS_19_20006102			
442	451				
443	452	SUPPORTING REQUIREMENTS : FMCS_19_21027005			
444	453				
445	454				
446	455	INPUT			VALUE
447	456	-----			-----
448	457	Test_Firstpass			boolean'(false)
449	458	Test_Steptype			CFP_PERF_STEP_IFTYPES.PastSpecStep
450	459	Test_LGB_Search			0
451	460	Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Stepptr			0
452	461	Perf_Preds_Lfdata.Navptr			1
453	462	Perf_Preds_Lfdata.Fltphase			FMCS_Base_Types.climb
454	463	Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Valid			boolean'(true)
455	464	Perf_Preds_Lfdata.PrevNavPtr			0
456	465	Perf_Integrators_Lfdata.TermBuf.TermArray(6).Active			boolean'(false)
457	466	Perf_Integrators_Lfdata.TermBuf.TermArray(6).Value			0.0
458	467	Perf_Integrators_Lfdata.IntProgBuf.Xprog			10000.0
459	468	true			
460	469	Step_Ptr			99
461	470				
462	471				
463	472	OUTPUT	EXPECTED	TOLERANCE	ACTUAL
		» P/F			
464	473	-----	-----	-----	-----
		» ----			
465	474	Perf_Integrators_Lfdata.TermBuf.TermArray(6).Active	boolean'(true)	(N/A)	true
		» P			
466	475	Perf_Integrators_Lfdata.TermBuf.TermArray(6).Value	10000.0	0.0001	10000.0
		» P			
467	476	Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Stepptr	1	(N/A)	1
		» P			
468	477	Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Valid	boolean'(false)	(N/A)	false
		» P			
469	478				
470	479				

File: CTP_B787_PERF_CRZINITE.ZIP\CTP_B787_PERF_CRZINITE.VER (continued)

471 480 ===== All 4 Comparisons Passed =====

472 481

473 482

474 483 TESTID: 10

475 484 This verify When If the Step type is PastSpecStep, the StepAlt termination value is set to

476 485 the current specified step altitude.

477 486

478 487 REQUIREMENTS UNDER EVALUATION: FMCS_19_20006025, FMCS_19_20006026, FMCS_19_20006027, FMCS_19_20006028, FMCS_19_20006029

479 488 FMCS_19_20006097, FMCS_19_20006098, FMCS_19_20006099, FMCS_19_20006102

480 489

481 490 SUPPORTING REQUIREMENTS : FMCS_19_21027005

482 491

483 492

484 493 INPUT

VALUE

485 494	-----	-----	-----
486 495	Test_Firstpass		boolean'(false)
487 496	Test_Steptype	CFP_PERF_STEP_IFTYPES.PastSpecStep	
488 497	Test_LGB_Search		0
489 498	Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Stepptr		0
490 499	Perf_Preds_Lfdata.Navptr		1
491 500	Perf_Preds_Lfdata.Fltphase	FMCS_Base_Types.Descent	
492 501	Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Valid		boolean'(true)
493 502	Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Data		500.0
494 503	Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Fixdistodest		1000.0
495 504	Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Valid		boolean'(true)
496 505	Perf_Integrators_Lfdata.TermBuf.TermArray(6).Active		boolean'(false)
497 506	Perf_Integrators_Lfdata.TermBuf.TermArray(6).Value		0.0

498 507

499 508

500 509 OUTPUT

EXPECTED

TOLERANCE

ACTUAL

» P/F

501 510	-----	-----	-----
---------	-------	-------	-------

» ----

502 511	Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Stepptr	1	(N/A)	1
---------	--	---	-------	---

» P

503 512	Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Valid	boolean'(false)	(N/A)	false
---------	--	-----------------	-------	-------

» P

504 513	Perf_Integrators_Lfdata.TermBuf.TermArray(6).Active	boolean'(true)	(N/A)	true
---------	---	----------------	-------	------

» P

505 514	Perf_Integrators_Lfdata.TermBuf.TermArray(6).Value	500.0	0.0001	500.0
---------	--	-------	--------	-------

» P

506 515

File: CTP_B787_PERF_CRZINITE.ZIP\CTP_B787_PERF_CRZINITE.VER (continued)

```

507 516
508 517 ===== All 4 Comparisons Passed =====
509 518
510 519
511 520 TESTID: 11
512 521 This verify When If the Step type is PastSpecStep, the StepAlt termination value is set to
513 522 the current specified step altitude.
514 523
515 524 REQUIREMENTS UNDER EVALUATION: FMCS_19_20006025, FMCS_19_20006026, FMCS_19_20006027, FMCS_19_20006028, FMCS_19_20006029
516 525 FMCS_19_20006097, FMCS_19_20006098, FMCS_19_20006099, FMCS_19_20006102
517 526
518 527 SUPPORTING REQUIREMENTS : FMCS_19_21027005
519 528
520 529
521 530 INPUT
522 531 -----
523 532 Test_Firstpass boolean'(false)
524 533 Test_Steptype CFP_PERF_STEP_IFTYPES.PastSpecStep
525 534 Test_LGB_Search 0
526 535 Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Stepptr 0
527 536 Perf_Preds_Lfdata.Navptr 1
528 537 Perf_Preds_Lfdata.Fltphase FMCS_Base_Types.Descent
529 538 Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Valid boolean'(false)
530 539 Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Data 500.0
531 540 Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Fixdistodest 1000.0
532 541 Perf_Integrators_Lfdata.TermBuf.TermArray(6).Active boolean'(false)
533 542 Perf_Integrators_Lfdata.TermBuf.TermArray(6).Value 0.0
534 543 Perf_Integrators_Lfdata.IntProgBuf.Xprog 999.0
535 544
536 545
537 546 OUTPUT
538 547 -----
539 548 Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Stepptr 1 (N/A) 1
540 549 Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Valid boolean'(false) (N/A) false
541 550 Perf_Integrators_Lfdata.TermBuf.TermArray(6).Active boolean'(true) (N/A) true
542 551 Perf_Integrators_Lfdata.TermBuf.TermArray(6).Value 999.0 0.0001 999.0

```

File: CTP_B787_PERF_CRZINITE.ZIP\CTP_B787_PERF_CRZINITE.VER (continued)

```

543 552
544 553
545 554 >>>> All 4 Comparisons Passed <<<<
546 555
547 556
548 557 TESTID: 12
549 558 This verify When If the Step type is PastSpecStep, the StepAlt termination value is set to
550 559 the current specified step altitude.
551 560
552 561 Robust test follows:
553 562 Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Data >
554 563 Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Fixdistodest.
555 564 Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Valid is true.
556 565
557 566 REQUIREMENTS UNDER EVALUATION: FMCS_19_20006025, FMCS_19_20006026, FMCS_19_20006027, FMCS_19_20006028, FMCS_19_20006029
558 567 FMCS_19_20006097, FMCS_19_20006098, FMCS_19_20006099, FMCS_19_20006102, FMCS_19_20006100
559 568
560 569 SUPPORTING REQUIREMENTS : FMCS_19_21027005
561 570
562 571
563 572 INPUT VALUE
564 573 -----
565 574 Test_Firstpass boolean'(false)
566 575 Test_Steptype CFP_PERF_STEP_IFTYPES.PastSpecStep
567 576 Test_LGB_Search 0
568 577 Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Stepptr 0
569 578 Perf_Preds_Lfdata.Navptr 1
570 579 Perf_Preds_Lfdata.Fltphase FMCS_Base_Types.Descent
571 580 Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Valid boolean'(true)
572 581 Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Data 1000.0
573 582 Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Fixdistodest 500.0
574 583 Perf_Integrators_Lfdata.TermBuf.TermArray(6).Active boolean'(false)
575 584 Perf_Integrators_Lfdata.TermBuf.TermArray(6).Value 0.0
576 585 Perf_Integrators_Lfdata.IntProgBuf.Xprog 999.0
577 586
578 587
579 588 OUTPUT EXPECTED TOLERANCE ACTUAL
580 589 -----
581 590 >> ----
581 590 Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Stepptr 1 (N/A) 1
581 590 >> P

```

File: CTP_B787_PERF_CRZINITE.ZIP\CTP_B787_PERF_CRZINITE.VER (continued)

582	591	Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Valid	boolean'(false)	(N/A)	false
		» P			
583	592	Perf_Integrators_Lfdata.TermBuf.TermArray(6).Active	boolean'(true)	(N/A)	true
		» P			
584	593	Perf_Integrators_Lfdata.TermBuf.TermArray(6).Value	999.0	0.0001	999.0
		» P			
585	594				
586	595				
587	596	====> All 4 Comparisons Passed <====			
588	597				
589	598				
590		Test End Time: 03/25/2013 Mon 9:46:00"			
	599	Test End Time: 07/02/2014 Wed 12:53:34"			
591	600	Test Generation System (TGS) Version v5.5, ps4082887-109			
592	601				
593		UserID: E527970 Node: CH71DT56F653X			
	602	UserID: E803143 Node: CH71DT517T0W1			
594	603				
595	604	Current Build			
596		C:\B787\BUILDS\ACMBLD_070_SBC			
	605	C:\B787\BUILDS\SBC2415_93C			

File: CTP_B787_PERF_CRZINITE.ZIP\CTP_B787_PERF_CRZINITE.xml

1	1	<!-- FMS Integ Release - Test Partition -->			
2	2	<ApplicationDescription xmlns="ARINC653"			
3	3	xmlns:xi="http://www.w3.org/2001/XInclude"			
4	4	xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"			
5	5	xsi:schemaLocation="ARINC653 ./schema/VXCR_Application.xsd"			
6	6	EntryPoint="ctp_b787_perf_crzinite"			
7	7	InitializationTime="1">			
8	8	<Description Name="Test_Partition" Version="1.0"			
9	9	BuildQualifier="PPC604gnu.debug"			
10	10	SourcePath=".\\ctp_b787_perf_crzinite"			
11	11	SourceModule="ctp_b787_perf_crzinite.out"			
12	12	SourceLanguage="Ada"			
13	13	Criticality="D0178B_E">			
14	14	<Notes/>			
15	15	</Description>			
16	16				
17	17	<MemorySize			
18	18	MemorySizeHeap="0x00020000"			
19	19	MemorySizeText="0x00f6f000"			

File: CTP_B787_PERF_CRZINITE.ZIP\CTP_B787_PERF_CRZINITE.xml (continued)

20	20	MemorySizeRoData="0x00161000"
21	21	MemorySizeData="0x00614000"
22	22	MemorySizeBss="0x0144b000"
23	23	MemorySizePersistentData="0x00001000"
24	24	MemorySizePersistentBss="0x00426000" />
25	25	<Ports>
26	26	</Ports>
27	27	</ApplicationDescription>
28	28	

File: CTP_B787_PERF_CRZINITE.ZIP\CTP_B787_PERF_CRZINITE_ADA.gpr

1	1	with "..\..\..\GPS\fm\fm.gpr";
2	2	project CTP_B787_PERF_CRZINITE_ADA is
3	3	
4	4	for Languages use ("Ada");
5	5	for Object_Dir use "..\OBJ";
6	6	for Exec_Dir use "..";
7	7	for Source_Dirs use ("..\SRC\ADA_SRC");
8	8	
9	9	for Main use ("CTP_B787_PERF_CRZINITE.ADA");
10	10	Tornado := external ("WIND_BASE");
11	11	Hi_Scoe := external ("SCOE_BASE");
12	12	Build_Use := external ("Build_Path");
13	13	Hi_Platform := "wrSbc750gx_scoe";
14	14	
15	15	package Ide is
16	16	for Compiler_Command ("ada") use "powerpc-wrs-vxworksae-gnatmake";
17	17	for Gnatlist use "powerpc-wrs-vxworksae-gnatls";
18	18	for Debugger_Command use "powerpc-wrs-vxworksae-gdb";
19	19	for Program_Host use "ISS-session";
20	20	for Communication_Protocol use "wtx";
21	21	end Ide;
22	22	
23	23	package Builder is
24	24	for Default_Switches ("ada") use ("--RTS=cert", "-j2", "-m",
25	25	"-I" & Hi_Scoe & "\platforms\" & Hi_Platform & "\include",
26	26	"-I"& Build_Use & "\CFG\CTP_B787_PERF_CRZINITE\SRC\Ada_src\StubSRC");
27	27	for Executable ("CTP_B787_PERF_CRZINITE.ada") use "CTP_B787_PERF_CRZINITE";
28	28	end Builder;
29	29	
30	30	package Binder is
31	31	for Default_Switches ("ada") use ("-E", "-t",

File: CTP_B787_PERF_CRZINITE.ZIP\CTP_B787_PERF_CRZINITE_ADA.gpr (continued)

```

32      32      "-aO" & Build_Use & "\PLATFORM\CSW\LIB",
33      33      "-aO" & Build_Use & "\PLATFORM\SCOE\LIB",
34      34      "-aO" & Build_Use & "\CFG\CTP_B787_PERF_CRZINITE\OBJ");
35      35  end Binder;
36      36
37      37  package Compiler is
38      38      for Default_Switches ("ada") use (
39      39      "-gdwarf-2",
40      40      "-ansi",
41      41      "-gnatf",
42      42      "-gnatn",
43      43      "-gnato",
44      44      "-fno-common",
45      45      "-mstrict-align",
46      46      "-fno-crossjumping",
47      47      "-fno-strict-aliasing",
48      48      "-fstack-check",
49      49      "-I" & Build_Use & "\CFG\CTP_B787_PERF_CRZINITE\src\Ada_src\StubSRC");
50      50  end Compiler;
51      51
52      52  package Linker is
53      53  for Default_Switches ("ada") use ("--LINK=ldppc",
54      54      "-nostdlib",
55      55      "-r",
56      56      "-d",
57      57      Hi_Scoe & "\platforms\" & Hi_Platform & "\lib\adaLCH.PPC604gnu.cert.o",
58      58      Hi_Scoe & "\platforms\" & Hi_Platform & "\lib\tftp.PPC604gnu.cert.o",
59      59
60      60      "-L" & Build_Use & "\LIB",
61      61      "-L" & Build_Use & "\LIB\IO",
62      62      "-L" & Build_Use & "\LIB\Bsvc",
63      63      "-L" & Build_Use & "\LIB\COM",
64      64      "-L" & Build_Use & "\LIB\FM",
65      65      "-L..\..\..\PLATFORM\CSW\LIB",
66      66      "--start-group",
67      67      "-l_bite_c_fmf",
68      68      "-l_bsvc_c_fmf",
69      69      "-l_ci_c",
70      70      "-l_dbam_c",
71      71      "-l_flxcore_c_fmf",
72      72      "-l_flxprj_c_fmf",
73      73      "-l_fpcore_c",

```

File: CTP_B787_PERF_CRZINITE.ZIP\CTP_B787_PERF_CRZINITE_ADA.gpr (continued)

```

74      74      "-l_fpprj_c",
75      75      "-l_hmi_c",
76      76      "-l_io_c_fmf",
77      77      "-l_ltc core_c",
78      78      "-l_psvc_c",
79      79      "--l_io",
80      80      "-l_io_tmf",
81      81      "-l_io_nav",
82      82      "-l_io_fmf",
83      83      "-l_com",
84      84      "-l_bsvc",
85      85      "-l_fm",
86      86      "-lCSW_V1_0",
87      87      "--end-group");
88      88      end Linker;
89      89
90      90      package Naming is
91      91          for Specification_Suffix ("ada") use ".ada";
92      92          for Implementation_Suffix ("ada") use ".ada";
93      93          for Separate_Suffix use ".ada";
94      94      end Naming;
95      95
96      96      end CTP_B787_PERF_CRZINITE_ADA;
97      97

```

File: CTP_B787_PERF_CRZINITE.ZIP\CTP_B787_PERF_CRZINITE_DRV.ada

```

1      1      with Perf_Preds_Lfdata;
2      2      with Perf_Max_Opt_Lfdata;
3      3      with Perf_Integrators_Lfdata;
4      4      with Perf_Profile_Lfdata;
5      5      with Perf_Crz_Pkg;
6      6      with Perf_Crztalt_Lfdata;
7      7
8      8      with FMF_IO_FMF_OUT_DPKG;
9      9      with Efis_661_Ildata;
10     10     with Fmf_Dual_Partition_Ildata;
11     11     with Fmci_Event_In_Dpkg;
12     12     with Nam_Waypoint_Ildata;
13     13     with Nam_Runway_Ildata;
14     14     with Nam_Corte_Ildata;
15     15     with Nam_Navaid_Ildata;
16     16     with OPS_CDK_Page_Data_Mgr_Pkg;

```

File: CTP_B787_PERF_CRZINITE.ZIP\CTP_B787_PERF_CRZINITE_DRV.ada (continued)

```

17 17 with Options_And_Data_Pkg;
18 18 with Fmci_Memory_Page_Pkg;
19 19 with CKY_Key_Pkg;
20 20 with Fmci_Display_Pkg;
21 21 with Flx_Semaphore_Pkg;
22 22
23 23 with Ops_Aedb_Ifdata;
24 24 with FMCS_AEDB_INIT;
25 25 with Fmcs_Partition_Data_Pkg;
26 26 with OPS_Data_Retained_Pkg;
27 27
28 28 Package Body CTP_B787_PERF_CRZINITE_DRV is
29 29
30 30 procedure end_dummy is
31 31 begin
32 32 null;
33 33 end end_dummy;
34 34
35 35 procedure CTP_B787_PERF_CRZINITE_D is
36 36 begin
37 37 -- execute SUT
38 38     if ( Start_SUT = 1 ) then
39 39
40 40         if ( load_ge_config ) then
41 41
42 42             Fmcs_Partition_Data_Pkg.Ops_Engine_Manufacturer      := Fmcs_Partition_Data_Pkg.ge;
43 43             Fmcs_Partition_Data_Pkg.Ops_Minor_Airframe_Model      := Fmcs_Partition_Data_Pkg.minor_airframe_dash_8 ;
44 44             Ops_Data_Retained_Pkg.Ops_Internal_EEC_Data.BOM_L_Data := OPS_DATA_RETAINED_PKG.Engine_BOM_1;
45 45             Ops_Data_Retained_Pkg.Ops_Internal_EEC_Data.BOM_R_Data := OPS_DATA_RETAINED_PKG.Engine_BOM_1;
46 46             Ops_Data_Retained_Pkg.OPS_Internal_EEC_Data.Rating_Data := OPS_Data_Retained_Pkg.GEnx_1B64;
47 47
48 48         else
49 49
50 50             Fmcs_Partition_Data_Pkg.Ops_Engine_Manufacturer      := Fmcs_Partition_Data_Pkg.rr;
51 51             Fmcs_Partition_Data_Pkg.Ops_Minor_Airframe_Model      := Fmcs_Partition_Data_Pkg.minor_airframe_dash_9;
52 52             Ops_Data_Retained_Pkg.Ops_Internal_EEC_Data.BOM_L_Data := OPS_DATA_RETAINED_PKG.Engine_BOM_1;
53 53             Ops_Data_Retained_Pkg.Ops_Internal_EEC_Data.BOM_R_Data := OPS_DATA_RETAINED_PKG.Engine_BOM_1;
54 54             Ops_Data_Retained_Pkg.OPS_Internal_EEC_Data.Rating_Data := OPS_Data_Retained_Pkg.Trent_1000_C;
55 55
56 56         end if;
57 57
58 58     Ops_Aedb_Ifdata.Aedb_Load_Signature := "FMCSAEDB";

```

File: CTP_B787_PERF_CRZINITE.ZIP\CTP_B787_PERF_CRZINITE_DRV.ada (continued)

```

59      59
60      60          FMCS_AEDB_INIT.INITIALIZE (config_found => CTP_CFG_Found,
61      61          no_aedb_loaded => CTP_AEDB_not_Loaded,
62      62          signature_fail => CTP_Sig_Fail,
63      63          version_compat => CTP_Ver_Compat);
64      64
65      65          Perf_Preds_Lfdata.VTPlogic.Firstpass := Test_Firstpass ;
66      66          Perf_Crzalt_Lfdata.LastCrzAlt.Valid := False ;
67      67          Perf_Preds_Lfdata.Vtpfplnindex := 0 ;
68      68          Perf_Max_Opt_Lfdata.Optalt_Is_LTOA(0) := True ;
69      69          Perf_Integrators_Lfdata.TermBuf.TermArray(5).Detected := False ;
70      70          Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Steptype := Test_Steptype ;
71      71
72      72          Perf_Crz_Pkg.Crz_Predexec;
73      73
74      74          end_dummy;
75      75
76      76      end if;
77      77
78      78 end   CTP_B787_PERF_CRZINITE_D;
79      79
80      80 end CTP_B787_PERF_CRZINITE_DRV;
```

File: CTP_B787_PERF_CRZINITE.ZIP\CTP_B787_PERF_CRZINITE_DRV_ada

```

1      1 with Cfp_Perf_Step_IFtypes;
2      2 with fmcs_base_types;
3      3 with AC_Position_Types;
4      4 with Portable_Types_Pkg;
5      5
6      6 package CTP_B787_PERF_CRZINITE_DRV is
7      7
8      8     Start_SUT           :   integer;
9      9     CTP_CFG_Found      :   boolean;
10     10     CTP_AEDB_not_Loaded :   boolean;
11     11     CTP_Sig_Fail      :   boolean;
12     12     CTP_Ver_Compat   :   boolean;
13     13     load_ge_config   :   boolean;
14     14
15     15 -- Global test variables go here
16     16 --
17     17     Test_Steptype       : Cfp_Perf_Step_IFtypes.Step_Types ;
18     18     Test_Firstpass     : Boolean ;
```

File: CTP_B787_PERF_CRZINITE.ZIP\CTP_B787_PERF_CRZINITE_DRV_ada (continued)

```

19      19 Test_LGB_Search      : Portable_Types_Pkg.Integer_32;
20      20 --
21      21 procedure CTP_B787_PERF_CRZINITE_D;
22      22 procedure end_dummy;
23      23
24      24 end CTP_B787_PERF_CRZINITE_DRV;
```

File: CTP_B787_PERF_CRZINITE.ZIP\module.xml

```

1      1 <!-- edited with XMLSpy v2005 rel. 3 U (http://www.altova.com) by Jerry (Honeywell) -->
2      2 <Module xmlns="ARINC653" xmlns:xi="http://www.w3.org/2001/XInclude"
3      3         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4      4         xsi:schemaLocation="ARINC653 ./schema/VXCR_Module.xsd" Name="vxWorks">
5      5     <CoreOS>
6      6         <xi:include href="$(SCOE_BASE)/Platforms/$(PLATFORM)/coreos.xml"/>
7      7     </CoreOS>
8      8     <Applications>
9      9
10     10         <Application Name="CTP_B787_PERF_CRZINITE">
11     11             <xi:include href="CTP_B787_PERF_CRZINITE.xml"/>
12     12         </Application>
13     13
14     14     </Applications>
15     15     <SharedDataRegions>
16     16         <SharedData Name="APP-AEDB_DB">
17     17             <SharedDataDescription Size="0x005DC000" CachePolicy="COPY_BACK" DataType="DATABASE" />
18     18         </SharedData>
19     19
20     20         <SharedData Name="APP-NAV_DB">
21     21             <SharedDataDescription Size="0x02800000" CachePolicy="COPY_BACK" DataType="DATABASE" />
22     22         </SharedData>
23     23
24     24         <SharedData Name="APP-FMS_AMI">
25     25             <SharedDataDescription Size="0x00001000" CachePolicy="COPY_BACK" DataType="DATABASE" />
26     26         </SharedData>
27     27
28     28         <SharedData Name="APP-FMS_OSS">
29     29             <SharedDataDescription Size="0x00001000" CachePolicy="COPY_BACK" DataType="DATABASE" />
30     30         </SharedData>
31     31
32     32     </SharedDataRegions>
33     33     <SharedLibraryRegions>
34     34         <SharedLibrary Name="ssl">
```

File: CTP_B787_PERF_CRZINITE.ZIP\module.xml (continued)

```

35      35      <xi:include href="$(SCOE_BASE)/Platforms/$(PLATFORM)/ssl.xml"/>
36      36      </SharedLibrary>
37      37      </SharedLibraryRegions>
38      38      <Partitions>
39      39          <Partition Id="1" Name="Test_Partition" Type="APP_PARTITION">
40      40              <PartitionDescription>
41      41                  <Application NameRef="CTP_B787_PERF_CRZINITE"/>
42      42                  <SharedDataRegion NameRef="APP-AEDB_DB" UserAccess="READ_ONLY" />
43      43                  <SharedDataRegion NameRef="APP-NAV_DB" UserAccess="READ_ONLY" />
44      44                  <SharedDataRegion NameRef="APP-FMS_AMI" UserAccess="READ_ONLY" />
45      45                  <SharedDataRegion NameRef="APP-FMS_OSS" UserAccess="READ_ONLY" />
46      46                  <SharedLibraryRegion NameRef="ssl"/>
47      47                  <Settings
48      48                      allocDisable="true" appsIdleRelinquishEnabled="0"
49      49                      appsPriority="-1" isrStackSize="0xffffffff"
50      50                      maxEventQStallDuration="INFINITE_TIME" maxGlobalFDs="10"
51      51                      numDrivers="0xffffffff" numFiles="0xffffffff"
52      52                      numLogMsgs="0xffffffff" numStackGuardPages="0xffffffff"
53      53                      numWorkerTasks="0" PartitionHMTTable="DefaultPartitionHM"
54      54                      selSvrQSize="0xffffffff" syscallPermissions="0x0007cfff"
55      55                      watchDogDuration="0" RequiredMemorySize="0x04000000"
56      56                      fpExcEnable="1" />
57      57              </PartitionDescription>
58      58          </Partition>
59      59      </Partitions>
60      60      <Schedules>
61      61          <Schedule Id="0" Name="schedule0" MajorFrame="0.0500" MinorFrame="0.000250">
62      62              <PartitionWindow PartitionNameRef="Test_Partition" Duration="0.030" />
63      63              <PartitionWindow PartitionNameRef="SPARE" Duration="0.020" />
64      64          </Schedule>
65      65      </Schedules>
66      66      <Connections>
67      67      </Connections>
68      68      <!-- Health Monitor Settings -->
69      69
70      70      <xi:include href="$(SCOE_BASE)/Platforms/$(PLATFORM)/HealthMonitorConfig.xml" />
71      71
72      72      <xi:include href="$(SCOE_BASE)/Platforms/$(PLATFORM)/ace.xml" />
73      73
74      74      </Module>
75      75
76      76

```

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_CREATE_POINT_SEP.ADA

```

1      1  --
2      2  -- DATA_RIGHTS: HONEYWELL CONFIDENTIAL & PROPRIETARY
3      3  --
4      4  -- THIS WORK CONTAINS VALUABLE CONFIDENTIAL AND PROPRIETARY
5      5  -- INFORMATION. DISCLOSURE, USE OR REPRODUCTION OUTSIDE OF
6      6  -- HONEYWELL INTERNATIONAL, INC. IS PROHIBITED EXCEPT AS
7      7  -- AUTHORIZED IN WRITING. THIS UNPUBLISHED WORK IS PROTECTED BY
8      8  -- THE LAWS OF THE UNITED STATES AND OTHER COUNTRIES. IN THE
9      9  -- EVENT OF PUBLICATION, THE FOLLOWING NOTICE SHALL APPLY:
10     10  -- COPR. 2007 HONEYWELL INTERNATIONAL, INC. ALL RIGHTS RESERVED.
11     11  --
12     12 with Perf_Point_Termination_Types; use Perf_Point_Termination_Types;
13     13 with standard_angle_pkg;          use standard_angle_pkg;
14     14 with fpp_interface_type;          use fpp_interface_type;
15     15 with Fmcs_Base_Types;             use Fmcs_Base_Types;
16     16 with Perf_Profile_Lfdata;         use Perf_Profile_Lfdata;
17     17 with Dst_Brg_Uilities_Pkg;
18     18
19     19 separate ( Perf_Lgb_Pkg )
20     20
21     21 procedure Create_Point
22     22 (
23     23     Event : in Perf_Point_Termination_Types.Termination_Type;
24     24     Display_Suppressed : boolean := false
25     25 ) is
26     26 --!
27     27 -- =====
28     28 -- PURPOSE:      Create an LGB Point Layer Object for The Current Predicted State
29     29 -- ANCHOR:       PERF_CODE_00021
30     30 -- SOURCE:      FMFSDD; PERF_SDD_00001 |
31     31 --
32     32 -- DESCRIPTION: This procedure creates a Core Flight Planning point layer object record for the
33     33 -- current predicted aircraft state and stores it for later output by
34     34 -- OUTPUT_PREDS.
35     35 --
36     36 -- SPECIAL_CONSIDERATIONS:
37     37 -- None.
38     38 --
39     39 -- REVISION_HISTORY (787):
40     40 -- Date      SCR      Engineer
41     41 -- 12/07/05  519.00  Pat Caulfield
42     42 -- Initial creation.

```


File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_CREATE_POINT_SEP.ADA (continued)

```

43      43  --
44      44  --   01/19/06   787.01   Pat Caulfield
45      45  --                               Updated based on point record changes; added lat/lon output.
46      46  --
47      47  --   06/04/07  1359.20   Pat Caulfield
48      48  --                               Modified to set new segment_data fields and calculate position.
49      49  --
50      50  --   07/20/07  4003.00   Ravish
51      51  --                               Added Logic for storing Latitude and Longitude of Top Of Descent
52      52  --                               for outputting to IO
53      53  --
54      54  --   04/28/08  6951.00   Added setting of Display_Suppressed.
55      55  --
56      56  -- =====
57      57  --!
58      58
59      59
60      60      Point_Data : Fpp_Interface_Type.Point_Type := Fpp_Interface_Type.Init_Point_Type;
61      61
62      62
63      63      procedure Calculate_Position (Point : in out Fpp_Interface_Type.Point_Type;
64      64                                     Minileg : in Perf_Lgb_Minileg_Types.Minileg_Rec_Type) is
65      65
66      66          Turn_Direction : constant array (Fpp_Interface_Type.Turn_Direction_Type) of
67      67          Fmcs_Base_Types.Turn_Direction_Type :=
68      68              ( Fpp_Interface_Type.Noturn => Fmcs_Base_Types.No_Turn,
69      69                Fpp_Interface_Type.Right => Fmcs_Base_Types.Right_Turn,
70      70                Fpp_Interface_Type.Left  => Fmcs_Base_Types.Left_Turn,
71      71                Fpp_Interface_Type.Either => Fmcs_Base_Types.Either);
72      72
73      73          -- define PI
74      74          PI : constant Portable_Types_Pkg.Float_32 := 3.14159265359;
75      75
76      76          -- Variables for Sodano calls
77      77          Dummy_Distance : Portable_Types_Pkg.Float_32;
78      78          Bearing        : Standard_Angle_Pkg.SAF_32;
79      79          Temp_Bearing    : Standard_Angle_Pkg.SAF_32;
80      80
81      81          -- Bearing from end to start of straight segments
82      82          Reverse_Bearing : Standard_Angle_Pkg.SAF_32;
83      83
84      84          Prof_Point_To_Seg_End_Dist : Portable_Types_Pkg.Float_32;

```

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_CREATE_POINT_SEP.ADA (continued)

```

85      85
86      86      --required for arc calculations
87      87      Temp_Angle      : Standard_Angle_Pkg.SAF_32;
88      88
89      89      begin
90      90      Prof_Point_To_Seg_End_Dist := point.aircraft_state.distance_to_destination - Minileg.Common_data.fixdistodest;
91      91      -- Lower limit Prof_Point_To_Seg_End_Dist to 0.0
92      92      if Prof_Point_To_Seg_End_Dist <= 0.0 then
93      93          Prof_Point_To_Seg_End_Dist := 0.0;
94      94      end if;
95      95
96      96      if (Minileg.Segment_Data.Segment_Path = Fpp_Interface_Type.Arc) then
97      97
98      98          -- Compute the bearing from turn center to segment end.
99      99          Dst_Brg_Uilities_Pkg.Sodanoinv
100     100          ( Lat1 => Minileg.Segment_Data.Segment_Arc_Center.Lat,
101     101              Lon1 => Minileg.Segment_Data.Segment_Arc_Center.Lon,
102     102              Lat2 => Minileg.Efis_Data.Tolatlon.Lat,
103     103              Lon2 => Minileg.Efis_Data.Tolatlon.Lon,
104     104              Dist => Dummy_Distance,
105     105              Brg12 => Bearing,
106     106              Brg21 => Temp_Bearing );
107     107
108     108      -- Calculate and normalize the course change
109     109      Course_Change := Standard_Angle_Pkg.SAF_32 (
110     110          ( Prof_Point_To_Seg_End_Dist / Minileg.Segment_Data.Segment_Arc_Radius ) * 180.0 / PI );
111     111      -- Limit course change; anything at or over 360 or at or under -360 is zero.
112     112      if (Course_Change >= 360.0) or else (Course_Change <= -360.0) then
113     113          Course_Change := 0.0;
114     114      end if;
115     115
116     116      if (Turn_Direction(Minileg.Segment_Data.Segment_Turn_Direction) = Fmcs_Base_Types.Right_Turn) then
117     117          Temp_Angle := Standard_Angle_Pkg.Normalize( Bearing - Course_Change );
118     118      else
119     119          Temp_Angle := Standard_Angle_Pkg.Normalize( Bearing + Course_Change );
120     120      end if;
121     121
122     122      -- Compute the coordinates of the point.
123     123      Dst_Brg_Uilities_Pkg.Sodanodir
124     124          ( Lat1 => Minileg.Segment_Data.Segment_Arc_Center.Lat,
125     125              Lon1 => Minileg.Segment_Data.Segment_Arc_Center.Lon,
126     126              Brg12 => Temp_Angle,

```

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_CREATE_POINT_SEP.ADA (continued)

```

127      127      Dist => abs (Minileg.Segment_Data.Segment_Arc_Radius),
128      128      Lat2 => Point.Aircraft_State.Position.Lat,
129      129      Lon2 => Point.Aircraft_State.Position.Lon,
130      130      Brg21 => Temp_Bearing );
131      131
132      132      Point.Point_Ac_State_Position_Valid := true;
133      133
134      134      elsif (Minileg.Segment_Data.Segment_Path = Fpp_Interface_Type.Straight) then
135      135
136      136          if Minileg.Efis_Data.Magnetic_North_Bearing then
137      137              Reverse_Bearing := Standard_Angle_Pkg.Normalize( 180.0 + Minileg.Efis_Data.Outcourse +
138      138                  Minileg.Efis_Data.Magvar );
139      139          else
140      140              Reverse_Bearing := Standard_Angle_Pkg.Normalize
141      141                  ( 180.0 + Minileg.Efis_Data.Incourse );
142      142          end if;
143      143
144      144          -- Compute the coordinates of the point.
145      145          Dst_Brg_Uutilities_Pkg.Sodanodir
146      146              ( Lat1 => Minileg.Efis_Data.ToLatLon.Lat,
147      147                  Lon1 => Minileg.Efis_Data.ToLatLon.Lon,
148      148                  Brg12 => Reverse_Bearing,
149      149                  Dist => Prof_Point_To_Seg_End_Dist,
150      150                  Lat2 => Point.Aircraft_State.Position.Lat,
151      151                  Lon2 => Point.Aircraft_State.Position.Lon,
152      152                  Brg21 => Temp_Bearing );
153      153
154      154          Point.Point_Ac_State_Position_Valid := true;
155      155
156      156          else
157      157              Point.Point_Ac_State_Position_Valid := false;
158      158          end if;
159      159
160      160      end Calculate_Position;
161      161
162      162      begin
163      163          if not Perf_Preds_Lfdata.Vtplogic.Hold_Multiple_Laps then
164      164              -- Populate a point record with current trajectory predictions data.
165      165
166      166              Point_Data.Aircraft_State.Predictions_Stable := not Perf_Preds_Lfdata.Vtplogic.Firstpass;
167      167              Point_Data.Aircraft_State.Cas := Perf_Integrators_Lfdata.Intprogbuf.Cas2;
168      168              Point_Data.Aircraft_State.Tas := Perf_Integrators_Lfdata.Intprogbuf.Tas2;

```

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_CREATE_POINT_SEP.ADA (continued)

```

169 169 Point_Data.Aircraft_State.Mach := Perf_Integrators_Lfdata.Intprogbuf.Mach2;
170 170 Point_Data.Aircraft_State.Wind := (Direction => Perf_Wind_Lfdata.Predwind.Dir, --
171 171 Speed => Perf_Wind_Lfdata.Predwind.Mag);
172 172 Point_Data.Aircraft_State.Isa_Deviation := Perf_Preds_Lfdata.Isadelta;
173 173 Point_Data.Aircraft_State.Flight_Phase := Perf_Preds_Lfdata.Desiredphase;
174 174 Point_Data.Aircraft_State.Ete := Perf_Integrators_Lfdata.Intprogbuf.Tprog;
175 175 Point_Data.Aircraft_State.Gross_Weight := Perf_Integrators_Lfdata.Intprogbuf.Gwprog;
176 176 Point_Data.Aircraft_State.Groundspeed := Perf_Integrators_Lfdata.Intprogbuf.Gndspd2;
177 177 Point_Data.Aircraft_State.Fuel_Weight :=
178 178 Perf_Integrators_Lfdata.Intprogbuf.Gwprog - Perf_Preds_Lfdata.Aircraft_State.Zfw.Data;
179 179 Point_Data.Aircraft_State.Distance_To_Destination := Perf_Integrators_Lfdata.Intprogbuf.Xprog;
180 180 Point_Data.Aircraft_State.Pressure_Altitude := Perf_Integrators_Lfdata.Intprogbuf.Hprog;
181 181 Point_Data.Aircraft_State.Pressure_Altitude_Rate := Perf_Integrators_Lfdata.Intprogbuf.Rateofclb2;
182 182 Point_Data.Aircraft_State.True_Track := Perf_Preds_Lfdata.Gsdata.Fplntrack;
183 183 Point_Data.Aircraft_State.Fuel_Flow := Perf_Integrators_Lfdata.Intprogbuf.Fuelflow2;
184 184 Point_Data.Aircraft_State.Gamma_Airmass := Perf_Integrators_Lfdata.Intprogbuf.Gamaair2;
185 185 Point_Data.Aircraft_State.Acceleration := Perf_Integrators_Lfdata.Intprogbuf.Accel2;
186 186
187 187 -- set the display suppressed flag true if:
188 188 -- 1. We're predicting an early descent.
189 189 -- 2. The display_suppressed input parameter is true.
190 190 -- 3. The predicted flight phase is descent or approach and there are no descent constraints.
191 191 Point_Data.Display_Suppressed := Perf_Preds_Lfdata.Early_Descent or else Display_Suppressed or else
192 192 (Perf_Preds_Lfdata.DesiredPhase >= Descent and then Perf_LGB_Lfdata.Last_Constraint_Index = 0);
193 193
194 194 Point_Data.Event := Event;
195 195 Point_Data.Priority := 1;
196 196
197 197 -- Output segment data to the point for VSD and Separation Assurance to use.
198 198 if Perf_Preds_Lfdata.Navptr in 1..Perf_Lgb_Minileg_Types.Max_Number_Minilegs then
199 199 Point_Data.Segment_Index := Perf_Lgb_Lfdata.Lgb(Perf_Preds_Lfdata.Navptr).Segment_Data.Segment_Index;
200 200 Point_Data.Segment_Arc_Center := Perf_Lgb_Lfdata.Lgb(Perf_Preds_Lfdata.Navptr).Segment_Data.Segment_Arc_Center;
201 201 Point_Data.Segment_Arc_Radius := Perf_Lgb_Lfdata.Lgb(Perf_Preds_Lfdata.Navptr).Segment_Data.Segment_Arc_Radius;
202 202 Point_Data.Segment_Path := Perf_Lgb_Lfdata.Lgb(Perf_Preds_Lfdata.Navptr).Segment_Data.Segment_Path;
203 203 Point_Data.Segment_Discon_Follows := Perf_Lgb_Lfdata.Lgb(Perf_Preds_Lfdata.Navptr).Segment_Data.Segment_Discon_Follows;
204 204 Point_Data.Segment_Turn_Direction := Perf_Lgb_Lfdata.Lgb(Perf_Preds_Lfdata.Navptr).Segment_Data.Segment_Turn_Direction;
205 205 Point_Data.Segment_Not_Computed_Trajectory :=
206 206 » Perf_Lgb_Lfdata.Lgb(Perf_Preds_Lfdata.Navptr).Segment_Data.Segment_Not_Computed_Trajectory;
207 207 end if;
208 208 -- if the termination is a waypoint or segment endpoint, fill in the position with the segment's
209 209 -- endpoint position from the current mini-leg.

```

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_CREATE_POINT_SEP.ADA (continued)

```

210 210
211 211     if (Event = Waypoint) or else (Event = Segment_Endpoint) or else (Event = Constrained_Des_Waypoint) then
212 212         Point_Data.Aircraft_State.Position.Lat := Perf_Lgb_Lfdata.Lgb(Perf_Preds_Lfdata.Navptr).Efis_Data.Tolatlon.Lat;
213 213         Point_Data.Aircraft_State.Position.Lon := Perf_Lgb_Lfdata.Lgb(Perf_Preds_Lfdata.Navptr).Efis_Data.Tolatlon.Lon;
214 214         Point_Data.Point_AC_State_Position_Valid := true;
215 215
216 216     elsif (Event = Start_Of_Predictions) then
217 217         Point_Data.Aircraft_State.Position.Lat := Perf_Preds_Lfdata.Aircraft_State.Lat_lon.Data.Lat;
218 218         Point_Data.Aircraft_State.Position.Lon := Perf_Preds_Lfdata.Aircraft_State.Lat_lon.Data.Lon;
219 219         Point_Data.Point_AC_State_Position_Valid := Perf_Preds_Lfdata.Aircraft_State.Lat_lon.Valid;
220 220
221 221         -- set the flight phase to the current aircraft phase.
222 222         Point_Data.Aircraft_State.Flight_Phase := Perf_Preds_Lfdata.Fltphase;
223 223
224 224     elsif not Point_Data.Point_AC_State_Position_Valid then
225 225
226 226         if Perf_Preds_Lfdata.Navptr in 1..Perf_Lgb_Minileg_Types.Max_Number_Minilegs then
227 227
228 228             -- need to calculate the point location directly
229 229             -- using distance to destination and segment data.
230 230             Calculate_Position (point_data, perf_lgb_lfdata.lgb(perf_preds_lfdata.navptr));
231 231         end if;
232 232         -- if the event is Top of Descent store Lat/Lon of the Position
233 233         -- to output it to IO.
234 234         -- SCR 4003.00
235 235         if (Event= Top_Of_Descent) and then
236 236             (Perf_LGB_Lfdata.Last_Constraint_Index >0) then
237 237
238 238             Perf_Profile_Lfdata.Todperfddata.Position.Lat := Point_Data.Aircraft_State.Position.Lat;
239 239             Perf_Profile_Lfdata.Todperfddata.Position.Lon := Point_Data.Aircraft_State.Position.Lon;
240 240             Perf_Profile_Lfdata.Todperfddata.Position_Valid := true;
241 241
242 242         end if;
243 243
244 244     end if;
245 245
246 246     -- Determine if the predicted speed should be tagged as a mach or a CAS value.
247 247
248 248     if Perf_Preds_Lfdata.Tgtspdrec.Tgtspdtag = Casonly or else ( Perf_Preds_Lfdata.Tgtspdrec.Tgtspdtag = Casmach and then
249 249         ( Point_Data.Aircraft_State.Pressure_Altitude < Perf_Preds_Lfdata.Tgtspdrec.Cmxalt ) ) then
250 250
251 251         Point_Data.Aircraft_State.Speed_Command := Perf_Preds_Lfdata.Tgtspdrec.Cas;

```

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_CREATE_POINT_SEP.ADA (continued)

```

252 252     else
253 253         Point_Data.Aircraft_State.Speed_Command := Perf_Preds_Lfdata.Tgtspdrec.Mach;
254 254     end if;
255 255
256 256     -- Copy the predictions data sequence counter from the snapshot of the flight plan header
257 257     -- currently being used by predictions into the point record. The header and point record
258 258     -- counters matching indicates the predicted data in general is valid.
259 259
260 260     Point_Data.Aircraft_State.Predictions_Data_Seq_Counter := Perf_Lgb_Lfdata.Lgb_Header.Prddataseq;
261 261
262 262     -- If this is the initial condition point, start at one. If not, increment the point count.
263 263
264 264     if Event = Start_Of_Predictions then
265 265         Point_Count := 1;
266 266     else
267 267         if Point_Count = Perf_Lgb_Minileg_Types.Point_Index_Type'Last then
268 268             Point_Count := 1;
269 269         else
270 270             Point_Count := Point_Count + 1;
271 271         end if;
272 272     end if;
273 273
274 274     -- Store the point record within this package; it will be output when we're done predicting the route.
275 275
276 276     Points( Point_Count ) := Point_Data;
277 277
278 278     end if;
279 279 end Create_Point;
```

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_OUTPUT_PREDS_SEP.ADA

```

1 1 --|
2 2 --| DATA_RIGHTS: HONEYWELL CONFIDENTIAL & PROPRIETARY
3 3 --|
4 4 --| THIS WORK CONTAINS VALUABLE CONFIDENTIAL AND PROPRIETARY
5 5 --| INFORMATION. DISCLOSURE, USE OR REPRODUCTION OUTSIDE OF
6 6 --| HONEYWELL INTERNATIONAL, INC. IS PROHIBITED EXCEPT AS
7 7 --| AUTHORIZED IN WRITING. THIS UNPUBLISHED WORK IS PROTECTED BY
8 8 --| THE LAWS OF THE UNITED STATES AND OTHER COUNTRIES. IN THE
9 9 --| EVENT OF PUBLICATION, THE FOLLOWING NOTICE SHALL APPLY:
10 10 --| COPR. 2007 HONEYWELL INTERNATIONAL, INC. ALL RIGHTS RESERVED.
11 11 --|
12 12 -- types
```

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_OUTPUT_PREDS_SEP.ADA (continued)

13	13	with Apex_Types; use Apex_Types;
14	14	with Cdk_Offpath_Iftypes;
15	15	with Cfp_Perf_Step_Iftypes; use Cfp_Perf_Step_Iftypes;
16	16	with Fix_Info_Iftypes;
17	17	with Perf_Offpath_Des_Lftypes;
18	18	with Scratch_Pad_Iftypes;
19	19	with Standard_Angle_Pkg; use Standard_Angle_Pkg;
20	20	-- global data objects
21	21	with Fix_Info_Ildata;
22	22	with Idx_Profile_Ildata;
23	23	-- perf data objects
24	24	with Perf_Crzalt_Lldata;
25	25	with Perf_Idx_Crzalt_Lldata;
26	26	with Perf_Idx_Msg_Flags_Lldata;
27	27	with Perf_Idx_Top_Of_Des_Lldata;
28	28	with Perf_Lgb_Lldata;
29	29	with Perf_Msg_Flags_Lldata;
30	30	with Perf_Offpath_Des_Lldata;
31	31	with Perf_Offpath_Descent_Ildata;
32	32	with Perf_Profile_Lldata;
33	33	with Perf_Rta_Lldata;
34	34	with Perf_Task_Control_Lldata;
35	35	with Perf_Top_Of_Des_Lldata;
36	36	with Perf_Vdu_Lldata;
37	37	with Perf_Vtp_Lldata;
38	38	-- global packages
39	39	with Apex_Processes;
40	40	with Dst_Brg_Uilities_Pkg;
41	41	with Fmf_IO_Fmf_Out_Dpkg;
42	42	with Fmci_Spad_Manager_Pkg;
43	43	with Ops_Cdk_Common_Mgr_Pkg;
44	44	with Ops_Cdk_Perf_Pdb_Mgr_Pkg;
45	45	with Ops_Perf_Change_Flags_Mgr_Pkg;
46	46	with Ops_Timer_Pkg;
47	47	-- perf packages
48	48	with Perf_Ads_Intent_Pkg;
	49	with Perf_Atc_Cond_Pkg;
49	50	with Perf_Efis_Lgb_Mgr_Pkg;
50	51	with Perf_Opd_Pkg;
51	52	with Perf_Vdu_Utils;
	53	with Perf_WTS_Lldata;
	54	with Ops_Perf_Rta_Data_Mgr_Pkg;

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_OUTPUT_PREDS_SEP.ADA (continued)

```

52      55
53      56 separate ( Perf_Lgb_Pkg )
54      57
55      58 procedure Output_Preds is
56      59 --!
57      60 -- =====
58      61 -- PURPOSE:      PERF-EFIS Lateral Guidance Buffer Manager Package
59      62 -- ANCHOR:       PERF_CODE_00020
60      63 -- SOURCE:      FMFSDD; PERF_SDD_00002 |
61      64 --
62      65 -- DESCRIPTION: This procedure copies the predicted flight plan data out to the PERF_EFIS_LGB_MGR_PKG
63      66 --                  where it will be output by EFIS PATH to the LGB.
64      67 --
65      68 -- SPECIAL_CONSIDERATIONS:
66      69 --   None.
67      70 --
68      71 -- REVISION_HISTORY (787):
69      72 --   Date      SCR      Engineer
70      73 --   12/07/05   519.00   Pat Caulfield
71      74 --                  Initial creation.
72      75 --
73      76 --   01/18/06   787.01   Pat Caulfield
74      77 --   Minor adjustments to point output.
75      78 --
76      79 --   02/08/06   865.00   Pat Caulfield
77      80 --   Switched to use Act_Prov_Index for Perf_Efis_Lgb_Mgr_Pkg accesses due to interface change.
78      81 --
79      82 --   02/02/07  2698.03   Keri Kalvelage
80      83 --   To fix SBC compiler warnings, removed declaratons for loop counters
81      84 --   Mini_Index and Point_Index.
82      85 --
83      86 --   06/01/07  1359.20   Pat Caulfield
84      87 --   Only output points for the active route.
85      88 --
86      89 --   07/18/07  3922.00   Pat Caulfield
87      90 --   I made changes for this SCR in here but then backed them out.
88      91 --
89      92 --   02/27/08  6677.00   Pat Caulfield
90      93 --   Check for the minileg fpln counts to match before outputing predictions.  Added preemption locking
91      94 --   and unlocking.  Moved the other predictions output here from the end of vtp_exec so that it would
92      95 --   be within this check.
93      96 --

```


File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_OUTPUT_PREDS_SEP.ADA (continued)

94	97	-- 03/11/08 5839.00 Pat Caulfield
95	98	-- Output the insufficient fuel messages (Help window and Eicas) based on the message flags.
96	99	-- This used to be done in DST_ESTIMATES.
97	100	--
98	101	-- =====
99	102	--!
100	103	
101	104	New_Lock_Level : Apex_Processes.Lock_Level_Type;
102	105	Lock_Status : Apex_Types.Return_Code_Type;
103	106	Stepclbrec : Cfp_Perf_Step_Iftypes.Step_Clb_Rec_Type;
104	107	
105	108	begin
106		lock_preemption so that we can get this out quickly.
	109	--VDU dump for window pass using PERF copy of LGB buffer.
	110	if (Perf_Preds_Lfdata.Dump_Window_Preds /= 0) then
	111	Perf_Vdu_Utils.Dump_Window_predictions;
	112	end if;
107	113	
108		Apex_Processes.Lock_Preemption (New_Lock_Level, Lock_Status);
	114	
	115	if (Perf_Rta_Lfdata.Rta_Window_Task) then
	116	
	117	if(Perf_Preds_Lfdata.Perf_Pass = Perf_preds_lftypes.Early) then
	118	» Ops_Perf_Rta_Data_Mgr_Pkg.Put_ETA_Earliest_Time(Perf_Rta_Lfdata.Rta_Windows(Perf_Preds_Lfdata.Vtpfplnindex).Min_Time,Perf_Pred
		» s_Lfdata.Vtpfplnindex);
	119	
	120	else
	121	
		» Ops_Perf_Rta_Data_Mgr_Pkg.Put_ETA_latest_Time(Perf_Rta_Lfdata.Rta_Windows(Perf_Preds_Lfdata.Vtpfplnindex).Max_Time,Perf_Preds_
		» Lfdata.Vtpfplnindex);
	122	
	123	end if;
	124	
	125	else
	126	-- lock preemption so that we can get this out quickly.
	127	Apex_Processes.Lock_Preemption (New_Lock_Level, Lock_Status);
109	128	
110		if Perf_Lgb_Lfdata.Minileg_Fpln_Count_Snapshot = Perf_Efis_Lgb_Mgr_Pkg.Minileg_Fpln_Count (Perf_Lgb_Lfdata.Act_Prov_Index)-
		» then
	129	if Perf_Lgb_Lfdata.Minileg_Fpln_Count_Snapshot = Perf_Efis_Lgb_Mgr_Pkg.Minileg_Fpln_Count (Perf_Lgb_Lfdata.Act_Prov_Index)
		» then

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_OUTPUT_PREDS_SEP.ADA (continued)

```

111 130
112 131     Perf_Lgb_Lfdata.Pass_Thrown_Out := false;
113 132
114 133     -- transfer the minileg flight plan from our copy in Perf_Lgb_Lfdata to the Perf_Efis_Lgb_Mgr_Pkg.
115 134
116 135     for Mini_Index in 1..Perf_Lgb_Lfdata.Lgb_Header.Lastfplnptr loop
117 136         Perf_Efis_Lgb_Mgr_Pkg.Put_Minileg ( Perf_Lgb_Lfdata.Act_Prov_Index, Perf_Lgb_Lfdata.Lgb( Mini_Index ), Mini_Index );
118 137     end loop;
119 138
120 139     -- Transfer the point layer data that was stored internally to this package by Create_Point calls to the
121 140     -- Perf_Efis_Lgb_Mgr_Pkg.
122 141
123 142     if Perf_Preds_Lfdata.Vtplogic.Haveactfpln then -- only for active flight plan
124 143         for Point_Index in 1..Point_Count loop
125 144             Perf_Efis_Lgb_Mgr_Pkg.Put_Point ( Points( Point_Index ), ( Point_Index = 1 ) );
126 145         end loop;
127 146     end if; -- active flight plan
128 147
129 148     -- Assuming we have a flight plan, signal EFIS PATH that we're done predicting either
130 149     -- the active or provisional route, and have output predictions to the PERF_EFIS_LGB_MGR_PKG
131 150     -- (done up above via the output_preds call) for them to store into the LGB.
132 151
133 152     if Perf_Preds_Lfdata.Vtplogic.Haveactfpln then
134 153         Ops_Perf_Change_Flags_Mgr_Pkg.Put_Perf_Efis_Do_Act( True );
135 154     else
136 155         Ops_Perf_Change_Flags_Mgr_Pkg.Put_Perf_Efis_Do_Prov( True );
137 156     end if;
138 157
139 158     -- Un-lock preemption to free up processing
140 159
141 160     Apex_Processes.Unlock_Preemption ( New_Lock_Level, Lock_Status );
142 161
143 162     -- Clear the EICAS message when we're predicting the active fpln, and
144 163     -- the most recent evaluation (up above) doesn't show insufficient fuel,
145 164     -- but the message is displayed. Clear the help window when the same is
146 165     -- true for the provisional flight plan.
147 166
148 167     if Perf_Preds_Lfdata.Vtplogic.Haveactfpln then
149 168
150 169         -- if the insufficient fuel Eicas message needs to be displayed, do so.
151 170         if Perf_Msg_Flags_Lfdata.Insufficient_Fuel_Eicas and then
152 171             not Perf_Idx_Msg_Flags_Lfdata.Iinsufffuel_Eicas then

```

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_OUTPUT_PREDS_SEP.ADA (continued)

153	172	
154	173	Fmf_Io_Fmf_Out_Dpkg.Insufficient_Fuel_Discr.Put (Data => True, Is_Valid => True);
155	174	end if;
156	175	
157	176	-- if the insufficient fuel message Eicas message needs to be cleared, do so.
158	177	if not Perf_Msg_Flags_Lfdata.Insufficient_Fuel_Eicas and then
159	178	Perf_Idx_Msg_Flags_Lfdata.Iinsuffuel_Eicas then
160	179	
161	180	Fmf_Io_Fmf_Out_Dpkg.Insufficient_Fuel_Discr.Put (Data => False, Is_Valid => True);
162	181	end if;
163	182	else -- provisional fpln predictions
164	183	
165	184	-- if the insufficient fuel Help Window message needs to be displayed, do so.
166	185	if Perf_Msg_Flags_Lfdata.Insufficient_Fuel_Help and then
167	186	not Perf_Idx_Msg_Flags_Lfdata.Iinsuffuel_Help then
168	187	
169	188	Fmci_Spad_Manager_Pkg.Display_Message (Message_Id => Scratch_Pad_Iftypes.Insufficient_Fuel);
170	189	end if;
171	190	
172	191	-- if the insufficient fuel Help Window message needs to be cleared, do so.
173	192	if not Perf_Msg_Flags_Lfdata.Insufficient_Fuel_Help and then
174	193	Perf_Idx_Msg_Flags_Lfdata.Iinsuffuel_Help then
175	194	
176	195	Fmci_Spad_Manager_Pkg.Clear_Message (Message_Id => Scratch_Pad_Iftypes.Insufficient_Fuel);
177	196	end if;
178	197	end if;
179	198	
	199	-- For active flight plan, perform ATC conditional clearance processing.
	200	if Perf_Preds_Lfdata.Vtplogic.Haveactfpln then
	201	Perf_Atc_Cond_Pkg.Atc_Cond_Clearance_Exec;
	202	end if;
	203	
180	204	-- mark the first-pass flag false
181	205	
182	206	Perf_Task_Control_Lfdata.Idofirstpass(Perf_Preds_Lfdata.Vtpfplnindex) := False;
183	207	
184	208	-- Save local Perf data items into interfunctional IDX data packages
185	209	
186	210	Idx_Profile_Ildata.Itocperfdata(Perf_Preds_Lfdata.Vtpfplnindex) := Perf_Profile_Lfdata.Tocperfdata;
187	211	Idx_Profile_Ildata.Itodperfdata(Perf_Preds_Lfdata.Vtpfplnindex) := Perf_Profile_Lfdata.Todperfdata;
188	212	
189		Perf_Idx_Top_Of_Des_Lfdata.Itoderzspeed(Perf_Preds_Lfdata.Vtpfplnindex) := Perf_Top_Of_Des_Lfdata.Toderzspeed;

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_OUTPUT_PREDS_SEP.ADA (continued)

190	213	
191	214	Idx_Profile_Ildata.Idestinrec(Perf_Preds_Lldata.Vtpfplnindex) := Perf_Profile_Lldata.Destinrec;
192	215	Idx_Profile_Ildata.Ialtprofptrec(Perf_Preds_Lldata.Vtpfplnindex) := Perf_Profile_Lldata.Altprofptrec;
193	216	Idx_Profile_Ildata.Ilvlaltaltrec(Perf_Preds_Lldata.Vtpfplnindex) := Perf_Profile_Lldata.Lvlaltaltrec;
194	217	Idx_Profile_Ildata.Ieodperpdata(Perf_Preds_Lldata.Vtpfplnindex) := Perf_Profile_Lldata.Eodperpdata;
195	218	
196	219	Perf_Idc_Crzalt_Lldata.Ilastcrzalt(Perf_Preds_Lldata.Vtpfplnindex) := Perf_Crzalt_Lldata.Lastcrzalt;
197	220	Perf_Idc_Msg_Flags_Lldata.Iunablecrzalt(Perf_Preds_Lldata.Vtpfplnindex) := Perf_Msg_Flags_Lldata.Unablecrzalt;
198	221	Perf_Idc_Msg_Flags_Lldata.Iinsuffuel_Help := Perf_Msg_Flags_Lldata.Insufficient_Fuel_Help;
199	222	Perf_Idc_Msg_Flags_Lldata.Iinsuffuel_Eicas := Perf_Msg_Flags_Lldata.Insufficient_Fuel_Eicas;
200	223	Perf_Idc_Msg_Flags_Lldata.Iabovemaxalt(Perf_Preds_Lldata.Vtpfplnindex) := Perf_Msg_Flags_Lldata.Abovemaxalt;
201	224	Perf_Idc_Msg_Flags_Lldata.Imax_Alt_Msg_Latch(Perf_Preds_Lldata.Vtpfplnindex) := Perf_Msg_Flags_Lldata.Max_Alt_Msg_Latch;
202	225	Perf_Idc_Msg_Flags_Lldata.Imax_Alt_Msg_Leg_Index(Perf_Preds_Lldata.Vtpfplnindex) :=
		» Perf_Msg_Flags_Lldata.Max_Alt_Msg_Leg_Index;
	226	Perf_Idc_Top_Of_Des_Lldata.Itoddata(Perf_Preds_Lldata.Vtpfplnindex) := Perf_Top_Of_Des_Lldata.Toddata;
	227	Perf_Idc_Top_Of_Des_Lldata.Itodcrzspeed(Perf_Preds_Lldata.Vtpfplnindex) := Perf_Top_Of_Des_Lldata.Todcrzspeed;
	228	Perf_Rta_Lldata.Idc_Data(Perf_Preds_Lldata.Vtpfplnindex) := Perf_Rta_Lldata.Perf_Rta_Data;
	229	Perf_WTS_Lldata.Idc_Rta_CI(Perf_Preds_Lldata.Vtpfplnindex) := Perf_WTS_Lldata.Rta_CI;
203	230	
204	231	-- output predictions that only are done for the active flight plan
205	232	
206	233	if Perf_Preds_Lldata.Vtplogic.Haveactfpln then
207	234	-- Output fixinfo predictions if active flight plan.
208	235	
209	236	for I in Fix_Info_Iftypes.Fixinfo_Array_Type' range loop
210	237	Ops_Cdk_Common_Mgr_Pkg.Put_Fix_Info_Pred_Data(Page_Number => I,
211	238	New_Data => Fix_Info_Ildata.Fix_Info(I));
212	239	end loop;
213	240	
214	241	-- Supply efisrad with new top of climb, step climb,
215	242	-- top of descent, & end of descent records if the
216	243	-- active flight plan was processed.
217	244	
218	245	Idx_Profile_Ildata.Inew_Profpts(Perf_Preds_Lldata.Vtpfplnindex) := True;
219	246	
220	247	--Stores Latitude/Longitude of TOD to IO from IDX
221	248	--IO uses Float_64 so type conversion is needed from SAF_32 type to Float_64
222	249	
223	250	-- Lock to prevent the Float_64 from being corrupted during the write.
224	251	Apex_Processes.Lock_Preemption (New_Lock_Level, Lock_Status);
225	252	
226	253	Fmf_IO_Fmf_out_dpkg.TOD_Position_Latitude.Put(Portable_Types_Pkg.Float_64(Idx_Profile_Ildata.Itodperpdata(

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_OUTPUT_PREDS_SEP.ADA (continued)

```

227 254 » Perf_Preds_Lfdata.Vtpfplnindex ).Position.Lat),
                                         Idx_Profile_Ifdata.Itodperfddata( Perf_Preds_Lfdata.Vtpfplnindex
228 255 » ).Position_Valid);
229 256 Fmf_IO_Fmf_out_dpkg.TOD_Position_Longitude.Put(Portable_Types_Pkg.Float_64(Idx_Profile_Ifdata.Itodperfddata(
» Perf_Preds_Lfdata.Vtpfplnindex ).Position.Lon),
230 257 Idx_Profile_Ifdata.Itodperfddata( Perf_Preds_Lfdata.Vtpfplnindex
» ).Position_Valid);
231 258
232 259 -- Unlock to free it up
233 260 Apex_Processes.Unlock_Preemption ( New_Lock_Level, Lock_Status );
234 261
235 262 end if;
236 263
237 264 -- write recommended takeoff time to manager
238 265
239 266 Ops_Cdk_Perf_Pdb_Mgr_Pkg.Put_Perf_Takeoff_Time(Takeoff_Time =>
» Perf_Rta_Lfdata.Idx_Data(Perf_Preds_Lfdata.Vtpfplnindex).Rcmd_Takeoff,
240 267 Fpln_Index => Perf_Preds_Lfdata.Vtpfplnindex);
241 268
242 269 -- Fetch a copy of the step climh data record from the shared object manager.
243 270
244 271 Ops_Cdk_Perf_Pdb_Mgr_Pkg.Get_Stepclbrec( Perf_Preds_Lfdata.Vtpfplnindex, Stepclbrec );
245 272
246 273 -- if we've just sequenced a step climb point...
247 274
248 275 if ( ( Perf_Profile_Lfdata.Step_Climb_Rec.Cfp_Step_Data.Steptype = Cfp_Perf_Step_Iftypes.Pastspecstep ) and then
249 276 ( Perf_Profile_Lfdata.Step_Climb_Rec.Cfp_Step_Data.Spec_Stepalt /=
250 277 Stepclbrec.Cfp_Step_Data.Spec_Stepalt ) ) then
251 278
252 279 -- update s/c record with s/c point just sequenced
253 280
254 281 Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Data := Stepclbrec.Cfp_Step_Data.Fixdistodest;
255 282
256 283 Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Valid := False;
257 284
258 285 -- allow preds to determine if it's maxalt limited
259 286
260 287 Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Eta.Valid := False;
261 288
262 289 end if;
263 290

```

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_OUTPUT_PREDS_SEP.ADA (continued)

264	291	-- store computed s/c data
265	292	Ops_Cdk_Perf_Pdb_Mgr_Pkg.Put_Stepclbrec_Perf(Perf_Preds_Lfdata.Vtpfplnindex,
		» Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data);
266	293	Ops_Cdk_Perf_Pdb_Mgr_Pkg.Put_Stepclbrec_Perfdataval(Perf_Preds_Lfdata.Vtpfplnindex,
		» Perf_Profile_Lfdata.Step_Climb_Rec.Perfdataval);
267	294	
	295	-- Do not update the indexed Rta_Iter_Counter until the end of preds
	296	Perf_WTS_Lfdata.Rta_Iter_Counter(Perf_Preds_Lfdata.Vtpfplnindex) :=
	297	Perf_WTS_Lfdata.New_Rta_Iter_Counter(Perf_Preds_Lfdata.Vtpfplnindex);
	298	
	299	-- Save the WTS Flat_Bias_Factor to be used for the next pass
	300	-- of predictions of this flight plan
	301	Perf_WTS_Lfdata.Idx_Flat_Bias_Factor(Perf_Preds_Lfdata.Vtpfplnindex) :=
	302	Perf_WTS_Lfdata.New_Flat_Bias_Factor;
	303	
	304	-- 'Push down' the WTS and RTA info saved from previous passes of predictions
	305	-- note: We do not currently plan to use data from passes other than the most recent
	306	-- pass, however it is helpful to have this data for debugging and it used to be
	307	-- stored for Perf_WTS_Lfdata.Num_Values_Type passes so we will continue to save it
	308	-- in case it is needed for something.
	309	for Push_Index in reverse Perf_WTS_Lfdata.Num_Stored_Passes loop
	310	if Push_index < Perf_WTS_Lfdata.Num_Stored_Passes'last then
	311	Perf_WTS_Lfdata.Idx_Pass_Info_Rec (Perf_Preds_Lfdata.Vtpfplnindex , Push_Index + 1) :=
	312	Perf_WTS_Lfdata.Idx_Pass_Info_Rec (Perf_Preds_Lfdata.Vtpfplnindex , Push_Index);
	313	end if;
	314	end loop;
	315	
	316	-- Save important WTS and RTA info from this pass of predictions,
	317	-- to be used in future passes [to support the following requirements that
	318	-- need data from the prior trip prediction pass: PERF_SRD_B_00413,
	319	-- PERF_SRD_B_00414, PERF_SRD_B_00415, PERF_SRD_B_00416]
	320	Perf_WTS_Lfdata.Current_Pass_Info_Rec.Pass_Info_Valid := True;
	321	Perf_WTS_Lfdata.Idx_Pass_Info_Rec (Perf_Preds_Lfdata.Vtpfplnindex , Perf_WTS_Lfdata.Num_Values_Type'first) :=
	322	Perf_WTS_Lfdata.Current_Pass_Info_Rec;
268	323	else -- the results of this pass of predictions is being thrown out (by not storing them out)
269	324	
270	325	-- Un-lock preemption to free up processing
271	326	
272	327	Apex_Processes.Unlock_Preemption (New_Lock_Level, Lock_Status);
273	328	
274	329	Perf_Lgb_Lfdata.Pass_Thrown_Out := true;
275	330	

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_OUTPUT_PREDS_SEP.ADA (continued)

276	331	Perf_Task_Control_Lfdata.Preds_Aborting_Reason (Perf_Preds_Lfdata.Vtpfplnindex) :=
277	332	Perf_Preds_Lftypes.Pass_Thrown_Out;
278	333	-- count the number of times this happens to aid in debugging.
279	334	
280	335	if Perf_Lgb_Lfdata.Pass_Thrown_Out_Count < 10000 then
281	336	Perf_Lgb_Lfdata.Pass_Thrown_Out_Count := Perf_Lgb_Lfdata.Pass_Thrown_Out_Count + 1;
282	337	else
283	338	Perf_Lgb_Lfdata.Pass_Thrown_Out_Count := 1;
284	339	end if;
285	340	end if;
286		
	341	end if;
287	342	end Output_Preds;

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_PKG.ADA

1	1	--
2	2	-- DATA_RIGHTS: HONEYWELL CONFIDENTIAL & PROPRIETARY
3	3	-- THIS WORK CONTAINS VALUABLE CONFIDENTIAL AND PROPRIETARY
4	4	-- INFORMATION. DISCLOSURE, USE OR REPRODUCTION OUTSIDE OF
5	5	-- HONEYWELL INTERNATIONAL, INC. IS PROHIBITED EXCEPT AS
6	6	-- AUTHORIZED IN WRITING. THIS UNPUBLISHED WORK IS PROTECTED BY
7	7	-- THE LAWS OF THE UNITED STATES AND OTHER COUNTRIES. IN THE
8	8	-- EVENT OF PUBLICATION, THE FOLLOWING NOTICE SHALL APPLY:
9	9	-- COPR. 2012 HONEYWELL INTERNATIONAL, INC. ALL RIGHTS RESERVED.
10	10	--
11	11	
12	12	with Fpp_Interface_Type;
13	13	with Fpp_Wrap_Point_Pkg;
14	14	with Fmcs_Base_Types;
15		with Fmcs_Fp_Guid_Btypes;
	15	with Fmcs_Fp_Guid_Btypes; use Fmcs_Fp_Guid_Btypes;
16	16	with Perf_Lgb_Lfdata;
17	17	with Flight_Pln_Leg_Types;
18	18	with Fpp_Status_Type_Tpkg;
19	19	with Perf_Integrators_Lfdata;
20	20	with Portable_Types_Pkg;
21	21	use Portable_Types_Pkg;
22	22	with Perf_Wind_Lfdata;
23	23	with Perf_Preds_Lfdata;
	24	with Perf_Top_Of_Des_Lfdata;
24	25	with Fpp_Common_Lgb_Wrap_Pkg;
25	26	with Flight_Pln_Hdr_Types;

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_PKG.ADA (continued)

```

26      27 with Perf_Lgb_Minileg_Types;
27      28 with Perf_Preds_Lftypes;
28      29 use Perf_Preds_Lftypes;
29      30 with Standard_Angle_Pkg;
30      31 use Standard_Angle_Pkg;
31      32 use Fmcs_Base_Types;
32      33
33      34 package body Perf_LGB_Pkg is
34      35     --!
35      36     -- ANCHOR:          FMCS_19_21023511
36      37     -- SOURCE:         FMFSDD; FMCS_19_21023000 |
37      38     -- | @DESCRIPTION: This package body contains the procedure bodies
38      39     -- |                   (declared as separates) for manipulation of PERF
39      40     -- |                   predictions copy of the LGB
40      41     -- |                   (Perf_LGB_Lfdata.LGB)
41      42     --
42      43     -- SPECIAL_CONSIDERATIONS:
43      44     --
44      45     -- REVISION_HISTORY:
45      46     --   DATE          SCR #          Programmer          DRCM#
46      47     -- =====
47      48     --   12/18/95      8011          B. O'Laughlin          M777B_FMF_00548
48      49     -- Added Lgb_Seq_Rta_Leg for processing predicted RTA leg sequence.
49      50     --
50      51     -- ===== 787 HISTORY STARTS HERE =====
51      52     --
52      53     --   12/06/05      519.00        Pat Caulfield
53      54     -- Added new procedures Create_Point and Output_Preds, as well as internal
54      55     -- temporary (during flight plan predictions) storage of the points.
55      56     --
56      57     --   01/18/06      787.01        Pat Caulfield
57      58     -- Cleaned up the point array declaration now that the event field is defined
58      59     -- in the point record. Also renamed Create_Point's parameter to Event.
59      60     --
60      61     --   07/28/08      7562.00       Pat Caulfield
61      62     -- Moved Course_Change here from Lgb_Seq_Leg to aid debugging.
62      63     --
63      64     --   08/07/08      6676.00       Pat Caulfield
64      65     -- Added new parameter Display_Suppressed to Create_Point.
65      66     --
66      67     --!
67      68

```


File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_PKG.ADA (continued)

68	69	-- V A R I A B L E D E C L A R A T I O N S --
69	70	
70	71	Points : Perf_Lgb_Minileg_Types.Point_Array_Type;
71	72	
72	73	Point_Count : Perf_Lgb_Minileg_Types.Point_Index_Type;
73	74	
74	75	-- DESCRIPTION Difference between current and previous flight plan tracks
75	76	Course_Change : Standard_Angle_Pkg.Saf_32;
76	77	
	78	Hold_Idx : Boolean := False;
	79	Hold_In_Descent : Boolean := False;
	80	Valid_Idx : Boolean := False;
	81	Valid_Spd_Idx : Boolean := False;
77	82	-- P R O C E D U R E D E C L A R A T I O N S --
	83	procedure Get_Num_Points (Num_Points : out Portable_Types_Pkg.Integer_32) is
	84	--!
	85	-- PURPOSE: This procedure get the total Number of Points in the Point Layer.
	86	--
	87	--
	88	-- PARAMETERS: Number of points in the point layer.
	89	-- RAISES: None
	90	--
	91	--!
	92	begin
	93	Num_Points := Point_Count;
	94	end Get_Num_Points;
78	95	
	96	procedure Get_Point_Data(Point_Index: in Portable_Types_Pkg.Integer_32;
	97	Point_Data : out Fpp_Interface_Type.Point_Type) is
	98	--!
	99	-- PURPOSE: This procedure get the point data in the Point Layer and copy into
100	100	-- the VDU buffer.
101	101	--
102	102	--
103	103	-- PARAMETERS: Point data index.
104	104	-- RAISES: None
105	105	--
106	106	--!
107	107	begin
108	108	Point_Data := Points(Point_Index);
109	109	end Get_Point_Data;
79	110	procedure LGB_Store_Data is separate;

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_PKG.ADA (continued)

80	111	
81	112	procedure LGB_Seq_Leg is separate;
82	113	
83		procedure LGB_Seq_Rta_Leg (Initial_Est : in boolean := False)
84		is separate;
85	114	
86	115	function LGB_Search (Starting_Leg_Index : Portable_Types_Pkg.Integer_32;
87	116	-- DESCRIPTION Starting Leg Index for the search.
88	117	--
89	118	Search_Thing : Search_Thing_Type;
90	119	-- DESCRIPTION The thing being searched for.
91	120	--
92	121	Search_Direction :
93	122	FMCS_Base_Types.Horizontal_Direction_Type
94	123	-- DESCRIPTION The direction of the search.
95	124	-- Regular predictions search forward while
96	125	-- Descent Path Generation searches backward.
97	126	--
98	127) return Portable_Types_Pkg.Integer_32 is separate;
99	128	
100	129	
101	130	procedure Lgb_Next_Hold (Starting_Leg_Index : in Portable_Types_Pkg.Integer_32;
102	131	Entry_Index : out Portable_Types_Pkg.Integer_32;
103	132	Exit_Index : out Portable_Types_Pkg.Integer_32) is
104	133	--!
105	134	-- PURPOSE: This procedure will search forward through Perf's copy of the LGB
106	135	-- from the provided starting index for the first HA (hold to altitude)
107	136	-- or HM (hold to manual termination) holding pattern in the flight
108	137	-- plan. It returns the "last_segment_of_leg" index for the holding
109	138	-- pattern as the exit index, and the waypoint prior to it as the
110	139	-- exit index. If not found, zero is returned.
111	140	--
112	141	-- PARAMETERS:
113	142	-- Starting_Leg_Index - The index of the starting segment in the minleg fpln.
114	143	-- Entry_Index - The index of the waypoint prior to the holding pattern.
115	144	-- Exit_Index - The index of the holding pattern waypoint.
116	145	--
117	146	-- RAISES: None
118	147	--
119	148	--!
120	149	
121	150	Idx : Portable_Types_Pkg.Integer_32 := Starting_Leg_Index;

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_PKG.ADA (continued)

122	151	
123	152	begin
124	153	-- Search forward from the starting index for the first HA or HM holding pattern.
125	154	While (Idx /= 0) and then
126	155	((Perf_Lgb_Lfdata.Lgb (Idx).Fpln_Data.Pathterm /= Fmcs_Base_Types.hm) and then
127	156	(Perf_Lgb_Lfdata.Lgb (Idx).Fpln_Data.Pathterm /= Fmcs_Base_Types.ha))
128	157	loop
129	158	Idx:= Lgb_Search (Starting_Leg_Index => Idx,
130	159	Search_Thing => Next_Waypoint,
131	160	Search_Direction => Fmcs_Base_Types.Forward);
132	161	end loop;
133	162	
134	163	-- Return the found hold as the exit index. If not found, this is zero.
135	164	Exit_Index := Idx;
136	165	
137	166	-- Now find the hold entry waypoint, which is the waypoint prior to the hold exit.
138	167	Entry_Index := Lgb_Search (Starting_Leg_Index => Idx,
139	168	Search_Thing => Next_Waypoint,
140	169	Search_Direction => Fmcs_Base_Types.Backward);
141	170	
142	171	end Lgb_Next_Hold;
143	172	
	173	--
	174	-----
	175	function Lgb_Next_Lowest_Spd_Cnstr (Search_Thing : Search_Thing_Type;
	176	Starting_Leg_Index : Portable_Types_Pkg.Integer_32;
	177	CnstrSpd : Portable_Types_Pkg.Float_32;
	178	Search_Direction : FMCS_Base_Types.Horizontal_Direction_Type :=
		» FMCS_Base_Types.Forward)
	179	return Portable_Types_Pkg.Integer_32 is
144	180	
	181	--!
	182	-- PURPOSE: This procedure will search through Perf's copy of the LGB
	183	-- from the provided starting index for the MOST RESTRICTIVE
	184	-- speed constraint in the flight plan. Default direction is forward.
	185	-- If no speed constraint is found in the specified direction,
	186	-- zero is returned.
	187	--
	188	-- PARAMETERS:
	189	-- Search_Thing - The type of speed constraint to be searched for
	190	-- (Next_Clb_Spd_Cstr, Next_Des_Spd_Cstr).
	191	-- Starting_Leg_Index - The index of the starting segment in the minleg fpln.

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_PKG.ADA (continued)

```

192      -- CnstrSpd      - The current constraint speed (for comparison against).
193      -- Search_Direction - The direction to search (default is forward).
194      --
195      -- RAISES: None
196      --
197      --!
145 198
199      Idx    : Portable_Types_Pkg.Integer_32 := Starting_Leg_Index;
200      SpdPtr : Portable_Types_Pkg.Integer_32 := 0;
201      Spd     : Portable_Types_Pkg.Float_32  := CnstrSpd;
202
203      begin
204          -- Return zero when end of flight plan detected or the Search_Thing
205          -- is not a constraint speed.
206          if (Idx = 0) or else
207              ((Search_Thing /= Next_Des_Spd_Cstr) and then
208               (Search_Thing /= Next_Clb_Spd_Cstr))
209          then
210              return(0);
211          end if;
212
213          loop
214
215              Valid_Idx := False;
216              Hold_Idx  := False;
217
218              -- move to the next speed constrained mini-leg
219              Idx := Perf_LGB_Pkg.LGB_Search(Starting_Leg_Index => Idx,
220                                           Search_Thing    => Search_Thing,
221                                           Search_Direction => Search_Direction);
222
223              Valid_Idx := (Idx /= 0) and then (not Perf_LGB_LFData.LGB(Idx).Fpln_Data.MissedAppr);
224
225              Hold_Idx := (Idx /= 0) and then Is_Hold_Leg(Idx);
226
227              if (Valid_Idx) and then
228                  (not Hold_Idx) and then
229                  (Perf_LGB_LFData.LGB(Idx).Perf_Data.SpcSpd < Spd) then
230
231                  -- this is the lowest speed constraint found so far, save it
232                  SpdPtr := Idx;
233                  Spd := Perf_LGB_LFData.LGB(Idx).Perf_Data.SpcSpd;

```

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_PKG.ADA (continued)

```

234
235     end if;
236
237     exit when not Valid_Idx;
238
239     end loop;
240     return SpdPtr;
241
242 end Lgb_Next_Lowest_Spd_Cnstr;
243
244 --|
245 -----
246 procedure Lgb_Next_Lowest_Spd_Cnstr_For_Descent (Starting_Leg_Index : in Portable_Types_Pkg.Integer_32;
247                                                  CnstrSpd                : in Portable_Types_Pkg.Float_32;
248                                                  Leg_Index                : out Portable_Types_Pkg.Integer_32;
249                                                  CSS_Found                : out Boolean) is
250
251 --!
252 --| PURPOSE: This procedure will search through Perf's copy of the LGB
253 --|           from the provided starting index for the MOST RESTRICTIVE
254 --|           descent-like speed constraint in the flight plan, including
255 --|           holds in the descent path. If no speed constraint is found,
256 --|           zero is returned.
257 --
258 -- PARAMETERS:
259 --   Starting_Leg_Index - The index of the starting segment in the minleg fpln.
260 --   CnstrSpd           - The current constraint speed (for comparison against).
261 --   Leg_Index          - The most restrictive descent-like speed constraint uppath
262 --                       from Starting_Leg_Index. Will return 0 if no constraint
263 --                       is found.
264 --   CSS_Found          - Will return true if the search was halted due to an
265 --                       encountered Cruise Speed Segment.
266 --
267 -- RAISES: None
268 --
269 --!
270
271     Idx      : Portable_Types_Pkg.Integer_32 := Starting_Leg_Index;
272     Spd      : Portable_Types_Pkg.Float_32  := CnstrSpd;
273     Hold_Bridge_Dist : constant := 10.0; -- NAUTICAL MILES
274
275 begin

```

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_PKG.ADA (continued)

```

276     Leg_Index := 0;
277     loop
278
279         Valid_Idx := False;
280         Hold_In_Descent := False;
281         Hold_Idx := False;
282         Valid_Spd_Idx := True;
283
284         -- move to the next waypoint. We are using the waypoint instead
285         -- of the speed constraint because we need to stop the search if we
286         -- encounter a cruise speed segment
287         Idx := Perf_LGB_Pkg.LGB_Search(Starting_Leg_Index => Idx,
288                                     Search_Thing      => Next_Waypoint,
289                                     Search_Direction   => FMCS_Base_Types.Backward);
290
291         -- It is not a valid Index and we should exit the search if we reach the
292         -- end of the flight plan, we are in missed approach, or if we reach a cruise
293         -- speed segment
294         Valid_Idx := (Idx /= 0) and then
295                     (not Perf_LGB_LFData.LGB(Idx).Fpln_Data.MissedAppr) and then
296                     (not Perf_LGB_LFData.LGB(Idx).Fpln_Data.Crzspdtgt_Val);
297
298         -- Since we are searching on 'waypoint', ensure that this waypoint contains
299         -- a speed constraint. This will include holds in descent, because the spcspd
300         -- for holds in descent will always contain at least the best hold speed
301         if (Valid_Idx) and then
302             (Perf_LGB_LFData.LGB(Idx).Perf_Data.SpcSpdVal) and then
303             (Perf_LGB_LFData.LGB(Idx).Perf_Data.FltMode = GbDescent) then
304
305             -- For descent, a hold is defined as an HF or HM leg
306             Hold_Idx := (Perf_LGB_LFData.LGB(Idx).Fpln_Data.PathTerm = HF) or else
307                         (Perf_LGB_LFData.LGB(Idx).Fpln_Data.PathTerm = HM);
308
309             -- If this hold is within 10nm of the top of descent, consider it when
310             -- determining the most limiting descent-like speed constraint
311             Hold_In_Descent :=
312                 (Perf_Top_Of_Des_Lfdata.TODdata.Valid) and then
313                 ((Perf_LGB_LFData.LGB(Idx).Common_Data.Fixdistodest + Hold_Bridge_Dist) <= Perf_Top_Of_Des_Lfdata.TODdata.Dist);
314
315             -- a speed constraint has been found
316             if (not Hold_Idx or else
317                 (Hold_Idx and then Hold_In_Descent)) and then

```

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_PKG.ADA (continued)

```

318         (Perf_LGB_LFData.LGB(IdX).Perf_Data.SpcSpd <= Spd) then
319
320         -- this is the lowest speed constraint found so far, save it
321         Leg_Index := Idx;
322         Spd := Perf_LGB_LFData.LGB(IdX).Perf_Data.SpcSpd;
323
324         end if;
325     end if;
326
327     CSS_Found := Perf_LGB_LFData.LGB(IdX).Fpln_Data.Crzspdtgt_Val;
328
329     exit when not Valid_IdX;
330
331 end loop;
332
333 end Lgb_Next_Lowest_Spd_Cnstr_For_Descent;
334
335 --|
336 -----
337 function Is_Hold_Leg (Leg_Index : in Portable_Types_Pkg.Integer_32) return Boolean is
338     --|
339     --| PURPOSE: This function returns true if the specified mini-leg index is an
340     --|           HM, HA, or HF leg.
341     --|
342     --| PARAMETERS:
343     --|           Leg_Index - The index to evaluate.
344     --|
345     --| SPECIAL CONSIDERATIONS: Assumes a valid leg index is supplied.
346     --|
347     --| RAISES: None.
348     --|
349
350     begin
351         return ((Perf_Lgb_Lfdata.Lgb(Leg_Index).Fpln_Data.Pathterm = HA) or else
352             (Perf_Lgb_Lfdata.Lgb(Leg_Index).Fpln_Data.Pathterm = HM) or else
353             (Perf_Lgb_Lfdata.Lgb(Leg_Index).Fpln_Data.Pathterm = HF));
354     end Is_Hold_Leg;
355
356 --|
357 -----
358 procedure Create_Point
359     (

```

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_PKG.ADA (continued)

```

148 360      Event : in Perf_Point_Termination_Types.Termination_Type;
149 361      Display_Suppressed : boolean := false
150 362      ) is separate;
151 363      --!
152 364      --| PURPOSE: This procedure will create an LGB point layer record using the
153 365      --|           passed-in termination type and other external predictions data
154 366      --|           objects to create a point layer record for the current predictions
155 367      --|           state. The point record will be kept internally to this package
156 368      --|           until Output_Preds is called do just that.
157 369      --
158 370      -- PARAMETERS: Termination_Type - The type of integration termination that
159 371      --               predictions stopped at, from Perf's point of view. This roughly
160 372      --               equates to the events checked for in the various PROCTERM routines.
161 373      --
162 374      -- RAISES: None
163 375      --
164 376      --!
165 377
166 378      procedure Output_Preds is separate;
167 379      --!
168 380      --| PURPOSE: This procedure outputs the predicted flight plan in Perf's copy
169 381      --|           of the route, as well as the point layer records generated during
170 382      --|           flight plan predictions, out to the interface with EFIS, who will
171 383      --|           then write them to the LGB.
172 384      --
173 385      -- PARAMETERS: None
174 386      -- RAISES: None
175 387      --
176 388      --!
177 389
178 390 end Perf_LGB_Pkg;
```

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_PKG_ADA

```

1 1 --|
2 2 --| DATA_RIGHTS: HONEYWELL CONFIDENTIAL & PROPRIETARY
3 3 --|
4 4 --| THIS WORK CONTAINS VALUABLE CONFIDENTIAL AND PROPRIETARY
5 5 --| INFORMATION. DISCLOSURE, USE OR REPRODUCTION OUTSIDE OF
6 6 --| HONEYWELL INTERNATIONAL, INC. IS PROHIBITED EXCEPT AS
7 7 --| AUTHORIZED IN WRITING. THIS UNPUBLISHED WORK IS PROTECTED BY
8 8 --| THE LAWS OF THE UNITED STATES AND OTHER COUNTRIES. IN THE
9 9 --| EVENT OF PUBLICATION, THE FOLLOWING NOTICE SHALL APPLY:
COPR. 2012 HONEYWELL INTERNATIONAL, INC. ALL RIGHTS RESERVED.
```


File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_PKG_.ADA (continued)

```

10 10 --|
11 11
12 12 with Portable_Types_Pkg;
13 13 with FMCS_Base_Types;
14 14 with Perf_Point_Termination_Types;
15 15 with Fpp_Interface_Type;
16 16
17 17 package Perf_LGB_Pkg is
18 18     --!
19 19     -- PURPOSE:      Performance Predictions LGB package specification.
20 20     --
21 21     -- ANCHOR:       FMCS_19_21023510
22 22     --
23 23     --| @DESCRIPTION: Makes visible the procedure declarations for the
24 24     --| predictions LGB related procedures.
25 25     --
26 26     -- SPECIAL_CONSIDERATIONS: None
27 27     --
28 28     -- REVISION_HISTORY:
29 29     --   DATE          SCR #          Programmer          DRCM#
30 30     -- =====
31 31     --   12/18/95      8011          B. O'Laughlin          M777B_FMF_00547
32 32     -- Added Lgb_Seq_Rta_Leg for processing predicted RTA leg sequence.
33 33     --
34 34     -- ===== 787 HISTORY STARTS HERE =====
35 35     --
36 36     --   12/06/05      519.00          Pat Caulfield
37 37     -- Added new procedures Create_Point and Output_Preds, and added Next_Wind
38 38     -- and Next_Temp to Search_Thing_Type.
39 39     --
40 40     --   01/05/06      519.04          Pat Caulfield
41 41     -- Replaced Next_Wind, Next_Temp with Next_Waypoint; Changed Next_Leg to
42 42     -- Next_Segment.
43 43     --
44 44     --   01/18/06      787.01          Pat Caulfield
45 45     -- Renamed Create_Point's parameter to Event.
46 46     --
47 47     --   08/07/08      6676.00          Pat Caulfield
48 48     -- Added new parameter Display_Suppressed to Create_Point.
49 49     --
49 50     --   07/18/2012    8301.02          Pat Caulfield
50 51     -- Added Lgb_Next_Hold.

```

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_PKG_.ADA (continued)

51	52	--!
52	53	-----
53	54	type Search_Thing_Type is (Next_Segment,
54	55	Next_Alt_Cstr,
55		Next_Spd_Cstr,
	56	Next_Clb_Spd_Cstr,
	57	Next_Des_Spd_Cstr,
56	58	Next_Alt_And_Spd_Cstr,
57	59	Next_Alt_Or_Spd_Cstr,
58	60	Next_Step_Alt_Term,
59	61	Next_Waypoint);
60	62	
61		for Search_Thing_Type use (Next_Segment => 0,
62		Next_Alt_Cstr => 1,
63		Next_Spd_Cstr => 2,
64		Next_Alt_And_Spd_Cstr => 3,
65		Next_Alt_Or_Spd_Cstr => 4,
66		Next_Step_Alt_Term => 5,
67		Next_Waypoint => 6);
	63	for Search_Thing_Type use (Next_Segment => 0,
	64	Next_Alt_Cstr => 1,
	65	Next_Clb_Spd_Cstr => 2,
	66	Next_Des_Spd_Cstr => 3,
	67	Next_Alt_And_Spd_Cstr => 4,
	68	Next_Alt_Or_Spd_Cstr => 5,
	69	Next_Step_Alt_Term => 6,
	70	Next_Waypoint => 7);
68	71	
69	72	
70	73	procedure LGB_Store_Data;
71	74	--!
72	75	-- @PURPOSE: This procedure stores predicted data from the integration
73	76	-- progress buffer into Perf's copy of the LGB (PERF_LGB_LFDATA.LGB).
74	77	--
75	78	-- PARAMETERS: None
76	79	-- RAISES: None
77	80	--
78	81	--!
79	82	
80	83	procedure LGB_Seq_Leg;
81	84	--!
82	85	-- @PURPOSE: This procedure changes the PERF LGB leg pointer

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_PKG_ADA (continued)

```

83      86      --| (PERF_PREDSED_LFDATA.NAVPTR) to point to the next leg in the flight
84      87      --| plan (in PERF_LGB_PKG.LGB). It also initializes the flight plan track
85      88      --| to be used for ground speed computations on the new leg, and some
86      89      --| other small miscellaneous implementation requirements for processing
87      90      --| the end of the flight plan (NAVPTR =0).

88      -----
89      -----PARAMETERS: None
90      -----RAISES: None
91      -----
92      -----+
93
94      -----procedure LGB_Seq_Rta_Leg (Initial_Est : in boolean := False);
95      -----+
96      -----| @PURPOSE: This procedure does all of the necessary processing when
97      -----| performance predictions sequence a leg with a RTA time constraint.

98      91      --
99      92      -- PARAMETERS: None
100     93      -- RAISES: None
101     94      --
102     95      --!
103     96
104     97      function LGB_Search (Starting_Leg_Index : Portable_Types_Pkg.Integer_32;
105     98          --| @DESCRIPTION Starting Leg Index for the search.
106     99          --
107    100          Search_Thing : Search_Thing_Type;
108    101          --| @DESCRIPTION The thing being searched for.
109    102          --
110    103          Search_Direction :
111    104              FMCS_Base_Types.Horizontal_Direction_Type
112    105          --| @DESCRIPTION The direction of the search.
113    106          --| Regular predictions search forward while
114    107          --| Descent Path Generation searches backward.
115    108          --
116    109          ) return Portable_Types_Pkg.Integer_32;
117    110      --!
118    111      --| @PURPOSE: This function is a performance predictions utility that
119    112      --| searches PERF_LGB_LFDATA.LGB for a desired leg index. The caller inputs
120    113      --| what to search for and the function returns the desired index, or 0 if
121    114      --| the index is not found.
122    115      --
123    116      -- PARAMETERS: None
124    117      -- RAISES: None

```

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_PKG_.ADA (continued)

```

125 118  --
126 119  --!
127 120
128 121  procedure Lgb_Next_Hold (Starting_Leg_Index : in Portable_Types_Pkg.Integer_32;
129 122      --| @DESCRIPTION: The index of the starting segment in
130 123      --| the minleg fpln.
131 124      Entry_Index : out Portable_Types_Pkg.Integer_32;
132 125      --| @DESCRIPTION: The index of the waypoint prior to
133 126      --| the holding pattern.
134 127      Exit_Index   : out Portable_Types_Pkg.Integer_32);
135 128      --| @DESCRIPTION: The index of the holding pattern waypoint.
136 129
130 130  --|
131 131  -----
132 132  function Lgb_Next_Lowest_Spd_Cnstr (Search_Thing      : Search_Thing_Type;
133 133      Starting_Leg_Index : Portable_Types_Pkg.Integer_32;
134 134      CnstrSpd           : Portable_Types_Pkg.Float_32;
135 135      Search_Direction   : FMCS_Base_Types.Horizontal_Direction_Type :=
136 136      » FMCS_Base_Types.Forward)
137 137      return Portable_Types_Pkg.Integer_32;
138 138  --!
139 139  --| @PURPOSE: This function is a performance predictions utility that searches
140 140  --| PERF_LGB_LFDATA.LGB for the next most restrictive waypoint
141 141  --| climb-like or descent-like speed constraint that is not on a
142 142  --| holding pattern.
143 143  --
144 144  -- PARAMETERS: Search_Thing (Next_Clb_Spd_Cstr or Next_Des_Spd_Cstr)
145 145  -- Starting_Leg_Index
146 146  -- CnstrSpd
147 147  -- Search_Direction (defaults to Forward)
148 148  -- Returns the index of the next most restrictive speed constraint (Clb or Des) or 0 if none found
149 149  -- RAISES: None
150 150  --
151 151  --!
152 152  --|
153 153  -----
154 154  procedure Lgb_Next_Lowest_Spd_Cnstr_For_Descent (Starting_Leg_Index : in Portable_Types_Pkg.Integer_32;
155 155      CnstrSpd           : in Portable_Types_Pkg.Float_32;
156 156      Leg_Index         : out Portable_Types_Pkg.Integer_32;
157 157      CSS_Found          : out Boolean);
158 158

```

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_PKG_.ADA (continued)

```

159  --!
160  --| PURPOSE: This procedure will search through Perf's copy of the LGB
161  --|           from the provided starting index for the MOST RESTRICTIVE
162  --|           descent-like speed constraint in the flight plan, including
163  --|           holds in the descent path. If no speed constraint is found,
164  --|           zero is returned.
165  --
166  -- PARAMETERS:
167  --   Starting_Leg_Index - The index of the starting segment in the minleg fpln.
168  --   CnstrSpd           - The current constraint speed (for comparison against).
169  --   Leg_Index          - The most restrictive descent-like speed constraint uppath
170  --                       from Starting_Leg_Index. Will return 0 if no constraint
171  --                       is found.
172  --   CSS_Found          - Will return true if the search was halted due to an
173  --                       encountered Cruise Speed Segment.
174  --
175  -- RAISES: None
176  --

```

```

177
178  --|
179  -----
180  function Is_Hold_Leg (Leg_Index : in Portable_Types_Pkg.Integer_32) return Boolean;
181  --!
182  --| PURPOSE: This function returns true if the specified mini-leg index is an
183  --|           HM, HA, or HF leg.
184  --|
185  --| PARAMETERS:
186  --|   Leg_Index - The index to evaluate.
187  --|
188  --| SPECIAL CONSIDERATIONS: Assumes a valid leg index is supplied.
189  --|
190  --| RAISES: None.
191  --!

```

```

192
193  --|
194  -----

```

```

137 195  procedure Create_Point
138 196  (
139 197      Event : in Perf_Point_Termination_Types.Termination_Type;
140 198      Display_Suppressed : boolean := false
141 199  );
142 200  --!

```

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_PKG_ADA (continued)

143	201	-- PURPOSE: This procedure will create an LGB point layer record using the
144	202	-- passed-in termination type and other external predictions data
145	203	-- objects to create a point layer record for the current predictions
146	204	-- state. The point record will be kept internally to this package
147	205	-- until Output_Preds is called do just that.
148	206	--
149	207	-- PARAMETERS: Termination_Type - The type of integration termination that
150	208	-- predictions stopped at, from Perf's point of view. This roughly
151	209	-- equates to the events checked for in the various PROCTERM routines.
152	210	--
153	211	-- RAISES: None
154	212	--
155	213	--!
156	214	
157	215	procedure Output_Preds;
158	216	--!
159	217	-- PURPOSE: This procedure outputs the predicted flight plan in Perf's copy
160	218	-- of the route, as well as the point layer records generated during
161	219	-- flight plan predictions, out to the interface with EFIS, who will
162	220	-- then write them to the LGB.
163	221	--
164	222	-- PARAMETERS: None
165	223	-- RAISES: None
166	224	--
167	225	--!
	226	procedure Get_Num_Points (Num_Points : out Portable_Types_Pkg.Integer_32);
	227	--!
	228	-- PURPOSE: This procedure get the Total Number of Points in the Point Layer.
	229	--
	230	--
	231	-- PARAMETERS: Number of points in the point layer.
	232	-- RAISES: None
	233	--
	234	--!
168	235	
	236	procedure Get_Point_Data(Point_Index: in Portable_Types_Pkg.Integer_32;
	237	Point_Data : out Fpp_Interface_Type.Point_Type);
	238	--!
	239	-- PURPOSE: This procedure get the point data in the Point Layer and copy into
	240	-- the VDU buffer.
	241	--
	242	--

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_PKG_.ADA (continued)

	243	-- PARAMETERS: Point data index.
	244	-- RAISES: None
	245	--
	246	--!
169	247	end Perf_LGB_Pkg;

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_SEARCH_SEP.ADA

1	1	--
2	2	-- STUB File
3	3	--
4	4	-- CTP_B787_PERF_LGBSEARCH_STB.ada
5	5	--
6	6	-- REASONS FOR STUBBING : The Procedure LGB_Search has been stubbed for return Starting Leg Index for the search.
7	7	--
8	8	--
9	9	-- DATA_RIGHTS: HONEYWELL CONFIDENTIAL & PROPRIETARY
10	10	-- THIS WORK CONTAINS VALUABLE CONFIDENTIAL AND PROPRIETARY
11	11	-- INFORMATION. DISCLOSURE, USE OR REPRODUCTION OUTSIDE OF
12	12	-- HONEYWELL INTERNATIONAL, INC. IS PROHIBITED EXCEPT AS
13	13	-- AUTHORIZED IN WRITING. THIS UNPUBLISHED WORK IS PROTECTED BY
14	14	-- THE LAWS OF THE UNITED STATES AND OTHER COUNTRIES. IN THE
15	15	-- EVENT OF PUBLICATION, THE FOLLOWING NOTICE SHALL APPLY:
16	16	-- COPR. 2005 HONEYWELL INTERNATIONAL, INC. ALL RIGHTS RESERVED.
17	17	--
18	18	with Portable_Types_Pkg;
19	19	with FMCS_Base_Types;
20	20	with FMCS_FP_Guid_Btypes;
21	21	with AC_Position_Types;
22	22	
23	23	with Perf_LGB_Lfdata;
24	24	
25	25	use Portable_Types_Pkg;
26	26	use FMCS_Base_Types;
27	27	use FMCS_FP_Guid_Btypes;
28	28	use AC_Position_Types;
29	29	-----
30	30	with CTP_B787_PERF_CRZINITE_DRV;
31	31	use CTP_B787_PERF_CRZINITE_DRV;
32	32	
33	33	separate (Perf_LGB_Pkg)
34	34	
35	35	function LGB_Search (Starting_Leg_Index : Portable_Types_Pkg.Integer_32;

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_SEARCH_SEP.ADA (continued)

```

36      36      --| DESCRIPTION Starting Leg Index for the search.
37      37      --
38      38      Search_Thing : Search_Thing_Type;
39      39      --| DESCRIPTION The thing being searched for.
40      40      --
41      41      Search_Direction :
42      42          FMCS_Base_Types.Horizontal_Direction_Type
43      43      --| DESCRIPTION The direction of the search.
44      44      --| Regular predictions search forward while
45      45      --| Descent Path Generation searches backward.
46      46      --
47      47      ) return Portable_Types_Pkg.Integer_32 is
48      48      --!
49      49      -- ANCHOR:          FMCS_19_21023514
50      50      -- SOURCE:       FMFSDD; FMCS_19_21023003 |
51      51      --| @DESCRIPTION:
52      52      --| This procedure is a performance predictions utility used to search the
53      53      --| LGB (PERF_LGB_LFDATA.LGB) for the next leg index that matches a desired
54      54      --| input. The calling procedure inputs as a parameter what to search for
55      55      --| (For example, the next altitude constraint).
56      56      --|
57      57      --| Possible search criteria are as follows :
58      58      --|     o Next segment
59      59      --|     o Next specified speed constraint
60      60      --|     o Next specified altitude constraint
61      61      --|     o Next specified speed and altitude constraint
62      62      --|     o Next specified speed or altitude constraint
63      63      --|     o Next specified step altitude constraint
64      64      --|     o Next waypoint (last segment of leg)
65      65      --|
66      66      --| The procedure checks the FLTMODE field of the leg to limit the search to
67      67      --| the current predicted flight phase, if applicable.
68      68      --|
69      69      --| The procedure also has a flag as an input parameter that tells it
70      70      --| whether to search forward or backward through the flight plan.
71      71      --
72      72      -- SPECIAL_CONSIDERATIONS:
73      73      --
74      74      --     SHARED_DATA_FOR:
75      75      --     IOBLK
76      76      --
77      77      -- REVISION_HISTORY:

```


File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_SEARCH_SEP.ADA (continued)

```

78      78      --      DATE          SCR #          Programmer          DRCM#
79      79      --      01-11-94      1777          D. Groethe          10016
80      80      --      Initial version - complete
81      81      --
82      82      --      DATE          SCR #          Programmer          DRCM#
83      83      --      03-22-94      1777          D. Groethe          11844
84      84      --      Wrong logic on step search was causing an erroneous index return of 0.
85      85      --
86      86      --      ===== 787 HISTORY =====
87      87      --
88      88      --      DATE          SCR #          Programmer
89      89      --      11-11-2005      519.00          Pat Caulfield
90      90      --      Added case for wind/temp entry searches, which amounts to searching
91      91      --      for the last lateral segment of the leg.
92      92      --
93      93      --      DATE          SCR #          Programmer
94      94      --      01-05-2006      519.04          Pat Caulfield
95      95      --      Changed Next_Wind and Next_Temp to be Next_Waypoint to simplify things
96      96      --      and avoid confusion, and renamed Next_leg to Next_Segment.
97      97      --
98      98      --!
99      99
100     100
101     101      -- L O C A L   V A R I A B L E S --
102     102
103     103      Return_Index : Portable_Types_Pkg.Integer_32;
104     104      --| DESCRIPTION Leg index returned to calling procedure
105     105
106     106      Starting_Fltmode : FMCS_FP_Guid_Btypes.Fltphasetyp;
107     107      --| DESCRIPTION Fltmode of the input mini-leg (or the first non-Nogbmode
108     108      --| value found in the direction of the search). This is used to limit
109     109      --| the searches for constraints to the relevant flight phase.
110     110 begin -- LGB_Search
111     111
112     112 -- protect against calls with a bad starting leg index
113     113
114     114 -- if Starting_Leg_Index = 0 then
115     115 --     return 0;
116     116 -- end if;
117     117 --
118     118 -- Return_Index := Starting_Leg_Index;
119     119 -- loop

```

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_SEARCH_SEP.ADA (continued)

```

120 120 --
121 121 --     if (Search_Direction = Forward) then
122 122 --         Return_Index := Perf_LGB_Lfdata.LGB(Return_Index).Nextfpn;
123 123 --     else
124 124 --         Return_Index := Perf_LGB_Lfdata.LGB(Return_Index).Prevfpn;
125 125 --     end if;
126 126 --
127 127 --     -- RETURN ZERO WHEN END OF FLIGHT PLAN DETECTED OR
128 128 --     -- MISSED APPROACH DETECTED OR
129 129 --     -- CHANGE IN PREDICTED GLEG FLIGHT PHASE
130 130 --     if (Return_Index = 0) or else
131 131 --
132 132 --         (Perf_LGB_Lfdata.LGB(Return_Index).Fpln_Data.MissedAppr) or else
133 133 --
134 134 --         ((Perf_LGB_Lfdata.LGB(Return_Index).Perf_Data.Fltmode) /=
135 135 --         (Perf_LGB_Lfdata.LGB(Starting_Leg_Index).Perf_Data.Fltmode) and
136 136 --         (Search_Thing /= Next_Step_Alt_Term) and
137 137 --         (Search_Thing /= Next_Segment) and
138 138 --         (Search_Thing /= Next_Waypoint))
139 139 --     then
140 140 --         return(0);
141 141 --     end if;
142 142 --
143 143 --     case Search_Thing is
144 144 --
145 145 --         when Next_Segment => -- already advanced the pointer up above, so just exit.
146 146 --             exit;
147 147 --
148 148 --         when Next_Alt_Cstr =>
149 149 --             exit when (Perf_LGB_Lfdata.LGB(Return_Index).Fpln_Data.HavePerf);
150 150 --
151 151 --         when Next_Spd_Cstr =>
152 152 --             exit when (Perf_LGB_Lfdata.LGB(Return_Index).Perf_Data.SpcSpdVal);
153 153 --
154 154 --         when Next_Alt_And_Spd_Cstr =>
155 155 --             exit when (Perf_LGB_Lfdata.LGB(Return_Index).Fpln_Data.HavePerf) and
156 156 --             (Perf_LGB_Lfdata.LGB(Return_Index).Perf_Data.SpcSpdVal);
157 157 --
158 158 --         when Next_Alt_Or_Spd_Cstr =>
159 159 --             exit when (Perf_LGB_Lfdata.LGB(Return_Index).Fpln_Data.HavePerf) or
160 160 --             (Perf_LGB_Lfdata.LGB(Return_Index).Perf_Data.SpcSpdVal);
161 161 --

```

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_SEARCH_SEP.ADA (continued)

```

162 162 --      when Next_Step_Alt_Term =>
163 163 --          exit when (Perf_LGB_Lfdata.LGB(Return_Index).Fpln_Data.SpAlt1Val) and
164 164 --              (Perf_LGB_Lfdata.
165 165 --                  LGB(Return_Index).Fpln_Data.SpAlt1Pos = SC);
166 166 --
167 167 --      -- Search for the next segment that is marked the last segment of the leg;
168 168 --      -- it contains data for the next waypoint.
169 169 --
170 170 --      when Next_Waypoint =>
171 171 --          exit when (Perf_LGB_Lfdata.LGB(Return_Index).Last_Segment_Of_Leg);
172 172 --
173 173 --      end case;
174 174 --
175 175 -- end loop;
176 176
177 177 Return_Index := Test_LGB_Search;
178 178
179 179 return(Return_Index);
180 180
181 181 end LGB_Search;
```

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_SEQ_LEG_SEP.ADA

```

1 1 --
2 2 -- DATA_RIGHTS: HONEYWELL CONFIDENTIAL & PROPRIETARY
3 3 --
4 4 -- THIS WORK CONTAINS VALUABLE CONFIDENTIAL AND PROPRIETARY
5 5 -- INFORMATION. DISCLOSURE, USE OR REPRODUCTION OUTSIDE OF
6 6 -- HONEYWELL INTERNATIONAL, INC. IS PROHIBITED EXCEPT AS
7 7 -- AUTHORIZED IN WRITING. THIS UNPUBLISHED WORK IS PROTECTED BY
8 8 -- THE LAWS OF THE UNITED STATES AND OTHER COUNTRIES. IN THE
9 9 -- EVENT OF PUBLICATION, THE FOLLOWING NOTICE SHALL APPLY:
10 10 -- COPR. 2005 HONEYWELL INTERNATIONAL, INC. ALL RIGHTS RESERVED.
11 11 with Portable_Types_Pkg;
12 12
13 13 with Perf_ADS_Intent_Pkg;
14 14 with Perf_Su_Spd_Utills_Pkg;
15 15 with Ops_Cdk_Perf_Pdb_Mgr_Pkg;
16 16 with Perf_Preds_Lfdata;
17 17 with Perf_LGB_Lfdata;
18 18 with Perf_Task_Control_Lfdata;
19 19 with Perf_Integrators_Lfdata;
20 20 with Perf_Wind_Lfdata;
```

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_SEQ_LEG_SEP.ADA (continued)

21	21	with Perf_Rta_Lfdata;
	22	with Perf_Rta_Pkg;
22	23	
23	24	use Portable_Types_Pkg;
24	25	
25	26	separate (Perf_LGB_Pkg)
26	27	
27	28	procedure LGB_Seq_Leg is
28	29	--!
29	30	-- DATA_RIGHTS: Honeywell ATSD Proprietary
30	31	-- ANCHOR: FMCS_19_21023513
31	32	-- SOURCE: FMFSDD; FMCS_19_21023002, FMCS_19_21023004
32	33	-- FMFSRD; FMCS_19_20006056, FMCS_19_20006062
33	34	-- FMFSRD; FMCS_19_20012452, FMCS_19_20012450
34	35	-- FMFSRD; FMCS_19_20006073, FMCS_19_20006075
35	36	-- FMFSRD; FMCS_19_20010030
36	37	-- FMFSRD; FMCS_19_20006316
37	38	-- @DESCRIPTION:
38	39	-- This procedure is a performance predictions utility. It is used to change
39	40	-- the leg index (Perf_Preds_Lfdata.NavPtr) that points to the current
40	41	-- predicted leg in Perf's copy of the LGB (PERF_LGB_LFDATA.LGB) to point
41	42	-- to the next leg.
42	43	--
43	44	-- It is also used to write out a set of waypoints to the real (FM Global)
44	45	-- LGB, once a certain number (see 'LGB_Leg_Output_Constant' declared in
45	46	-- package body) have been predicted.
46	47	--
47	48	-- SPECIAL_CONSIDERATIONS:
48	49	--
49	50	-- Use of Perf_BG_Waiting_On_Semaphore to signal the Preds Restart
50	51	-- function not to restart Perf_BG while Perf_BG is accessing
51	52	-- the LGB. The semaphore associated with accessing the LGB needs to be
52	53	-- restored (by releasing access to the LGB) before a process restart can
53	54	-- be done.
54	55	--
55	56	-- If a restart was needed while the LGB was being accessed (as indicated
56	57	-- by the Unsuccessful_Restart flag), PERF BG restarts its self after
57	58	-- regaining processor control and signalling the semaphore (releasing LGB
58	59	-- access).
59	60	--
60	61	--
61	62	-- SHARED_DATA_FOR:

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_SEQ_LEG_SEP.ADA (continued)

```

62 63 -- IOBLK
63 64 --
64 65 -- TRANSLATION HISTORY:
65 66 -- This module was translated from the 747 SEQNAVLEG.PAS, B7X7FMS, Gen. 2
66 67 --
67 68 -- REVISION_HISTORY:
68 69 -- DATE SCR # Programmer DRCM#
69 70 -- 01-11-94 1777 D. Groethe 10015
70 71 -- Initial version - complete
71 72 --
72 73 -- 01-11-94 1777, 2402 D. Groethe 10541
73 74 -- Initial value of First_Output_Leg_Index was NavPtr. Should be
74 75 -- PrevNavPtr because NavPtr is already updated to the next leg by
75 76 -- the time First_Output_Leg_Index is set. Also made LGB object update
76 77 -- (MagVar) because of a type change in LGB manager (2402).
77 78 --
78 79 -- 11-21-94 5686 B. O'Laughlin 19484
79 80 -- Update groundspeed when there is a turn at the waypoint being sequenced,
80 81 -- because a turn represents a step change in groundspeed if there is a wind.
81 82 --
82 83 -- 01/17/96 8011 B. O'Laughlin M777B_FMF_00549
83 84 -- Added call to Lgb_Seq_Rta_Leg for processing predicted RTA leg sequence.
84 85 --
85 86 -- 04/14/96 8030.08 Karen Hegeman M777B_FMF_01596
86 87 -- Added call to Perf_ADS_Intent_Pkg.Calc_Intermediate_Point to store
87 88 -- intermediate intent data at a flight plan track change, or due to
88 89 -- sequencing the RTA fix.
89 90 --
90 91 -- ===== 787 HISTORY =====
91 92 --
92 93 -- 12/01/05 519.00 Pat Caulfield
93 94 -- Reworked for segment predictions; the logic for writing predictions to the
94 95 -- LGB has been removed - PERF_EFIS_LGB_MGR_PKG does all of that now.
95 96 --
96 97 --!
97 98 -----
98 99
99 100 -- L O C A L V A R I A B L E S --
100 101
101 102 -- DESCRIPTION Local copy of Perf_Preds_Lfdata.PrevNavPtr used for
102 103 -- efficiency to reduce global memory access. Initialized to the current
103 104 -- NavPtr, which advances its current value to point to the next leg.

```

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_SEQ_LEG_SEP.ADA (continued)

```

104 105   PrevNavPtr : Portable_Types_Pkg.Integer_32 := Perf_Preds_Lfdata.NavPtr;
105 106
106 107   -- DESCRIPTION Local copy of Perf_Preds_Lfdata.NavPtr used for
107 108   -- efficiency to reduce global memory access. Initialized to the Nextfpn
108 109   -- field of the current leg, which advances its current value to point to
109 110   -- the next leg.
110 111   NavPtr      : Portable_Types_Pkg.Integer_32 :=
111 112               Perf_LGB_Lfdata.LGB(Perf_Preds_Lfdata.NavPtr).NextFpn;
112 113
113 114   -- DESCRIPTION Leg track in.
114 115   Track_In    : Standard_Angle_Pkg.Saf_32;
115 116
116 117   -- DESCRIPTION Leg track out.
117 118   Track_Out   : Standard_Angle_Pkg.Saf_32;
118 119
119 120 begin -- LGB_Seq_Leg
120 121
121 122   -- IF NOT END OF FLIGHT PLAN
122 123   if (NavPtr > 0) then
123 124
124 125       -- SET UP GROUND SPEED DATA (FLIGHT PLAN TRACK AND PATH TERM)
125 126       Track_In  := Perf_LGB_Lfdata.LGB(NavPtr).Efis_Data.Incourse +
126 127               Perf_LGB_Lfdata.LGB(NavPtr).Efis_Data.MagVar;
127 128       Track_Out := Perf_LGB_Lfdata.LGB(NavPtr).Efis_Data.Outcourse +
128 129               Perf_LGB_Lfdata.LGB(PrevNavPtr).Efis_Data.MagVar;
129 130       Perf_Preds_Lfdata.Gsdata.Fplntrack :=
130 131       Track_Out + (Track_In - Track_Out) / Portable_Types_Pkg.Float_32(2.0);
131 132       Perf_Preds_Lfdata.Gsdata.Pathterm :=
132 133       Perf_LGB_Lfdata.LGB(NavPtr).Fpln_Data.Pathterm;
133 134
134 135       -- STOP PREDICTING UPON SEQUENCING MISSED APPROACH
135 136       if Perf_LGB_Lfdata.LGB(NavPtr).Fpln_Data.MissedAppr
136 137       then
137 138           -- SET NAVPTR TO 0 TO INDICATE END OF FLIGHT PLAN
138 139           NavPtr := 0;
139 140       end if;
140 141
141 142   end if; -- IF NOT END OF FLIGHT PLAN
142 143
143 144   -- if we've reached the next waypoint, find the next one.  If there are no more,
144 145   -- lgb_search will return zero.
145 146   If PrevNavPtr >= Perf_Preds_Lfdata.Next_Waypoint_NavPtr then

```

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_SEQ_LEG_SEP.ADA (continued)

```

146 147     Perf_Preds_Lfdata.Next_Waypoint_Navptr := Lgb_Search (Starting_Leg_Index => PrevNavPtr,
147 148                                     Search_Thing => Next_Waypoint,
148 149                                     Search_Direction => Fmcs_Base_Types.Forward);
149 150 end if;
150 151
151 152 -- RESTORE GLOBAL VALUES OF NAVPTR AND PREVNAVPTR
152 153 -- this also sequences the segment/leg (see above in local var declarations)
153 154 Perf_Preds_Lfdata.NavPtr      := NavPtr;
154 155 Perf_Preds_Lfdata.PrevNavPtr := PrevNavPtr;
155 156
156 157 -- IF FLIGHT PLAN HAS TURNED, UPDATE GROUNDSPED FOR NEW TRACK
157 158 -- and determine if ADS intermediate intent data needs to be sent
158 159 -- Note that PrevNavPtr points to the segment/leg we're sequencing.  Only do this
159 160 -- at a waypoint sequence (last segment of leg), not every segment.
160 161 if (PrevNavPtr > 0) and then Perf_LGB_Lfdata.LGB(PrevNavPtr).Last_Segment_Of_Leg then
161 162
162 163     -- compute the course change at the waypoint being sequenced
163 164     if Perf_Preds_Lfdata.Next_Waypoint_Navptr > 0 then
164 165         -- course change = next leg's outcourse minus current leg's incourse
165 166         Course_Change := abs(Perf_LGB_Lfdata.LGB(Perf_Preds_Lfdata.Next_Waypoint_Navptr).Efis_Data.Leg_Outcourse -
166 167                             Perf_LGB_Lfdata.LGB(PrevNavPtr).Efis_Data.Leg_Incourse);
167 168     else
168 169         Course_Change := 0.0;
169 170     end if;
170 171
171 172 -- recompute the ground speed if the track has changed more than 5 degrees
172 173 if (Course_Change > 5.0) then
173 174     -- (may want to call atmosphere model here if the waypoint
174 175     -- termination caused significant rollback in CHEKINTERMS)
175 176     Perf_Su_Spd_Utils_Pkg.Su_Compgrndspd
176 177         (Perf_Preds_Lfdata.Gsdata,
177 178         Perf_Integrators_Lfdata.Intprogbuf.Tas2,
178 179         Perf_Wind_Lfdata.Predwind,
179 180         Perf_Integrators_Lfdata.Intprogbuf.Gndspd2);
180 181 end if;
181 182
182 183 if (Course_Change > 1.5) then
183 184     -- Store ADS intermediate intent data if the track has changed more than 1.5 degrees
184 185     Perf_ADS_Intent_Pkg.Calc_Intermediate_Point;
185 186 end if;
186 187 end if;
187 188

```

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_SEQ_LEG_SEP.ADA (continued)

188	CHECK TO SEE IF LEG HAS A RTA TIME CONSTRAINT
189	if ((Ops_Cdk_Perf_Pdb_Mgr_Pkg.RTA_Is_Active(Perf_Preds_Lfdata.Vtpfplnindex)) and then
190	(PrevNavPtr = Perf_LGB_Lfdata.LGB_Header.RTA_Fix_Ptr) and then
191	(PrevNavPtr > 0) and then
192	(Perf_LGB_Lfdata.LGB(PrevNavPtr).Fpln_Data.RTA_Time_Val)) then
193	-----
194	set ads intent point (speed change), see FMCS_19_20006075
195	note Pred_Ksa will be 0.0 after call to Perf_Lgb_Pkg.Lgb_Seq_Rta_Leg so
196	you'll need to use a different variable if this line is moved
197	-----
198	if (Perf_Rta_Lfdata.Pred_Ksa /= 0.0) then
199	Perf_Ads_Intent_Pkg.Calc_Intermediate_Point;
200	end if;
189	if((PrevNavPtr > 0) and then (PrevNavPtr = Perf_LGB_Lfdata.LGB_Header.RTA_Fix_Ptr) and then
190	(Perf_Rta_Lfdata.Rta_Window_Task or else
191	((Ops_Cdk_Perf_Pdb_Mgr_Pkg.RTA_Is_Active(Perf_Preds_Lfdata.Vtpfplnindex)) and then
192	(Perf_LGB_Lfdata.LGB(PrevNavPtr).Fpln_Data.RTA_Time_Val)))) then
193	
194	
195	
196	if Perf_Rta_Lfdata.Rta_Window_Task then
197	-- Added a logic to stop ETA window predictions when it reaches at RTA waypoint.
198	-- This logic will ensure the storing of the MAX/MIN data.
199	-- At the end of logics set stop preds flag to TRUE.
200	-- It ensure to come out of the VTP exec . PERF_SRD_B_00067
201	if(Perf_Preds_Lfdata.Perf_Pass = Early) then
202	Perf_Rta_Lfdata.Rta_Windows(Perf_Preds_Lfdata.Vtpfplnindex).Min_Time.valid :=True;
203	Perf_Rta_Lfdata.Rta_Windows(Perf_Preds_Lfdata.Vtpfplnindex).Min_Time.Data :=
204	» Portable_Types_Pkg.Integer_32(Perf_Integrators_Lfdata.IntProgBuf.TProg) +
205	Perf_Preds_Lfdata.Aircraft_State.GMT.Data;
206	else
207	Perf_Rta_Lfdata.Rta_Windows(Perf_Preds_Lfdata.Vtpfplnindex).Max_Time.valid :=True;
208	Perf_Rta_Lfdata.Rta_Windows(Perf_Preds_Lfdata.Vtpfplnindex).Max_Time.Data :=
209	» Portable_Types_Pkg.Integer_32(Perf_Integrators_Lfdata.IntProgBuf.TProg) +
210	Perf_Preds_Lfdata.Aircraft_State.GMT.Data;
211	end if;
212	Perf_Preds_Lfdata.Vtplogic.Stop_Preds :=True; -- PERF_SRD_B_00068
213	else
214	-----
215	-- set ads intent point (speed change), see FMCS_19_20006075

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_SEQ_LEG_SEP.ADA (continued)

216	-- note Pred_Ksa will be 0.0 after call to Perf_Lgb_Pkg.Lgb_Seq_Rta_Leg so
217	-- you'll need to use a different variable if this line is moved
218	-----
219	if (Perf_Rta_Lfdata.Pred_Ksa /= 0.0) then
220	Perf_Ads_Intent_Pkg.Calc_Intermediate_Point;
221	end if;
201	222
202	if (Perf_Preds_Lfdata.Aircraft_State.GMT.Valid) then
203	 PROCESS THE RTA LEG SEQUENCE, COMPUTE NEW KSA, SET/CLEAR SPAD MSG
204	 Perf_Lgb_Pkg.Lgb_Seq_Rta_Leg;
205	end if;
206	Perf_Rta_Lfdata.Pred_Pastrta := True;
207	Perf_Rta_Lfdata.Pred_Ksa := 0.0;
223	if (Perf_Preds_Lfdata.Aircraft_State.GMT.Valid) then
224	-- PROCESS THE RTA LEG SEQUENCE, COMPUTE NEW KSA, SET/CLEAR SPAD MSG
225	Perf_Rta_Pkg.Rta_Seq_Rta_Leg;
226	end if;
227	Perf_Rta_Lfdata.Pred_Pastrta := True;
228	Perf_Rta_Lfdata.Pred_Ksa := 0.0;
229	end if;
208	230 end if;
209	231
210	232 end LGB_Seq_Leg;

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_SEQ_RTA_LEG_SEP.ADA

1	1	--
2	2	-- DATA_RIGHTS: HONEYWELL CONFIDENTIAL & PROPRIETARY
3	3	--
4	4	-- THIS WORK CONTAINS VALUABLE CONFIDENTIAL AND PROPRIETARY
5	5	-- INFORMATION. DISCLOSURE, USE OR REPRODUCTION OUTSIDE OF
6	6	-- HONEYWELL INTERNATIONAL, INC. IS PROHIBITED EXCEPT AS
7	7	-- AUTHORIZED IN WRITING. THIS UNPUBLISHED WORK IS PROTECTED BY
8	8	-- THE LAWS OF THE UNITED STATES AND OTHER COUNTRIES. IN THE
9	9	-- EVENT OF PUBLICATION, THE FOLLOWING NOTICE SHALL APPLY:
10	10	-- COPR. 2012 HONEYWELL INTERNATIONAL, INC. ALL RIGHTS RESERVED.
11	11	with Portable_Types_Pkg; use Portable_Types_Pkg;
12	12	with Ac_Position_Types; use Ac_Position_Types;
13	13	with Fmcs_Base_Types; use Fmcs_Base_Types;
14	14	with Fmcs_Fp_Guid_Btypes; use Fmcs_Fp_Guid_Btypes;
15	15	with Scratch_Pad_Iftypes;
16	16	with Cfp_Perf_Rta_Iftypes;
17	17	with Perf_Integrators_Lftypes;

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_SEQ_RTA_LEG_SEP.ADA (continued)

```

18      18 with Flight_Pln_Leg_Types;
19      19 with Flight_Pln_Hdr_Types;
20      20 with Alt_Profile_Iftypes;
21      21 with VGB_Iftypes;
22      22 with Perf_Air_Data_Pkg;
23      23 with FMCS_AEDB_Constants_Ifdata;
24      24 with OPS_CDK_Common_Mgr_Pkg;
25      25 with Perf_Aero_Speed_Pkg;
26      26 with Perf_Crz_Pkg;      use Perf_Crz_Pkg;
27      27
28      28 with Perf_Rp_Guidprms_Ifdata;
29      29
30      30 with Perf_Preds_Lfdata;
31      31 with Perf_Cmd_Spd_Pkg;
32      32 with Perf_LGB_Lfdata;
33      33 with Perf_Integrators_Lfdata;
34      34 with Perf_Wind_Lfdata;
35      35 with Perf_Rta_Lfdata;
36      36 with Perf_Idx_Msg_Flags_Lfdata;
37      37 with Perf_Crzalt_Lfdata;
38      38 with Idx_Profile_Ifdata;
39      39 with Perf_Profile_Lfdata;
40      40 with Perf_WTS_Lfdata;
41      41
42      42 with Math_Pkg; use Math_Pkg;
43      43 with Math_Rad_Pkg; use Math_Rad_Pkg;
44      44 with Ops_Cdk_Perf_Pdb_Mgr_Pkg;
45      45 with Fmci_Spad_Manager_Pkg;
46      46 with Act_Prov_Index_Manager;
47      47 with Ops_Lateral_Guidance_Buffer_Manager;
48      48 with Fpp_Common_Lgb_Wrap_Pkg;
49      49 with Fpp_Status_Type_Tpkg; use Fpp_Status_Type_Tpkg;
50      50
51      51
52      52 separate (Perf_LGB_Pkg)
53      53
54      54 procedure LGB_Seq_Rta_Leg (Initial_Est : in boolean := False) is
55      55     --!
56      56     -- ANCHOR:      FMCS_19_21023515
57      57     --
58      58     -- | @DESCRIPTION:
59      59     -- | This procedure is a performance predictions utility. It is used to process

```

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_SEQ_RTA_LEG_SEP.ADA (continued)

```

60 60  -- the predicted sequence of a leg with a RTA time constraint. It computes
61 61  -- the speed adjustment factor (Ksa) to attempt to meet the required time of
62 62  -- arrival, and sets status booleans to make it easier to find out information
63 63  -- about this computation without access to a debugger. It also controls part
64 64  -- of the display logic for the 'UNABLE RTA' and the 'UNABLE FLXXX AT RTA FIX'
65 65  -- scratchpad messages.
66 66  --
67 67  -- SPECIAL_CONSIDERATIONS:
68 68  --   Don't use 'Mach', 'Tas' or 'Ta' for local variable names because they are
69 69  --   complicated to display in the HADS debugger: (e.g. display mach * 1.0
70 70  --   (-or- display perf_lgb_pkg.lgb_seq_rta_leg:body.mach).
71 71  --   Many of the constants that control the new Ksa computation have been made
72 72  --   into variables in Perf_Rta_Lfdata. This is so that they can be changed
73 73  --   on the Memory Readout page (W/address/value) when memory write is enabled.
74 74  --   Read the description of local procedure Initial_Estimate. Procedure
75 75  --   Initial_Estimate should only be run when this module is called from
76 76  --   Restart_Check, so there is no chance of being restarted when we have
77 77  --   access to the LGB.
78 78  --   Because of the complexity of this module and the fact that it is only run
79 79  --   once during a predicitions pass, code readability is more important than
80 80  --   code execution efficiency.
81 81  --
82 82  --   SHARED_DATA_FOR:
83 83  --   IOBLK
84 84  --
85 85  -- TRANSLATION HISTORY:
86 86  -- This module is the equivalent of 747FANS RTALEGSEQ.PAS
87 87  --
88 88  -- REVISION_HISTORY:
89 89  --   DATE          SCR #          Programmer          DRCM#
90 90  --   01/24/96      8011.04      B. O'Laughlin      M777B_FMF_00550
91 91  -- Initial version for 777 Mkt-B.
92 92  --
93 93  --   02/02/96      8011.04      B. O'Laughlin      M777B_FMF_00948
94 94  -- Fixes to recommended takeoff time computation after first lab build.
95 95  --
96 96  --   02/13/96      8011.04      B. O'Laughlin      M777B_FMF_01091
97 97  -- Fixed 'UNABLE FLXXX AT RTA FIX' message logic and saved the altitude.
98 98  -- Moved all of the Rta initialization code from Restart_Check to here
99 99  -- (to be run when Initial_Est is input true).
100 100 -- Turn on the VG RTA speed filter when the solution is stable and the
101 101 -- aircraft is in cruise.

```

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_SEQ_RTA_LEG_SEP.ADA (continued)

```

102 102 --
103 103 -- 02/26/96      8011.04      B. O'Laughlin      M777B_FMF_01165
104 104 -- Fixes to turn off the VG RTA speed filter when the speed target changes
105 105 -- by a large amount.
106 106 --
107 107 -- 06/14/96      8544,8578,8579      B. O'Laughlin      M777B_FMF_02143
108 108 -- Extensive changes to module to improve RTA behavior near speed limits
109 109 -- and to improve readability. Added Max/Min_Ksa.
110 110 -- Fixed case where recommended takeoff time was in the past (instead of NOW).
111 111 -- Check if PE T/O time is in past before using it to compute ETA.
112 112 --
113 113 -- 08/04/96      8544      B. O'Laughlin      M777B_FMF_02637
114 114 -- Added Perf_Rta_Lfdata.k8 and k9 to make algorithm more controllable.
115 115 --
116 116 -- 04/02/97      9204.08      D. Turner/A. Comaduran M777B_FMF_04122
117 117 -- Extensive changes made to integrate required time of arrival and wind
118 118 -- trade step climb functions. Alternate method of computing KSA based on
119 119 -- previous guesses for KSA and the resulting ATE added. Computation of
120 120 -- "RTA" Cost Index - corresponding to the computed KSA - added. Additional
121 121 -- limits on the change in KSA between subsequent calls to this module added.
122 122 --
123 123 -- 04/15/97      9204.08      D. Turner/A. Comaduran M777B_FMF_04293
124 124 -- Fix errors in the Compute_Ksa module. Incorporated the pre-DRCM#4122
125 125 -- compute ksa logic into the logic added for Wind Trade Step Climbs (4122).
126 126 --
127 127 -- 04/17/97      9204.08      D. Turner      M777B_FMF_04349
128 128 -- The CRZ Alt used to compute the Rta Cost Index was changed to
129 129 -- Perf_Preds_Lfdata.VGB(CRZ2L).CmdSpdAlt so that it would always match
130 130 -- the T/C Cruise altitude.
131 131 --
132 132 -- 04/18/97      9204.08      D. Turner      M777B_FMF_04368
133 133 -- Fixed problem with Perf_WTS_Lfdata.Rta_Iter_Counter not being updated
134 134 -- until after the second (not first) pass through Predictions. Problem
135 135 -- was in the call to Common_Init from Predicted_Sequence. Also expanded
136 136 -- data sorting logic in Compute_Ksa so that "Bad_data" points will
137 137 -- automatically be moved to the lowest array location so that they can
138 138 -- be eliminated from consideration first.
139 139 --
140 140 -- 04/21/97      9204.08      D. Turner      M777B_FMF_04431
141 141 -- The cost biasing logic has been changed so that the step point data
142 142 -- is always stored in array location Rta iteration counter + 1, except
143 143 -- for case of first time thru predictions with an active rta.

```

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_SEQ_RTA_LEG_SEP.ADA (continued)

```

144 144 -- Added logic to not compute a new Rta_Ci when already on time.
145 145 --
146 146 -- 04/23/97 9204.08 D. Turner M777B_FMF_04460
147 147 -- Fixed a potential reset which would occur if tried to assign the local
148 148 -- Iter_Ctr a value of the rta iteration counter which is outside it's range.
149 149 -- Also fixed the ksa computation logic so that it would work properly
150 150 -- without sorting the ksa-ate data.
151 151 --
152 152 -- 04/24/97 9204.08 D. Turner M777B_FMF_04488
153 153 -- Added logic to copy the step-point storage and other data which is indexed
154 154 -- to a flight plan to the other flight plan (Act or Prov) when entering this
155 155 -- module in Initial_Est.
156 156 --
157 157 -- 05/01/97 9204.08 D. Turner M777B_FMF_04618
158 158 -- Added limiting logic to the KSA_Guess sub-unit to control the value of
159 159 -- Ksa_ATE_Slope.
160 160 --
161 161 -- 05/04/97 9204.08 D. Turner M777B_FMF_04642
162 162 -- Fixed a divide by zero reset in the Compute_Ksa module in the logic to sort
163 163 -- through the date to determine if a point of the opposite sign as current data
164 164 -- exists.
165 165 --
166 166 -- 05/12/97 9313 D. Turner M777B_FMF_04763
167 167 -- Added a Reset of the Max_Speed_Up and Max_Slow_Down booleans so that they could
168 168 -- not be true upon entry into this module.
169 169 --
170 170 -- 05/13/97 9313 D. Turner M777B_FMF_04763
171 171 -- Added logic to allow for negative cost indicies. The data used for the negative
172 172 -- cost index was obtained from running HSS cases with slow-down Rta's
173 173 --
174 174 -- 10/29/97 9629 D. Turner M777B_FMF_05977
175 175 -- Added checks to verify that we have a good rta pointer before attempting to
176 176 -- access the LGB with it. If it is bad, we simply set the Ksa to zero. This will
177 177 -- ruin the rta solution, but should prevent a reset.
178 178 --
179 179 -- 787 MODIFICATION_HISTORY
180 180 -- =====
181 181 -- 6/02/05 480.00 Rob Celesnik
182 182 -- Updated Crew Interface signal names from "Fmf_Ioc" to "Fmci".
183 183 --
184 184 -- 9/12/06 1594.00 Pat Caulfield
185 185 -- Cast ETA assignment to integer_32 since the predicted ETA in the leg is now float.

```

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_SEQ_RTA_LEG_SEP.ADA (continued)

```

186 186 --
187 187 -- 02/02/07      26984.03      Keri Kalvelage
188 188 -- To fix SBC compiler warnings, removed unused declarations for
189 189 -- Sav_Step_Dist_Data, Sav_Step_Alt_Data & rec_ctr.
190 190 --
191 191 -- 04/22/08      6276.06      Pat Caulfield
192 192 -- Switched to using get_lat_leg_Status to avoid resets and have a leg validity.
193 193 --
194 194 -- 07/07/2008    5295.00      Rajesh Chaubey
195 195 -- Updated the code to display the 'UNABLE RTA' message for the active route
196 196 -- when the MOD is erased.
197 197 --
198 198 -- 04/04/12      14535.02      Keri Kalvelage
199 199 -- In procedure Compute_Ksa changed the check for abs(Ate_Result > Spd_Adj_Tol)
200 200 -- to now be >=. The comparison of the Ate_Result and Spd_Adj_Tol = 0 was not
201 201 -- being handled and was causing the Ksa_Adj to be uninitialized value of 0.
202 202 -- This was causing the spd target to go to the unadj econ value and causing a spike.
203 203 --
204 204 -- 07/19/12      11939.02      B. O'Laughlin/Kevin Corbett
205 205 -- Several updates:
206 206 -- 1) Use the Pegasus version of the code related to negative cost index values
207 207 -- (in procedure Limit_Ksa) and add min and max limits to the Ci_Mach_Slope.
208 208 -- 2) Do not update the indexed Rta_Iter_Counter (that is being used by other
209 209 -- predictions files) until the end of a pass of preds.
210 210 -- 3) Modify the way that Flat_Bias_Factor works so that it does not rely on
211 211 -- At_Max_Bias and Apply_Flat_Bias, and Rta_Iter_Counter. Get rid of variables
212 212 -- At_Max_Bias and Apply_Flat_Bias, since they are not indexed by flight plan.
213 213 -- Replaced uses of Perf_WTS_Lfdata.Apply_Flat_Bias with the following logic
214 214 -- (Perf_WTS_Lfdata.Flat_Bias_Factor >= Perf_WTS_Lfdata.Initial_Flat_Bias_Factor).
215 215 -- Created indexed copy of Flat_Bias_Factor to hold the data for each flight
216 216 -- plan independently and new variable New_Flat_Bias_Factor to hold the
217 217 -- value to be used by the next pass, so that the value is not changed in the
218 218 -- middle of a pass at the RTA waypoint.
219 219 -- 4) Created indexed copy of Rta_CI (it is ok to update this in the middle
220 220 -- of a pass at the RTA waypoint since it shouldn't be used for the remainder
221 221 -- of that pass).
222 222 --
223 223 --!
224 224
225 225 ----- R E N A M E S -----
226 226 package r_Msg_Flags renames Perf_Idx_Msg_Flags_Lfdata; -- shortens name
227 227 package R_WTS_Pkg renames Perf_WTS_Lfdata;

```

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_SEQ_RTA_LEG_SEP.ADA (continued)

```

228 228 package R_Portable renames Portable_Types_Pkg;
229 229 ----- L O C A L   V A R I A B L E S -----
230 230
231 231 Eta           : R_Portable.Integer_32; -- est time of arrival (GMT in sec)
232 232 Flight_Time   : R_Portable.Integer_32; -- in flight time to rta waypoint
233 233 Fpln_Index    : Fmcs_Fp_Guid_Btypes.Act_Or_Prov_Type; -- pred's flight plan index
234 234 Init_Valid    : Boolean := True;      -- initialization valid flag
235 235 Good_Pointer  : Boolean := True;      -- RTA pointer in LGB valid flag (SCR 9629)
236 236 Ksa_New       : R_Portable.Float_32;  -- ksa for the next trip preds
237 237 Ksa_Old       : R_Portable.Float_32;  -- ksa used for current trip preds
238 238 RtaPtr        : R_Portable.Integer_32 :=
239 239             Perf_Preds_Lfdata.PrevNavPtr; -- local copy for efficiency
240 240 Rta_Time      : R_Portable.Integer_32; -- required time of arrival
241 241 Rta_Idx_Data  : Perf_Rta_Lfdata.Rta_Data_Rec_Type; -- local copy for efficiency
242 242 Rta_Type      : Ac_Position_Types.Time_Constraint_Type; -- After,Before,AT_Time
243 243 Time_Target   : R_Portable.Integer_32; -- targeted arrival time
244 244 Tko_Time     : Ops_Cdk_Perf_Pdb_Mgr_Pkg.
245 245             Int_32_Tko_Entry_Stat.State; -- takeoff time and a status
246 246 Loop_Ctr     : R_Portable.Integer_32; -- stores ksa, eta data: compare new Ksa guess to prev. ones
247 247 Iter_Ctr     : R_WTS_Pkg.Num_Values_Type := 1; -- local var to hold RTA Ctr
248 248 KSA_ATE_Slope : R_Portable.Float_32; -- slope of Ate vs. Ksa curve based on current iteration
249 249             -- and next closedt point
250 250 Min_ATE      : R_Portable.Integer_32; -- used to keep track of the minimum ATE, considering 9
251 251             -- most previous iterations
252 252 Max_ATE      : R_Portable.Integer_32; -- used to keep track of the maximum ATE, considering 9
253 253             -- most previous iterations
254 254 Num_Places   : R_Portable.Integer_32; -- used to limit how many Perf_RTA_Lfdata.Ksa_ETA_Rec
255 255             -- array locations will be examined to find the Min_ATE
256 256             -- and Max_ATE
257 257 Min_Location : R_Portable.Integer_32; -- Perf_RTA_Lfdata.Ksa_ETA_Rec array location for the
258 258             -- Min_ATE
259 259 Max_Location : R_Portable.Integer_32; -- Perf_RTA_Lfdata.Ksa_ETA_Rec array location for the
260 260             -- Max_ATE
261 261 Save_ATE     : R_Portable.Integer_32; -- local storage for an ATE record from
262 262             -- Perf_RTA_Lfdata.Ksa_ETA_Rec array
263 263 Save_Ksa     : R_Portable.Float_32;  -- local storage for an Ksa record from
264 264             -- Perf_RTA_Lfdata.Ksa_ETA_Rec array
265 265 Save_BadData : Boolean;              -- local storage for a Bad_Data record from
266 266             -- Perf_RTA_Lfdata.Ksa_ETA_Rec array
267 267 Min_ATE_Opp_Sign : R_Portable.Integer_32; -- Value of the Minimum ATE, considering 9 most
268 268             -- previous iterations, which is of the opposite sign
269 269             -- as the current iteration's ATE

```

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_SEQ_RTA_LEG_SEP.ADA (continued)

```

270 270 Max_ATE_Opp_Sign : R_Portable.Integer_32; -- Value of the Maximum ATE, considering 9 most
271 271                                     -- previous iterations, which is of the opposite sign
272 272                                     -- as the current iteration's ATE
273 273 Opposite_Sign   : Boolean; -- If this boolean is true, the ATE for the previous iteration which
274 274                                     -- is under consideration is of the opposite sign as the current
275 275                                     -- iteration's ATE
276 276 Opposite_Sign_Exist : Boolean; -- If this boolean is true, there exists at least one ATE from a
277 277                                     -- previous pass which is of the opposite sign as the current
278 278                                     -- iteration's ATE
279 279 CI_Guess          : R_Portable.Integer_32; -- Initial guess for "RTA" Cost Index
280 280 WOD               : R_Portable.Float_32; -- Gross weight / delta
281 281 Press_Ratio       : R_Portable.Float_32; -- Atmospheric pressure ratio
282 282 Theta             : R_Portable.Float_32; -- Atmospheric temperature ratio
283 283 Theta_Std         : R_Portable.Float_32; -- standard day temperature ratio
284 284 Vel_Sound         : R_Portable.Float_32; -- speed of sound
285 285 Corr_Theta        : R_Portable.Float_32; -- corrected temperature ratio
286 286 Base_CI           : R_Portable.Float_32; -- Cost Index at Top of Climb altitude
287 287 Corr_Base_CI      : R_Portable.Float_32; -- Corrected Cost Index at Top of Climb altitude
288 288 Base_Mach         : R_Portable.Float_32; -- Mach number at Top of Climb altitude based on Base_CI
289 289 Tgt_Mach          : R_Portable.Float_32; -- Mach Number at Top of Climb altitude with computed Ksa
290 290                                     -- applied
291 291 Previous_Leg      : FMCS_Fp_Guid_Btypes.GuidanceIdx;
292 292 Next_Leg          : FMCS_Fp_Guid_Btypes.GuidanceIdx;
293 293 Prddataseq       : Portable_Types_Pkg.Integer_32;
294 294 Corefp_Status     : Fpp_Status_Type_Tpkg.Fpp_Status_Type_T;
295 295
296 296 Type Sav_Step_Dist_Type is Array(Alt_Profile_Iftypes.NumScPts) of
297 297                                     FMCS_Base_Types.Float_32_Valid.Normal; -- make of the same type as
298 298                                     -- Perf_Rta_Lfdata.Step_Dist_Rec
299 299 CrzAlt : R_Portable.Float_32; -- Local copy of Top of Climb Cruise altitude
300 300 Initialize_To_Iter_Ctr : Boolean; -- If true then the Step point records will be rolled back
301 301                                     -- from Iter_Ctr and not (Iter_Ctr + 1)
302 302 Copy_From_Opposite_Fpln : Boolean; -- If true, when roll back from Iter_Ctr or Iter_Ctr+1 in
303 303                                     -- Common_Init, get the data from the opposite flightplan
304 304 Rollback_Fpln : Fmcs_Fp_Guid_Btypes.Act_Or_Prov_Type; -- The correct Fplnindex to use in rollback
305 305
306 306
307 307
308 308 ----- L O C A L   C O N S T A N T S -----
309 309 -- most of the local constants were moved to Perf_Rta_Lfdata so that they would
310 310 -- be global and could be changed from the memory readout page for debug.
311 311 One_Day          : R_Portable.Integer_32 := 86400; -- 86400 sec = 24 hr

```


File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_SEQ_RTA_LEG_SEP.ADA (continued)

```

312 312 Half_Day      : R_Portable.Integer_32 := 43200; -- 43200 sec = 12 hr
313 313
314 314 ----- L O C A L   F U N C T I O N   (Time_Map) -----
315 315 -- Local function Time_Map exists to make the code more readable.  It takes
316 316 -- an input time (in seconds) and maps it to be (0 <= time < One_Day).
317 317 -- NOTE: if the module dosen't compile, the pragma inline will generate:
318 318 -- "Inline expansion of TIME_MAP is not achieved here: body is not available"
319 319 -- don't worry about this, it will go away when the module compiles.
320 320
321 321 function Time_Map(Time_1 : R_Portable.Integer_32)
322 322     return R_Portable.Integer_32 is
323 323 begin
324 324     return (Time_1 mod One_Day);
325 325 end;
326 326 pragma inline (Time_Map);
327 327
328 328 function Time_Map(Time_1 : R_Portable.Integer_32)
329 329     return R_Portable.Float_32 is
330 330 begin
331 331     return R_Portable.Float_32(Time_1 mod One_Day);
332 332 end;
333 333 pragma inline (Time_Map);
334 334
335 335 ----- L O C A L   P R O C E D U R E   (Common_Init) -----
336 336 -- This procedure is used because Initial_Estimate and Predicted_Sequence
337 337 -- both use the following initilizations.
338 338 --
339 339 procedure Common_Init is
340 340 begin
341 341
342 342     if Copy_From_Opposite_Fpln then
343 343         if (Fpln_Index = Act_Prov_Index_Manager.Prov_Index) then
344 344             Rollback_Fpln := Act_Prov_Index_Manager.Act_Index;
345 345         else
346 346             Rollback_Fpln := Act_Prov_Index_Manager.Prov_Index;
347 347         end if;
348 348     else
349 349         Rollback_Fpln := Fpln_Index;
350 350     end if;
351 351
352 352     if Initialize_To_Iter_Ctr then
353 353         -- before rolling the counter back to 1, copy the step point locations

```

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_SEQ_RTA_LEG_SEP.ADA (continued)

```

354 354 -- into the first array locations so that we have them
355 355 for Loop_Ctr in Alt_Profile_Iftypes.NumScPts loop
356 356     R_WTS_Pkg.Step_Dist_Rec(Fpln_Index,Loop_Ctr,1).Data :=
357 357         R_WTS_Pkg.Step_Dist_Rec(Rollback_Fpln,Loop_Ctr, Iter_Ctr).Data;
358 358     R_WTS_Pkg.Step_Dist_Rec(Fpln_Index,Loop_Ctr,1).Valid :=
359 359         R_WTS_Pkg.Step_Dist_Rec(Rollback_Fpln,Loop_Ctr, Iter_Ctr).Valid;
360 360 end loop;
361 361 else
362 362     for Loop_Ctr in Alt_Profile_Iftypes.NumScPts loop
363 363         R_WTS_Pkg.Step_Dist_Rec(Fpln_Index,Loop_Ctr,1).Data :=
364 364             R_WTS_Pkg.Step_Dist_Rec(Rollback_Fpln,Loop_Ctr, Iter_Ctr+1).Data;
365 365         R_WTS_Pkg.Step_Dist_Rec(Fpln_Index,Loop_Ctr,1).Valid :=
366 366             R_WTS_Pkg.Step_Dist_Rec(Rollback_Fpln,Loop_Ctr, Iter_Ctr+1).Valid;
367 367     end loop;
368 368 end if;
369 369
370 370 -- copy step climb data from current 'pass' to index 1
371 371 R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, 1).Ksa_Guess :=
372 372     R_WTS_Pkg.KSA_ETA_REC(Rollback_Fpln, Iter_Ctr).Ksa_Guess;
373 373 R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, 1).Ate_Result :=
374 374     R_WTS_Pkg.KSA_ETA_REC(Rollback_Fpln, Iter_Ctr).Ate_Result;
375 375 R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, 1).Bad_Data :=
376 376     R_WTS_Pkg.KSA_ETA_REC(Rollback_Fpln, Iter_Ctr).Bad_Data;
377 377
378 378 for Loop_Ctr in Alt_Profile_Iftypes.NumScPts loop
379 379     for Rec_Ctr in 2..R_WTS_Pkg.Num_Values_Type'last loop
380 380         R_WTS_Pkg.Step_Dist_Rec(Fpln_Index,Loop_Ctr, Rec_Ctr).Data := 0.0;
381 381         R_WTS_Pkg.Step_Dist_Rec(Fpln_Index,Loop_Ctr, Rec_Ctr).Valid := False;
382 382     end loop;
383 383 end loop;
384 384
385 385 R_WTS_Pkg.Rta_Iter_Counter(Fpln_Index) := 1;
386 386
387 387 Iter_Ctr := R_WTS_Pkg.Rta_Iter_Counter(Fpln_Index);
388 388 Perf_WTS_Lfdata.Flat_Bias_Factor := 0.0;
389 389 Perf_WTS_Lfdata.New_Flat_Bias_Factor := 0.0;
390 390 -- The value of New_Flat_Bias_Factor is now the main way of managing the logic
391 391
392 392 -- 1/23/97 refresh the ksa_eta_rec storage array
393 393 Loop_Ctr := 2;
394 394
395 395 while Loop_Ctr < R_WTS_Pkg.Num_Ksa_ETA_Records loop

```

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_SEQ_RTA_LEG_SEP.ADA (continued)

```

396 396 R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, Loop_Ctr).Ksa_Guess := 0.0;
397 397 R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, Loop_Ctr).Ate_Result := 0;
398 398 Loop_Ctr := Loop_Ctr + 1;
399 399 end loop;
400 400
401 401 end Common_Init;
402 402
403 403 ----- L O C A L   P R O C E D U R E   (Initial_Est) -----
404 404 -- Local procedure Initial_Estimate is run when the Initial_Est input boolean
405 405 -- is True. This means that RTA has just been activated and an initial estimate
406 406 -- of the Ksa should be attempted. Since RTA has just been activated, there
407 407 -- are no ETA predictions for the RTA waypoint in the provisional route.
408 408 -- However, if the RTA waypoint exists in the active route, then we can use
409 409 -- it's ETA to make an initial guess at the Ksa.
410 410 --
411 411 -- Initial_Estimate is called when Lgb_Seq_Rta_Leg is called from Restart_Check
412 412 -- so this is running from CDK's task (but it shouldn't interrupt the Perf_BG
413 413 -- task in the middle of Leg_Seq_Rta_Leg, because RTA isn't active).
414 414 -- Since this is running from Restart_Check, there is no chance that it will
415 415 -- be restarted while it has access to the LGB.
416 416 --
417 417 -- This code is in this module because the initial estimate uses the same Ksa
418 418 -- and recommended takeoff time computation logic that is run when trip
419 419 -- predictions sequence the RTA waypoint.
420 420
421 421 procedure Initial_Estimate is
422 422   -- local variables
423 423   Fpln_ID      : FMCS_Fp_Guid_Btypes.Actfpln_Type;
424 424   Route_ID     : Fmcs_Fp_Guid_Btypes.Gbthreadtype;
425 425   Act_Gleg     : Flight_Pln_Leg_Types.Leg_Rec;
426 426   Prov_Gleg    : Flight_Pln_Leg_Types.Leg_Rec;
427 427   Act_Access_Id : OPS_Lateral_Guidance_Buffer_Manager.Access_ID_Type;
428 428   Prov_Access_Id : OPS_Lateral_Guidance_Buffer_Manager.Access_ID_Type;
429 429   Act_Index    : Fmcs_Fp_Guid_Btypes.Act_Or_Prov_Type;
430 430
431 431 begin
432 432
433 433   -- use provisional flight plan index for rest of module
434 434   Fpln_Index := Act_Prov_Index_Manager.Prov_Index;
435 435   Good_Pointer := True;
436 436
437 437   -- Before going into Common_Init, have to determine where latest step point

```

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_SEQ_RTA_LEG_SEP.ADA (continued)

```

438 438 -- data is stored. In this case, its Iter_Ctr+1 since the non-rta step point
439 439 -- data is always stored in array location rta iteration counter + 1.
440 440 Initialize_To_Iter_Ctr := False;
441 441 Copy_From_Opposite_Fpln := True;
442 442
443 443 -- Initialize steps data, ksa ate storage array, rta iter counter and
444 444 -- biasing flags
445 445 Common_Init;
446 446
447 447 -- initialize the VG rta speed filter (off)
448 448 Perf_Rp_Guidprms_Ildata.RTA_Spd_Tgt_Filter_On := False;
449 449
450 450 -- initialize the message flags
451 451 r_Msg_Flags.Unable_Rta_Msg_Issued(Fpln_Index) := False;
452 452 r_Msg_Flags.Unable_Rta_Msg_Cleared(Fpln_Index) := False;
453 453 r_Msg_Flags.Unable_Flxxx_At_Rta_Fix_Msg_Issued(Fpln_Index) := False;
454 454 r_Msg_Flags.Unable_Flxxx_At_Rta_Fix_Msg_Cleared(Fpln_Index) := False;
455 455
456 456 -- Set up local copy of rta data and initialize only those values
457 457 -- that will be read from before they are written to.
458 458 Rta_Idx_Data := Perf_Rta_Lfdata.Idx_Data(Fpln_Index); -- OLD data
459 459 Rta_Idx_Data.Preds_Status.Rta_Happy := False;
460 460 Rta_Idx_Data.Msg_Counter := 0;
461 461
462 462 Rta_Idx_Data.Ksa := 0.0; -- rta was not active when the act preds were done
463 463 -- WTS 3/16/97: Refresh the Rta_CostIndex
464 464 R_WTS_Pkg.Rta_CI := Ops_CDK_Common_Mgr_Pkg.Cost_Index.Data;
465 465 Rta_Idx_Data.Max_Ksa := 200.0; -- don't limit first guess
466 466 Rta_Idx_Data.Min_Ksa := -200.0; -- don't limit first guess
467 467 Rta_Idx_Data.Preds_Status.At_Max_Speed_Up := False;
468 468 Rta_Idx_Data.Preds_Status.At_Max_Slow_Down := False;
469 469 -- Note: It would be nice (but difficult) to compute the actual thrust
470 470 -- limited speeds by calling Max_Spd_At_Alt, we may want to do this someday
471 471 -- so the initial VGB speeds will be thrust limited.
472 472 Rta_Idx_Data.Min_Spd_At_Crzalt := (Speed => 0.0, Speed_Is_Thrust_Lim => False,
473 473 Valid => False, Alt => 0.0, Gwt => 0.0);
474 474 Rta_Idx_Data.Max_Spd_At_Crzalt := (Speed => 1.0, Speed_Is_Thrust_Lim => False,
475 475 Valid => False, Alt => 0.0, Gwt => 0.0);
476 476 -- Rta_Idx_Data.Spd_Gen_Status will be set when the speed generators are run
477 477 -- Rta_Idx_Data.Preds_Status, Old_Ksa, Ate, Tup, Tdn, Crz_Dist, and Crz_Time
478 478 -- will be set in this module
479 479 -- Rta_Idx_Data.Rcmd_Takeoff was initialized in Restart_Clear_Perf_Data

```

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_SEQ_RTA_LEG_SEP.ADA (continued)

```
480 480
481 481 -- get access to prov lgb
482 482 Ops_Lateral_Guidance_Buffer_Manager.Requestlgb
483 483   (Lgb_Process_Id => Ops_Lateral_Guidance_Buffer_Manager.Preds_Restart,
484 484    Access_Id      => Prov_Access_Id,
485 485    Lgb_Operation  => Flight_Pln_Hdr_Types.Rte_Read,
486 486    Route_Id      => Fmcs_Fp_Guid_Btypes.Gbprov);
487 487
488 488 -- get provisional header rta pointer
489 489 RtaPtr := Ops_Lateral_Guidance_Buffer_Manager.RTA_Fix_Ptr(Prov_Access_Id);
490 490
491 491 -- SCR 9629 Only attempt to access the LGB if there is a good rta pointer.
492 492 if RtaPtr /= 0 then
493 493   -- get rta leg from the provisional lgb
494 494   Fpp_Common_Lgb_Wrap_Pkg.Get_Lat_Leg_Status
495 495     (Process_ID   => Fmcs_Fp_Guid_Btypes.Preds_Restart_C,
496 496      Leg_Index    => RtaPtr,
497 497      Lateral_Leg  => Prov_Gleg,
498 498      Previous_Leg => Previous_Leg,
499 499      Next_Leg    => Next_Leg,
500 500      Check_Status => true,
501 501      Corefp_Status => Corefp_Status,
502 502      Hdr_Prddataseq => Prddataseq);
503 503
504 504   -- make sure we actually got a leg
505 505   Init_Valid := Corefp_Status = Fppsucces;
506 506
507 507   -- make local copy of rta time constraint
508 508   Rta_Time := Prov_Gleg.Fpln_Data.RTA_Time;
509 509   Rta_Type := Prov_Gleg.Fpln_Data.RTA_Time_Type;
510 510   -- WTS store this rta target as old rta
511 511   R_WTS_Pkg.Old_RTA_Tgt(Fpln_Index) := Rta_Time;
512 512
513 513 else -- SCR 9629 bad RTA pointer
514 514   Init_Valid := False; -- get out of module
515 515 end if;
516 516
517 517 -- release access to prov lgb
518 518 Ops_Lateral_Guidance_Buffer_Manager.Releaselgb
519 519   (Access_Id => Prov_Access_Id);
520 520
521 521 if (Init_Valid) then
```

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_SEQ_RTA_LEG_SEP.ADA (continued)

```

522 522
523 523      -- determine which lgb is act lgb
524 524      Fpln_ID := Ops_Lateral_Guidance_Buffer_Manager.Actfpln;
525 525      if (Fpln_ID = FMCS_Fp_Guid_Btypes.Fpln1) then
526 526          Route_ID := FMCS_Fp_Guid_Btypes.Gbrtel;
527 527      elsif (Fpln_ID = FMCS_Fp_Guid_Btypes.Fpln2) then
528 528          Route_ID := FMCS_Fp_Guid_Btypes.Gbrte2;
529 529      else
530 530          Init_Valid := False;  -- get out of module
531 531      end if;
532 532  end if;
533 533
534 534  if (Init_Valid) then
535 535
536 536      -- get access to act lgb
537 537      Ops_Lateral_Guidance_Buffer_Manager.Requestlgb
538 538      (Lgb_Process_Id => Ops_Lateral_Guidance_Buffer_Manager.Preds_Restart,
539 539       Access_Id      => Act_Access_Id,
540 540       Lgb_Operation  => Flight_Pln_Hdr_Types.Rte_Read,
541 541       Route_Id       => Route_ID);
542 542
543 543      -- get active fpln rta pointer
544 544      RtaPtr := Ops_Lateral_Guidance_Buffer_Manager.RTA_Fix_Ptr(Act_Access_Id);
545 545
546 546      if RtaPtr /= 0 then
547 547          Fpp_Common_Lgb_Wrap_Pkg.Get_Lat_Leg_Status
548 548          (Process_ID   => Fmcs_Fp_Guid_Btypes.Preds_Restart_C,
549 549           Leg_Index    => RtaPtr,
550 550           Lateral_Leg  => Act_Gleg,
551 551           Previous_Leg => Previous_Leg,
552 552           Next_Leg    => Next_Leg,
553 553           Check_Status => true,
554 554           Corefp_Status => Corefp_Status,
555 555           Hdr_Prddataseq => Prddataseq);
556 556      else
557 557          Init_Valid := false;
558 558      end if;
559 559
560 560      -- release access to act lgb
561 561      Ops_Lateral_Guidance_Buffer_Manager.Releaselgb
562 562      (Access_Id => Act_Access_Id);
563 563

```

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_SEQ_RTA_LEG_SEP.ADA (continued)

```

564 564 -- check if the act and prov routes are similar (we could add more
565 565 -- checks here: does the act wypt disttodest match, etc.)
566 566 if not (Corefp_Status = Fppsucceed) or else ((not Act_Gleg.Perf_Data.PrdETAFixVal) or else
567 567 (Act_Gleg.Fpln_Data.FixIdent /= Prov_Gleg.Fpln_Data.FixIdent) or else
568 568 (not Prov_Gleg.Fpln_Data.Matchactfpln)) then
569 569 Init_Valid := False; -- get out of module
570 570 end if;
571 571 end if;
572 572
573 573 if (Init_Valid) then -- setup the rta data
574 574 -- get the rta waypoint eta in the active route
575 575 Eta := Portable_Types_Pkg.Integer_32 ( Act_Gleg.Perf_Data.PrdETAToFix );
576 576
577 577 -- get the index of the active route for: toc, tod, rcmd tko time
578 578 Act_Index := Act_Prov_Index_Manager.Act_Index;
579 579
580 580 -- compute distance and time spent in cruise before the rta waypoint
581 581 -- note that t/c data is current a/c data when a/c is in cruise and beyond
582 582 if (Act_Gleg.Perf_Data.Fltmode <= Fmcs_Fp_Guid_Btypes.GBClimb) then
583 583 Rta_Idx_Data.Crz_Time := 0.0;
584 584 Rta_Idx_Data.Crz_Dist := 0.0;
585 585 elsif (Act_Gleg.Perf_Data.Fltmode <= Fmcs_Fp_Guid_Btypes.GBCruise) then
586 586 Rta_Idx_Data.Crz_Time := Time_Map(
587 587 Eta - Idx_Profile_Ildata.Itocperldata(Act_Index).Eta.Data);
588 588 Rta_Idx_Data.Crz_Dist :=
589 589 Idx_Profile_Ildata.Itocperldata(Act_Index).Disttodest.Data -
590 590 Act_Gleg.Common_Data.Fixdistodest;
591 591 else
592 592 Rta_Idx_Data.Crz_Time := Time_Map(
593 593 Idx_Profile_Ildata.Itodperldata (Act_Index).Eta.Data -
594 594 Idx_Profile_Ildata.Itocperldata (Act_Index).Eta.Data);
595 595 Rta_Idx_Data.Crz_Dist :=
596 596 Idx_Profile_Ildata.Itocperldata(Act_Index).Disttodest.Data -
597 597 Idx_Profile_Ildata.Itodperldata(Act_Index).Disttodest.Data;
598 598 end if;
599 599
600 600 -- assume that entire time in cruise is available for speed up/slow down
601 601 Rta_Idx_Data.Tup := Rta_Idx_Data.Crz_Time;
602 602 Rta_Idx_Data.Tdn := Rta_Idx_Data.Crz_Time;
603 603 Rta_Idx_Data.Tup_At_Ksa_Minus := Rta_Idx_Data.Crz_Time;
604 604 Rta_Idx_Data.Tdn_At_Ksa_Plus := Rta_Idx_Data.Crz_Time;
605 605

```

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_SEQ_RTA_LEG_SEP.ADA (continued)

```

606 606 -- note: Perf_Preds_Lfdata.Aircraft_State.Airborne and
607 607 -- Perf_Rta_Lfdata.Pred_Gmt_Unmodified.Data should have been set
608 608 -- fairly recently if the preds for the active route are valid, so
609 609 -- go ahead and use them (instead of getting a current copy)
610 610
611 611 -- compute the time spent flying to the rta waypoint
612 612 if ((not Perf_Preds_Lfdata.Aircraft_State.Airborne) and then
613 613   (Ops_Cdk_Perf_Pdb_Mgr_Pkg.Takeoff_Time_Is_Pilot_Entered)) then
614 614   Ops_Cdk_Perf_Pdb_Mgr_Pkg.Get_Takeoff_Time(Tko_Time,Act_Index);
615 615   -- there is only one pilot entered takeoff time, fpln_index dosen't matter
616 616   Flight_Time := Time_Map(Eta - Tko_Time.Data);
617 617 else
618 618   Flight_Time := Time_Map(Eta - Perf_Rta_Lfdata.Pred_Gmt_Unmodified.Data);
619 619 end if;
620 620 end if;
621 621
622 622 end Initial_Estimate;
623 623
624 624 ----- L O C A L   P R O C E D U R E   (Predicted_Sequence) -----
625 625 procedure Predicted_Sequence is
626 626   -- local variables
627 627   Time_Delta      : R_Portable.Integer_32; -- time difference
628 628 begin
629 629   -- use predictions flight plan index for rest of module
630 630   Fpln_Index := Perf_Preds_Lfdata.Vtpfplnindex;
631 631   Rta_Idx_Data := Perf_Rta_Lfdata.Idx_Data(Fpln_Index);
632 632
633 633   -- save predicted Tup, Tdn, Max_Ksa, Min_Ksa to indexed copies
634 634   Rta_Idx_Data.Tup := Perf_Rta_Lfdata.Pred_Tup;
635 635   Rta_Idx_Data.Tdn := Perf_Rta_Lfdata.Pred_Tdn;
636 636   Rta_Idx_Data.Tup_At_Ksa_Minus := Perf_Rta_Lfdata.Pred_Tup_At_Ksa_Minus;
637 637   Rta_Idx_Data.Tdn_At_Ksa_Plus := Perf_Rta_Lfdata.Pred_Tdn_At_Ksa_Plus;
638 638   Rta_Idx_Data.Max_Ksa := Perf_Rta_Lfdata.Max_Ksa;
639 639   Rta_Idx_Data.Min_Ksa := Perf_Rta_Lfdata.Min_Ksa;
640 640
641 641   Flight_Time := R_Portable.Integer_32(
642 642     Perf_Integrators_Lfdata.IntProgBuf.TProg);
643 643
644 644   -- SCR 9629 This SCR was for a bad RTA pointer during the initial call
645 645   --           but we always want to protect against a bad rta pointer
646 646   if Rtaptr /= 0 then
647 647     Good_Pointer := True;

```


File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_SEQ_RTA_LEG_SEP.ADA (continued)

```

648 648
649 649 -- make local copy of rta time constraint
650 650 Rta_Time := Perf_LGB_Lfdata.LGB(RtaPtr).Fpln_Data.RTA_Time;
651 651 Rta_Type := Perf_LGB_Lfdata.LGB(RtaPtr).Fpln_Data.RTA_Time_Type;
652 652
653 653 -- reset or increment the rta loopcounter (and reset rta_happy on a firstpass)
654 654 if (Perf_Preds_Lfdata.Vtplogic.Firstpass) then
655 655     Rta_Idx_Data.Msg_Counter := 1;
656 656     Rta_Idx_Data.Preds_Status.Rta_Happy := False;
657 657
658 658     -- Before calling Common_Init, must determine where the latest step point
659 659     -- data is stored. In this case, its Iter_Ctr + 1
660 660     If R_WTS_Pkg.Rta_Iter_Counter(Fpln_Index) >= R_WTS_Pkg.Num_Ksa_ETA_Records then
661 661         -- Set iter_ctr to maximum - 1 because will be resetting to (iter_ctr + 1)
662 662         -- in common_init
663 663         Iter_Ctr := R_WTS_Pkg.Num_Ksa_ETA_Records - 1;
664 664     else
665 665         Iter_Ctr := R_WTS_Pkg.Rta_Iter_Counter(Fpln_Index);
666 666     end if;
667 667     Initialize_To_Iter_Ctr := False;
668 668     Copy_From_Opposite_Fpln := False;
669 669
670 670     if (Perf_WTS_Lfdata.Old_Rta_Tgt(Fpln_Index) /= RTA_Time) then
671 671         -- Re-Initialize steps data, ksa ate storage array, rta iter counter etc.
672 672         -- Steps will be unlatched and free to move
673 673         Common_Init;
674 674     else
675 675         -- for other mods, the arrays will not be cleared, because the step location
676 676         -- checking logic will filter out any bad data before curve fitting to find
677 677         -- the next ksa. However, steps will be unlatched for any Mods (which
678 678         -- as of 7/18/12 is now done in Vtp_Init by setting Flat_Bias_Factor to
679 679         -- 0.0 when the Firstpass flag is true).
680 680         Perf_WTS_Lfdata.Flat_Bias_Factor := 0.0; -- not necessary
681 681         Perf_WTS_Lfdata.New_Flat_Bias_Factor := 0.0; -- not necessary
682 682         Perf_WTS_Lfdata.Rta_Iter_Counter(Fpln_Index) :=
683 683             Perf_WTS_Lfdata.Rta_Iter_Counter(Fpln_Index) + 1;
684 684     end if;
685 685
686 686     elsif (Rta_Idx_Data.Msg_Counter >= 30000) then -- 30000 is less than maxint
687 687         Rta_Idx_Data.Msg_Counter := 6; -- prevent overflow
688 688
689 689     elsif (R_WTS_Pkg.Rta_Iter_Counter(Fpln_Index) >= 30000) then

```

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_SEQ_RTA_LEG_SEP.ADA (continued)

```

690 690      -- 30000 is less than maxint
691 691      R_WTS_Pkg.Rta_Iter_Counter(Fpln_Index) := 11;
692 692      -- prevent overflow
693 693      -- go to 11 so that array locations not overwritten
694 694  else
695 695      Rta_Idx_Data.Msg_Counter := Rta_Idx_Data.Msg_Counter + 1;
696 696      -- WTS : increment iteration counter
697 697      R_WTS_Pkg.Rta_Iter_Counter(Fpln_Index) :=
698 698          R_WTS_Pkg.Rta_Iter_Counter(Fpln_Index) + 1;
699 699  end if;
700 700
701 701  -- The previous setup of Iter_Ctr was for the purposes of Common_Init only.
702 702  -- This setup is also necessary. Don't move this code.
703 703  If R_WTS_Pkg.Rta_Iter_Counter(Fpln_Index) >= R_WTS_Pkg.Num_Ksa_ETA_Records then
704 704      Iter_Ctr := R_WTS_Pkg.Num_Ksa_ETA_Records;
705 705  else
706 706      Iter_Ctr := R_WTS_Pkg.Rta_Iter_Counter(Fpln_Index);
707 707  end if;
708 708
709 709  -- there is no guaranty that lgb_store_data has run to store the eta, so
710 710  -- compute the predicted eta for the leg; besides, if there is a recommended
711 711  -- takeoff time, we don't want the eta to be based on it
712 712  if ( (not Perf_Preds_Lfdata.Aircraft_State.Airborne) and then
713 713      Ops_Cdk_Perf_Pdb_Mgr_Pkg.Takeoff_Time_Is_Pilot_Entered ) then
714 714
715 715      Ops_Cdk_Perf_Pdb_Mgr_Pkg.Get_Takeoff_Time(Tko_Time,Fpln_Index);
716 716      Time_Delta := Tko_Time.Data - Perf_Rta_Lfdata.Pred_Gmt_Unmodified.Data;
717 717
718 718      -- make sure pilot entered takeoff time is in the future
719 719      if ( ((Time_Delta > 0) and then (Time_Delta < Half_Day)) or else
720 720          ((Time_Delta < -Half_Day) and then (Time_Delta > -One_Day)) )
721 721      then -- pilot entered takeoff time is in future, use it
722 722          Eta := Time_Map(Tko_Time.Data + Flight_Time);
723 723      else -- use the GMT variable that was not adjusted for pilot entered time
724 724          Eta := Time_Map(Perf_Rta_Lfdata.Pred_Gmt_Unmodified.Data + Flight_Time);
725 725      end if;
726 726
727 727  else
728 728      -- use the GMT variable that was not adjusted for recommended takeoff time
729 729      Eta := Time_Map(
730 730          Perf_Rta_Lfdata.Pred_Gmt_Unmodified.Data + Flight_Time);
731 731  end if;

```

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_SEQ_RTA_LEG_SEP.ADA (continued)

```

732 732
733 733 -- compute distance and time spent in cruise before the rta waypoint
734 734 -- note that t/c data is current a/c data when a/c is in cruise and beyond
735 735 if (Perf_Preds_Lfdata.Desiredphase <= Fmcs_Base_Types.Climb) then
736 736     Rta_Idx_Data.Crz_Time := 0.0;
737 737     Rta_Idx_Data.Crz_Dist := 0.0;
738 738 elsif (Perf_Preds_Lfdata.Desiredphase <= Fmcs_Base_Types.Cruise) then
739 739     Rta_Idx_Data.Crz_Time := Time_Map(R_Portable.Integer_32(
740 740         Eta) - Perf_Profile_Lfdata.Tocperfddata.Eta.Data);
741 741     Rta_Idx_Data.Crz_Dist :=
742 742         Perf_Profile_Lfdata.Tocperfddata.Disttodest.Data -
743 743         Perf_Integrators_Lfdata.IntProgBuf.XProg;
744 744 else
745 745     Rta_Idx_Data.Crz_Time := Time_Map(
746 746         Perf_Profile_Lfdata.Todperfddata.Eta.Data -
747 747         Perf_Profile_Lfdata.Tocperfddata.Eta.Data);
748 748     Rta_Idx_Data.Crz_Dist :=
749 749         Perf_Profile_Lfdata.Tocperfddata.Disttodest.Data -
750 750         Perf_Profile_Lfdata.Todperfddata.Disttodest.Data;
751 751 end if;
752 752
753 753 else -- SCR 9629 Bad RTA Pointer
754 754     Good_Pointer := False;
755 755 end if;
756 756
757 757 end Predicted_Sequence;
758 758
759 759 ----- L O C A L   P R O C E D U R E   (Compute_Ate) -----
760 760 procedure Compute_Ate is
761 761     -- local variables
762 762     Data_Use_Tolerance : R_Portable.Integer_32 := 1;
763 763 begin
764 764     -- Do NOT mess with the Fpln_Index variable.  It has been set already
765 765
766 766     -- make local copy of ksa that was used to generate the current trip preds
767 767     Ksa_Old := Rta_Idx_Data.Ksa;
768 768
769 769     -- determine time target based on rta type
770 770     if (Rta_Type = Ac_Position_Types.At_Time) then
771 771         Time_Target := Rta_Time;
772 772     elsif (Rta_Type = Ac_Position_Types.After) then
773 773         Time_Target := Rta_Time + Perf_Rta_Lfdata.Message_Tol;

```

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_SEQ_RTA_LEG_SEP.ADA (continued)

```

774 774 elsif (Rta_Type = Ac_Position_Types.Before) then
775 775     Time_Target := Rta_Time - Perf_Rta_Lfdata.Message_Tol;
776 776 end if;
777 777
778 778 -- compute arrival time error (ate) at the rta waypoint
779 779 Rta_Idx_Data.Ate := Eta - Time_Target;
780 780 if (Rta_Idx_Data.Ate > Half_Day) then
781 781     Rta_Idx_Data.Ate := Rta_Idx_Data.Ate - One_Day;
782 782 elsif (Rta_Idx_Data.Ate < -Half_Day) then
783 783     Rta_Idx_Data.Ate := Rta_Idx_Data.Ate + One_Day;
784 784 end if;
785 785
786 786 -- WTS : Store Ksa_ETA record data.  If neccessary, slide all
787 787 -- elements back one place to store last R_WTS_Pkg.Num_Ksa_ETA_Records
788 788 -- number of records.
789 789 if (R_WTS_Pkg.Rta_Iter_Counter(Fpln_Index) > R_WTS_Pkg.Num_Ksa_ETA_Records) then
790 790     Loop_Ctr := 1;
791 791     While Loop_Ctr < R_WTS_Pkg.Num_Ksa_ETA_Records loop
792 792         R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, Loop_Ctr).Ksa_Guess :=
793 793             R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, Loop_Ctr + 1).Ksa_Guess;
794 794         R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, Loop_Ctr).ATE_Result :=
795 795             R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, Loop_Ctr + 1).ATE_Result;
796 796         R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, Loop_Ctr).Bad_Data :=
797 797             R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, Loop_Ctr + 1).Bad_Data;
798 798         Loop_Ctr := Loop_Ctr + 1;
799 799     end loop;
800 800 end if;
801 801
802 802 R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, Iter_Ctr).Ksa_Guess := Rta_Idx_Data.Ksa;
803 803 R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, Iter_Ctr).ATE_Result := Rta_Idx_Data.Ate;
804 804
805 805 -- determine if aircraft is on time (for speed adjustment logic)
806 806 if ( ((Rta_Type = Ac_Position_Types.AT_Time) and then
807 807     (abs(Rta_Idx_Data.Ate) < Perf_Rta_Lfdata.Spd_Adj_Tol))
808 808     or else
809 809     ((Rta_Type = Ac_Position_Types.After) and then
810 810         ( (abs(Rta_Idx_Data.Ate) < Perf_Rta_Lfdata.Spd_Adj_Tol) and then
811 811             (Ksa_Old <= 0.0))
812 812         or else
813 813             ((Rta_Idx_Data.Ate > Perf_Rta_Lfdata.Spd_Adj_Tol) and then
814 814                 (Ksa_Old = 0.0)) ))
815 815     or else

```

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_SEQ_RTA_LEG_SEP.ADA (continued)

```

816 816      ((Rta_Type = Ac_Position_Types.Before) and then
817 817      ( ((abs(Rta_Idx_Data.Ate) < Perf_Rta_Lfdata.Spd_Adj_Tol) and then
818 818      (Ksa_Old >= 0.0))
819 819      or else
820 820      ((Rta_Idx_Data.Ate < -Perf_Rta_Lfdata.Spd_Adj_Tol) and then
821 821      (Ksa_Old = 0.0)) ))
822 822  )
823 823  then -- aircraft is on time, do not adjust ksa
824 824      Rta_Idx_Data.Ate := 0;
825 825      Rta_Idx_Data.Preds_Status.Within_Spd_Adj_Tol := True;
826 826
827 827      -- WTS: Before calling Common_Init, have to determine where the latest
828 828      -- step point data is stored
829 829      Initialize_To_Iter_Ctr := True;
830 830      Copy_From_Opposite_Fpln := False;
831 831
832 832      -- WTS : Initialize data storage array, ksa ate records, RTA iter counter and
833 833      -- reset biasing flags, copy the step point locations into the first
834 834      -- array locations so that we have them roll counter back to 1
835 835      Common_Init;
836 836
837 837      -- WTS : Max the biasing factor out to freeze steps when on time.
838 838      if (R_WTS_Pkg.Want_To_Use_Flat_Bias) then
839 839          R_WTS_Pkg.New_Flat_Bias_Factor := R_WTS_Pkg.Cost_Bias_Factor;
840 840      end if;
841 841
842 842  else
843 843      Rta_Idx_Data.Preds_Status.Within_Spd_Adj_Tol := False;
844 844
845 845      -- If we started biasing the cost data on this pass thru perf, sort through
846 846      -- all of the ksa-ate data and "throw out" the data which has a step point
847 847      -- significantly different than where we're trying to hold it at.
848 848      if R_WTS_Pkg.Want_To_Use_Flat_Bias and
849 849          (R_WTS_Pkg.Flat_Bias_Factor >= R_WTS_Pkg.Initial_Flat_Bias_Factor) and
850 850          (Iter_Ctr >= 2) and
851 851          R_WTS_Pkg.Want_To_Sort_Data then
852 852
853 853          for Loop_Ctr in 1..(Iter_Ctr - 1) loop
854 854              -- Should check all of the steps, not just the first one
855 855              for Rec_Ctr in Alt_Profile_Iftypes.NumScPts loop
856 856                  if R_WTS_Pkg.Step_Dist_Rec
857 857                      (Fpln_Index,Rec_Ctr,Loop_Ctr).Valid

```

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_SEQ_RTA_LEG_SEP.ADA (continued)

```

858 858         and then
859 859         (abs(R_WTS_Pkg.Step_Dist_Rec
860 860         (Fpln_Index,Rec_Ctr, Loop_Ctr).Data -
861 861         R_WTS_Pkg.Step_Dist_Rec
862 862         (Fpln_Index,Rec_Ctr,Iter_Ctr).Data) >
863 863         R_WTS_Pkg.Max_Step_Movement)
864 864     then
865 865         R_WTS_Pkg.KSA_ETA_REC
866 866         (Fpln_Index, Loop_Ctr).Bad_Data := true;
867 867         exit;
868 868         -- when step is out of wack, get out of inner (all step points) loop
869 869     else -- if this data was previously thrown out and is now good, use it
870 870         R_WTS_Pkg.KSA_ETA_REC
871 871         (Fpln_Index, Loop_Ctr).Bad_Data := false;
872 872     end if;
873 873     end loop; -- loop through all step points
874 874 end loop;
875 875 end if;
876 876
877 877 end if;
878 878
879 879 end Compute_Ate;
880 880
881 881 ----- L O C A L   P R O C E D U R E   (Compute_Ksa) -----
882 882 -- compute speed adjustment factor (Ksa) for next pass of trip predictions
883 883 -- Note: Ksa is a constant TAS adjustment, in knots
884 884 -- Note: When reducing the (abs) speed adjustment, the appropriate Tavail
885 885 -- to use is somewhere between Tup and Tdn (e.g. for reducing a speed up,
886 886 -- small Ate's should use a Tavail near Tup, large Ate's should use a
887 887 -- Tavail near Tdn). If we assume a ratio of delta(Tavail/Ksa) = Slope
888 888 -- then we can compute a quadratic equation for Tavail. This equation
889 889 -- can be approximated with it's most significant term, giving:
890 890 -- Tavail = Tavail(ksa_old) + (x1 * Sqrt(Slope*Ate*Avg_Crz_Gndspd)).
891 891 -- where x1 is between 0.707 (Sqrt(2)/2) and 1.0; 0.707 works best for
892 892 -- small Ate's and 1.0 is better for large Ate's.
893 893 -- Of course this works best when the Ate is small, since the computation
894 894 -- of the Slope is only valid for a small range of Ksa.
895 895
896 896 procedure Compute_Ksa is
897 897     -- local variables
898 898     Avg_Crz_Gndspd : R_Portable.Float_32; -- average cruise groundspeed
899 899     Ate             : R_Portable.Float_32; -- arrival time error (float)

```

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_SEQ_RTA_LEG_SEP.ADA (continued)

```

900 900 Minimum      : R_Portable.Float_32;  -- min tavail
901 901 Slope       : R_Portable.Float_32;  -- slope of tavail/ksa curve
902 902 Invalid_Ate_Tolerance : R_Portable.Integer_32 := 1;
903 903 Good_Curve_Fit_Data : Boolean := False; -- if all of the ate_result data is
904 904 -- bad, then this variable will remain false and we will use the existing ksa_new
905 905 begin
906 906
907 907   if (Rta_Idx_Data.Crz_Time <= 0.0) then  -- RTA in clb or a/c in des
908 908     -- do not adjust speed but may adjust rcmd tko time
909 909     -- solution is finished if ksa_old is zero too
910 910     Rta_Idx_Data.Preds_Status.Rta_Happy := (Ksa_Old = 0.0);
911 911     Ksa_New := 0.0;
912 912
913 913   elsif ((Rta_Idx_Data.Ate = 0) or else
914 914     (Rta_Idx_Data.Crz_Time < Perf_Rta_Lfdata.Min_Crz_Time)) then
915 915     -- aircraft is on time or RTA fix is too near a/c or t/c, do not adjust ksa
916 916     -- (this also protects Avg_Crz_Gndspd computation below)
917 917     Rta_Idx_Data.Preds_Status.Rta_Happy := True;  -- solution is finished
918 918     Ksa_New := Ksa_Old;  -- do not change ksa
919 919
920 920   else  -- adjust ksa
921 921     -- solution is not finished, unless
922 922     -- 1) a rcmd tko time is being computed (Rta_Happy is set later)
923 923     -- 2) there is no time available to adjust speed (Rta_Happy is set later)
924 924     Rta_Idx_Data.Preds_Status.Rta_Happy := False;
925 925
926 926     -- If we are at max speed up or max slow down and still can't make RTA, we
927 927     -- shouldn't look for new ksa tries.  It may cause the code to blow up.
928 928
929 929     If Rta_Idx_Data.Preds_Status.At_Max_Speed_Up or
930 930       Rta_Idx_Data.Preds_Status.At_Max_Slow_Down then
931 931
932 932       if abs(R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, Iter_Ctr).Ate_Result) >=
933 933         Perf_Rta_Lfdata.Spd_Adj_Tol then
934 934         if R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, Iter_Ctr).Ate_Result > 0
935 935           then
936 936           -- at max speed up and still late, so can't make RTA.  Bump Ksa old
937 937           -- up by a little so that it will trip the at max speed up logic
938 938           -- again. This should set it to the same value on the next pass
939 939           Ksa_New := Ksa_Old + 1.0;
940 940         elsif
941 941           R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, Iter_Ctr).Ate_Result < 0 then

```

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_SEQ_RTA_LEG_SEP.ADA (continued)

```
942 942      -- at max slow down and still early, so can't make RTA.  Lower ksa
943 943      -- old some so that we trip the max slow down logic.  This should
944 944      -- reset ksa to the same value on the next pass
945 945      Ksa_New := Ksa_Old - 1.0;
946 946      end if;
947 947      end if; -- at max speed up or slow down
948 948      elsif (R_WTS_Pkg.Rta_Iter_Counter(Fpln_Index) > R_WTS_Pkg.Num_Ksa_ETA_Records and
949 949      (R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, 1).Ate_Result = 0 or
950 950      R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, R_WTS_Pkg.Num_Ksa_ETA_Records).Ate_Result = 0)) then
951 951      -- this is a case where you're doing a slowdown while on the ground,
952 952      -- so Iter_Ctr will be allowed to go past 1 even though the ate
953 953      -- result is 0.  Skip the ksa guess logic to prevent a divide by zero
954 954      Ksa_New := 0.0;
955 955      else
956 956      -- perform normal ksa guess logic
957 957      -- if there are less than two data points available, we can't do a
958 958      -- guess with this method.  If there are two points available, we
959 959      -- will do a linear fit.  If there are more than two, we will do the
960 960      -- linear fit using the two points closest to the RTA target only.
961 961      if (Iter_Ctr = 2) then
962 962          KSA_ATE_Slope := R_Portable.Float_32
963 963          (R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, 2).ATE_Result -
964 964          R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, 1).ATE_Result);
965 965          Good_Curve_Fit_Data := True;
966 966          if (KSA_ATE_Slope = 0.0) then
967 967              -- limit the slope and prevent a possible divide by zero
968 968              KSA_ATE_Slope := -1.0 * R_WTS_Pkg.Default_KSA_Ate_Slope;
969 969          else
970 970              KSA_ATE_Slope := (R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, 2).Ksa_Guess -
971 971              R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, 1).Ksa_Guess) /
972 972              KSA_ATE_Slope;
973 973          end if;
974 974
975 975      -- we cannot allow a positive slope, and we want to bound the slope
976 976      if (Ksa_ATE_Slope >= 0.0) then
977 977          KSA_ATE_Slope := -1.0 * R_WTS_Pkg.Default_KSA_Ate_Slope;
978 978      elsif (Ksa_ATE_Slope > (-1.0 * R_WTS_Pkg.KSA_Ate_Slope_Upper_Bound)) then
979 979          KSA_ATE_Slope := -1.0 * R_WTS_Pkg.KSA_Ate_Slope_Upper_Bound;
980 980      elsif (Ksa_ATE_Slope < (-1.0 * R_WTS_Pkg.KSA_Ate_Slope_Lower_Bound)) then
981 981          KSA_ATE_Slope := -1.0 * R_WTS_Pkg.KSA_Ate_Slope_Lower_Bound;
982 982      end if;
983 983
```


File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_SEQ_RTA_LEG_SEP.ADA (continued)

```

984      984      Ksa_New := R_WTS_Pkg.KSA_ETA_REC(Fpln_Index,Iter_Ctr).Ksa_Guess
985      985          - KSA_ATE_Slope * R_Portable.Float_32
986      986          (R_WTS_Pkg.KSA_ETA_REC(Fpln_Index,Iter_Ctr)
987      987          .ATE_Result);
988      988
989      989      elsif (Iter_Ctr /= 1) then
990      990
991      991          -- loop through the array, keeping the last data point, plus the
992      992          -- other data point closest to the RTA.  If there is a data point
993      993          -- on "the other side" of the RTA, we want to use this point
994      994          -- because it will give the best next guess, even if it isn't
995      995          -- the closest point to the rta target.  The points furthest
996      996          -- from the RTA will be placed in the lowest array locations so
997      997          -- that they will be dropped off as array locations are overwritten.
998      998      if R_WTS_Pkg.Want_To_Sort_Data then
999      999          Num_Places := Iter_Ctr - 1;
1000     1000          Min_ATE := 100000000;
1001     1001          Max_ATE := 0;
1002     1002          Min_ATE_Opp_Sign := 10000000;
1003     1003          Max_ATE_Opp_Sign := 0;
1004     1004          Opposite_Sign_Exist := False;
1005     1005          Loop_Ctr := 1;
1006     1006          Max_Location := 1;
1007     1007          Min_Location := Num_Places;
1008     1008
1009     1009          Find_Min : For Loop_Ctr in 1..Num_Places loop
1010     1010
1011     1011              if not(R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, Loop_Ctr).Bad_Data)
1012     1012              then
1013     1013                  Good_Curve_Fit_Data := True; -- Have at least one good data point
1014     1014                  -- only do the sorting logic if this is good data
1015     1015                  if R_Portable.Float_32(R_WTS_Pkg.KSA_ETA_REC
1016     1016                      (Fpln_Index, Loop_Ctr).ATE_Result) /
1017     1017                      R_Portable.Float_32(R_WTS_Pkg.KSA_ETA_REC
1018     1018                          (Fpln_Index,Iter_Ctr).ATE_Result)
1019     1019                      < 0.0 then
1020     1020
1021     1021                      Opposite_Sign := True;
1022     1022                      Opposite_Sign_Exist := True;
1023     1023                  else
1024     1024                      Opposite_Sign := False;
1025     1025                  end if;

```

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_SEQ_RTA_LEG_SEP.ADA (continued)

```

1026 1026
1027 1027         if (abs(R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, Loop_Ctr)
1028 1028             .ATE_Result) < abs(Min_Ate) and
1029 1029             (not Opposite_Sign_Exist)) or
1030 1030             (abs(R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, Loop_Ctr)
1031 1031             .ATE_Result) < abs(Min_Ate_Opp_Sign) and
1032 1032             (Opposite_Sign))
1033 1033     then
1034 1034         Min_ATE := R_WTS_Pkg.KSA_ETA_REC(Fpln_Index,
1035 1035             Loop_Ctr).ATE_Result;
1036 1036         Min_Location := Loop_Ctr;
1037 1037     end if;
1038 1038     if abs(R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, Loop_Ctr).ATE_Result) >
1039 1039         abs(Max_Ate) and (not Opposite_Sign)
1040 1040     then
1041 1041         Max_ATE := R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, Loop_Ctr).ATE_Result;
1042 1042         Max_Location := Loop_Ctr;
1043 1043     end if;
1044 1044     else -- not good data, so move this location to the lowest array location
1045 1045         -- so that it can be dropped from consideration eventually
1046 1046         Max_Location := Loop_Ctr;
1047 1047     end if; -- good_data check
1048 1048 end loop Find_Min;
1049 1049
1050 1050 -- only perfrom the Min_Location movement if there was at least one good data
1051 1051 -- point
1052 1052 if Good_Curve_Fit_Data then
1053 1053     if (Min_Location /= Num_Places) then
1054 1054         if (Iter_Ctr = 3 and
1055 1055             Max_Location /= 1) then
1056 1056             null;
1057 1057         else
1058 1058             Save_ATE := R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, Num_Places).ATE_Result;
1059 1059             R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, Num_Places).ATE_Result :=
1060 1060                 R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, Min_Location).ATE_Result;
1061 1061             R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, Min_Location).ATE_Result :=
1062 1062                 Save_ATE;
1063 1063             Save_Ksa := R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, Num_Places).Ksa_Guess;
1064 1064             R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, Num_Places).Ksa_Guess :=
1065 1065                 R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, Min_Location).Ksa_Guess;
1066 1066             R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, Min_Location).Ksa_Guess :=
1067 1067                 Save_Ksa;

```

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_SEQ_RTA_LEG_SEP.ADA (continued)

```

1068      1068          Save_BadData := R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, Num_Places).Bad_Data;
1069      1069          R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, Num_Places).Bad_Data :=
1070      1070              R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, Min_Location).Bad_Data;
1071      1071          R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, Min_Location).Bad_Data :=
1072      1072              Save_BadData;
1073      1073      end if;
1074      1074  end if;
1075      1075  end if; -- the Min data movement
1076      1076
1077      1077  -- The max location movement is don whether there is good data or not
1078      1078  if Max_Location /= 1 then
1079      1079      Save_ATE := R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, 1).ATE_Result;
1080      1080      R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, 1).ATE_Result :=
1081      1081          R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, Max_Location).ATE_Result;
1082      1082      R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, Max_Location).ATE_Result :=
1083      1083          Save_ATE;
1084      1084      Save_Ksa := R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, 1).Ksa_Guess;
1085      1085      R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, 1).Ksa_Guess :=
1086      1086          R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, Max_Location).Ksa_Guess;
1087      1087      R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, Max_Location).Ksa_Guess :=
1088      1088          Save_Ksa;
1089      1089      Save_BadData := R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, 1).Bad_Data;
1090      1090      R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, 1).Bad_Data :=
1091      1091          R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, Max_Location).Bad_Data;
1092      1092      R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, Max_Location).Bad_Data :=
1093      1093          Save_BadData;
1094      1094  end if;
1095      1095  end if; -- sorting array
1096      1096
1097      1097  -- 4/23/97 Want to be able to use this logic if not sorting data
1098      1098  if (R_WTS_Pkg.Want_To_Sort_Data and Good_Curve_Fit_Data) or
1099      1099      (not R_WTS_Pkg.Want_To_Sort_Data) then
1100      1100      -- use standard method to compute ksa_new
1101      1101      KSA_ATE_Slope := R_Portable.Float_32
1102      1102          (R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, Iter_Ctr).ATE_Result -
1103      1103          R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, Iter_Ctr - 1).ATE_Result);
1104      1104      if (KSA_ATE_Slope = 0.0) then
1105      1105          -- limit the slope and prevent a possible divide by zero
1106      1106          KSA_ATE_Slope := -1.0 * R_WTS_Pkg.Default_KSA_Ate_Slope;
1107      1107      else
1108      1108          KSA_ATE_Slope :=
1109      1109          (R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, Iter_Ctr).Ksa_Guess -

```

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_SEQ_RTA_LEG_SEP.ADA (continued)

```

1110 1110      R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, Iter_Ctr - 1).Ksa_Guess) /
1111 1111      KSA_ATE_Slope;
1112 1112  end if;
1113 1113  -- We cannot allow a positive slope to be used.  If positive, use a default
1114 1114  -- negative value (chosen so that it will probably undershoot solution).
1115 1115  if KSA_ATE_Slope > 0.0 then
1116 1116      KSA_ATE_Slope := -1.0 * R_WTS_Pkg.Default_KSA_Ate_Slope;
1117 1117  elsif (Ksa_ATE_Slope > (-1.0 * R_WTS_Pkg.KSA_Ate_Slope_Upper_Bound)) then
1118 1118      KSA_ATE_Slope := -1.0 * R_WTS_Pkg.KSA_Ate_Slope_Upper_Bound;
1119 1119  elsif (Ksa_ATE_Slope < (-1.0 * R_WTS_Pkg.KSA_Ate_Slope_Lower_Bound)) then
1120 1120      KSA_ATE_Slope := -1.0 * R_WTS_Pkg.KSA_Ate_Slope_Lower_Bound;
1121 1121  end if;
1122 1122  Ksa_New := R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, Iter_Ctr).Ksa_Guess -
1123 1123      KSA_ATE_Slope * R_Portable.Float_32
1124 1124      (R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, Iter_Ctr).ATE_Result);
1125 1125  else -- use the alternate ksa prediction method
1126 1126      -- compute average cruise ground speed
1127 1127      Avg_Crz_Gndspd := 3600.0 * Rta_Idx_Data.Crz_Dist / Rta_Idx_Data.Crz_Time;
1128 1128      if ((Avg_Crz_Gndspd < 100.0) or else (Avg_Crz_Gndspd > 1000.0)) then
1129 1129          Avg_Crz_Gndspd := 100.0;
1130 1130      end if;
1131 1131
1132 1132      -- make a floating point copy of Ate, since it's used a lot
1133 1133      Ate := R_Portable.Float_32(Rta_Idx_Data.Ate);
1134 1134
1135 1135      -- find time available to adjust speed
1136 1136      if (Ksa_Old >= 0.0) then -- previous predictions had a speed up
1137 1137          if (Rta_Idx_Data.Ate >= 0) then -- eta is late, speed up more
1138 1138              Rta_Idx_Data.Tavail := Rta_Idx_Data.Tup;
1139 1139          else -- (Ate < 0) -- eta is early, reduce speed up
1140 1140              -- compute slope of tavail/ksa (Tup_At_Ksa_Minus is Tup at Ksa-k3)
1141 1141              Slope := (Rta_Idx_Data.Tup_At_Ksa_Minus - Rta_Idx_Data.Tup) /
1142 1142                  Perf_Rta_Lfdata.k3;
1143 1143              if (Slope < Perf_Rta_Lfdata.k1) then
1144 1144                  Slope := Perf_Rta_Lfdata.k1;
1145 1145              end if;
1146 1146              Rta_Idx_Data.Tavail := Rta_Idx_Data.Tup +
1147 1147                  (Perf_Rta_Lfdata.k8 * Math_Pkg.Sqrt(Slope * Avg_Crz_Gndspd * (-Ate)));
1148 1148          -- compute minimum tavail to prevent overshoot when slope is small
1149 1149          -- and ate is large (since the slope was only computed at one point)
1150 1150          if (Ksa_Old > 1.0) then
1151 1151              Minimum := Rta_Idx_Data.Tup +

```

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_SEQ_RTA_LEG_SEP.ADA (continued)

```

1152 1152             (2.0 * (-Ate) * Avg_Crz_Gndspd / Ksa_Old);
1153 1153             if (Rta_Idx_Data.Tavail < Minimum) then
1154 1154                 Rta_Idx_Data.Tavail := Minimum;
1155 1155             end if;
1156 1156         end if;
1157 1157         if (Rta_Idx_Data.Tavail > Rta_Idx_Data.Tdn) then
1158 1158             Rta_Idx_Data.Tavail := Rta_Idx_Data.Tdn;
1159 1159         end if;
1160 1160     end if;
1161 1161 else -- ((Ksa_Old < 0.0) -- previous predictions had a slow down
1162 1162     if (Rta_Idx_Data.Ate <= 0) then -- eta is early, slow down more
1163 1163         Rta_Idx_Data.Tavail := Rta_Idx_Data.Tdn;
1164 1164     else -- (Ate > 0) -- eta is late, reduce slow down
1165 1165         -- compute slope of tavail/ksa (Tdn_At_Ksa_Plus is Tdn at Ksa+k3)
1166 1166         Slope := (Rta_Idx_Data.Tdn_At_Ksa_Plus - Rta_Idx_Data.Tdn) /
1167 1167             Perf_Rta_Lfdata.k3;
1168 1168         if (Slope < Perf_Rta_Lfdata.k2) then
1169 1169             Slope := Perf_Rta_Lfdata.k2;
1170 1170         end if;
1171 1171         Rta_Idx_Data.Tavail := Rta_Idx_Data.Tdn +
1172 1172             (Perf_Rta_Lfdata.k9 * Math_Pkg.Sqrt(Slope * Avg_Crz_Gndspd * Ate));
1173 1173         -- compute minimum tavail to prevent overshoot when slope is small
1174 1174         -- and ate is large (since the slope was only computed at one point)
1175 1175         if (Ksa_Old < -1.0) then
1176 1176             Minimum := Rta_Idx_Data.Tdn +
1177 1177                 (2.0 * Ate * Avg_Crz_Gndspd / (-Ksa_Old));
1178 1178             if (Rta_Idx_Data.Tavail < Minimum) then
1179 1179                 Rta_Idx_Data.Tavail := Minimum;
1180 1180             end if;
1181 1181         end if;
1182 1182         if (Rta_Idx_Data.Tavail > Rta_Idx_Data.Tup) then
1183 1183             Rta_Idx_Data.Tavail := Rta_Idx_Data.Tup;
1184 1184         end if;
1185 1185     end if;
1186 1186 end if;
1187 1187
1188 1188 -- compute new speed adjustment factor (ksa)
1189 1189 if (Rta_Idx_Data.Tavail > (2.0 * Ate)) then
1190 1190     if abs(Rta_Idx_Data.Tavail) > 10.0 then
1191 1191         Ksa_New := Ksa_Old + Avg_Crz_Gndspd * (Ate / (Rta_Idx_Data.Tavail));
1192 1192     else
1193 1193         Ksa_New := Ksa_Old + Avg_Crz_Gndspd * (Ate / (Rta_Idx_Data.Tavail - Ate));

```

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_SEQ_RTA_LEG_SEP.ADA (continued)

```

1194 1194         end if;
1195 1195         else
1196 1196             Ksa_New := Ksa_Old + Avg_Crz_Gndspd;
1197 1197         end if;
1198 1198         -- end of alternate ksa prediction method
1199 1199     end if; -- end of standard ksa computation
1200 1200 else -- if get to here, were not able to do a curve fit, so use alternate method
1201 1201     -- compute average cruise ground speed
1202 1202     Avg_Crz_Gndspd := 3600.0 * Rta_Idx_Data.Crz_Dist / Rta_Idx_Data.Crz_Time;
1203 1203     if ((Avg_Crz_Gndspd < 100.0) or else (Avg_Crz_Gndspd > 1000.0)) then
1204 1204         Avg_Crz_Gndspd := 100.0;
1205 1205     end if;
1206 1206
1207 1207     -- make a floating point copy of Ate, since it's used a lot
1208 1208     Ate := R_Portable.Float_32(Rta_Idx_Data.Ate);
1209 1209
1210 1210     -- find time available to adjust speed
1211 1211     if (Ksa_Old >= 0.0) then -- previous predictions had a speed up
1212 1212         if (Rta_Idx_Data.Ate >= 0) then -- eta is late, speed up more
1213 1213             Rta_Idx_Data.Tavail := Rta_Idx_Data.Tup;
1214 1214         else -- (Ate < 0) -- eta is early, reduce speed up
1215 1215             -- compute slope of tavail/ksa (Tup_At_Ksa_Minus is Tup at Ksa-k3)
1216 1216             Slope := (Rta_Idx_Data.Tup_At_Ksa_Minus - Rta_Idx_Data.Tup) /
1217 1217                 Perf_Rta_Lfdata.k3;
1218 1218             if (Slope < Perf_Rta_Lfdata.k1) then
1219 1219                 Slope := Perf_Rta_Lfdata.k1;
1220 1220             end if;
1221 1221             Rta_Idx_Data.Tavail := Rta_Idx_Data.Tup +
1222 1222                 (Perf_Rta_Lfdata.k8 * Math_Pkg.Sqrt(Slope * Avg_Crz_Gndspd * (-Ate)));
1223 1223             -- compute minimum tavail to prevent overshoot when slope is small
1224 1224             -- and ate is large (since the slope was only computed at one point)
1225 1225             if (Ksa_Old > 1.0) then
1226 1226                 Minimum := Rta_Idx_Data.Tup +
1227 1227                     (2.0 * (-Ate) * Avg_Crz_Gndspd / Ksa_Old);
1228 1228                 if (Rta_Idx_Data.Tavail < Minimum) then
1229 1229                     Rta_Idx_Data.Tavail := Minimum;
1230 1230                 end if;
1231 1231             end if;
1232 1232             if (Rta_Idx_Data.Tavail > Rta_Idx_Data.Tdn) then
1233 1233                 Rta_Idx_Data.Tavail := Rta_Idx_Data.Tdn;
1234 1234             end if;
1235 1235         end if;

```

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_SEQ_RTA_LEG_SEP.ADA (continued)

```

1236 1236      else -- ((Ksa_Old < 0.0) -- previous predictions had a slow down
1237 1237          if (Rta_Idx_Data.Ate <= 0) then -- eta is early, slow down more
1238 1238              Rta_Idx_Data.Tavail := Rta_Idx_Data.Tdn;
1239 1239      else -- (Ate > 0) -- eta is late, reduce slow down
1240 1240          -- compute slope of tavail/ksa (Tdn_At_Ksa_Plus is Tdn at Ksa+k3)
1241 1241          Slope := (Rta_Idx_Data.Tdn_At_Ksa_Plus - Rta_Idx_Data.Tdn) /
1242 1242              Perf_Rta_Lfdata.k3;
1243 1243          if (Slope < Perf_Rta_Lfdata.k2) then
1244 1244              Slope := Perf_Rta_Lfdata.k2;
1245 1245          end if;
1246 1246          Rta_Idx_Data.Tavail := Rta_Idx_Data.Tdn +
1247 1247              (Perf_Rta_Lfdata.k9 * Math_Pkg.Sqrt(Slope * Avg_Crz_Gndspd * Ate));
1248 1248          -- compute minimum tavail to prevent overshoot when slope is small
1249 1249          -- and ate is large (since the slope was only computed at one point)
1250 1250          if (Ksa_Old < -1.0) then
1251 1251              Minimum := Rta_Idx_Data.Tdn +
1252 1252                  (2.0 * Ate * Avg_Crz_Gndspd / (-Ksa_Old));
1253 1253              if (Rta_Idx_Data.Tavail < Minimum) then
1254 1254                  Rta_Idx_Data.Tavail := Minimum;
1255 1255              end if;
1256 1256          end if;
1257 1257          if (Rta_Idx_Data.Tavail > Rta_Idx_Data.Tup) then
1258 1258              Rta_Idx_Data.Tavail := Rta_Idx_Data.Tup;
1259 1259          end if;
1260 1260      end if;
1261 1261  end if;
1262 1262
1263 1263      -- compute new speed adjustment factor (ksa)
1264 1264      if (Rta_Idx_Data.Tavail > (2.0 * Ate)) then
1265 1265          if abs(Rta_Idx_Data.Tavail) > 10.0 then
1266 1266              Ksa_New := Ksa_Old + Avg_Crz_Gndspd * (Ate / (Rta_Idx_Data.Tavail));
1267 1267          else
1268 1268              Ksa_New := Ksa_Old + Avg_Crz_Gndspd * (Ate / (Rta_Idx_Data.Tavail - Ate));
1269 1269          end if;
1270 1270      else
1271 1271          Ksa_New := Ksa_Old + Avg_Crz_Gndspd;
1272 1272      end if;
1273 1273      -- end of alternate ksa prediction method
1274 1274  end if;
1275 1275  end if;
1276 1276
1277 1277      -- If the RTA is obviously unachievable, go ahead and issue the messages,

```

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_SEQ_RTA_LEG_SEP.ADA (continued)

```

1278 1278 -- but don't issue the messages in marginal cases where wind gusts may
1279 1279 -- be making the solution oscillate between achievable and unachievable.
1280 1280 -- This margin is currently controlled by k10.
1281 1281 if ( (Rta_Idx_Data.Tavail <= 0.0) and then
1282 1282     ((abs(Rta_Idx_Data.Ate) > Perf_Rta_Lfdata.k10) or else
1283 1283     (Rta_Idx_Data.Msg_Counter >= Perf_Rta_Lfdata.Max_Rta_Loops)) ) then
1284 1284     Rta_Idx_Data.Preds_Status.Rta_Happy := True; -- issue 'unable' message
1285 1285 end if;
1286 1286
1287 1287 end if; -- end of KSA adjustment
1288 1288
1289 1289 -- Do not speed up for At/After RTA's or slow down for At/Before RTA's
1290 1290 if ( ((Rta_Type = Ac_Position_Types.After) and then (Ksa_New > 0.0)) or else
1291 1291     ((Rta_Type = Ac_Position_Types.Before) and then (Ksa_New < 0.0)) ) then
1292 1292     Ksa_New := 0.0;
1293 1293 end if;
1294 1294
1295 1295 end Compute_Ksa;
1296 1296
1297 1297 ----- L O C A L   P R O C E D U R E   (Compute_Rcmd_Tko_Time) -----
1298 1298 -- Note: be very, very careful when modifying this code
1299 1299
1300 1300 procedure Compute_Rcmd_Tko_Time is
1301 1301 -- local variables
1302 1302 Time_Delta : R_Portable.Integer_32; -- time difference
1303 1303 begin
1304 1304 -- Never predict a slow down to meet a rta time if a delayed takeoff can be
1305 1305 -- predicted instead. Trip predictions must be performed at the unadjusted
1306 1306 -- econ speed (ksa = 0) in order to compute a recommended takeoff time, so
1307 1307 -- set ksa = 0 to be able to compute a recommended takeoff time next pass.
1308 1308 if (Ksa_New < 0.0) then
1309 1309     Ksa_New := 0.0;
1310 1310 end if;
1311 1311
1312 1312 -- When (current or next) predictions reflect a rta speed up (ksa > 0) or
1313 1313 -- when the rta time is an 'at or before', the recommended takeoff time
1314 1314 -- must be 'NOW'.
1315 1315 if ((Ksa_Old > 0.0) or else
1316 1316     (Ksa_New > 0.0) or else
1317 1317     (Rta_Type = Ac_Position_Types.Before))
1318 1318 then
1319 1319     -- recommended takeoff time is 'NOW'

```


File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_SEQ_RTA_LEG_SEP.ADA (continued)

```

1320 1320   Rta_Idx_Data.Rcmd_Takeoff.Data.Is_Now := True;
1321 1321   Rta_Idx_Data.Rcmd_Takeoff.Data.Time :=
1322 1322       Perf_Rta_Lfdata.Pred_Gmt_Unmodified.Data;
1323 1323   -- if pred's are stable, set recommended takeoff time valid
1324 1324   -- do not display "NOW" after a MOD until we are sure of it
1325 1325   if ( (Rta_Type = Ac_Position_Types.Before) or else
1326 1326       (Rta_Idx_Data.Preds_Status.Rta_Happy) or else
1327 1327       (Rta_Idx_Data.Msg_Counter >= Perf_Rta_Lfdata.Max_Rta_Loops) ) then
1328 1328       Rta_Idx_Data.Rcmd_Takeoff.Valid := True;
1329 1329   end if;
1330
1331 1331   -- To compute a recommended takeoff time other than 'NOW', predictions
1332 1332   -- using the unadjusted econ speed (Ksa_Old = 0) must arrive at the RTA
1333 1333   -- waypoint before the RTA time (Ksa_New <= 0, Ate <= 0.0).
1334 1334   elsif ((Ksa_Old = 0.0) and then (Ksa_New <= 0.0) and then
1335 1335       (Rta_Idx_Data.Ate <= 0)) then
1336 1336       -- compute recommended takeoff time
1337 1337       Rta_Idx_Data.Rcmd_Takeoff.Valid := True;
1338 1338       Rta_Idx_Data.Rcmd_Takeoff.Data.Time := Time_Map(Time_Target - Flight_Time);
1339 1339       -- If rta time is 'at or after' and the recommended takeoff time is
1340 1340       -- more than 12 hours in the future or less than 12 hours in the past,
1341 1341       -- then the recommended takeoff time is 'NOW'.
1342 1342       Time_Delta := Rta_Idx_Data.Rcmd_Takeoff.Data.Time -
1343 1343           Perf_Rta_Lfdata.Pred_Gmt_Unmodified.Data;
1344 1344       if ((Rta_Type = Ac_Position_Types.After) and then
1345 1345           ( (Time_Delta > Half_Day) or else
1346 1346               ((Time_Delta < 0) and then
1347 1347                   (Time_Delta > -Half_Day)) )) then
1348 1348           Rta_Idx_Data.Rcmd_Takeoff.Data.Is_Now := True;
1349 1349       else
1350 1350           Rta_Idx_Data.Rcmd_Takeoff.Data.Is_Now := False;
1351 1351           Rta_Idx_Data.Preds_Status.Rta_Happy := True;
1352 1352       end if;
1353 1353   end if;
1354 1354   -- note that if ((Ksa_Old < 0) and (Ksa_New = 0)) then the recommended
1355 1355   -- takeoff time is unchanged (except for 'Before' rta)
1356
1357 1357   -- note that the rcmd tko time is not output to manager until the end of
1358 1358   -- predictions so that it will not conflict with the predicted ETA's
1359
1360 1360 end Compute_Rcmd_Tko_Time;
1361

```

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_SEQ_RTA_LEG_SEP.ADA (continued)

```

1362 1362 ----- L O C A L   P R O C E D U R E   (Limit_Ksa) -----
1363 1363 procedure Limit_Ksa is
1364 1364   -- local variables
1365 1365   Ksa_Change   : R_Portable.Float_32; -- Ksa_Change = Ksa_New - Ksa_Old
1366 1366   Upper_Limit : R_Portable.Float_32; -- Upper_Limit = Max_Ksa + k6
1367 1367   Lower_Limit : R_Portable.Float_32; -- Lower_Limit = Min_Ksa - k6
1368 1368   Max_Ksa_Step_Local : R_Portable.Float_32;
1369 1369   Zero_CI_Mach   : R_Portable.Float_32; -- Mach number with CI=0
1370 1370   CI_Mach_Slope : R_Portable.Float_32; -- Slope of Cost Index vs Mach
1371 1371   Corr_Hundred_Ci : Portable_Types_Pkg.Float_32; -- corrected cost index
1372 1372   Hundred_Ci_Mach : Portable_Types_Pkg.Float_32; -- Mach number at step altitude with CI=100
1373 1373
1374 1374 begin
1375 1375   -- Fpln_index & Iter_Ctr variables set in Initialization: Do Not Modify
1376 1376
1377 1377   -- reduce ksa change to help prevent overshoot, but do not alter ksa when it
1378 1378   -- has been set to 0.0 (it is very unlikely to be computed as exactly 0.0)
1379 1379   if (Ksa_New /= 0.0) then -- limit ksa rate of change
1380 1380     Ksa_Change := Ksa_New - Ksa_Old;
1381 1381     if ((Ksa_Old * Ksa_Change) > 0.0) then -- check for same sign
1382 1382       -- approaching the ksa limit, the natural tendency of Tavail to become
1383 1383       -- smaller already provides good rate limiting so use a small factor
1384 1384       Ksa_Change := Perf_Rta_Lfdata.k4 * Ksa_Change;
1385 1385     else -- coming down from the ksa limit, the unreliability of Tavail
1386 1386       -- may require a larger rate limit to help prevent overshoot
1387 1387       Ksa_Change := Perf_Rta_Lfdata.k5 * Ksa_Change;
1388 1388     end if;
1389 1389     Ksa_New := Ksa_Old + Ksa_Change;
1390 1390   end if;
1391 1391
1392 1392   -- WTS : Flat Cost Biasing (new version as of 7/18/2012)
1393 1393   if R_WTS_Pkg.Want_To_Use_Flat_Bias then
1394 1394     if R_WTS_Pkg.Flat_Bias_Factor < R_WTS_Pkg.Initial_Flat_Bias_Factor then
1395 1395       -- increment the bias by a very tiny (negligible) amount each pass before
1396 1396       -- the Iter_To_Start_Biasing, then on the Iter_To_Start_Biasing set the
1397 1397       -- bias to Initial_Flat_Bias_Factor
1398 1398       -- (note: the (0.1 * R_WTS_Pkg.Tiny_Bias) is there to deal with any numerical issues)
1399 1399       if R_WTS_Pkg.Flat_Bias_Factor <
1400 1400         (R_WTS_Pkg.Tiny_Bias * R_Portable.Float_32(R_WTS_Pkg.Iter_To_Start_Biasing) -
1401 1401         (0.1 * R_WTS_Pkg.Tiny_Bias)) then
1402 1402         -- increment the value by a tiny amount, the number of multiples of the
1403 1403         -- tiny amount indicate the number of times the value has been incremented

```

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_SEQ_RTA_LEG_SEP.ADA (continued)

```

1404 1404      -- (i.e. how many passes have been done).
1405 1405      R_WTS_Pkg.New_Flat_Bias_Factor := R_WTS_Pkg.Flat_Bias_Factor +
1406 1406          R_WTS_Pkg.Tiny_Bias;
1407 1407      else
1408 1408          R_WTS_Pkg.New_Flat_Bias_Factor := R_WTS_Pkg.Initial_Flat_Bias_Factor;
1409 1409      end if;
1410 1410      elsif R_WTS_Pkg.Flat_Bias_Factor < R_WTS_Pkg.Cost_Bias_Factor then
1411 1411          -- increment the bias by one increment
1412 1412          R_WTS_Pkg.New_Flat_Bias_Factor := R_WTS_Pkg.Flat_Bias_Factor +
1413 1413              (R_WTS_Pkg.Cost_Bias_Factor - R_WTS_Pkg.Initial_Flat_Bias_Factor) /
1414 1414              R_Portable.Float_32(R_WTS_Pkg.Iter_To_Max_Biasing - R_WTS_Pkg.Iter_To_Start_Biasing);
1415 1415      else
1416 1416          R_WTS_Pkg.New_Flat_Bias_Factor := R_WTS_Pkg.Cost_Bias_Factor;
1417 1417      end if;
1418 1418  end if; -- end WTS
1419 1419
1420 1420  -- WTS : Been having a lot of trouble with overshoots, so the following
1421 1421  -- logic is designed to limit the ksa step
1422 1422  -- don't mess with a ksa that has been set to zero
1423 1423  if (Ksa_New /= 0.0) then -- limit ksa rate of change
1424 1424      Ksa_Change := Ksa_New - Ksa_Old;
1425 1425      -- first passes after a new rta time are especially bad overshooters
1426 1426      if Iter_Ctr = 1 and ((R_WTS_Pkg.Old_Rta_Tgt(Fpln_Index) /= Rta_Time) or
1427 1427          (not Rta_Idx_Data.Preds_Status.Within_Spd_Adj_Tol))
1428 1428      then
1429 1429          -- this is a first pass with new rta time, or just a first pass.
1430 1430          -- limit the ksa change
1431 1431          -- to a percentage of the computed step
1432 1432          Ksa_Change := R_WTS_Pkg.First_Pass_Ksa_Step_Percent * Ksa_Change;
1433 1433      elsif (abs(Rta_Idx_Data.Ate) >= R_WTS_Pkg.Large_Ate_Determiner) then
1434 1434          -- If we are dealing with a large ate, the code often overshoots, so
1435 1435          -- limit the ksa change. This is an elsif because we don't want to
1436 1436          -- so this limiting we we are doing the first pass one.
1437 1437          Ksa_Change := R_WTS_Pkg.Ksa_Step_Factor_For_Large_Ate *
1438 1438              Ksa_Change;
1439 1439      elsif (R_WTS_Pkg.Rta_Iter_Counter(Fpln_Index) =
1440 1440          (R_WTS_Pkg.Iter_To_Start_Biasing)) then
1441 1441          -- we always seem to overshoot on the first iteration after the
1442 1442          -- biasing is applied
1443 1443          Ksa_Change := R_WTS_Pkg.First_Pass_Ksa_Step_Percent * Ksa_Change;
1444 1444      end if;
1445 1445      -- the ksa-change will be further limited to a maximum for all cases

```

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_SEQ_RTA_LEG_SEP.ADA (continued)

```

1446 1446   if (abs(Rta_Idx_Data.Ate) <= R_WTS_Pkg.Low_Ksa_Step_Determiner) and
1447 1447       (abs(Ksa_Change) > R_WTS_Pkg.Max_Ksa_Step_Allowed_Low) then
1448 1448       if Ksa_Change > 0.0 then
1449 1449           Ksa_Change := R_WTS_Pkg.Max_Ksa_Step_Allowed_Low;
1450 1450       else
1451 1451           Ksa_Change := -1.0 * R_WTS_Pkg.Max_Ksa_Step_Allowed_Low;
1452 1452       end if;
1453 1453   elsif (abs(Rta_Idx_Data.Ate) > R_WTS_Pkg.Low_Ksa_Step_Determiner) and
1454 1454       (abs(Rta_Idx_Data.Ate) < R_WTS_Pkg.High_Ksa_Step_Determiner) then
1455 1455       Max_Ksa_Step_Local := (R_Portable.Float_32(abs(Rta_Idx_Data.Ate) -
1456 1456           R_WTS_Pkg.Low_Ksa_Step_Determiner))/
1457 1457           (R_Portable.Float_32(R_WTS_Pkg.High_Ksa_Step_Determiner -
1458 1458           R_WTS_Pkg.Low_Ksa_Step_Determiner));
1459 1459       Max_Ksa_Step_Local := (R_WTS_Pkg.Max_Ksa_Step_Allowed_High -
1460 1460           R_WTS_Pkg.Max_Ksa_Step_Allowed_Low) * Max_Ksa_Step_Local;
1461 1461       Max_Ksa_Step_Local := R_WTS_Pkg.Max_Ksa_Step_Allowed_Low +
1462 1462           Max_Ksa_Step_Local;
1463 1463       if (abs(Ksa_Change) > Max_Ksa_Step_Local) then
1464 1464           if Ksa_Change > 0.0 then
1465 1465               Ksa_Change := Max_Ksa_Step_Local;
1466 1466           else
1467 1467               Ksa_Change := -1.0 * Max_Ksa_Step_Local;
1468 1468           end if;
1469 1469       end if;
1470 1470   elsif (abs(Rta_Idx_Data.Ate) >= R_WTS_Pkg.High_Ksa_Step_Determiner) and
1471 1471       (abs(Ksa_Change) > R_WTS_Pkg.Max_Ksa_Step_Allowed_High) then
1472 1472       if Ksa_Change > 0.0 then
1473 1473           Ksa_Change := R_WTS_Pkg.Max_Ksa_Step_Allowed_High;
1474 1474       else
1475 1475           Ksa_Change := -1.0 * R_WTS_Pkg.Max_Ksa_Step_Allowed_High;
1476 1476       end if;
1477 1477   end if;
1478 1478   Ksa_New := Ksa_Old + Ksa_Change;
1479 1479 end if;
1480 1480 -- end WTS
1481 1481
1482 1482 -- compute upper and lower limits for ksa
1483 1483 Upper_Limit := Rta_Idx_Data.Max_Ksa + Perf_Rta_Lfdata.k6;
1484 1484 Lower_Limit := Rta_Idx_Data.Min_Ksa - Perf_Rta_Lfdata.k6;
1485 1485
1486 1486 -- upper limit ksa
1487 1487 if (Ksa_New >= Upper_Limit) then

```

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_SEQ_RTA_LEG_SEP.ADA (continued)

```

1488 1488 Ksa_New := Upper_Limit;
1489 1489 Rta_Idx_Data.Preds_Status.At_Max_Speed_Up := True;
1490 1490 Rta_Idx_Data.Preds_Status.At_Max_Slow_Down := False;
1491 1491
1492 1492 -- lower limit ksa
1493 1493 elsif (Ksa_New <= Lower_Limit) then
1494 1494 Ksa_New := Lower_Limit;
1495 1495 Rta_Idx_Data.Preds_Status.At_Max_Speed_Up := False;
1496 1496 Rta_Idx_Data.Preds_Status.At_Max_Slow_Down := True;
1497 1497
1498 1498 else
1499 1499 Rta_Idx_Data.Preds_Status.At_Max_Speed_Up := False;
1500 1500 Rta_Idx_Data.Preds_Status.At_Max_Slow_Down := False;
1501 1501 end if;
1502 1502
1503 1503 -- WTS : Split Ksa
1504 1504 -- To do split ksa correctly, we should compute one cost index and use this for the
1505 1505 -- entire next pass through Perf. We will use that data to compute the "RTA"
1506 1506 -- cost index for the next pass based on the Ksa_New for the next pass.
1507 1507 -- If we're converged and the counter is rolled back
1508 1508 -- to 1 then we don't have to compute a new Rta Cost Index.
1509 1509 if (not Rta_Idx_Data.Preds_Status.Within_Spd_Adj_Tol) then
1510 1510 Crzalt := Perf_Preds_Lfdata.VGB(VGB_Iftypes.CRZ2L).CMDSpdAlt;
1511 1511 Perf_Air_Data_Pkg.Air_Static_Data(Crzalt,
1512 1512 Perf_Profile_Lfdata.TOCPerfData.IsaDelta,
1513 1513 Press_Ratio, Theta, Theta_Std, Vel_Sound);
1514 1514 WOD := Perf_Profile_Lfdata.TOCPerfData.GrossWeight.Data / Press_Ratio;
1515 1515 Corr_Theta := Theta **
1516 1516 Portable_Types_Pkg.Float_32(FMCS_AEDB_CONSTANTS_IFDATA.XTheta);
1517 1517 Base_CI := Portable_Types_Pkg.Float_32(Ops_CDK_Common_Mgr_Pkg.Cost_Index.Data);
1518 1518 Corr_Base_CI := Base_CI / (Press_Ratio * Corr_Theta);
1519 1519 Perf_Aero_Speed_Pkg.EconCrzSpd (WOD, Corr_Base_CI, Base_Mach);
1520 1520 Tgt_Mach := Base_Mach + Ksa_New/Vel_Sound;
1521 1521 -- If always using T/C for RTA CI, we need this if block to speed convergence
1522 1522 -- by using the best guess available for CI
1523 1523 if (R_WTS_Pkg.Rta_Iter_Counter(Fpln_Index) = 1) then
1524 1524 CI_Guess := Ops_CDK_Common_Mgr_Pkg.Cost_Index.Data;
1525 1525 else
1526 1526 CI_Guess := R_WTS_Pkg.Rta_CI;
1527 1527 end if;
1528 1528 R_WTS_Pkg.Rta_CI := Perf_Crz_Pkg.Crz_CIFroMach(CI_Guess => CI_Guess,
1529 1529 Alt_Tgt => Crzalt,
```

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_SEQ_RTA_LEG_SEP.ADA (continued)

```

1530 1530      PredWind => Perf_Profile_Lfdata.TOCPerfData.PredWind,
1531 1531      WOD => WOD,
1532 1532      Press_Ratio => Press_Ratio,
1533 1533      Isa_Delta => Perf_Profile_Lfdata.TOCPerfData.IsaDelta,
1534 1534      Corr_Theta => Corr_Theta,
1535 1535      Vel_Sound => Vel_Sound,
1536 1536      FplnTrack => Perf_Profile_Lfdata.TOCPerfData.FplnTrack,
1537 1537      Mach_Tgt => Tgt_Mach);
1538 1538  end if;
1539 1539
1540 1540
1541 1541
1542 1542  -- Pegasus version:
1543 1543  -- If a zero cost index is computed, the odds are that it is in fact negative
1544 1544  -- The econ cruise tables in the PDB only contain positive cost index values.
1545 1545  if (Perf_Wts_Lfdata.Rta_Ci <= 0) and (not Rta_Idx_Data.Preds_Status.Within_Spd_Adj_Tol) then
1546 1546      -- Compute the negative cost index corresponding to the Tgt_Mach
1547 1547      -- This slope is assumed to hold for CI=0 to CI=-Infinity, even though
1548 1548      -- we know that this is not true.
1549 1549
1550 1550      -- compute the Mach Number at zero CI
1551 1551      Perf_Aero_Speed_Pkg.Econcrzspd (Wod, 0.0, Zero_Ci_Mach);
1552 1552
1553 1553      -- compute the Mach Number at 100 CI
1554 1554      Corr_Hundred_Ci := 100.0 / (Press_Ratio * Corr_Theta);
1555 1555      Perf_Aero_Speed_Pkg.Econcrzspd (Wod, Corr_Hundred_Ci, Hundred_Ci_Mach);
1556 1556
1557 1557      -- compute the cost index vs mach slope
1558 1558      Ci_Mach_Slope := (Hundred_Ci_Mach - Zero_Ci_Mach) / 100.0;
1559 1559
1560 1560      -- Protect against unreasonable values and ensure a positive slope
1561 1561      if (Ci_Mach_Slope > R_WTS_Pkg.Max_Ci_Mach_Slope_Limit) then
1562 1562          Ci_Mach_Slope := R_WTS_Pkg.Max_Ci_Mach_Slope_Limit;
1563 1563      elsif (Ci_Mach_Slope < R_WTS_Pkg.Min_Ci_Mach_Slope_Limit) then
1564 1564          Ci_Mach_Slope := R_WTS_Pkg.Min_Ci_Mach_Slope_Limit;
1565 1565      end if;
1566 1566
1567 1567      -- compute the cost index based on linear extrapolation from CI=100 to CI=0
1568 1568      Perf_Wts_Lfdata.Rta_Ci := Portable_Types_Pkg.Integer_32 ((Tgt_Mach - Zero_Ci_Mach) / Ci_Mach_Slope);
1569 1569  end if;
1570 1570
1571 1571  -- WTS : set the old_rta_Tgt = current

```

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_SEQ_RTA_LEG_SEP.ADA (continued)

```

1572 1572 R_WTS_Pkg.Old_RTA_Tgt(Fpln_Index) := Rta_Time;
1573 1573 -- Reset the Found Opt Step Flag to False
1574 1574 R_WTS_Pkg.Found_Opt_Step := False;
1575 1575 -- it will be set to true is a step is found on next pass through Perf
1576 1576
1577 1577 end Limit_Ksa;
1578 1578
1579 1579 ----- L O C A L   P R O C E D U R E   (Do_Message_Logic) -----
1580 1580 procedure Do_Message_Logic is
1581 1581   -- local variables
1582 1582   Ate           : R_Portable.Integer_32; -- arrival time error (ETA - RTA)
1583 1583   Msg_Display   : Boolean;               -- current predictions are for the displayed route
1584 1584   Pred_Alt      : R_Portable.Float_32;   -- predicted altitude
1585 1585   Step_Alt      : R_Portable.Float_32;   -- step altitude
1586 1586   Step_Ptr      : R_Portable.Integer_32; -- planned step climb leg index
1587 1587
1588 1588 begin
1589 1589   -- find out if these predictions are for the displayed route
1590 1590   if ((not Perf_Preds_Lfdata.Vtplogic.Haveactfpln) or else
1591 1591     (Ops_Lateral_Guidance_Buffer_Manager.Provfpln =
1592 1592     Fmcs_Fp_Guid_Btypes.Noprov)) then
1593 1593     -- these predictions are for the provisional route (which must be displayed)
1594 1594     -- or they are for the active route and there is no provisional route
1595 1595     Msg_Display := True;
1596 1596   else
1597 1597     Msg_Display := False;
1598 1598   end if;
1599 1599
1600 1600 ----- U N A B L E   R T A -----
1601 1601   -- set/clear 'UNABLE RTA' scratchpad message
1602 1602   -- note the local variable Eta is reused with a different definition
1603 1603   -- if there is no rcmd tko time, compute Eta from this predictions pass,
1604 1604   -- if there is a rcmd tko time, compute what the Eta would be if the rcmd tko
1605 1605   -- time had been used.
1606 1606   if (Rta_Idx_Data.Rcmd_Takeoff.Valid) then
1607 1607     if (Rta_Idx_Data.Rcmd_Takeoff.Data.Is_Now) then
1608 1608       Eta := Perf_Rta_Lfdata.Pred_Gmt_Unmodified.Data + Flight_Time;
1609 1609     else
1610 1610       Eta := Rta_Idx_Data.Rcmd_Takeoff.Data.Time + Flight_Time;
1611 1611     end if;
1612 1612   else -- not Rta_Idx_Data.Rcmd_Takeoff.Valid
1613 1613     Eta := Perf_Preds_Lfdata.Aircraft_State.GMT.Data + Flight_Time;

```

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_SEQ_RTA_LEG_SEP.ADA (continued)

```
1614 1614     end if;
1615 1615
1616 1616     -- compute arrival time error (ate) at the rta waypoint
1617 1617     Ate := Eta - Rta_Time;
1618 1618     if (Ate > Half_Day) then
1619 1619         Ate := Ate - One_Day;
1620 1620     elsif (Ate < -Half_Day) then
1621 1621         Ate := Ate + One_Day;
1622 1622     end if;
1623 1623
1624 1624     -- determine if aircraft is on time (for message logic)
1625 1625     Rta_Idx_Data.Preds_Status.Within_Msg_Tol :=
1626 1626         ( ((Rta_Type = Ac_Position_Types.After) and then (Ate >= 0))
1627 1627         or else
1628 1628         ((Rta_Type = Ac_Position_Types.Before) and then (Ate <= 0))
1629 1629         or else
1630 1630         ((Rta_Type = Ac_Position_Types.AT_Time) and then
1631 1631         (abs(Ate) <= Perf_Rta_Lfdata.Message_Tol)) );
1632 1632
1633 1633     -- reset the message counter if the rta is on time
1634 1634     if (Rta_Idx_Data.Preds_Status.Within_Msg_Tol) then
1635 1635         Rta_Idx_Data.Msg_Counter := 0;
1636 1636     end if;
1637 1637
1638 1638     -- reset the 'UNABLE RTA' message Issued flag
1639 1639     if (r_Msg_Flags.Unable_Rta_Msg_Cleared(Fpln_Index)) then
1640 1640         r_Msg_Flags.Unable_Rta_Msg_Issued(Fpln_Index) := False;
1641 1641     end if;
1642 1642
1643 1643     -- 'UNABLE RTA' message set logic
1644 1644     if ( ((Rta_Idx_Data.Preds_Status.Rta_Happy) or else
1645 1645         (Rta_Idx_Data.Msg_Counter >= Perf_Rta_Lfdata.Max_Rta_Loops)) and then
1646 1646         (not Rta_Idx_Data.Preds_Status.Within_Msg_Tol) and then
1647 1647         (not r_Msg_Flags.Unable_Rta_Msg_Issued(Fpln_Index)) )
1648 1648     then
1649 1649         -- set flag indicating that the RTA is unable for this route
1650 1650         r_Msg_Flags.Unable_Rta_Msg_Issued(Fpln_Index) := True;
1651 1651         r_Msg_Flags.Unable_Rta_Msg_Cleared(Fpln_Index) := False;
1652 1652         -- if predictions are for the currently displayed route
1653 1653         if (Msg_Display) then
1654 1654             -- display 'UNABLE RTA' message in scratchpad
1655 1655             Fmci_Spad_Manager_Pkg.Display_Message
```


File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_SEQ_RTA_LEG_SEP.ADA (continued)

```

1656 1656      (Message_Id => Scratch_Pad_Iftypes.Unable_RTA);
1657 1657      end if;
1658 1658  end if;
1659 1659
1660 1660  -- 'UNABLE RTA' message clear logic
1661 1661  if (Rta_Idx_Data.Preds_Status.Within_Msg_Tol) then
1662 1662      -- set flag indicating that the RTA is able for this route
1663 1663      r_Msg_Flags.Unable_Rta_Msg_Issued(Fpln_Index) := False;
1664 1664      r_Msg_Flags.Unable_Rta_Msg_Cleared(Fpln_Index) := False;
1665 1665      -- if predictions are for the currently displayed route
1666 1666      if (Msg_Display) then
1667 1667          -- clear 'UNABLE RTA' message from scratchpad
1668 1668          -- (it doesn't hurt to try to clear it when it's not there)
1669 1669          Fmci_Spad_Manager_Pkg.Clear_Message
1670 1670          (Message_Id => Scratch_Pad_Iftypes.Unable_RTA);
1671 1671      end if;
1672 1672  end if;
1673 1673
1674 1674  ----- U N A B L E   F L X X X   A T   R T A   F I X   -----
1675 1675  -- set/clear 'UNABLE FLxxx AT RTA FIX' scratchpad message
1676 1676  if (Rta_Idx_Data.Preds_Status.Rta_Happy) then
1677 1677      -- search for a planned (specified) step on a waypoint before the rta wypt
1678 1678      Step_Ptr := Perf_LGB_Pkg.LGB_Search
1679 1679          (Starting_Leg_Index => RtaPtr,
1680 1680           Search_Thing => Next_Step_Alt_Term,
1681 1681           Search_Direction => FMCS_Base_Types.Backward);
1682 1682
1683 1683  if (Step_Ptr > 0) then
1684 1684      Step_Alt := Perf_LGB_Lfdata.LGB(Step_Ptr).Fpln_Data.SpAlt1;
1685 1685      Pred_Alt := Perf_LGB_Lfdata.LGB(RtaPtr).Perf_Data.PrdAlt;
1686 1686
1687 1687      -- 'UNABLE FLxxx AT RTA FIX' message set logic
1688 1688      if ((Perf_Preds_Lfdata.DesiredPhase <= FMCS_Base_Types.Cruise) and then
1689 1689          (not r_Msg_Flags.Unable_Rta_Msg_Issued(Fpln_Index)) and then
1690 1690          (not r_Msg_Flags.Unable_Flxxx_At_Rta_Fix_Msg_Issued(Fpln_Index))
1691 1691          and then (Pred_Alt < (Step_Alt - 50.0)))
1692 1692      then
1693 1693          -- did not make it to planned step altitude before the rta waypoint
1694 1694          -- set flag indicating that the FLXXX is unable for this route
1695 1695          r_Msg_Flags.Unable_Flxxx_At_Rta_Fix_Msg_Issued(Fpln_Index) := True;
1696 1696          r_Msg_Flags.Unable_Flxxx_At_Rta_Fix_Msg_Cleared(Fpln_Index) := False;
1697 1697          r_Msg_Flags.Unable_Flxxx_At_Rta_Fix_Altitude(Fpln_Index) := Step_Alt;

```

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_SEQ_RTA_LEG_SEP.ADA (continued)

```

1698 1698      -- if predictions are for the currently displayed route
1699 1699      if (Msg_Display) then
1700 1700          -- display 'UNABLE FLXXX AT RTA FIX' message in scratchpad
1701 1701          Fmci_Spad_Manager_Pkg.Display_Message
1702 1702              (Message_Id => Scratch_Pad_Iftypes.Unable_FLXXX_At_RTA_Fix,
1703 1703              Number => Step_Alt);
1704 1704      end if;
1705 1705  end if;
1706 1706
1707 1707      -- 'UNABLE FLxxx AT RTA FIX' message clear logic
1708 1708      if ( ( (Perf_Preds_Lfdata.DesiredPhase <= FMCS_Base_Types.Cruise) and then
1709 1709          (Pred_Alt >= (Step_Alt - 50.0)) )
1710 1710          or else
1711 1711          ( (Perf_Preds_Lfdata.DesiredPhase > FMCS_Base_Types.Cruise) and then
1712 1712          (Perf_Crzalt_Lfdata.LastCrzAlt.Data >= (Step_Alt - 50.0)) ) )
1713 1713      then
1714 1714          -- set flag indicating that the FLXXX is able for this route
1715 1715          r_Msg_Flags.Unable_Flxxx_At_Rta_Fix_Msg_Issued(Fpln_Index) := False;
1716 1716          r_Msg_Flags.Unable_Flxxx_At_Rta_Fix_Msg_Cleared(Fpln_Index) := False;
1717 1717          -- if predictions are for the currently displayed route
1718 1718          if (Msg_Display) then
1719 1719              -- clear 'UNABLE FLXXX AT RTA FIX' message from scratchpad
1720 1720              -- (it doesn't hurt to try to clear it when it's not there)
1721 1721              Fmci_Spad_Manager_Pkg.Clear_Message
1722 1722                  (Message_Id => Scratch_Pad_Iftypes.Unable_FLXXX_At_RTA_Fix);
1723 1723          end if;
1724 1724      end if;
1725 1725
1726 1726      end if; -- if (Step_Ptr > 0)
1727 1727  end if; -- if (Rta_Idx_Data.Preds_Status.Rta_Happy)
1728 1728 end Do_Message_Logic;
1729 1729
1730 1730 ----- L O C A L   P R O C E D U R E   (Do_Filter_Logic) -----
1731 1731 procedure Do_Filter_Logic is
1732 1732 begin
1733 1733     -- turn on VG rta speed filter once solution is stable and a/c is in crz
1734 1734     -- turn off VG rta speed filter if speed adjustment changes significantly
1735 1735     if ((not Perf_Rta_Lfdata.Vg_Rta_Spd_Filter_Inhibit) and then
1736 1736         (Perf_Rp_Guidprms_Ildata.Fltphase = Fmcs_Base_Types.Cruise)) then
1737 1737         if (Rta_Idx_Data.Preds_Status.Within_Spd_Adj_Tol or else
1738 1738             Rta_Idx_Data.Msg_Counter >= Perf_Rta_Lfdata.Max_Rta_Loops) then
1739 1739             Perf_Rp_Guidprms_Ildata.RTA_Spd_Tgt_Filter_On := True;

```

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_SEQ_RTA_LEG_SEP.ADA (continued)

```

1740 1740     elsif (abs(Perf_Rta_Lfdata.Act_Data.Ksa - Ksa_New) > Perf_Rta_Lfdata.k7) then
1741 1741         Perf_Rp_Guidprms_Ildata.RTA_Spd_Tgt_Filter_On := False;
1742 1742     end if;
1743 1743     else
1744 1744         Perf_Rp_Guidprms_Ildata.RTA_Spd_Tgt_Filter_On := False;
1745 1745     end if;
1746 1746 end Do_Filter_Logic;
1747 1747
1748 1748 -----
1749 1749 ----- M A I N   P R O C E D U R E -----
1750 1750 -----
1751 1751 begin -- LGB_Seq_Rta_Leg
1752 1752
1753 1753     -- 7/19/12: Quick fix to not update the indexed Rta_Iter_Counter until the end of preds
1754 1754     -- since Common_Init has not yet been run to set the Fpln_Index, just save both indexes
1755 1755     R_WTS_Pkg.Old_Rta_Iter_Counter := R_WTS_Pkg.Rta_Iter_Counter;
1756 1756
1757 1757     if (Initial_Est) then
1758 1758         Initial_Estimate; -- setup data when called from Restart_Check
1759 1759     else
1760 1760         Predicted_Sequence; -- setup data when called from Lgb_Seq_Leg
1761 1761     end if;
1762 1762
1763 1763     if (Init_Valid) and (Good_Pointer) then
1764 1764         Compute_Ate; -- compute arrival time error and time avail to adj speed
1765 1765         Compute_Ksa; -- compute the new speed adjustment factor
1766 1766
1767 1767         if ((not Perf_Preds_Lfdata.Aircraft_State.Airborne) and then
1768 1768             (not Ops_Cdk_Perf_Pdb_Mgr_Pkg.Takeoff_Time_Is_Pilot_Entered)) then
1769 1769             Compute_Rcmd_Tko_Time; -- convert ksa to recommended takeoff time
1770 1770         else
1771 1771             Rta_Idx_Data.Rcmd_Takeoff.Valid := False;
1772 1772         end if;
1773 1773
1774 1774         Limit_Ksa; -- limit the ksa rate of change
1775 1775
1776 1776         if (not Initial_Est) then
1777 1777             Do_Message_Logic; -- set/clear scratchpad messages
1778 1778
1779 1779             if (Perf_Preds_Lfdata.VTPlogic.Haveactfpln) then
1780 1780                 Do_Filter_Logic; -- turn the VG speed filter on/off
1781 1781             end if;

```

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_SEQ_RTA_LEG_SEP.ADA (continued)

```

1782 1782     end if; -- if (not Initial_Est)
1783 1783
1784 1784     else -- not Init_Valid or SCR 9629 bad RTA Pointer
1785 1785         Ksa_New := 0.0;
1786 1786     end if; -- if (Init_Valid)
1787 1787
1788 1788     -- output data
1789 1789     Rta_Idx_Data.Old_Ksa := Ksa_Old;
1790 1790     Rta_Idx_Data.Ksa := Ksa_New;
1791 1791     Rta_Idx_Data.Preds_Status.Heartbeat := not(Rta_Idx_Data.Preds_Status.Heartbeat);
1792 1792     Perf_Rta_Lfdata.Idx_Data(Fpln_Index) := Rta_Idx_Data;
1793 1793     R_WTS_Pkg.Idx_Rta_CI(Fpln_Index) := R_WTS_Pkg.Rta_CI;
1794 1794     if (Fpln_Index = Act_Prov_Index_Manager.Act_Index) then
1795 1795         -- save active rta data in a separate location for easy stripcharting
1796 1796         Perf_Rta_Lfdata.Act_Index := Fpln_Index;
1797 1797         Perf_Rta_Lfdata.Act_Data := Rta_Idx_Data;
1798 1798     end if;
1799 1799
1800 1800     -- 7/19/12, quick fix to not update the indexed Rta_Iter_Counter until the end of preds
1801 1801     -- Save the new value of R_WTS_Pkg.Rta_Iter_Counter then set the value back
1802 1802     -- to what it was when this procedure was called. The saved value will be
1803 1803     -- used to update the value at the end of predictions.
1804 1804     R_WTS_Pkg.New_Rta_Iter_Counter(Fpln_Index) := R_WTS_Pkg.Rta_Iter_Counter(Fpln_Index);
1805 1805     R_WTS_Pkg.Rta_Iter_Counter(Fpln_Index) := R_WTS_Pkg.Old_Rta_Iter_Counter(Fpln_Index);
1806 1806
1807 1807 end LGB_Seq_Rta_Leg;
```

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_STORE_DATA_SEP.ADA

```

1 1 --|
2 2 --|DATA_RIGHTS: HONEYWELL CONFIDENTIAL & PROPRIETARY
3 3 --|
4 4 --|THIS WORK CONTAINS VALUABLE CONFIDENTIAL AND PROPRIETARY
5 5 --|INFORMATION. DISCLOSURE, USE OR REPRODUCTION OUTSIDE OF
6 6 --|HONEYWELL INTERNATIONAL, INC. IS PROHIBITED EXCEPT AS
7 7 --|AUTHORIZED IN WRITING. THIS UNPUBLISHED WORK IS PROTECTED BY
8 8 --|THE LAWS OF THE UNITED STATES AND OTHER COUNTRIES. IN THE
9 9 --|EVENT OF PUBLICATION, THE FOLLOWING NOTICE SHALL APPLY:
10 10 --|COPR. 2007 HONEYWELL INTERNATIONAL, INC. ALL RIGHTS RESERVED.
11 11 with Portable_Types_Pkg;
12 12 with Flight_Pln_Leg_Types;
12 13 with FMCS_Base_Types;
13 14 with FMCS_FP_Guid_BTypes;
```

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_STORE_DATA_SEP.ADA (continued)

```

14      15 with Perf_Preds_Lftypes;
15      16
16      17 with Perf_LGB_Lfdata;
17      18 with Perf_Preds_Lfdata;
18      19 with Perf-Origin_Dest_Lfdata;
19      20 with Perf_Integrators_Lfdata;
20      21 with Perf_VTP_Lfdata;
21      22 with Perf_Msg_Flags_Lfdata;
22      23
23      24 with Perf_Pred_Spd_Env_Pkg;
24      25 with Perf_SG_Spd_Gen_Pkg;
25      26 with Perf_Air_Data_Pkg;
26      27 with Perf_ADS_Intent_Pkg;
28      28 with Perf_Rta_Lfdata;
27      29
28      30 use Portable_Types_Pkg;
31      31 use Flight_Pln_Leg_Types;
29      32 use FMCS_Base_Types;
30      33 use FMCS_FP_Guid_BTypes;
31      34 use Perf_Preds_Lftypes;
32      35
33      36 separate (Perf_LGB_Pkg)
34      37
35      38 procedure LGB_Store_Data is
36      39     --!
37      40     -- ANCHOR:          FMCS_19_21023512
38      41     -- SOURCE:         FMFSDD; FMCS_19_21023000, FMCS_19_21023001, FMCS_19_21023005,
39      42     --                      FMCS_19_21023006
40      43     --                      FMFSRD; FMCS_19_20012454, FMCS_19_20006076, FMCS_19_20010030 |
41      44     --| @DESCRIPTION:
42      45     --| This procedure is responsible for storing predicted data from the
43      46     --| integration progress buffer into PERF's copy of the LGB
44      47     --| (PERF_LGB_LFDATA.LGB).
45      48     --|
46      49     --| When flight plan predictions reach (by distance) a waypoint, the required
47      50     --| aircraft state variables (integration progress buffer) at that point are
48      51     --| stored into the corresponding guidance leg in PERF_LGB_LFDATA.LGB.
49      52     --|
50      53     --
51      54     -- SPECIAL_CONSIDERATIONS:
52      55     --
53      56     -- SHARED_DATA_FOR:

```

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_STORE_DATA_SEP.ADA (continued)

```

54 57 -- IOBLK
55 58 --
56 59 -- REVISION_HISTORY:
57 60 -- DATE SCR # Programmer DRCM#
58 61 -- 01-11-94 1777 D. Groethe 10014
59 62 -- Initial version - complete.
60 63 --
61 64 -- 05-18-94 3269 D. Groethe 12763
62 65 -- Validate PrdAirSpd for holds and procedure turns.
63 66 --
64 67 -- 08-25-94 4693 D. Jiles 16198
65 68 -- Replaced transition_alt with Fl_MSL_Trans_Alt since baro correction
66 69 -- should be removed in relation to Fl/MSL alt. instead of speed tranistion
67 70 -- altitude.
68 71 --
69 72 -- 04/17/96 8030.08 Karen Hegeman M777B_FMF_01635
70 73 -- Added call to Perf_ADS_Intent_Pkg.Calc_Intermediate_Point to store
71 74 -- intermediate intent data when sequencing a hold leg. When the flight
72 75 -- phase is Climb or Cruise, only store the data if sequencing the hold
73 76 -- leg will cause a change in the target speed.
74 77 --
75 78 -- 06/25/97 9326.00 Mark Webb M777B_FMF_05097
76 79 -- Added the Max Alt message latch and message index to the invalidate
77 80 -- predicted gross weight if block. This is to prevent the gross weights
78 81 -- from being invalidated on legs that are before a Max Alt condition
79 82 -- occurs.
80 83 --
81 84 -- ===== 787 HISTORY =====
82 85 --
83 86 -- 12/07/05 519.00 Pat Caulfield
84 87 -- Added output of Prddataseq, Predisadev, Predfuelwgt, Pred_Wind,
85 88 -- and Preds_Stable.
86 89 --
87 90 -- 05/02/06 788.00 Pat Caulfield
88 91 -- Added output of predicted flight phase.
89 92 --
90 93 -- 03/29/07 2676.00 Pat Caulfield
91 94 -- Added output of predicted flight path angle (FPA).
92 95 --!
93 96
94 97
95 98 -- L O C A L V A R I A B L E S --

```

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_STORE_DATA_SEP.ADA (continued)

```

96      99
97     100      -- DESCRIPTION Tolerance to be used for comparing CAS speeds
98     101      CAS_Tolerance : constant Portable_Types_Pkg.Float_32 := 1.0;
99     102
100    103      -- DESCRIPTION Local copy of Perf_Preds_Lfdata.NavPtr used for
101    104      -- efficiency to reduce global memory access.
102    105      NavPtr          : Portable_Types_Pkg.Integer_32 := Perf_Preds_Lfdata.NavPtr;
103    106
104    107      -- DESCRIPTION Predicted altitude from the integration progress buffer;
105    108      -- used for efficiency to reduce global memory access.
106    109      Predicted_Alt : Portable_Types_Pkg.Float_32 :=
107    110          Perf_Integrators_Lfdata.IntProgBuf.Hprog;
108    111
109    112      -- DESCRIPTION Predicted gross weight from the integration progress buffer;
110    113      -- used for efficiency to reduce global memory access.
111    114      Predicted_GWT : Portable_Types_Pkg.Float_32 :=
112    115          Perf_Integrators_Lfdata.IntProgBuf.GWprog;
113    116
114    117      -- DESCRIPTION Predicted flight phase; used for efficiency
115    118      -- to reduce global memory access.
116    119      Predicted_Flight_Phase : FMCS_Base_Types.Flight_Phase_Type :=
117    120          Perf_Preds_Lfdata.DesiredPhase;
118    121
119    122      -- DESCRIPTION Transition Altitude
120    123      Trans_Alt : Portable_Types_Pkg.Float_32;
121    124
122    125      -- DESCRIPTION PathTerm for the leg - used for efficiency purposes
123    126      LegTerm : FMCS_Base_Types.PathType;
124    127
125    128      -- DESCRIPTION Indicates use of flaps extended for best hold speed.
126    129      Use_Flaps : Boolean;
127    130
128    131      -- DESCRIPTION Minimum Mach - used in hold speed computation
129    132      MinMach : Portable_Types_Pkg.Float_32;
130    133
131    134      -- DESCRIPTION Maximum Mach - used in hold speed computation
132    135      MaxMach : Portable_Types_Pkg.Float_32;
133    136
134    137      -- DESCRIPTION Not needed - output from Pred Spd Env call
135    138      Mbfnglo : Portable_Types_Pkg.Float_32;
136    139
137    140      -- DESCRIPTION: Best Hold Speed

```

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_STORE_DATA_SEP.ADA (continued)

138	141	Hold_Spd : Portable_Types_Pkg.Float_32;
139	142	
140	143	-- DESCRIPTION: T=Hold_Spd has been calculated and is being used in place of CAS2.
141	144	Using_Hold_Spd : boolean := false;
142	145	
143	146	Satnorm : Portable_Types_Pkg.Float_32;
144	147	-- DESCRIPTION: NORMALIZED STAT AIR TEMPERATURE
145	148	
146	149	Adat_Delta : Portable_Types_Pkg.Float_32;
147	150	-- DESCRIPTION: ATMOSPHERIC PRESSURE RATIO
148	151	
149	152	Satnormstd : Portable_Types_Pkg.Float_32;
150	153	-- DESCRIPTION: NORMALIZED STAT AIR TEMPERATURE
151	154	
152	155	Velsound : Portable_Types_Pkg.Float_32;
153	156	-- DESCRIPTION: VELOCITY OF SOUND
154	157	
155	158	HX_PI_Leg : boolean;
156	159	-- DESCRIPTION: LEG TYPE IS A HOLD OR PROCEDURE TURN
157	160	
	161	TempCAS : Portable_Types_Pkg.Float_32;
	162	-- DESCRIPTION: Temporary variable to hold the conversion of a Mach speed to a CAS
	163	-- for the calculation of the hold speed
158	164	
159	165	begin -- LGB_Store_Data
160	166	
161	167	-- DETERMINE WHICH TRANSITION ALTITUDE TO USE FOR WPT PRED ALT OUTPUT
162	168	if (Predicted_Flight_Phase <= Cruise) then
163	169	Trans_Alt := Perf-Origin_Dest_Lfdata.Origin.FL_MSL_Trans_Alt;
164	170	else
165	171	Trans_Alt := Perf-Origin_Dest_Lfdata.Destination.FL_MSL_Trans_Alt;
166	172	end if;
167	173	
168	174	-- SET UP THE NODODESPRED LEG FIELD FOR CDK TO NOT DISPLAY PREDICTIONS
169	175	-- FOR A DEFAULT DESCENT PATH.
170	176	
171	177	if (Predicted_Flight_Phase = Descent) and (Perf_LGB_Lfdata.Last_Constraint_Index = 0) then
172	178	Perf_LGB_Lfdata.LGB(NavPtr).Perf_Data.NoDoDesPred := True;
173	179	else
174	180	Perf_LGB_Lfdata.LGB(NavPtr).Perf_Data.NoDoDesPred := False;
175	181	end if;
176	182	

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_STORE_DATA_SEP.ADA (continued)

177	183	-- OVERWRITE THE FLIGHT MODE FIELD FOR THE LEG IF NOT AN ALT CONSTRAINT
178	184	
179	185	if not Perf_LGB_Lfdata.LGB(NavPtr).Fpln_Data.HavePerf then
180		if (Predicted_Flight_Phase < Cruise) then
181		Perf_LGB_Lfdata.LGB(NavPtr).Perf_Data.Fltmode := GBClimb;
182		elsif (Predicted_Flight_Phase = Cruise) then
183		Perf_LGB_Lfdata.LGB(NavPtr).Perf_Data.Fltmode := GBCruise;
184		else -- DESCENT
185		Perf_LGB_Lfdata.LGB(NavPtr).Perf_Data.Fltmode := GBDescent;
186	186	-- If there is an untagged FltMode on a speed constraint, this
187	187	-- means that FPLN was unable to tag it and PERF will have to
188	188	if Perf_LGB_Lfdata.LGB(NavPtr).Perf_Data.SpcSpdVal and then
189	189	Perf_LGB_Lfdata.LGB(NavPtr).Perf_Data.Fltmode = NoGBMode then
190	190	if (Predicted_Flight_Phase < Cruise) then
191	191	Perf_LGB_Lfdata.LGB(NavPtr).Perf_Data.Fltmode := GBClimb;
192	192	else -- DESCENT
193	193	Perf_LGB_Lfdata.LGB(NavPtr).Perf_Data.Fltmode := GBDescent;
194	194	end if;
195	195	-- Once a speed constraint has been tagged for FltMode, it cannot
196	196	-- change. Mini-legs that do not contain speed or altitude constraints
197	197	-- however can be tagged accordingly
198	198	elsif not Perf_LGB_Lfdata.LGB(NavPtr).Perf_Data.SpcSpdVal then
199	199	if (Predicted_Flight_Phase < Cruise) then
200	200	Perf_LGB_Lfdata.LGB(NavPtr).Perf_Data.Fltmode := GBClimb;
201	201	elsif (Predicted_Flight_Phase = Cruise) then
202	202	Perf_LGB_Lfdata.LGB(NavPtr).Perf_Data.Fltmode := GBCruise;
203	203	else -- DESCENT
204	204	Perf_LGB_Lfdata.LGB(NavPtr).Perf_Data.Fltmode := GBDescent;
205	205	end if;
186	206	end if;
187	207	end if;
188	208	
189	209	-- WRITE OUT PREDICTED ETA FOR THE LEG
190	210	
191	211	Perf_LGB_Lfdata.LGB(NavPtr).Perf_Data.PrdETAToFix :=
192	212	Portable_Types_Pkg.Integer_32(Perf_Integrators_Lfdata.IntProgBuf.TProg) +
193	213	Perf_Preds_Lfdata.Aircraft_State.GMT.Data;
194	214	Perf_LGB_Lfdata.LGB(NavPtr).Perf_Data.PrdETAFixVal :=
195	215	Perf_Preds_Lfdata.Aircraft_State.GMT.Valid;
196	216	
197	217	
198	218	-- WRITE OUT PREDICTED GROSS WEIGHT FOR THE LEG

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_STORE_DATA_SEP.ADA (continued)

```

199 219
200 220 -- INVALIDATE THE PREDICTED GROSS WEIGHT FOR THE LEG WHEN CRUISE ALTITUDE
201 221 -- EXCEEDS THE MAXIMUM ALTITUDE AND AIRCRAFT FLIGHT PHASE IS BEFORE DESCENT.
202 222
203 223 if ((Perf_Preds_Lfdata.Fltphase < Descent) and then
204 224   (Perf_Msg_Flags_Lfdata.AboveMaxAlt or else
205 225    ((Perf_Msg_Flags_Lfdata.Max_Alt_Msg_Leg_Indx > 0) and then
206 226     (Perf_Integrators_Lfdata.Intprogbuf.Xprog <
207 227      Perf_Lgb_Lfdata.Lgb(Perf_Msg_Flags_Lfdata.Max_Alt_Msg_Leg_Indx).Common_Data.Fixdistodest)))) then
208 228   Perf_LGB_Lfdata.LGB(NavPtr).Perf_Data.PrdGwtFixVal := False;
209 229 else
210 230   Perf_LGB_Lfdata.LGB(NavPtr).Perf_Data.PrdGwtFixVal := True;
211 231 end if;
212 232
213 233 Perf_LGB_Lfdata.LGB(NavPtr).Perf_Data.PrdGwtToFix := Predicted_GWT;
214 234
215 235 -- WRITE OUT PREDICTED ALTITUDE FOR THE LEG
216 236
217 237 Perf_LGB_Lfdata.LGB(NavPtr).Perf_Data.PrdAltVal := True;
218 238
219 239 if (Predicted_Alt < Trans_Alt) then
220 240   Perf_LGB_Lfdata.LGB(NavPtr).Perf_Data.PrdAlt :=
221 241   Predicted_Alt - Perf_Preds_Lfdata.Aircraft_State.Barocorr;
222 242 else
223 243   Perf_LGB_Lfdata.LGB(NavPtr).Perf_Data.PrdAlt := Predicted_Alt;
224 244 end if;
225 245
226 246 -- WRITE OUT THE PREDICTED AIR SPEED AND GROUND SPEED FOR THE LEG
227 247
228 248 LegTerm := Perf_LGB_Lfdata.LGB(NavPtr).Fpln_Data.PathTerm;
229 249 HX_PI_Leg := (LegTerm = HA) or (LegTerm = HF) or (LegTerm = HM) or (LegTerm = PI);
230 250 HX_PI_Leg := Perf_Lgb_Pkg.Is_Hold_Leg(NavPtr) or (LegTerm = PI);
231 251
232 252 if HX_PI_Leg then -- IF HOLD OR PROCEDURE TURN
233 253
234 254   -- COMPUTE BEST HOLD SPEED TO OUTPUT IN THE PREDICTED AIR SPEED FIELD
235 255
236 256   -- Use flaps is true if the flaps are out for real and we're either in descent
237 257   -- or predicting the active leg.
238 258   Use_Flaps := Perf_VTP_Lfdata.Act_VTP_Flaps and then ((Perf_Preds_Lfdata.Next_Waypoint_NavPtr =
239 259   Perf_Preds_Lfdata.Active_Waypoint_NavPtr) or else (Perf_Preds_Lfdata.Fltphase > Cruise));

```

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_STORE_DATA_SEP.ADA (continued)

```

260      -- COMPUTE THE STATIC AIR DATA FOR THE CURRENT PREDICTIONS STATE
261      Perf_Air_Data_Pkg.Air_Static_Data (Pressalt => Perf_Integrators_Lfdata.IntProgBuf.Hprog,
262                                         Isadelta => Perf_Preds_Lfdata.Isadelta,
263                                         Pressnorm => Adat_Delta,
264                                         Satnorm => Satnorm,
265                                         Satnormstd => Satnormstd,
266                                         Velsound => Velsound);
267
268      -- If there is a specified speed for the current leg:
269      if (Perf_Lgb_Lfdata.Lgb(Navptr).Perf_Data.Spcspdval) and then
270         (Perf_Lgb_Lfdata.Lgb(Navptr).Fpln_Data.Spcspdpe) and then
271         (Perf_Lgb_Lfdata.Lgb(Navptr).Fpln_Data.Hold_Src = Flight_Pln_Leg_Types.PILOT_ENTERED) then
272
273         -- The hold CAS is the specified speed
274         Hold_Spd := Perf_Lgb_Lfdata.Lgb(Navptr).Perf_Data.Spcspd;
275
276         -- Limit the hold speed to the profile speed for the active flight mode
277         -- If the profile speed is in CAS, the speed can be compared directly to the hold speed
278         if (Perf_Preds_Lfdata.VGB(Perf_Preds_Lfdata.VGBPtr).CmdCASVal) and then
279            (Hold_Spd > Perf_Preds_Lfdata.VGB(Perf_Preds_Lfdata.VGBPtr).CmdCAS) then
280
281             Hold_Spd := Perf_Preds_Lfdata.VGB(Perf_Preds_Lfdata.VGBPtr).CmdCAS;
282
283         elsif (Perf_Preds_Lfdata.VGB(Perf_Preds_Lfdata.VGBPtr).CmdMachVal) then
284
285             -- If the profile speed is in Mach, convert the Mach speed to a CAS before
286             -- making the comparison
287             TempCas := Perf_Air_Data_Pkg.Air_Mcas
288                        (Mach      => Perf_Preds_Lfdata.VGB(Perf_Preds_Lfdata.VGBPtr).CmdMach,
289                         Pressnorm => Adat_Delta);
290
291             -- Limit the hold speed to the profile speed
292             if (Hold_Spd > TempCas) then
293                 Hold_Spd := TempCas;
294             end if;
295         end if;
296     else
297
298         -- There is no specified speed on the hold, so calculate the best hold
299         -- speed
300
301     Perf_Pred_Spd_Env_Pkg.Pred_Spd_Env
302     (Fltphase      => Predicted_Flight_Phase,

```

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_STORE_DATA_SEP.ADA (continued)

242	302	Despathgen => False,
243	303	Flapsexist => Use_Flaps,
244	304	Grossweight => Predicted_GWT,
245	305	Pressalt => Predicted_Alt,
246	306	Minmach => MinMach,
247	307	Maxmach => MaxMach,
248	308	Mbfnglo => Mbfnglo);
249	309	
250	310	Perf_SG_Spd_Gen_Pkg.SG_Hold_Spd
251	311	(Flaps_Exist => Use_Flaps,
252	312	Press_Alt => Predicted_Alt,
253	313	Gw => Predicted_GWT,
254	314	Min_Mach => MinMach,
255	315	Max_Mach => MaxMach,
256	316	Hold_CAS => Hold_Spd);
257	317	
	318	-- limit the BHS to a NavDB defined speed on the hold
	319	if (Perf_Lgb_Lfdata.Lgb(Navptr).Perf_Data.Spcspdval) and then
	320	(Perf_Lgb_Lfdata.Lgb(Navptr).Fpln_Data.Spcspdpe) and then
	321	(Hold_Spd > Perf_Lgb_Lfdata.Lgb(Navptr).Perf_Data.Spcspd) then
	322	Hold_Spd := Perf_Lgb_Lfdata.Lgb(Navptr).Perf_Data.Spcspd;
	323	end if;
	324	end if;
258	325	if Predicted_Flight_Phase < Descent or else
259	326	Perf_Integrators_Lfdata.IntProgBuf.CAS2 > Hold_Spd then
260	327	
261	328	Perf_LGB_Lfdata.LGB(NavPtr).Perf_Data.PrdAirSpd.Value := Hold_Spd;
262	329	Perf_LGB_Lfdata.LGB(NavPtr).Perf_Data.PrdAirSpd.Speed_Type := CAS;
263	330	Using_Hold_Spd := true;
264	331	
265		----- COMPUTE THE STATIC AIR DATA FOR THE CURRENT PREDICTIONS STATE
266		----- Perf_Air_Data_Pkg.Air_Static_Data (Pressalt => Perf_Integrators_Lfdata.IntProgBuf.Hprog,
267		Isadelta => Perf_Preds_Lfdata.Isadelta,
268		Pressnorm => Adat_Delta,
269		Satnorm => Satnorm,
270		Satnormstd => Satnormstd,
271		Velound => Velound);
272	332	
273	333	-- COMPUTE THE PREDICTED TAS FOR THE LEG BASED ON THE BEST HOLD SPEED
274	334	-- USING THE MACH EQUIVALENT OF THE CAS SPEED TIMES THE SPEED OF SOUND
275	335	Perf_LGB_Lfdata.LGB(NavPtr).Perf_Data.Predicted_TAS := Velound *
276	336	Perf_Air_Data_Pkg.Air_Casm (Hold_Spd, Adat_Delta);

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_STORE_DATA_SEP.ADA (continued)

```

277 337     Perf_LGB_Lfdata.LGB(NavPtr).Perf_Data.PrdSpdVal := True;
278 338     end if;
279 339
280 340     -- INVALIDATE PREDICTED GROUND SPEED FOR HOLD OR PROCEDURE TURN
281 341     Perf_LGB_Lfdata.LGB(NavPtr).Perf_Data.PrdGndSpdVal := False;
282 342
283 343     if Using_Hold_Spd and then ((Predicted_FLight_Phase = Descent) or else
284 344         (not Perf_LGB_Lfdata.LGB(NavPtr).Perf_Data.SpcSpdVal and then
285 345             (abs(Perf_LGB_Lfdata.LGB(NavPtr).Perf_Data.PrdAirSpd.Value -
286 346                 Perf_Integrators_Lfdata.IntProgBuf.CAS2) > CAS_Tolerance))) then
287 347         Perf_ADS_Intent_Pkg.Calc_Intermediate_Point;
288 348     end if;
289 349 else
290 350     -- NOT A HOLD OR PROCEDURE TURN - WRITE OUT THE PREDICTED GROUND SPEED FOR THE LEG.
291 351     Perf_LGB_Lfdata.LGB(NavPtr).Perf_Data.PrdGndSpdVal := True;
292 352     Perf_LGB_Lfdata.LGB(NavPtr).Perf_Data.PrdGndSpd := Perf_Integrators_Lfdata.IntProgBuf.GndSpd2;
293 353 end if;
294 354
295 355 -- SELECT CAS OR MACH TO WRITE OUT AS PREDICTED AIRSPEED FOR THE LEG
296 356 if not Using_Hold_Spd then
297 357     if ((Perf_Preds_Lfdata.TgtSpdRec.TgtSpdTag = MachOnly) or
298 358         ((Perf_Preds_Lfdata.TgtSpdRec.TgtSpdTag = CasMach) and
299 359             (Predicted_Alt >= Perf_Preds_Lfdata.TgtSpdRec.CMXalt))) and not
300 360         HX_PI_Leg -- don't output mach for these legs.
301 361     then -- MACH
302 362         Perf_LGB_Lfdata.LGB(NavPtr).Perf_Data.PrdAirSpd.Speed_Type := Mach;
303 363         Perf_LGB_Lfdata.LGB(NavPtr).Perf_Data.PrdAirSpd.Value := Perf_Integrators_Lfdata.IntProgBuf.Mach2;
304 364     else -- CAS
305 365         Perf_LGB_Lfdata.LGB(NavPtr).Perf_Data.PrdAirSpd.Speed_Type := CAS;
306 366         Perf_LGB_Lfdata.LGB(NavPtr).Perf_Data.PrdAirSpd.Value := Perf_Integrators_Lfdata.IntProgBuf.CAS2;
307 367     end if;
308 368
309 369     -- WRITE OUT THE PREDICTED TAS FOR THE LEG.
310 370     Perf_LGB_Lfdata.LGB(NavPtr).Perf_Data.Predicted_TAS := Perf_Integrators_Lfdata.IntProgBuf.Tas2;
311 371     Perf_LGB_Lfdata.LGB(NavPtr).Perf_Data.PrdSpdVal := True;
312 372 end if; -- IF NOT USING HOLD SPEED
313 373
314 374 -- Copy the guidance header's prediction data sequence counter into the leg to validate
315 375 -- the data. Core FP tells that predictions are valid when the header and the leg/segment
316 376 -- values are the same, and can invalidate them by incrementing the header version.
317 377
318 378 Perf_LGB_Lfdata.LGB(NavPtr).Fpln_Data.Prddataseq := Perf_Lgb_Lfdata.Lgb_Header.Prddataseq;

```

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_STORE_DATA_SEP.ADA (continued)

319	379
320	380
321	381
322	382
323	383
324	384
325	385
326	386
327	387
328	388
329	389
330	390
331	391
332	392
333	393
334	394
335	395
336	396
337	397
338	398

```

-- These values are added for 787 to make full output of point layer data easier.

Perf_LGB_Lfdata.LGB(NavPtr).Perf_Data.Predisadev := Perf_Preds_Lfdata.Isadelta;

Perf_LGB_Lfdata.LGB(NavPtr).Perf_Data.Predfuelwgt := Perf_Integrators_Lfdata.Intprogbuf.Gwprog -
    Perf_Preds_Lfdata.Aircraft_State.Zfw.Data;

Perf_LGB_Lfdata.LGB(NavPtr).Perf_Data.Pred_Wind := (Direction => Perf_Wind_Lfdata.Predwind.Dir,
    Speed      => Perf_Wind_Lfdata.Predwind.Mag);

Perf_LGB_Lfdata.LGB(NavPtr).Perf_Data.Preds_Stable := not Perf_Preds_Lfdata.Vtplogic.Firstpass;

Perf_LGB_Lfdata.LGB(NavPtr).Perf_Data.Prdfltphase := Predicted_Flight_Phase;

Perf_LGB_Lfdata.LGB(NavPtr).Perf_Data.Pred_FPA := Portable_Types_Pkg.Float_32 (
    Perf_Integrators_Lfdata.Intprogbuf.Gamaair2);

end LGB_Store_Data;

```

File: CTP_B787_PERF_CRZINITE.ZIP\stubs.gpr

1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	10
11	11
12	12
13	13
14	14
15	15
16	16
17	17
18	18
19	19
20	20

```

with "../.../gps/fm/fm_naming.gpr";
with "../.../gps/io/io.gpr";

project stubs is

    for Source_Dirs use ("..\SRC\ADA_SRC",
        "..\SRC\ADA_SRC\StubSRC",
        "..\..\..\SRC\fm",
        "..\..\..\SRC\fm\stubs",
        "..\..\..\SRC\com",
        "..\..\..\SRC\ci_c\auto\ADA",
        "..\..\..\SRC\com\Stubs");

package Naming is
    --com naming
    for Spec ("ops_timer_pkg") use "OPS_TIMER_PKG.ADA";
    for Spec ("fmci_widget_event_constant_tpkg") use "FMCI_WIDGET_EVENT_CONSTANT_TPKG.ADA";
    for Spec ("wind_iftypes") use "WIND_IFTYPES.ADA";
    for Spec ("windrec_types") use "WINDREC_TYPES.ADA";
    for Spec ("viatype_set_pkg") use "VIATYPE_SET_PKG.ADA";

```

File: CTP_B787_PERF_CRZINITE.ZIP\stubs.gpr (continued)

```

21      21      for Spec ("vgb_ifotypes") use "VGB_IFTYPES_.ADA";
22      22      for Spec ("updatewind_ifotypes") use "UPDATEWIND_IFTYPES_.ADA";
23      23      for Spec ("transition_tpkg") use "TRANSITION_TPKG_.ADA";
24      24      for Spec ("step_climb_ifdata") use "STEP_CLIMB_IFDATA_.ADA";
25      25      for Spec ("simsoft_interface_dpkg") use "SIMSOFT_INTERFACE_DPKG_.ADA";
26      26      for Spec ("set_package") use "SET_PACKAGE_.ADA";
27      27      for Body ("set_package") use "SET_PACKAGE.ADA";
28      28      for Spec ("scratch_pad_ifotypes") use "SCRATCH_PAD_IFTYPES_.ADA";
29      29      for Spec ("radio_tuning_ifotypes") use "RADIO_TUNING_IFTYPES_.ADA";
30      30      for Spec ("perf_tko_ref_spd_ifdata") use "PERF_TKO_REF_SPD_IFDATA_.ADA";
31      31      for Spec ("perf_st_spdtape_ifdata") use "PERF_ST_SPDTAPE_IFDATA_.ADA";
32      32      for Spec ("perf_sg_spd_gen_ifdata") use "PERF_SG_SPD_GEN_IFDATA_.ADA";
33      33      for Spec ("perf_rt_ifotypes") use "PERF_RT_IFTYPES_.ADA";
34      34      for Spec ("perf_rp_guidprms_lfdata") use "PERF_RP_GUIDPRMS_LFDATA_.ADA";
35      35      for Spec ("perf_rp_guidprms_ifdata") use "PERF_RP_GUIDPRMS_IFDATA_.ADA";
36      36      for Spec ("perf_preds_btypes") use "PERF_PREDS_BTYPES_.ADA";
37      37      for Spec ("perf_offpath_descent_ifotypes") use "PERF_OFFPATH_DESCENT_IFTYPES_.ADA";
38      38      for Spec ("perf_offpath_descent_ifdata") use "PERF_OFFPATH_DESCENT_IFDATA_.ADA";
39      39      for Spec ("perf_max_opt_ifotypes") use "PERF_MAX_OPT_IFTYPES_.ADA";
40      40      for Spec ("perf_max_opt_ifdata") use "PERF_MAX_OPT_IFDATA_.ADA";
41      41      for Spec ("perf_hold_time_ifotypes") use "PERF_HOLD_TIME_IFTYPES_.ADA";
42      42      for Spec ("perf_hold_time_ifdata") use "PERF_HOLD_TIME_IFDATA_.ADA";
43      43      for Spec ("perf_change_flags_ifotypes") use "PERF_CHANGE_FLAGS_IFTYPES_.ADA";
44      44      for Spec ("perf_act_spd_env_lfdata") use "PERF_ACT_SPD_ENV_LFDATA_.ADA";
45      45      for Spec ("perf_act_spd_env_ifdata") use "PERF_ACT_SPD_ENV_IFDATA_.ADA";
46      46      for Spec ("pdb_types_pkg") use "PDB_TYPES_PKG_.ADA";
47      47      for Spec ("pdb_table_interp_3_pkg") use "PDB_TABLE_INTERP_3_PKG_.ADA";
48      48      for Body ("pdb_table_interp_3_pkg") use "PDB_TABLE_INTERP_3_PKG.ADA";
49      49      for Spec ("pdb_table_interp_2_pkg") use "PDB_TABLE_INTERP_2_PKG_.ADA";
50      50      for Body ("pdb_table_interp_2_pkg") use "PDB_TABLE_INTERP_2_PKG.ADA";
51      51      for Spec ("pdb_table_interp_1_pkg") use "PDB_TABLE_INTERP_1_PKG_.ADA";
52      52      for Body ("pdb_table_interp_1_pkg") use "PDB_TABLE_INTERP_1_PKG.ADA";
53      53      for Spec ("pdb_selection_pkg") use "PDB_SELECTION_PKG_.ADA";
54      54      for Body ("pdb_selection_pkg") use "PDB_SELECTION_PKG.ADA";
55      55      for Spec ("pdb_parameter_block_data_pkg") use "PDB_PARAMETER_BLOCK_DATA_PKG_.ADA";
56      56      for Spec ("pdb_initialization_pkg") use "PDB_INITIALIZATION_PKG_.ADA";
57      57      for Body ("pdb_initialization_pkg") use "PDB_INITIALIZATION_PKG.ADA";
58      58      for Spec ("pdb_data_pkg") use "PDB_DATA_PKG_.ADA";
59      59      for Spec ("pdb_data_access_pkg") use "PDB_DATA_ACCESS_PKG_.ADA";
60      60      for Body ("pdb_data_access_pkg") use "PDB_DATA_ACCESS_PKG.ADA";
61      61      for Spec ("pdb_application_types_pkg") use "PDB_APPLICATION_TYPES_PKG_.ADA";
62      62      for Spec ("pdb_address_translation_pkg") use "PDB_ADDRESS_TRANSLATION_PKG_.ADA";

```

File: CTP_B787_PERF_CRZINITE.ZIP\stubs.gpr (continued)

```

63      63      for Body ("pdb_address_translation_pkg") use "PDB_ADDRESS_TRANSLATION_PKG.ADA";
64      64      for Spec ("path_term_types") use "PATH_TERM_TYPES_.ADA";
65      65      for Body ("ops_timer_pkg") use "OPS_TIMER_PKG.ADA";
66      66      for Spec ("fmcs_partnumber_pkg") use "OPS_PARTNUMBER_PKG_.ADA";
67      67      for Spec ("ops_opc_ami_ifdata") use "OPS_OPC_AMI_IFDATA_.ADA";
68      68      for Spec ("ops_lateral_guidbuff_mgr_iftypes") use "OPS_LATERAL_GUIDBUFF_MGR_IFTYPES_.ADA";
69      69      for Spec ("ops_hs_buffer_gnrc") use "OPS_HS_BUFFER_GNRC_.ADA";
70      70      for Body ("ops_hs_buffer_gnrc") use "OPS_HS_BUFFER_GNRC.ADA";
71      71      for Spec ("ops_db_const_pkg") use "OPS_DB_CONST_PKG_.ADA";
72      72      for Body ("ops_db_const_pkg") use "OPS_DB_CONST_PKG.ADA";
73      73      for Spec ("ops_date_compare") use "OPS_DATE_COMP_PKG_.ADA";
74      74      for Body ("ops_date_compare") use "OPS_DATE_COMP_PKG.ADA";
75      75      for Spec ("ops_data_retained_pkg") use "OPS_DATA_RETAINED_PKG_.ADA";
76      76      for Body ("ops_data_retained_pkg") use "OPS_DATA_RETAINED_PKG.ADA";
77      77      for Spec ("ops_cmn_utilities_pkg") use "OPS_CMN_UTILITIES_PKG_.ADA";
78      78      for Body ("ops_cmn_utilities_pkg") use "OPS_CMN_UTILITIES_PKG.ADA";
79      79      for Spec ("ops_cdl_buffer_mgr_iftypes") use "OPS_CDL_BUFFER_MGR_IFTYPES_.ADA";
80      80      for Spec ("ops_cdk_altn_object_iftypes") use "OPS_CDK_ALTN_OBJECT_IFTYPES_.ADA";
81      81      for Spec ("ops_cdk_altn_init_constants_pkg") use "OPS_CDK_ALTN_INIT_CONSTANTS_PKG_.ADA";
82      82      for Spec ("ops_aedb_ifdata") use "OPS_AEDB_IFDATA_.ADA";
83      83      for Body ("nav_wind_pkg.relative_wind") use "NAV_WIND_PKG_RELATIVE_WIND.ADA";
84      84      for Body ("nav_wind_pkg.measure_wind") use "NAV_WIND_PKG_MEASURE_WIND.ADA";
85      85      for Body ("nav_wind_pkg.filter_non_gusty_wind") use "NAV_WIND_PKG_FILTER_NON_GUSTY_WIND.ADA";
86      86      for Body ("nav_wind_pkg.filter_gusty_wind") use "NAV_WIND_PKG_FILTER_GUSTY_WIND.ADA";
87      87      for Spec ("nav_wind_pkg") use "NAV_WIND_PKG_.ADA";
88      88      for Body ("nav_wind_pkg") use "NAV_WIND_PKG.ADA";
89      89      for Body ("wgs84_geoid.elevation") use "NAV_WGS84_GEOID_ELEVATION.ADA";
90      90      for Spec ("wgs84_geoid") use "NAV_WGS84_GEOID_.ADA";
91      91      for Body ("wgs84_geoid") use "NAV_WGS84_GEOID.ADA";
92      92      for Spec ("navigation_types") use "NAV_TYPES_.ADA";
93      93      for Body ("navigation_types") use "NAV_TYPES.ADA";
94      94      for Spec ("third_order_vector") use "NAV_THIRD_ORDER_VECTOR_.ADA";
95      95      for Body ("third_order_vector") use "NAV_THIRD_ORDER_VECTOR.ADA";
96      96      for Body ("nav_sensor_computations.velocity_vector") use "NAV_SENSOR_COMPUTATIONS_VELOCITY_VECTO.ADA";
97      97      for Body ("nav_sensor_computations.position_vector") use "NAV_SENSOR_COMPUTATIONS_POSITION_VECTO.ADA";
98      98      for Body ("nav_sensor_computations.ecef_xyz_to_geodetic_velocity") use "NAV_SENSOR_COMPUTATIONS_GEODETTIC_VELOC.ADA";
99      99      for Body ("nav_sensor_computations.ecef_xyz_to_geodetic_position") use "NAV_SENSOR_COMPUTATIONS_GEODETTIC_POSIT.ADA";
100     100     for Spec ("nav_sensor_computations") use "NAV_SENSOR_COMPUTATIONS_.ADA";
101     101     for Body ("nav_sensor_computations") use "NAV_SENSOR_COMPUTATIONS.ADA";
102     102     for Spec ("sem_types") use "NAV_SEM_TYPES_.ADA";
103     103     for Spec ("nav_hot_spare_iftypes") use "NAV_HOT_SPARE_IFTYPES_.ADA";
104     104     for Spec ("nav_filter_blocks_pkg") use "NAV_FILTER_BLOCKS_.ADA";

```


File: CTP_B787_PERF_CRZINITE.ZIP\stubs.gpr (continued)

```

105 105      for Body ("nav_filter_blocks_pkg") use "NAV_FILTER_BLOCKS.ADA";
106 106      for Spec ("navigation_constants") use "NAV_CONSTANTS_.ADA";
107 107      for Body ("navigation_utilities.rotate_bias") use "NAVIGATION_UTILITIES_ROTATE_BIAS.ADA";
108 108      for Spec ("navigation_utilities") use "NAVIGATION_UTILITIES_.ADA";
109 109      for Body ("navigation_utilities") use "NAVIGATION_UTILITIES.ADA";
110 110      for Spec ("nam_waypoint_ifdata") use "NAM_WAYPOINT_IFDATA_.ADA";
111 111      for Spec ("nam_userrec_iftypes") use "NAM_USERREC_IFTYPES_.ADA";
112 112      for Spec ("nam_runway_ifdata") use "NAM_RUNWAY_IFDATA_.ADA";
113 113      for Spec ("nam_navaid_iftypes") use "NAM_NAVAID_IFTYPES_.ADA";
114 114      for Spec ("nam_navaid_ifdata") use "NAM_NAVAID_IFDATA_.ADA";
115 115      for Spec ("nam_image_types") use "NAM_IMAGE_TYPES_.ADA";
116 116      for Spec ("nam_iftypes") use "NAM_IFTYPES_.ADA";
117 117      for Spec ("nam_holding_pattern_tpkg") use "NAM_HOLDING_PATTERN_TPKG_.ADA";
118 118      for Spec ("nam_efis_iftypes") use "NAM_EFIS_IFTYPES_.ADA";
119 119      for Spec ("nam_efis_ifdata") use "NAM_EFIS_IFDATA_.ADA";
120 120      for Body ("nam_efis_ifdata") use "NAM_EFIS_IFDATA.ADA";
121 121      for Spec ("nam_corte_ifdata") use "NAM_CORTE_IFDATA_.ADA";
122 122      for Spec ("nam_base_types") use "NAM_BASE_TYPES_.ADA";
123 123      for Spec ("measurement_valid_types") use "MEASUREMENT_VALID_TYPES_.ADA";
124 124      for Spec ("math_rad_pkg") use "MATH_RAD_PKG.ADA";
125 125      for Spec ("math_rad_64_pkg") use "MATH_RAD_64_PKG.ADA";
126 126      for Spec ("math_primitives_pkg") use "MATH_PRIMITIVES_PKG.ADA";
127 127      for Spec ("math_primitives_64_pkg") use "MATH_PRIMITIVES_64_PKG.ADA";
128 128      for Spec ("math_pkg") use "MATH_PKG_.ADA";
129 129      for Body ("math_pkg") use "MATH_PKG.ADA";
130 130      for Spec ("lg_slow_out_ifdata_pkg") use "LG_SLOW_OUT_IFDATA_PKG_.ADA";
131 131      for Spec ("lg_segment_manager_pkg") use "LG_SEGMENT_MANAGER_PKG_.ADA";
132 132      for Body ("lg_segment_manager_pkg") use "LG_SEGMENT_MANAGER_PKG.ADA";
133 133      for Spec ("lg_iftypes") use "LG_IFTYPES_.ADA";
134 134      for Spec ("lg_group_id_pkg") use "LG_GROUP_ID_PKG_.ADA";
135 135      for Spec ("lg_fast_out_ifdata_pkg") use "LG_FAST_OUT_IFDATA_PKG_.ADA";
136 136      for Spec ("legseg_types") use "LEGSEG_TYPES_.ADA";
137 137      for Spec ("lateral_segment_tpkg") use "LATERAL_SEGMENT_TPKG_.ADA";
138 138      for Spec ("idx_profile_ifdata") use "IDX_PROFILE_IFDATA_.ADA";
139 139      for Spec ("holdupdate_iftypes") use "HOLDUPDATE_IFTYPES_.ADA";
140 140      for Spec ("fprequest_iftypes") use "FPREQUEST_IFTYPES_.ADA";
141 141      for Spec ("fm_sync_imm_types_pkg") use "FM_SYNC_IMM_TYPES_PKG_.ADA";
142 142      for Spec ("fm_navigation_types") use "FM_NAVIGATION_TYPES_.ADA";
143 143      for Spec ("fm_hotspare_iftypes") use "FM_HOTSPARE_IFTYPES_.ADA";
144 144      for Spec ("fm_hotspare_ifdata") use "FM_HOTSPARE_IFDATA_.ADA";
145 145      for Spec ("vg_vert_dev_ifdata") use "FMF_VG_VERT_DEV_IFDATA_.ADA";
146 146      for Spec ("vg_tgt_source_iftypes") use "FMF_VG_TGT_SOURCE_IFTYPES_.ADA";

```

File: CTP_B787_PERF_CRZINITE.ZIP\stubs.gpr (continued)

```

147 147      for Spec ("vg_tgt_source_ifdata") use "FMF_VG_TGT_SOURCE_IFDATA_.ADA";
148 148      for Spec ("vg_speed_tgt_sel_ifdata") use "FMF_VG_SPEED_TGT_SEL_IFDATA_.ADA";
149 149      for Spec ("vg_leg_setup_ifdata") use "FMF_VG_LEG_SETUP_IFDATA_.ADA";
150 150      for Spec ("vg_io_output_iftypes") use "FMF_VG_IO_OUTPUT_IFTYPES_.ADA";
151 151      for Spec ("vg_io_output_ifdata") use "FMF_VG_IO_OUTPUT_IFDATA_.ADA";
152 152      for Spec ("vg_fm_output_iftypes") use "FMF_VG_FM_OUTPUT_IFTYPES_.ADA";
153 153      for Spec ("vg_fast_logic_ifdata") use "FMF_VG_FAST_LOGIC_IFDATA_.ADA";
154 154      for Spec ("vg_engagement_ifdata") use "FMF_VG_ENGAGEMENT_IFDATA_.ADA";
155 155      for Spec ("vg_cur_lim_spds_ifdata") use "FMF_VG_CUR_LIM_SPDS_IFDATA_.ADA";
156 156      for Spec ("fmf_opc_access_pkg") use "FMF_OPC_ACCESS_PKG_.ADA";
157 157      for Body ("fmf_opc_access_pkg") use "FMF_OPC_ACCESS_PKG.ADA";
158 158      for Spec ("fmcs_fp_guid_btipes") use "FMCS_FP_GUID_BTYPES_.ADA";
159 159      for Spec ("b787_pdb_constants") use "FMCS_B787_PDB_CONSTANTS_.ADA";
160 160      for Spec ("fmcs_aem_types_pkg") use "FMCS_AEM_TYPES_PKG.ADA";
161 161      for Spec ("fmcs_aem_trm_psp_pkg") use "FMCS_AEM_TRM_PSP_PKG_.ADA";
162 162      for Body ("fmcs_aem_trm_psp_pkg") use "FMCS_AEM_TRM_PSP_PKG.ADA";
163 163      for Body ("fmcs_aem_trm_psp_pkg.dat_calc") use "FMCS_AEM_TRM_PSP_DAT_CALC_SEP.ADA";
164 164      for Spec ("fmcs_aem_temp_calc_pkg") use "FMCS_AEM_TEMP_CALC_PKG_.ADA";
165 165      for Body ("fmcs_aem_temp_calc_pkg") use "FMCS_AEM_TEMP_CALC_PKG.ADA";
166 166      for Spec ("fmcs_aem_press_calc_pkg") use "FMCS_AEM_PRESS_CALC_PKG_.ADA";
167 167      for Body ("fmcs_aem_press_calc_pkg") use "FMCS_AEM_PRESS_CALC_PKG.ADA";
168 168      for Spec ("fmcs_aem_interpolate_pkg") use "FMCS_AEM_INTERPOLATE_PKG_.ADA";
169 169      for Body ("fmcs_aem_interpolate_pkg") use "FMCS_AEM_INTERPOLATE_PKG.ADA";
170 170      for Spec ("fmcs_aedb_tables_lfdata") use "FMCS_AEDB_TABLES_LFDATA_.ADA";
171 171      for Spec ("fmcs_aedb_init_mgr") use "FMCS_AEDB_INIT_MGR_.ADA";
172 172      for Body ("fmcs_aedb_init_mgr") use "FMCS_AEDB_INIT_MGR.ADA";
173 173      for Spec ("fmcs_aedb_init") use "FMCS_AEDB_INIT_.ADA";
174 174      for Body ("fmcs_aedb_init") use "FMCS_AEDB_INIT.ADA";
175 175      for Spec ("fmcs_aedb_ident_ifdata") use "FMCS_AEDB_IDENT_IFDATA_.ADA";
176 176      for Spec ("perf_constant_init_pkg") use "FMCS_AEDB_CONSTANT_INIT_PKG_.ADA";
177 177      for Body ("perf_constant_init_pkg") use "FMCS_AEDB_CONSTANT_INIT_PKG.ADA";
178 178      for Spec ("fmcs_aedb_constants_ifdata") use "FMCS_AEDB_CONSTANTS_IFDATA_.ADA";
179 179      for Spec ("fmcs_ada_to_c_iface") use "FMCS_ADA_TO_C_IFACE_.ADA";
180 180      for Spec ("fmci_message_tpkg") use "FMCI_MESSAGE_TPKG_.ADA";
181 181      for Spec ("fmci_memory_tpkg") use "FMCI_MEMORY_TPKG_.ADA";
182 182      for Spec ("fmci_memory_saved_dpkg") use "FMCI_MEMORY_SAVED_DPKG_.ADA";
183 183      for Spec ("fmci_memory_dpkg") use "FMCI_MEMORY_DPKG_.ADA";
184 184      for Body ("fmci_memory_dpkg") use "FMCI_MEMORY_DPKG.ADA";
185 185      for Spec ("fmci_bp_req_iftypes") use "FMCI_BP_REQ_IFTYPES_.ADA";
186 186      for Spec ("fmci_annunciator_manager_pkg") use "FMCI_ANNUNCIATOR_MANAGER_PKG_.ADA";
187 187      for Body ("fmci_annunciator_manager_pkg") use "FMCI_ANNUNCIATOR_MANAGER_PKG.ADA";
188 188      for Spec ("flx_waypoint_tpkg") use "FLX_WAYPOINT_TPKG_.ADA";

```

File: CTP_B787_PERF_CRZINITE.ZIP\stubs.gpr (continued)

```

189      189      for Spec ("flx_waypoint_pkg") use "FLX_WAYPOINT_PKG_.ADA";
190      190      for Body ("flx_waypoint_pkg") use "FLX_WAYPOINT_PKG.ADA";
191      191      for Spec ("flx_vhf_navaid_tpkg") use "FLX_VHF_NAVAID_TPKG_.ADA";
192      192      for Spec ("flx_vhf_navaid_pkg") use "FLX_VHF_NAVAID_PKG_.ADA";
193      193      for Body ("flx_vhf_navaid_pkg") use "FLX_VHF_NAVAID_PKG.ADA";
194      194      for Spec ("flx_utils_pkg") use "FLX_UTILS_PKG_.ADA";
195      195      for Body ("flx_utils_pkg") use "FLX_UTILS_PKG.ADA";
196      196      for Spec ("flx_tap_legs_tpkg") use "FLX_TAP_LEGS_TPKG_.ADA";
197      197      for Spec ("flx_tap_legs_pkg") use "FLX_TAP_LEGS_PKG_.ADA";
198      198      for Body ("flx_tap_legs_pkg") use "FLX_TAP_LEGS_PKG.ADA";
199      199      for Spec ("flx_star_tpkg") use "FLX_STAR_TPKG_.ADA";
200      200      for Spec ("flx_star_pkg") use "FLX_STAR_PKG_.ADA";
201      201      for Body ("flx_star_pkg") use "FLX_STAR_PKG.ADA";
202      202      for Spec ("flx_sid_tpkg") use "FLX_SID_TPKG_.ADA";
203      203      for Spec ("flx_sid_pkg") use "FLX_SID_PKG_.ADA";
204      204      for Body ("flx_sid_pkg") use "FLX_SID_PKG.ADA";
205      205      for Spec ("flx_runway_tpkg") use "FLX_RUNWAY_TPKG_.ADA";
206      206      for Spec ("flx_runway_pkg") use "FLX_RUNWAY_PKG_.ADA";
207      207      for Body ("flx_runway_pkg") use "FLX_RUNWAY_PKG.ADA";
208      208      for Spec ("flx_ndrb_pkg") use "FLX_NDRB_PKG_.ADA";
209      209      for Body ("flx_ndrb_pkg") use "FLX_NDRB_PKG.ADA";
210      210      for Spec ("flx_nav_airport_pkg") use "FLX_NAV_AIRPORT_PKG_.ADA";
211      211      for Body ("flx_nav_airport_pkg") use "FLX_NAV_AIRPORT_PKG.ADA";
212      212      for Body ("flx_navigation_pkg.get_closest_two_nav aids_by_freq") use "FLX_NAVIGATION_PKG_CLO_2_NAV_BY_FRQ.ADA";
213      213      for Body ("flx_navigation_pkg.build_nav_list") use "FLX_NAVIGATION_PKG_BUILD_NAV_LIST.ADA";
214      214      for Body ("flx_navigation_pkg.build_nav_tuning_list") use "FLX_NAVIGATION_PKG_BLD_NAV_TUNE_LST.ADA";
215      215      for Spec ("flx_navigation_pkg") use "FLX_NAVIGATION_PKG_.ADA";
216      216      for Body ("flx_navigation_pkg") use "FLX_NAVIGATION_PKG.ADA";
217      217      for Spec ("flx_navigation_ifdata") use "FLX_NAVIGATION_IFDATA_.ADA";
218      218      for Body ("flx_navigation_ifdata") use "FLX_NAVIGATION_IFDATA.ADA";
219      219      for Spec ("flx_localizer_tpkg") use "FLX_LOCALIZER_TPKG_.ADA";
220      220      for Spec ("flx_localizer_pkg") use "FLX_LOCALIZER_PKG_.ADA";
221      221      for Body ("flx_localizer_pkg") use "FLX_LOCALIZER_PKG.ADA";
222      222      for Spec ("flx_iftypes") use "FLX_IPTYPES_.ADA";
223      223      for Spec ("flx_holding_pattern_tpkg") use "FLX_HOLDING_PATTERN_TPKG_.ADA";
224      224      for Spec ("flx_holding_pattern_pkg") use "FLX_HOLDING_PATTERN_PKG_.ADA";
225      225      for Body ("flx_holding_pattern_pkg") use "FLX_HOLDING_PATTERN_PKG.ADA";
226      226      for Spec ("flx_gate_tpkg") use "FLX_GATE_TPKG_.ADA";
227      227      for Spec ("flx_gate_pkg") use "FLX_GATE_PKG_.ADA";
228      228      for Body ("flx_gate_pkg") use "FLX_GATE_PKG.ADA";
229      229      for Body ("flx_efis_pkg.build_wpt_candidate_list") use "FLX_EFIS_PKG_BUILD_WPT_CANDIDATE_LIST.ADA";
230      230      for Body ("flx_efis_pkg.build_ndrb_candidate_list") use "FLX_EFIS_PKG_BUILD_NDRB_CANDIDATE_LIST.ADA";

```

File: CTP_B787_PERF_CRZINITE.ZIP\stubs.gpr (continued)

```

231 231      for Body ("flx_efis_pkg.build_nav_candidate_list") use "FLX_EFIS_PKG_BUILD_NAV_CANDIDATE_LIST.ADA";
232 232      for Body ("flx_efis_pkg.build Apt_candidate_list") use "FLX_EFIS_PKG_BUILD_APT_CANDIDATE_LIST.ADA";
233 233      for Spec ("flx_efis_pkg") use "FLX_EFIS_PKG.ADA";
234 234      for Body ("flx_efis_pkg") use "FLX_EFIS_PKG.ADA";
235 235      for Spec ("flx_db_utils_pkg") use "FLX_DB_UTILS_PKG.ADA";
236 236      for Body ("flx_db_utils_pkg") use "FLX_DB_UTILS_PKG.ADA";
237 237      for Spec ("flx_corte_tpkg") use "FLX_CORTE_TPKG.ADA";
238 238      for Spec ("flx_corte_pkg") use "FLX_CORTE_PKG.ADA";
239 239      for Body ("flx_corte_pkg") use "FLX_CORTE_PKG.ADA";
240 240      for Spec ("flx_bite_pkg") use "FLX_BITE_PKG.ADA";
241 241      for Body ("flx_bite_pkg") use "FLX_BITE_PKG.ADA";
242 242      for Spec ("flx_approach_tpkg") use "FLX_APPROACH_TPKG.ADA";
243 243      for Spec ("flx_approach_pkg") use "FLX_APPROACH_PKG.ADA";
244 244      for Body ("flx_approach_pkg") use "FLX_APPROACH_PKG.ADA";
245 245      for Spec ("flx_any_navaid_tpkg") use "FLX_ANY_NAVAID_TPKG.ADA";
246 246      for Spec ("flx_any_navaid_pkg") use "FLX_ANY_NAVAID_PKG.ADA";
247 247      for Body ("flx_any_navaid_pkg") use "FLX_ANY_NAVAID_PKG.ADA";
248 248      for Spec ("flx_airway_tpkg") use "FLX_AIRWAY_TPKG.ADA";
249 249      for Spec ("flx_airway_pkg") use "FLX_AIRWAY_PKG.ADA";
250 250      for Body ("flx_airway_pkg") use "FLX_AIRWAY_PKG.ADA";
251 251      for Spec ("flx_airport_tpkg") use "FLX_AIRPORT_TPKG.ADA";
252 252      for Spec ("flx_airport_pkg") use "FLX_AIRPORT_PKG.ADA";
253 253      for Body ("flx_airport_pkg") use "FLX_AIRPORT_PKG.ADA";
254 254      for Spec ("flight_pln_leg_types") use "FLIGHT_PLN_LEG_TYPES.ADA";
255 255      for Spec ("flight_pln_hold_ent_types") use "FLIGHT_PLN_HOLD_ENT_TYPES.ADA";
256 256      for Spec ("flight_pln_hdr_types") use "FLIGHT_PLN_HDR_TYPES.ADA";
257 257      for Spec ("fix_info_iftypes") use "FIX_INFO_IFTYPES.ADA";
258 258      for Spec ("fix_info_ifdata") use "FIX_INFO_IFDATA.ADA";
259 259      for Spec ("ops_bite_ifdata_pkg") use "FCS_OPS_BITE_IFDATA.ADA";
260 260      for Spec ("fcs_iin_generic_data_def_pkg") use "FCS_IIN_GENERIC_DATA_DEF_PKG.ADA";
261 261      for Spec ("assert_pkg") use "FCS_ASSERT_PKG.ADA";
262 262      for Body ("assert_pkg") use "FCS_ASSERT_PKG.ADA";
263 263      for Spec ("efis_ops_ifdata") use "EFIS_OPS_IFDATA.ADA";
264 264      for Spec ("dtrrec_types") use "DTRREC_TYPES.ADA";
265 265      for Spec ("dst_brg_utilities_pkg") use "DST_BRG_UTILITIES_PKG.ADA";
266 266      for Body ("dst_brg_utilities_pkg") use "DST_BRG_UTILITIES_PKG.ADA";
267 267      for Spec ("dirto_wpt_iftypes") use "DIRTO_WPT_IFTYPES.ADA";
268 268      for Spec ("descent_path_iftypes") use "DESCENT_PATH_IFTYPES.ADA";
269 269      for Spec ("deldirect_iftypes") use "DELDIRECT_IFTYPES.ADA";
270 270      for Spec ("definefixrec_iftypes") use "DEFINEFIXREC_IFTYPES.ADA";
271 271      for Spec ("cs_oss_access_gnrc") use "CS_OSS_ACCESS_GNRC.ADA";
272 272      for Body ("cs_oss_access_gnrc") use "CS_OSS_ACCESS_GNRC.ADA";

```

File: CTP_B787_PERF_CRZINITE.ZIP\stubs.gpr (continued)

```

273 273      for Spec ("cs_database_access_iftypes") use "CS_DATABASE_ACCESS_IFTYPES_.ADA";
274 274      for Spec ("cs_database_access_gnrc") use "CS_DATABASE_ACCESS_GNRC_.ADA";
275 275      for Body ("cs_database_access_gnrc") use "CS_DATABASE_ACCESS_GNRC.ADA";
276 276      for Spec ("compute_phinom_pkg") use "COMPUTE_PHINOM_PKG_.ADA";
277 277      for Body ("compute_phinom_pkg") use "COMPUTE_PHINOM_PKG.ADA";
278 278      for Spec ("compat_check_result_pkg") use "COMPAT_CHECK_RESULT_PKG_.ADA";
279 279      for Body ("compat_check_result_pkg") use "COMPAT_CHECK_RESULT_PKG.ADA";
280 280      for Spec ("common_io_pkg") use "COMMON_IO_PKG_.ADA";
281 281      for Body ("common_io_pkg") use "COMMON_IO_PKG.ADA";
282 282      for Spec ("close_intc_crs_from_iftypes") use "CLOSE_INTC_CRS_FROM_IFTYPES_.ADA";
283 283      for Spec ("cfp_vg_iftypes") use "CFP_VG_IFTYPES_.ADA";
284 284      for Spec ("cfp_perf_step_iftypes") use "CFP_PERF_STEP_IFTYPES_.ADA";
285 285      for Spec ("cfp_io_download_iftypes") use "CFP_IO_DOWNLOAD_IFTYPES_.ADA";
286 286      for Spec ("cfp_io_download_ifdata") use "CFP_IO_DOWNLOAD_IFDATA_.ADA";
287 287      for Spec ("cfp_efis_iftypes") use "CFP_EFIS_IFTYPES_.ADA";
288 288      for Spec ("cfp_efis_ifdata") use "CFP_EFIS_IFDATA_.ADA";
289 289      for Spec ("cfp_directories_iftypes") use "CFP_DIRECTORIES_IFTYPES_.ADA";
290 290      for Spec ("cfp_cdk_ifdata") use "CFP_CDK_IFDATA_.ADA";
291 291      for Spec ("cfp_calcmagvar_pkg") use "CFP_CALCMAGVAR_PKG_.ADA";
292 292      for Body ("cfp_calcmagvar_pkg") use "CFP_CALCMAGVAR_PKG.ADA";
293 293      for Spec ("cex_bite_iftypes") use "CEX_BITE_IFTYPES_.ADA";
294 294      for Spec ("cex_bite_ifdata") use "CEX_BITE_IFDATA_.ADA";
295 295      for Spec ("cdu_output_iftypes") use "CDU_OUTPUT_IFTYPES_.ADA";
296 296      for Spec ("cdl_dltype_iftypes") use "CDL_DLTYPE_IFTYPES_.ADA";
297 297      for Spec ("cdl_dltokentype_data") use "CDL_DLTOKENTYPE_DATA_.ADA";
298 298      for Spec ("cdl_cdk_takeoff_iftypes") use "CDL_CDK_TAKEOFF_UPLINK_IFTYPES_.ADA";
299 299      for Spec ("cdl_cdk_perf_init_iftypes") use "CDL_CDK_PERF_INIT_IFTYPES_.ADA";
300 300      for Spec ("cdl_cdk_perf_init_ifdata") use "CDL_CDK_PERF_INIT_IFDATA_.ADA";
301 301      for Spec ("cdl_cdk_fpln_iftypes") use "CDL_CDK_FPLN_IFTYPES_.ADA";
302 302      for Spec ("cdl_cdk_flight_plan_uplink_ifdata") use "CDL_CDK_FLIGHT_PLAN_UPLINK_IFDATA_.ADA";
303 303      for Spec ("cdl_cdk_alternates_iftypes") use "CDL_CDK_ALTERNATES_IFTYPES_.ADA";
304 304      for Spec ("cdl_cdk_alternates_ifdata") use "CDL_CDK_ALTERNATES_IFDATA_.ADA";
305 305      for Spec ("cdk_reference_iftypes") use "CDK_REFERENCE_IFTYPES_.ADA";
306 306      for Spec ("cdk_page_iftypes") use "CDK_PAGE_IFTYPES_.ADA";
307 307      for Spec ("cdk_offpath_iftypes") use "CDK_OFFPATH_IFTYPES_.ADA";
308 308      for Spec ("cdk_nav_radio_preselect_iftypes") use "CDK_NAV_RADIO_PRESELECT_IFTYPES_.ADA";
309 309      for Spec ("cdk_nav_radio_iftypes") use "CDK_NAV_RADIO_IFTYPES_.ADA";
310 310      for Spec ("cdk_key_data_iftypes") use "CDK_KEY_DATA_IFTYPES_.ADA";
311 311      for Spec ("cdk_internal_ltypes") use "CDK_INTERNAL_LTYPES_.ADA";
312 312      for Spec ("cdk_fix_entry_iftypes") use "CDK_FIX_ENTRY_IFTYPES_.ADA";
313 313      for Body ("cdk_fix_entry_iftypes") use "CDK_FIX_ENTRY_IFTYPES.ADA";
314 314      for Spec ("cabin_pressure_ifdata") use "CABIN_PRESSURE_IFDATA_.ADA";

```

File: CTP_B787_PERF_CRZINITE.ZIP\stubs.gpr (continued)

```

315 315      for Spec ("bld_intc_crs_from_iftypes") use "BLD_INTC_CRS_FROM_IFTYPES_.ADA";
316 316      for Spec ("bldvia_iftypes") use "BLDVIA_IFTYPES_.ADA";
317 317      for Spec ("bldvert_iftypes") use "BLDVERT_IFTYPES_.ADA";
318 318      for Spec ("bldrta_iftypes") use "BLDRTA_IFTYPES_.ADA";
319 319      for Spec ("blddeparr_iftypes") use "BLDDEPARR_IFTYPES_.ADA";
320 320      for Spec ("bldcorte_iftypes") use "BLDCORTE_IFTYPES_.ADA";
321 321      for Spec ("bldapt_iftypes") use "BLDAPT_IFTYPES_.ADA";
322 322      for Spec ("bldalternate_iftypes") use "BLDALTERNATE_IFTYPES_.ADA";
323 323      for Spec ("bldaltcnstr_iftypes") use "BLDALTCNSTR_IFTYPES_.ADA";
324 324      for Spec ("ioc_wst_wordstring_types") use "BITE_IOC_WST_WORDSTRING_TYPES_.ADA";
325 325      for Spec ("bite_advanced_fault_record_func_pkg") use "BITE_ADVANCED_FAULT_RECORD_FUNC_PKG_.ADA";
326 326      for Body ("bite_advanced_fault_record_func_pkg") use "BITE_ADVANCED_FAULT_RECORD_FUNC_PKG.ADA";
327 327      for Spec ("base_domain_services_tpkg") use "BASE_DOMAIN_SERVICES_TPKG_.ADA";
328 328      for Spec ("options_and_data_pkg") use "B787_OPTIONS_AND_DATA_PKG_.ADA";
329 329      for Body ("options_and_data_pkg") use "B787_OPTIONS_AND_DATA_PKG.ADA";
330 330      for Spec ("flx_user_id_dpkg") use "B787_FLX_USER_ID_DPKG_.ADA";
331 331      for Spec ("arr_dep_iftypes") use "ARR_DEP_IFTYPES_.ADA";
332 332      for Spec ("arinc_base_types_pkg") use "ARINC_BASE_TYPES_PKG_.ADA";
333 333      for Spec ("arinc_629_rep_constants_pkg") use "ARINC_629_REP_CONSTANTS_PKG_.ADA";
334 334      for Spec ("ami_iftypes") use "AMI_IFTYPES_.ADA";
335 335      for Spec ("alt_profile_iftypes") use "ALT_PROFILE_IFTYPES_.ADA";
336 336      for Spec ("alternate_airport_iftypes") use "ALTERNATE_AIRPORT_IFTYPES_.ADA";
337 337      for Spec ("ac_position_types") use "AC_POSITION_TYPES_.ADA";
338 338      for Spec ("ac_config_types") use "AC_CONFIG_TYPES_.ADA";
339 339      for Spec ("acars_buffer_iftypes") use "ACARS_BUFFER_IFTYPES_.ADA";
340 340
341 341      --fm naming
342 342      for Body ("ops_fm_partition_init_pkg") use "OPS_FM_PARTITION_INIT_PKG.ADA";
343 343      for Body ("ops_fm_init_status_pkg") use "OPS_FM_INIT_STATUS_PKG.ADA";
344 344      for Body ("ops_fm_database_verification_pkg") use "OPS_FM_DATABASE_VERIFICATION_PKG.ADA";
345 345      for Body ("ops_efis_bg_pkg") use "OPS_EFIS_BG_PKG.ADA";
346 346      for Body ("bite_exec_pkg") use "FMF_BITE_EXEC_PKG.ADA";
347 347      for Body ("efis_store_buff_pkg") use "EFIS_STORE_BUF.ADA";
348 348      for Spec ("bite_large_bp_service_pkg") use "BITE_LARGE_BP_SERVICE_PKG_.ADA";
349 349      for Body ("bite_large_bp_service_pkg") use "BITE_LARGE_BP_SERVICE_PKG.ADA";
350 350      for Spec ("perf_wts_lfdata") use "PERF_WTS_LFDATA_.ADA";
351 351      for Spec ("perf_wind_lftypes") use "PERF_WIND_LFTYPES_.ADA";
352 352      for Spec ("perf_wind_lfdata") use "PERF_WIND_LFDATA_.ADA";
353 353      for Body ("perf_vtp_pkg.vtp_post_processing") use "PERF_VTP_POST_PROCESSING_SEP.ADA";
354 354      for Spec ("perf_vtp_pkg") use "PERF_VTP_PKG_.ADA";
355 355      for Body ("perf_vtp_pkg") use "PERF_VTP_PKG.ADA";
356 356      for Spec ("perf_vtp_lfdata") use "PERF_VTP_LFDATA_.ADA";

```

File: CTP_B787_PERF_CRZINITE.ZIP\stubs.gpr (continued)

```

357 357      for Body ("perf_vtp_pkg.vtp_init") use "PERF_VTP_INIT_SEP.ADA";
358 358      for Body ("perf_vtp_pkg.vtp_exec") use "PERF_VTP_EXEC_SEP.ADA";
359 359      for Body ("perf_vdu_utils.make_buffer") use "PERF_VDU_UTILS_MAKE_BUFFER_SEP.ADA";
360 360      for Spec ("perf_vdu_utils") use "PERF_VDU_UTILS_.ADA";
361 361      for Body ("perf_vdu_utils") use "PERF_VDU_UTILS.ADA";
362 362      for Spec ("perf_vdu_lftypes") use "PERF_VDU_LFTYPES_.ADA";
363 363      for Spec ("perf_vdu_lfdata") use "PERF_VDU_LFDATA_.ADA";
364 364      for Spec ("perf_tto_allow_pkg") use "PERF_TTO_ALLOW_PKG_.ADA";
365 365      for Body ("perf_tto_allow_pkg") use "PERF_TTO_ALLOW_PKG.ADA";
366 366      for Spec ("perf_top_of_des_lfdata") use "PERF_TOP_OF_DES_LFDATA_.ADA";
367 367      for Spec ("perf_task_control_lfdata") use "PERF_TASK_CONTROL_LFDATA_.ADA";
368 368      for Body ("perf_su_spd_utils_pkg.su_machterm") use "PERF_SU_MACHTERM_SEP.ADA";
369 369      for Body ("perf_su_spd_utils_pkg.su_frmtgtspdrec") use "PERF_SU_FRMTGTSPDREC_SEP.ADA";
370 370      for Body ("perf_su_spd_utils_pkg.su_comptgtspd") use "PERF_SU_COMPTGTSPD_SEP.ADA";
371 371      for Body ("perf_su_spd_utils_pkg.su_compgndspd") use "PERF_SU_COMPGNDSPD_SEP.ADA";
372 372      for Body ("perf_st_spdtape_pkg.st_ten_hz") use "PERF_ST_TEN_HZ_SEP.ADA";
373 373      for Body ("perf_st_spdtape_pkg.st_one_hz") use "PERF_ST_ONE_HZ_SEP.ADA";
374 374      for Body ("perf_st_spdtape_pkg.st_five_hz") use "PERF_ST_FIVE_HZ_SEP.ADA";
375 375      for Body ("perf_act_spd_env_pkg.slow_act_spd_env") use "PERF_SLOW_ACT_SPD_ENV_SEP.ADA";
376 376      for Body ("perf_act_spd_env_pkg.slow_act_spd_env_put_data") use "PERF_SLOW_ACT_SPD_ENV_PUT_DATA_SEP.ADA";
377 377      for Body ("perf_act_spd_env_pkg.slow_act_spd_env_init_data") use "PERF_SLOW_ACT_SPD_ENV_INIT_DATA_SEP.ADA";
378 378      for Body ("perf_act_spd_env_pkg.slow_act_spd_env_get_data") use "PERF_SLOW_ACT_SPD_ENV_GET_DATA_SEP.ADA";
379 379      for Body ("perf_sg_spd_gen_pkg.sg_spd_gen") use "PERF_SG_SPD_GEN_SEP.ADA";
380 380      for Spec ("perf_sg_spd_gen_pkg") use "PERF_SG_SPD_GEN_PKG_.ADA";
381 381      for Body ("perf_sg_spd_gen_pkg") use "PERF_SG_SPD_GEN_PKG.ADA";
382 382      for Body ("perf_sg_spd_gen_pkg.sg_max_angle_spd") use "PERF_SG_MAX_ANGLE_SPD_SEP.ADA";
383 383      for Body ("perf_sg_spd_gen_pkg.sg_hold_spd") use "PERF_SG_HOLD_SPD_SEP.ADA";
384 384      for Spec ("perf_rta_lfdata") use "PERF_RTA_LFDATA_.ADA";
385 385      for Body ("perf_rp_guidprms_pkg.rp_vnav_ref_params") use "PERF_RP_VNAV_REF_PARAMS_SEP.ADA";
386 386      for Body ("perf_rp_guidprms_pkg.rp_unablenxtalt_msg") use "PERF_RP_UNABLENXTALT_MSG_SEP.ADA";
387 387      for Body ("perf_rp_guidprms_pkg.rp_thrust_targets") use "PERF_RP_THRUST_TARGETS_SEP.ADA";
388 388      for Body ("perf_rp_guidprms_pkg.rp_thrust_roll_limit") use "PERF_RP_THRUST_ROLL_LIMIT_SEP.ADA";
389 389      for Body ("perf_rp_guidprms_pkg.rp_roll_limit") use "PERF_RP_ROLL_LIMIT_SEP.ADA";
390 390      for Body ("perf_rp_guidprms_pkg.rp_refalt") use "PERF_RP_REFALT_SEP.ADA";
391 391      for Body ("perf_rp_guidprms_pkg.rp_put_data") use "PERF_RP_PUT_DATA_SEP.ADA";
392 392      for Body ("perf_rp_guidprms_pkg.rp_pitch_limit") use "PERF_RP_PITCH_LIMIT_SEP.ADA";
393 393      for Body ("perf_rp_guidprms_pkg.rp_manlimalt") use "PERF_RP_MANLIMALT_SEP.ADA";
394 394      for Body ("perf_rp_guidprms_pkg.rp_guidprms") use "PERF_RP_GUIDPRMS_SEP.ADA";
395 395      for Spec ("perf_rp_guidprms_pkg") use "PERF_RP_GUIDPRMS_PKG_.ADA";
396 396      for Body ("perf_rp_guidprms_pkg") use "PERF_RP_GUIDPRMS_PKG.ADA";
397 397      for Body ("perf_rp_guidprms_pkg.rp_get_data") use "PERF_RP_GET_DATA_SEP.ADA";
398 398      for Body ("perf_rp_guidprms_pkg.rp_fltphase") use "PERF_RP_FLTPHASE_SEP.ADA";

```

File: CTP_B787_PERF_CRZINITE.ZIP\stubs.gpr (continued)

```

399      399      for Body ("perf_rp_guidprms_pkg.rp_command_tsp") use "PERF_RP_COMMAND_TSP_SEP.ADA";
400      400      for Spec ("perf_restart_preds_pkg") use "PERF_RESTART_PREDS_PKG_.ADA";
401      401      for Body ("perf_restart_preds_pkg") use "PERF_RESTART_PREDS_PKG.ADA";
402      402      for Body ("perf_restart_preds_pkg.restart_clear_perf_data") use "PERF_RESTART_CLEAR_PERF_DATA_SEP.ADA";
403      403      for Body ("perf_restart_preds_pkg.restart_check") use "PERF_RESTART_CHECK_SEP.ADA";
404      404      for Spec ("perf_recmd_crz_fl_pkg") use "PERF_RECMD_CRZ_FL_PKG_.ADA";
405      405      for Body ("perf_recmd_crz_fl_pkg") use "PERF_RECMD_CRZ_FL_PKG.ADA";
406      406      for Body ("perf_recmd_crz_fl_pkg.recmd_crz_fl_comp") use "PERF_RECMD_CRZ_FL_COMP_SEP.ADA";
407      407      for Spec ("perf_punt_pkg") use "PERF_PUNT_PKG_.ADA";
408      408      for Body ("perf_punt_pkg") use "PERF_PUNT_PKG.ADA";
409      409      for Spec ("perf_punt_lftypes") use "PERF_PUNT_LFTYPES_.ADA";
410      410      for Spec ("perf_punt_lfdata") use "PERF_PUNT_LFDATA_.ADA";
411      411      for Body ("perf_pte_pkg.pte_sel_task") use "PERF_PTE_SEL_TASK_SEP.ADA";
412      412      for Body ("perf_pte_pkg.pte_sel_priority_task") use "PERF_PTE_SEL_PRIORITY_TASK_SEP.ADA";
413      413      for Spec ("perf_pte_pkg") use "PERF_PTE_PKG_.ADA";
414      414      for Body ("perf_pte_pkg") use "PERF_PTE_PKG.ADA";
415      415      for Body ("perf_pte_pkg.pte_init_timers") use "PERF_PTE_INIT_TIMERS_SEP.ADA";
416      416      for Body ("perf_pte_pkg.pte_init_powerup_data") use "PERF_PTE_INIT_POWERUP_DATA_SEP.ADA";
417      417      for Body ("perf_pte_pkg.pte_get_subtask_data") use "PERF_PTE_GET_SUBTASK_DATA_SEP.ADA";
418      418      for Body ("perf_pte_pkg.pte_get_lgb") use "PERF_PTE_GET_LGB_SEP.ADA";
419      419      for Body ("perf_pte_pkg.pte_get_exec_data") use "PERF_PTE_GET_EXEC_DATA_SEP.ADA";
420      420      for Body ("perf_pte_pkg.pte_exec") use "PERF_PTE_EXEC_SEP.ADA";
421      421      for Body ("perf_pte_pkg.pte_calc_timers") use "PERF_PTE_CALC_TIMERS_SEP.ADA";
422      422      for Body ("perf_pte_pkg.calc_preds_available") use "PERF_PTE_CALC_PREDS_AVAILABLE_SEP.ADA";
423      423      for Spec ("perf_profile_lfdata") use "PERF_PROFILE_LFDATA_.ADA";
424      424      for Spec ("perf_pred_spd_env_pkg") use "PERF_PRED_SPD_ENV_PKG_.ADA";
425      425      for Body ("perf_pred_spd_env_pkg") use "PERF_PRED_SPD_ENV_PKG.ADA";
426      426      for Spec ("perf_preds_lftypes") use "PERF_PREDS_LFTYPES_.ADA";
427      427      for Spec ("perf_preds_lfdata") use "PERF_PREDS_LFDATA_.ADA";
428      428      for Spec ("perf_point_termination_types") use "PERF_POINT_TERMINATION_TYPES_.ADA";
429      429      for Spec ("perf_persistent_lfdata") use "PERF_PERSISTENT_LFDATA_.ADA";
430      430      for Spec ("perf_origin_dest_lfdata") use "PERF_ORIGIN_DEST_LFDATA_.ADA";
431      431      for Body ("perf_opd_pkg.opd_proctermes") use "PERF_OPD_PROCTERMS_SEP.ADA";
432      432      for Body ("perf_opd_pkg.opd_predexec") use "PERF_OPD_PREDEXEC_SEP.ADA";
433      433      for Spec ("perf_opd_pkg") use "PERF_OPD_PKG_.ADA";
434      434      for Body ("perf_opd_pkg") use "PERF_OPD_PKG.ADA";
435      435      for Body ("perf_opd_pkg.opd_init") use "PERF_OPD_INIT_SEP.ADA";
436      436      for Body ("perf_opd_pkg.opd_calcradius") use "PERF_OPD_CALCRADIUS_SEP.ADA";
437      437      for Spec ("perf_opa_pkg") use "PERF_OPA_PKG_.ADA";
438      438      for Body ("perf_opa_pkg") use "PERF_OPA_PKG.ADA";
439      439      for Spec ("perf_onpath_lfdata") use "PERF_ONPATH_LFDATA_.ADA";
440      440      for Spec ("perf_offpath_des_lftypes") use "PERF_OFFPATH_DES_LFTYPES.ADA";

```


File: CTP_B787_PERF_CRZINITE.ZIP\stubs.gpr (continued)

441	441	for Spec ("perf_offpath_des_lfdata") use "PERF_OFFPATH_DES_LFDATA.ADA";
442	442	for Body ("perf_oa_optalt_pkg.oa_short_trip") use "PERF_OA_SHORT_TRIP_SEP.ADA";
443	443	for Body ("perf_oa_optalt_pkg.oa_optalt") use "PERF_OA_OPTALT_SEP.ADA";
444	444	for Spec ("perf_oa_optalt_pkg") use "PERF_OA_OPTALT_PKG.ADA";
445	445	for Body ("perf_oa_optalt_pkg") use "PERF_OA_OPTALT_PKG.ADA";
446	446	for Body ("perf_oa_optalt_pkg.oa_optalt_comp") use "PERF_OA_OPTALT_COMP_SEP.ADA";
447	447	for Body ("perf_oa_optalt_pkg.oa_long_trip") use "PERF_OA_LONG_TRIP_SEP.ADA";
448	448	for Body ("perf_oa_optalt_pkg.oa_eqv_trip_dist") use "PERF_OA_EQV_TRIP_DIST_SEP.ADA";
449	449	for Spec ("perf_msg_flags_lfdata") use "PERF_MSG_FLAGS_LFDATA.ADA";
450	450	for Body ("perf_max_alt_pkg.max_spd_at_alt") use "PERF_MAX_SPD_AT_ALT_SEP.ADA";
451	451	for Spec ("perf_max_opt_lftypes") use "PERF_MAX_OPT_LFTYPES.ADA";
452	452	for Spec ("perf_max_opt_lfdata") use "PERF_MAX_OPT_LFDATA.ADA";
453	453	for Body ("perf_max_alt_pkg.max_maroot") use "PERF_MAX_MAROOT_SEP.ADA";
454	454	for Body ("perf_max_alt_pkg.max_margincrv") use "PERF_MAX_MARGINCRV_SEP.ADA";
455	455	for Body ("perf_max_alt_pkg.max_alt") use "PERF_MAX_ALT_SEP.ADA";
456	456	for Spec ("perf_max_alt_pkg") use "PERF_MAX_ALT_PKG.ADA";
457	457	for Body ("perf_max_alt_pkg") use "PERF_MAX_ALT_PKG.ADA";
458	458	for Body ("perf_max_alt_pkg.max_alt_comp") use "PERF_MAX_ALT_COMP_SEP.ADA";
459	459	for Body ("perf_max_alt_pkg.max_alt_comp_eo") use "PERF_MAX_ALT_COMP_EO_SEP.ADA";
460	460	for Body ("perf_lgb_pkg.lgb_store_data") use "PERF_LGB_STORE_DATA_SEP.ADA";
461	461	for Body ("perf_lgb_pkg.lgb_seq_rta_leg") use "PERF_LGB_SEQ_RTA_LEG_SEP.ADA";
462	461	for Body ("perf_lgb_pkg.lgb_seq_leg") use "PERF_LGB_SEQ_LEG_SEP.ADA";
463	462	for Body ("perf_lgb_pkg.lgb_search") use "PERF_LGB_SEARCH_SEP.ADA";
464	463	for Spec ("perf_lgb_pkg") use "PERF_LGB_PKG.ADA";
465	464	for Body ("perf_lgb_pkg") use "PERF_LGB_PKG.ADA";
466	465	for Body ("perf_lgb_pkg.output_preds") use "PERF_LGB_OUTPUT_PREDS_SEP.ADA";
467	466	for Spec ("perf_lgb_minileg_types") use "PERF_LGB_MINILEG_TYPES.ADA";
468	467	for Spec ("perf_lgb_lfdata") use "PERF_LGB_LFDATA.ADA";
469	468	for Body ("perf_lgb_pkg.create_point") use "PERF_LGB_CREATE_POINT_SEP.ADA";
470	469	for Body ("perf_int_pkg.int_vpath") use "PERF_INT_VPATH_SEP.ADA";
471	470	for Body ("perf_int_pkg.int_vertspd") use "PERF_INT_VERTSPD_SEP.ADA";
472	471	for Body ("perf_int_pkg.int_pthdes") use "PERF_INT_PTHDES_SEP.ADA";
473	472	for Spec ("perf_int_pkg") use "PERF_INT_PKG.ADA";
474	473	for Body ("perf_int_pkg") use "PERF_INT_PKG.ADA";
475	474	for Body ("perf_int_pkg.int_lvltgtspd") use "PERF_INT_LVLTGTSPD_SEP.ADA";
476	475	for Body ("perf_int_pkg.int_lvlaclcl") use "PERF_INT_LVLACLDCL_SEP.ADA";
477	476	for Body ("perf_int_pkg.int_driftdown") use "PERF_INT_DRIFTDOWN_SEP.ADA";
478	477	for Body ("perf_int_pkg.int_decellimit") use "PERF_INT_DECELLIMIT_SEP.ADA";
479	478	for Body ("perf_int_pkg.int_clbaccel") use "PERF_INT_CLBACCEL_SEP.ADA";
480	479	for Body ("perf_int_pkg.int_chekintterms") use "PERF_INT_CHEKINTTERMS_SEP.ADA";
481	480	for Body ("perf_int_pkg.int_cdtgtspd") use "PERF_INT_CDTGTSPD_SEP.ADA";
482	481	for Spec ("perf_integrators_lftypes") use "PERF_INTEGRATORS_LFTYPES.ADA";

File: CTP_B787_PERF_CRZINITE.ZIP\stubs.gpr (continued)

```

483 482      for Spec ("perf_integrators_lfdata") use "PERF_INTEGRATORS_LFDATA_.ADA";
484 483      for Spec ("perf_idx_wind_lfdata") use "PERF_IDX_WIND_LFDATA_.ADA";
485 484      for Spec ("perf_idx_top_of_des_lfdata") use "PERF_IDX_TOP_OF_DES_LFDATA_.ADA";
486 485      for Spec ("perf_idx_origin_dest_lfdata") use "PERF_IDX_ORIGIN_DEST_LFDATA_.ADA";
487 486      for Spec ("perf_idx_msg_flags_lfdata") use "PERF_IDX_MSG_FLAGS_LFDATA_.ADA";
488 487      for Spec ("perf_idx_crzalt_lfdata") use "PERF_IDX_CRZALT_LFDATA_.ADA";
489 488      for Spec ("perf_hld_pkg") use "PERF_HLD_PKG_.ADA";
490 489      for Body ("perf_hld_pkg") use "PERF_HLD_PKG.ADA";
491 490      for Spec ("perf_hdu_pkg") use "PERF_HDU_PKG_.ADA";
492 491      for Body ("perf_hdu_pkg") use "PERF_HDU_PKG.ADA";
493 492      for Spec ("perf_guid_header_lftypes") use "PERF_GUID_HEADER_LFTYPES_.ADA";
494 493      for Spec ("perf_fwt_pkg") use "PERF_FWT_PKG_.ADA";
495 494      for Body ("perf_fwt_pkg") use "PERF_FWT_PKG.ADA";
496 495      for Spec ("perf_fwt_lfdata") use "PERF_FWT_LFDATA_.ADA";
497 496      for Body ("perf_fwt_pkg.fwt_init") use "PERF_FWT_INIT_SEP.ADA";
498 497      for Body ("perf_fwt_pkg.fwt_get_data") use "PERF_FWT_GET_DATA_SEP.ADA";
499 498      for Body ("perf_fwt_pkg.fwt_exec") use "PERF_FWT_EXEC_SEP.ADA";
500 499      for Spec ("perf_fuelintime_pkg") use "PERF_FUELINTIME_PKG_.ADA";
501 500      for Body ("perf_fuelintime_pkg") use "PERF_FUELINTIME_PKG.ADA";
502 501      for Spec ("perf_fpi_pkg") use "PERF_FPI_PKG_.ADA";
503 502      for Body ("perf_fpi_pkg") use "PERF_FPI_PKG.ADA";
504 503      for Body ("perf_fpi_pkg.fpi_exec") use "PERF_FPI_EXEC_SEP.ADA";
505 504      for Body ("perf_fpi_pkg.fpi_estwindtemp") use "PERF_FPI_ESTWINDTEMP_SEP.ADA";
506 505      for Body ("perf_fpi_pkg.fpi_estdistgw") use "PERF_FPI_ESTDISTGW_SEP.ADA";
507 506      for Body ("perf_fix_info_pkg.fix_setterms") use "PERF_FIX_SETTERMS_SEP.ADA";
508 507      for Body ("perf_fix_info_pkg.fix_procterm") use "PERF_FIX_PROCTERMS_SEP.ADA";
509 508      for Spec ("perf_fix_info_pkg") use "PERF_FIX_INFO_PKG_.ADA";
510 509      for Body ("perf_fix_info_pkg") use "PERF_FIX_INFO_PKG.ADA";
511 510      for Spec ("perf_fix_info_lfdata") use "PERF_FIX_INFO_LFDATA_.ADA";
512 511      for Body ("perf_fg_exec_pkg.fg_sim_gwt_step") use "PERF_FG_SIM_GWT_STEP_SEP.ADA";
513 512      for Body ("perf_fg_exec_pkg.fg_powerup_init") use "PERF_FG_POWERUP_INIT_SEP.ADA";
514 513      for Spec ("perf_fg_lfdata") use "PERF_FG_LFDATA_.ADA";
515 514      for Body ("perf_fg_exec_pkg.fg_init") use "PERF_FG_INIT_SEP.ADA";
516 515      for Body ("perf_fg_exec_pkg.fg_get_data") use "PERF_FG_GET_DATA_SEP.ADA";
517 516      for Body ("perf_fg_exec_pkg.fg_flight_complete") use "PERF_FG_FLIGHT_COMPLETE_SEP.ADA";
518 517      for Body ("perf_fg_exec_pkg.fg_exec") use "PERF_FG_EXEC_SEP.ADA";
519 518      for Spec ("perf_fg_exec_pkg") use "PERF_FG_EXEC_PKG_.ADA";
520 519      for Body ("perf_fg_exec_pkg") use "PERF_FG_EXEC_PKG.ADA";
521 520      for Body ("perf_act_spd_env_pkg.fast_act_spd_env") use "PERF_FAST_ACT_SPD_ENV_SEP.ADA";
522 521      for Body ("perf_act_spd_env_pkg.fast_act_spd_env_put_data") use "PERF_FAST_ACT_SPD_ENV_PUT_DATA_SEP.ADA";
523 522      for Body ("perf_act_spd_env_pkg.fast_act_spd_env_get_data") use "PERF_FAST_ACT_SPD_ENV_GET_DATA_SEP.ADA";
524 523      for Spec ("perf_efis_lgb_mgr_pkg") use "PERF_EFIS_LGB_MGR_PKG_.ADA";

```

File: CTP_B787_PERF_CRZINITE.ZIP\stubs.gpr (continued)

```

525 524      for Body ("perf_efis_lgb_mgr_pkg") use "PERF_EFIS_LGB_MGR_PKG.ADA";
526 525      for Body ("perf_dst_estimates_pkg.perf_dst_insufficient_fuel") use "PERF_DST_INSUFFICIENT_FUEL_SEP.ADA";
527 526      for Spec ("perf_dst_estimates_pkg") use "PERF_DST_ESTIMATES_PKG.ADA";
528 527      for Body ("perf_dst_estimates_pkg") use "PERF_DST_ESTIMATES_PKG.ADA";
529 528      for Body ("perf_dpc_pkg.dpc_proctermes") use "PERF_DPC_PROCTERMS_SEP.ADA";
530 529      for Body ("perf_dpc_pkg.dpc_procflapdecel") use "PERF_DPC_PROCFLAPDECEL_SEP.ADA";
531 530      for Body ("perf_dpc_pkg.dpc_proconconstraint") use "PERF_DPC_PROCONCONSTRAINT_SEP.ADA";
532 531      for Spec ("perf_dpc_pkg") use "PERF_DPC_PKG.ADA";
533 532      for Body ("perf_dpc_pkg") use "PERF_DPC_PKG.ADA";
534 533      for Body ("perf_dpc_pkg.dpc_initpath") use "PERF_DPC_INITPATH_SEP.ADA";
535 534      for Body ("perf_dpc_pkg.dpc_getnextleg") use "PERF_DPC_GETNEXTLEG_SEP.ADA";
536 535      for Body ("perf_dpc_pkg.dpc_getlastleg") use "PERF_DPC_GETLASTLEG_SEP.ADA";
537 536      for Body ("perf_dpc_pkg.dpc_genpath") use "PERF_DPC_GENPATH_SEP.ADA";
538 537      for Body ("perf_dpc_pkg.dpc_checkpath") use "PERF_DPC_CHECKPATH_SEP.ADA";
539 538      for Body ("perf_des_pkg.des_setterms") use "PERF_DES_SETTERMS_SEP.ADA";
540 539      for Body ("perf_des_pkg.des_proctermes") use "PERF_DES_PROCTERMS_SEP.ADA";
541 540      for Body ("perf_des_pkg.des_predexec") use "PERF_DES_PREDEXEC_SEP.ADA";
542 541      for Spec ("perf_des_pkg") use "PERF_DES_PKG.ADA";
543 542      for Body ("perf_des_pkg") use "PERF_DES_PKG.ADA";
544 543      for Body ("perf_des_pkg.des_calcprofpts") use "PERF_DES_CALCPROFPTS_SEP.ADA";
545 544      for Body ("perf_crz_pkg.crz_splitksa") use "PERF_CRZ_SPLIT_KSA.ADA";
546 545      for Body ("perf_crz_pkg.crz_setterms") use "PERF_CRZ_SETTERMS_SEP.ADA";
547 546      for Body ("perf_crz_pkg.crz_selectint") use "PERF_CRZ_SELECTINT_SEP.ADA";
548 547      for Body ("perf_crz_pkg.crz_proctermes") use "PERF_CRZ_PROCTERMS_SEP.ADA";
549 548      for Body ("perf_crz_pkg.crz_procstep") use "PERF_CRZ_PROCTSTEP_SEP.ADA";
550 549      for Body ("perf_crz_pkg.crz_predexec") use "PERF_CRZ_PREDEXEC_SEP.ADA";
551 550      for Spec ("perf_crz_pkg") use "PERF_CRZ_PKG.ADA";
552 551      for Body ("perf_crz_pkg") use "PERF_CRZ_PKG.ADA";
553 552      for Body ("perf_crz_pkg.crz_initstepterms") use "PERF_CRZ_INITSTEPTERMS_SEP.ADA";
554 553      for Body ("perf_crz_pkg.crz_icaotrack") use "PERF_CRZ_ICAOTRACK_SEP.ADA";
555 554      for Body ("perf_crz_pkg.crz_geticaoalt") use "PERF_CRZ_GETICAOALT_SEP.ADA";
556 555      for Body ("perf_crz_pkg.crz_domaxaltmsg") use "PERF_CRZ_DOMAXALTMMSG_SEP.ADA";
557 556      for Body ("perf_crz_pkg.crz_comp_step_cost") use "PERF_CRZ_COMP_STEP_COST_SEP.ADA";
558 557      for Body ("perf_crz_pkg.crz_cifromach") use "PERF_CRZ_CI_FRO_MACH.ADA";
559 558      for Body ("perf_crz_pkg.crz_check_above_maxalt") use "PERF_CRZ_CHECK_ABOVE_MAXALT.ADA";
560 559      for Body ("perf_crz_pkg.crz_best_costfl") use "PERF_CRZ_BEST_COSTFL.ADA";
561 560      for Spec ("perf_crzalt_lfdata") use "PERF_CRZALT_LFDATA.ADA";
562 561      for Body ("perf_cmd_spd_pkg.cmd_spd") use "PERF_CMD_SPD_SEP.ADA";
563 562      for Spec ("perf_cmd_spd_pkg") use "PERF_CMD_SPD_PKG.ADA";
564 563      for Body ("perf_cmd_spd_pkg") use "PERF_CMD_SPD_PKG.ADA";
565 564      for Body ("perf_cmd_spd_pkg.cmd_min_drag_spd") use "PERF_CMD_MIN_DRAG_SPD_SEP.ADA";
566 565      for Body ("perf_cmd_spd_pkg.cmd_des_spd") use "PERF_CMD_DES_SPD_SEP.ADA";

```

File: CTP_B787_PERF_CRZINITE.ZIP\stubs.gpr (continued)

```

567 566      for Body ("perf_cmd_spd_pkg.cmd_descent_cas") use "PERF_CMD_DESCENT_CAS_SEP.ADA";
568 567      for Body ("perf_cmd_spd_pkg.cmd_crz_spd") use "PERF_CMD_CRZ_SPD_SEP.ADA";
569 568      for Body ("perf_cmd_spd_pkg.cmd_crz_delta_ci") use "PERF_CMD_CRZ_DELTA_CI_SEP.ADA";
570 569      for Body ("perf_cmd_spd_pkg.cmd_climb_cas") use "PERF_CMD_CLIMB_CAS_SEP.ADA";
571 570      for Body ("perf_cmd_spd_pkg.cmd_clb_spd") use "PERF_CMD_CLB_SPD_SEP.ADA";
572 571      for Body ("perf_clb_pkg.clb_setwptterms") use "PERF_CLB_SETWPTTERMS_SEP.ADA";
573 572      for Body ("perf_clb_pkg.clb_setterms") use "PERF_CLB_SETTERMS_SEP.ADA";
574 573      for Body ("perf_clb_pkg.clb_proctermes") use "PERF_CLB_PROCTERMS_SEP.ADA";
575 574      for Body ("perf_clb_pkg.clb_predexec") use "PERF_CLB_PREDEXEC_SEP.ADA";
576 575      for Spec ("perf_clb_pkg") use "PERF_CLB_PKG.ADA";
577 576      for Body ("perf_clb_pkg") use "PERF_CLB_PKG.ADA";
578 577      for Spec ("perf_bld_pkg") use "PERF_BLD_PKG.ADA";
579 578      for Body ("perf_bld_pkg") use "PERF_BLD_PKG.ADA";
580 579      for Body ("perf_atm_pkg.atm_preddeswind") use "PERF_ATM_PREDDESWIND_SEP.ADA";
581 580      for Body ("perf_atm_pkg.atm_predcrzwind") use "PERF_ATM_PREDCRZWIND_SEP.ADA";
582 581      for Body ("perf_atm_pkg.atm_predclbwind") use "PERF_ATM_PREDCLBWIND_SEP.ADA";
583 582      for Spec ("perf_atm_pkg") use "PERF_ATM_PKG.ADA";
584 583      for Body ("perf_atm_pkg") use "PERF_ATM_PKG.ADA";
585 584      for Body ("perf_atm_pkg.atm_model") use "PERF_ATM_MODEL_SEP.ADA";
586 585      for Body ("perf_atm_pkg.atm_interpwind") use "PERF_ATM_INTERPWIND_SEP.ADA";
587 586      for Body ("perf_atm_pkg.atm_forcwind") use "PERF_ATM_FORCWND_SEP.ADA";
588 587      for Body ("perf_atm_pkg.atm_forctempdev") use "PERF_ATM_FORCTEMPDEV_SEP.ADA";
589 588      for Body ("perf_atm_pkg.atm_forcdeswind") use "PERF_ATM_FORCDESWIND_SEP.ADA";
590 589      for Body ("perf_atm_pkg.atm_calctempdev") use "PERF_ATM_CALCTEMPDEV_SEP.ADA";
591 590      for Body ("perf_atm_pkg.atm_calcpredwind") use "PERF_ATM_CALCPRDWND_SEP.ADA";
592 591      for Body ("perf_atm_pkg.atm_alt_forcwind") use "PERF_ATM_ALT_FORCWND_SEP.ADA";
593 592      for Body ("perf_atm_pkg.atm_altnwind") use "PERF_ATM_ALTNWIND_SEP.ADA";
594 593      for Body ("perf_atm_pkg.atm_altntemp") use "PERF_ATM_ALTNTEMP_SEP.ADA";
595 594      for Body ("perf_altn_pkg.altn_pred_to_overhead") use "PERF_ALTN_PRED_TO_OVERHEAD_SEP.ADA";
596 595      for Body ("perf_altn_pkg.altn_predict") use "PERF_ALTN_PREDICT_SEP.ADA";
597 596      for Spec ("perf_altn_pkg") use "PERF_ALTN_PKG.ADA";
598 597      for Body ("perf_altn_pkg") use "PERF_ALTN_PKG.ADA";
599 598      for Spec ("perf_altn_lfdata") use "PERF_ALTN_LFDATA.ADA";
600 599      for Body ("perf_altn_pkg.altn_init") use "PERF_ALTN_INIT_SEP.ADA";
601 600      for Body ("perf_altn_pkg.altn_est_despath") use "PERF_ALTN_EST_DESPATH_SEP.ADA";
602 601      for Body ("perf_altn_pkg.altn_calc_alt") use "PERF_ALTN_CALC_ALT_SEP.ADA";
603 602      for Spec ("perf_altd_pkg") use "PERF_ALTD_PKG.ADA";
604 603      for Body ("perf_altd_pkg") use "PERF_ALTD_PKG.ADA";
605 604      for Spec ("perf_air_data_pkg") use "PERF_AIR_DATA_PKG.ADA";
606 605      for Body ("perf_air_data_pkg") use "PERF_AIR_DATA_PKG.ADA";
607 606      for Body ("perf_aero_takeoff_pkg.wind_comp") use "PERF_AERO_TKO_WIND_COMP_SEP.ADA";
608 607      for Body ("perf_aero_takeoff_pkg.temp_dialback") use "PERF_AERO_TKO_TEMP_DIALBACK_SEP.ADA";

```

File: CTP_B787_PERF_CRZINITE.ZIP\stubs.gpr (continued)

```

609      608      for Body ("perf_aero_takeoff_pkg.takeoff_speeds") use "PERF_AERO_TKO TAKEOFF_SPEEDS_SEP.ADA";
610      609      for Body ("perf_aero_takeoff_pkg.stabtrimset") use "PERF_AERO_TKO STABTRIMSET_SEP.ADA";
611      610      for Body ("perf_aero_takeoff_pkg.speed_monitor_initialize") use "PERF_AERO_TKO SPD_MONITOR_INIT_SEP.ADA";
612      611      for Body ("perf_aero_takeoff_pkg.select_database") use "PERF_AERO_TKO_SELECT_DATABASE_SEP.ADA";
613      612      for Body ("perf_aero_takeoff_pkg.scheduled_speeds") use "PERF_AERO_TKO_SCHEDULED_SPEEDS_SEP.ADA";
614      613      for Body ("perf_aero_takeoff_pkg.put_vspeeds") use "PERF_AERO_TKO_PUT_VSPEEDS_SEP.ADA";
615      614      for Body ("perf_aero_takeoff_pkg.input_proc") use "PERF_AERO_TKO_INPUT_PROC_SEP.ADA";
616      615      for Body ("perf_aero_takeoff_pkg.gen_engine_data") use "PERF_AERO_TKO_GEN_ENGINE_DATA_SEP.ADA";
617      616      for Body ("perf_aero_takeoff_pkg.eng_data_interp") use "PERF_AERO_TKO_ENG_DATA_INTERP_SEP.ADA";
618      617      for Body ("perf_aero_takeoff_pkg.balanced_field.bal_fld_st_eqn") use "PERF_AERO_TKO_BAL_FLD_ST_EQN_SEP.ADA";
619      618      for Body ("perf_aero_takeoff_pkg.balanced_field.bal_fld_integ") use "PERF_AERO_TKO_BAL_FLD_INTEG_SEP.ADA";
620      619      for Body ("perf_aero_takeoff_pkg.balanced_field.bal_fld_comp_vlmcg") use "PERF_AERO_TKO_BAL_FLD_COMP_VLMCG_SEP.ADA";
621      620      for Body ("perf_aero_takeoff_pkg.balanced_field") use "PERF_AERO_TKO_BALANCED_FIELD_SEP.ADA";
622      621      for Body ("perf_aero_takeoff_pkg.atmos_comp") use "PERF_AERO_TKO_ATMOS_COMP_SEP.ADA";
623      622      for Spec ("perf_aero_takeoff_pkg") use "PERF_AERO_TAKEOFF_PKG_.ADA";
624      623      for Body ("perf_aero_takeoff_pkg") use "PERF_AERO_TAKEOFF_PKG.ADA";
625      624      for Spec ("perf_aero_speed_pkg") use "PERF_AERO_SPEED_PKG_.ADA";
626      625      for Body ("perf_aero_speed_pkg") use "PERF_AERO_SPEED_PKG.ADA";
627      626      for Spec ("perf_aero_reference_pkg") use "PERF_AERO_REFERENCE_PKG_.ADA";
628      627      for Body ("perf_aero_reference_pkg") use "PERF_AERO_REFERENCE_PKG.ADA";
629      628      for Spec ("perf_aero_limit_pkg") use "PERF_AERO_LIMIT_PKG_.ADA";
630      629      for Body ("perf_aero_limit_pkg") use "PERF_AERO_LIMIT_PKG.ADA";
631      630      for Body ("perf_ads_intent_pkg.ads_set_terms") use "PERF_ADS_SET_TERMS_SEP.ADA";
632      631      for Body ("perf_ads_intent_pkg.ads_process_terms") use "PERF_ADS_PROCESS_TERMS_SEP.ADA";
633      632      for Spec ("perf_ads_intent_pkg") use "PERF_ADS_INTENT_PKG_.ADA";
634      633      for Body ("perf_ads_intent_pkg") use "PERF_ADS_INTENT_PKG.ADA";
635      634      for Body ("perf_ads_intent_pkg.ads_get_init_data") use "PERF_ADS_GET_INIT_DATA_SEP.ADA";
636      635      for Body ("perf_ads_intent_pkg.calc_off_route_fixed_intent_points") use "PERF_ADS_CALC_OFF_RTE_FIXED_INTENT_SEP.ADA";
637      636      for Body ("perf_ads_intent_pkg.calc_intermediate_point") use "PERF_ADS_CALC_INTERMEDIATE_POINT_SEP.ADA";
638      637      for Body ("perf_ads_intent_pkg.calc_fixed_point") use "PERF_ADS_CALC_FIXED_POINT_SEP.ADA";
639      638      for Spec ("perf_act_spd_env_pkg") use "PERF_ACT_SPD_ENV_PKG_.ADA";
640      639      for Body ("perf_act_spd_env_pkg") use "PERF_ACT_SPD_ENV_PKG.ADA";
641      640      for Spec ("perf_acstate_mgr") use "PERF_ACSTATE_MGR_.ADA";
642      641      for Body ("perf_acstate_mgr") use "PERF_ACSTATE_MGR.ADA";
643      642      for Spec ("ops_vgb_mgr_pkg") use "OPS_VGB_MGR_PKG_.ADA";
644      643      for Body ("ops_vgb_mgr_pkg") use "OPS_VGB_MGR_PKG.ADA";
645      644      for Spec ("ops_tob_pkg") use "OPS_TOB_PKG_.ADA";
646      645      for Body ("ops_tob_pkg") use "OPS_TOB_PKG.ADA";
647      646      for Spec ("ops_tob_ifdata") use "OPS_TOB_IFDATA_.ADA";
648      647      for Spec ("ops_tob_hs_mgr_pkg") use "OPS_TOB_HS_MGR_PKG_.ADA";
649      648      for Body ("ops_tob_hs_mgr_pkg") use "OPS_TOB_HS_MGR_PKG.ADA";
650      649      for Spec ("ops_standby_one_ltypes") use "OPS_STANDBY_ONE_LTYPES_.ADA";

```

File: CTP_B787_PERF_CRZINITE.ZIP\stubs.gpr (continued)

```

651 650      for Spec ("ops_printer_pkg") use "OPS_PRINTER_PKG_.ADA";
652 651      for Body ("ops_printer_pkg") use "OPS_PRINTER_PKG.ADA";
653 652      for Spec ("ops_perf_fg_pkg") use "OPS_PERF_FG_PKG_.ADA";
654 653      for Body ("ops_perf_fg_pkg") use "OPS_PERF_FG_PKG.ADA";
655 654      for Spec ("ops_perf_change_flags_mgr_pkg") use "OPS_PERF_CHANGE_FLAGS_MGR_PKG_.ADA";
656 655      for Body ("ops_perf_change_flags_mgr_pkg") use "OPS_PERF_CHANGE_FLAGS_MGR_PKG.ADA";
657 656      for Spec ("ops_perf_bg_pkg") use "OPS_PERF_BG_PKG_.ADA";
658 657      for Body ("ops_perf_bg_pkg") use "OPS_PERF_BG_PKG.ADA";
659 658      for Spec ("ops_onehertz_pkg") use "OPS_ONEHERTZ_PKG_.ADA";
660 659      for Body ("ops_onehertz_pkg") use "OPS_ONEHERTZ_PKG.ADA";
661 660      for Spec ("ops_nvm_manager") use "OPS_NVM_MANAGER_.ADA";
662 661      for Body ("ops_nvm_manager") use "OPS_NVM_MANAGER.ADA";
663 662      for Spec ("ops_lateral_guidance_buffer_manager") use "OPS_LATERAL_GUIDANCE_BUFFER_MANAGER_.ADA";
664 663      for Body ("ops_lateral_guidance_buffer_manager") use "OPS_LATERAL_GUIDANCE_BUFFER_MANAGER.ADA";
665 664      for Spec ("ops_icbout_pkg") use "OPS_ICBOUT_PKG_.ADA";
666 665      for Body ("ops_icbout_pkg") use "OPS_ICBOUT_PKG.ADA";
667 666      for Spec ("ops_hs_sync_mon") use "OPS_HS_SYNC_MON_PKG_.ADA";
668 667      for Body ("ops_hs_sync_mon") use "OPS_HS_SYNC_MON_PKG.ADA";
669 668      for Spec ("ops_hs_chkpnt_control_pkg") use "OPS_HS_CHKPNT_CONTROL_PKG_.ADA";
670 669      for Body ("ops_hs_chkpnt_control_pkg") use "OPS_HS_CHKPNT_CONTROL_PKG.ADA";
671 670      for Spec ("ops_hs_buffer_pkg") use "OPS_HS_BUFFER_PKG_.ADA";
672 671      for Body ("ops_hs_buffer_pkg") use "OPS_HS_BUFFER_PKG.ADA";
673 672      for Spec ("ops_hotspare_exec_pkg") use "OPS_HOTSPARE_EXEC_PKG_.ADA";
674 673      for Body ("ops_hotspare_exec_pkg") use "OPS_HOTSPARE_EXEC_PKG.ADA";
675 674      for Spec ("ops_foreground_pkg") use "OPS_FOREGROUND_PKG_.ADA";
676 675      for Body ("ops_foreground_pkg") use "OPS_FOREGROUND_PKG.ADA";
677 676      for Spec ("ops_fm_zombie_pkg") use "OPS_FM_ZOMBIE_PKG_.ADA";
678 677      for Body ("ops_fm_zombie_pkg") use "OPS_FM_ZOMBIE_PKG.ADA";
679 678      for Spec ("ops_fm_utilities_pkg") use "OPS_FM_UTILITIES_PKG_.ADA";
680 679      for Body ("ops_fm_utilities_pkg") use "OPS_FM_UTILITIES_PKG.ADA";
681 680      for Spec ("ops_fm_semaphore_id_pkg") use "OPS_FM_SEMAPHORE_ID_PKG_.ADA";
682 681      for Spec ("ops_process_id_pkg") use "OPS_FM_PROCESS_ID_PKG_.ADA";
683 682      for Spec ("ops_fm_partition_init_pkg") use "OPS_FM_PARTITION_INIT_PKG_.ADA";
684 683      for Body ("ops_fm_partition_init_pkg") use "OPS_FM_PARTITION_INIT_PKG.ADA";
685 684      for Body ("ops_fm_master_determination.determine_transition_allowance") use "OPS_FM_MASTER_DET_TRANS_ALLOW.ADA";
686 685      for Body ("ops_fm_master_determination.transitions_exec") use "OPS_FM_MASTER_DET_TRANSITIONS_EXEC.ADA";
687 686      for Body ("ops_fm_master_determination.power_up_partition") use "OPS_FM_MASTER_DET_POWERUP.ADA";
688 687      for Body ("ops_fm_master_determination.determine_offside_master_status") use "OPS_FM_MASTER_DET_OFFS_MS.ADA";
689 688      for Body ("ops_fm_master_determination.determine_offside_kill_signal") use "OPS_FM_MASTER_DET_OFFS_KILL.ADA";
690 689      for Body ("ops_fm_master_determination.calculate_tm_status") use "OPS_FM_MASTER_DET_CALC_TM_MS.ADA";
691 690      for Body ("ops_fm_master_determination.calculate_offside_health") use "OPS_FM_MASTER_DET_CALC_OFFS_HEALTH.ADA";
692 691      for Body ("ops_fm_master_determination.auto_processing") use "OPS_FM_MASTER_DET_AUTO_PROC.ADA";

```

File: CTP_B787_PERF_CRZINITE.ZIP\stubs.gpr (continued)

```

693 692      for Spec ("ops_fm_master_determination") use "OPS_FM_MASTER_DETERMINATION_PKG_.ADA";
694 693      for Body ("ops_fm_master_determination") use "OPS_FM_MASTER_DETERMINATION_PKG.ADA";
695 694      for Spec ("ops_fm_latch_states_pkg") use "OPS_FM_LATCH_STATES_PKG_.ADA";
696 695      for Body ("ops_fm_latch_states_pkg") use "OPS_FM_LATCH_STATES_PKG.ADA";
697 696      for Spec ("ops_fm_init_status_pkg") use "OPS_FM_INIT_STATUS_PKG_.ADA";
698 697      for Spec ("ops_fm_fault_response_pkg") use "OPS_FM_FAULT_RESPONSE_PKG_.ADA";
699 698      for Body ("ops_fm_fault_response_pkg") use "OPS_FM_FAULT_RESPONSE_PKG.ADA";
700 699      for Spec ("ops_fm_event_id_pkg") use "OPS_FM_EVENT_ID_PKG_.ADA";
701 700      for Spec ("ops_fm_erp_pkg") use "OPS_FM_ERROR_RECOVERY_PROCESS_PKG_.ADA";
702 701      for Body ("ops_fm_erp_pkg") use "OPS_FM_ERROR_RECOVERY_PROCESS_PKG.ADA";
703 702      for Spec ("ops_fm_database_verification_pkg") use "OPS_FM_DATABASE_VERIFICATION_PKG_.ADA";
704 703      for Body ("ops_fm_database_verification_pkg") use "OPS_FM_DATABASE_VERIFICATION_PKG.ADA";
705 704      for Spec ("ops_fm_database_utils_pkg") use "OPS_FM_DATABASE_UTILS_PKG_.ADA";
706 705      for Spec ("ops_fm_bite_pkg") use "OPS_FM_BITE_PKG_.ADA";
707 706      for Body ("ops_fm_bite_pkg") use "OPS_FM_BITE_PKG.ADA";
708 707      for Spec ("ops_efis_path_pkg") use "OPS_EFIS_PATH_PKG_.ADA";
709 708      for Body ("ops_efis_path_pkg") use "OPS_EFIS_PATH_PKG.ADA";
710 709      for Spec ("ops_efis_fg_pkg") use "OPS_EFIS_FG_PKG_.ADA";
711 710      for Body ("ops_efis_fg_pkg") use "OPS_EFIS_FG_PKG.ADA";
712 711      for Spec ("ops_efis_bg_pkg") use "OPS_EFIS_BG_PKG_.ADA";
713 712      for Body ("ops_efis_bg_pkg") use "OPS_EFIS_BG_PKG.ADA";
714 713      for Spec ("ops_dual_simsoft_pkg") use "OPS_DUAL_SIMSOFT_PKG_.ADA";
715 714      for Body ("ops_dual_simsoft_pkg") use "OPS_DUAL_SIMSOFT_PKG.ADA";
716 715      for Spec ("ops_cfp_nav_mgr") use "OPS_CFP_NAV_MGR_.ADA";
717 716      for Body ("ops_cfp_nav_mgr") use "OPS_CFP_NAV_MGR.ADA";
718 717      for Spec ("ops_cfp_internal_mgr_pkg") use "OPS_CFP_INTERNAL_MGR_PKG_.ADA";
719 718      for Body ("ops_cfp_internal_mgr_pkg") use "OPS_CFP_INTERNAL_MGR_PKG.ADA";
720 719      for Spec ("ops_cex_cdukey_pkg") use "OPS_CEX_CDUKEY_PKG_.ADA";
721 720      for Body ("ops_cex_cdukey_pkg") use "OPS_CEX_CDUKEY_PKG.ADA";
722 721      for Spec ("ops_cdu_guid_mgr") use "OPS_CDU_GUID_MGR_.ADA";
723 722      for Body ("ops_cdu_guid_mgr") use "OPS_CDU_GUID_MGR.ADA";
724 723      for Spec ("ops_cdl_interface_mgr_pkg") use "OPS_CDL_INTERFACE_MGR_PKG_.ADA";
725 724      for Body ("ops_cdl_interface_mgr_pkg") use "OPS_CDL_INTERFACE_MGR_PKG.ADA";
726 725      for Spec ("ops_cdl_buffer_mgr_pkg") use "OPS_CDL_BUFFER_MGR_PKG_.ADA";
727 726      for Body ("ops_cdl_buffer_mgr_pkg") use "OPS_CDL_BUFFER_MGR_PKG.ADA";
728 727      for Spec ("ops_cdk_perf_pdb_mgr_pkg") use "OPS_CDK_PERF_PDB_MGR_PKG_.ADA";
729 728      for Body ("ops_cdk_perf_pdb_mgr_pkg") use "OPS_CDK_PERF_PDB_MGR_PKG.ADA";
730 729      for Spec ("ops_cdk_page_data_mgr_pkg") use "OPS_CDK_PAGE_DATA_MGR_PKG_.ADA";
731 730      for Body ("ops_cdk_page_data_mgr_pkg") use "OPS_CDK_PAGE_DATA_MGR_PKG.ADA";
732 731      for Spec ("ops_cdk_io_mgr_pkg") use "OPS_CDK_IO_MGR_PKG_.ADA";
733 732      for Body ("ops_cdk_io_mgr_pkg") use "OPS_CDK_IO_MGR_PKG.ADA";
734 733      for Spec ("ops_cdk_internal_cdk_mgr_pkg") use "OPS_CDK_INTERNAL_CDK_MGR_PKG_.ADA";

```

File: CTP_B787_PERF_CRZINITE.ZIP\stubs.gpr (continued)

```

735 734      for Body ("ops_cdk_internal_cdk_mgr_pkg") use "OPS_CDK_INTERNAL_CDK_MGR_PKG.ADA";
736 735      for Spec ("ops_cdk_fuel_weight_mgr_pkg") use "OPS_CDK_FUEL_WEIGHT_MGR_PKG.ADA";
737 736      for Body ("ops_cdk_fuel_weight_mgr_pkg") use "OPS_CDK_FUEL_WEIGHT_MGR_PKG.ADA";
738 737      for Spec ("ops_cdk_fm_nav_mgr_pkg") use "OPS_CDK_FM_NAV_MGR_PKG.ADA";
739 738      for Body ("ops_cdk_fm_nav_mgr_pkg") use "OPS_CDK_FM_NAV_MGR_PKG.ADA";
740 739      for Spec ("ops_cdk_flt_plan_mgr_pkg") use "OPS_CDK_FLT_PLAN_MGR_PKG.ADA";
741 740      for Body ("ops_cdk_flt_plan_mgr_pkg") use "OPS_CDK_FLT_PLAN_MGR_PKG.ADA";
742 741      for Spec ("ops_cdk_duplicate_fix_mgr_pkg") use "OPS_CDK_DUPLICATE_FIX_MGR_PKG.ADA";
743 742      for Body ("ops_cdk_duplicate_fix_mgr_pkg") use "OPS_CDK_DUPLICATE_FIX_MGR_PKG.ADA";
744 743      for Spec ("ops_cdk_common_mgr_pkg") use "OPS_CDK_COMMON_MGR_PKG.ADA";
745 744      for Body ("ops_cdk_common_mgr_pkg") use "OPS_CDK_COMMON_MGR_PKG.ADA";
746 745      for Body ("ops_cdk_altn_data_mgr_pkg.put_preds") use "OPS_CDK_ALTN_DATA_MGR_PKG__PUT_PREDS.ADA";
747 746      for Body ("ops_cdk_altn_data_mgr_pkg.gen_list_ul") use "OPS_CDK_ALTN_DATA_MGR_PKG__GEN_LIST_UL.ADA";
748 747      for Body ("ops_cdk_altn_data_mgr_pkg.enter_arpt") use "OPS_CDK_ALTN_DATA_MGR_PKG__ENTER_ARPT.ADA";
749 748      for Body ("ops_cdk_altn_data_mgr_pkg.sort_by_time") use "OPS_CDK_ALTN_DATA_MGR_PKG_SORT_BY_TIME.ADA";
750 749      for Body ("ops_cdk_altn_data_mgr_pkg.sort_by_dist") use "OPS_CDK_ALTN_DATA_MGR_PKG_SORT_BY_DIST.ADA";
751 750      for Body ("ops_cdk_altn_data_mgr_pkg.gen_list_ndb") use "OPS_CDK_ALTN_DATA_MGR_PKG_GEN_LIST_NDB.ADA";
752 751      for Spec ("ops_cdk_altn_data_mgr_pkg") use "OPS_CDK_ALTN_DATA_MGR_PKG.ADA";
753 752      for Body ("ops_cdk_altn_data_mgr_pkg") use "OPS_CDK_ALTN_DATA_MGR_PKG.ADA";
754 753      for Spec ("offset_segment_set_pkg") use "OFFSET_SEGMENT_SET_PKG.ADA";
755 754      for Spec ("lt_initialize_wrapper_pkg") use "LT_INITIALIZE_WRAPPER_PKG.ADA";
756 755      for Body ("lt_initialize_wrapper_pkg") use "LT_INITIALIZE_WRAPPER_PKG.ADA";
757 756      for Body ("lg_data_managers_pkg.lg_write_slow_to_fast_data") use "LG_WRITE_SLOW_TO_FAST_DATA_SEP.ADA";
758 757      for Body ("lg_data_managers_pkg.lg_write_slow_interface_data") use "LG_WRITE_SLOW_INTERFACE_DATA_SEP.ADA";
759 758      for Body ("lg_data_managers_pkg.lg_write_fast_to_slow_data") use "LG_WRITE_FAST_TO_SLOW_DATA_SEP.ADA";
760 759      for Body ("lg_data_managers_pkg.lg_write_fast_interface_data") use "LG_WRITE_FAST_INTERFACE_DATA_SEP.ADA";
761 760      for Body ("lg_path_def_pkg.lg_updatelegs") use "LG_UPDATELEGS_SEP.ADA";
762 761      for Spec ("lg_unable_hold_airspace_message") use "LG_UNABLE_HOLD_AIRSPACE_MESSAGE.ADA";
763 762      for Body ("lg_unable_hold_airspace_message") use "LG_UNABLE_HOLD_AIRSPACE_MESSAGE.ADA";
764 763      for Body ("lg_mode_control_pkg.lg_testlnaveng") use "LG_TESTLNAVENG_SEP.ADA";
765 764      for Body ("lg_path_def_pkg.lg_tactical_segments") use "LG_TACTICAL_SEGMENTS_SEP.ADA";
766 765      for Body ("lg_slow_progress_data_pkg.lg_sumlegdists") use "LG_SUMLEGDISTSEP.ADA";
767 766      for Body ("lg_seq_legs_pkg.lg_storewptdata") use "LG_STOREWPTDATA_SEP.ADA";
768 767      for Spec ("lg_slow_to_fast_lfdata") use "LG_SLOW_TO_FAST_LFDATA.ADA";
769 768      for Spec ("lg_slow_task_lfdata") use "LG_SLOW_TASK_LFDATA.ADA";
770 769      for Spec ("lg_slow_progress_data_pkg") use "LG_SLOW_PROGRESS_DATA_PKG.ADA";
771 770      for Body ("lg_slow_progress_data_pkg") use "LG_SLOW_PROGRESS_DATA_PKG.ADA";
772 771      for Spec ("lg_slow_in_lfdata") use "LG_SLOW_IN_LFDATA.ADA";
773 772      for Body ("lg_mode_control_pkg.lg_set_ref_point") use "LG_SET_REF_POINT_SEP.ADA";
774 773      for Body ("lg_slow_progress_data_pkg.lg_setnrplrp") use "LG_SETNRPLRP_SEP.ADA";
775 774      for Spec ("lg_seq_legs_pkg") use "LG_SEQ_LEGS_PKG.ADA";
776 775      for Body ("lg_seq_legs_pkg") use "LG_SEQ_LEGS_PKG.ADA";

```


File: CTP_B787_PERF_CRZINITE.ZIP\stubs.gpr (continued)

```

777 776      for Body ("lg_seq_legs_pkg.lg_seq_lat_leg") use "LG_SEQ_LAT_LEG_SEP.ADA";
778 777      for Body ("lg_seq_legs_pkg.lg_sequencing_paramaters") use "LG_SEQUENCING_PARAMATERS_SEP.ADA";
779 778      for Body ("lg_seq_legs_pkg.lg_sequencelegs") use "LG_SEQUENCELEGS_SEP.ADA";
780 779      for Body ("lg_slow_progress_data_pkg.lg_sel_vgdist_comp") use "LG_SEL_VGDIST_COMP_SEP.ADA";
781 780      for Body ("lg_slow_progress_data_pkg.lg_sel_control_mode") use "LG_SEL_CONTROL_MODE_SEP.ADA";
782 781      for Body ("lg_path_def_pkg.lg_select_hx_speed") use "LG_SELECT_HX_SPEED_SEP.ADA";
783 782      for Body ("lg_data_managers_pkg.lg_read_slow_to_fast_data") use "LG_READ_SLOW_TO_FAST_DATA_SEP.ADA";
784 783      for Body ("lg_data_managers_pkg.lg_read_slow_interface_data") use "LG_READ_SLOW_INTERFACE_DATA_SEP.ADA";
785 784      for Body ("lg_data_managers_pkg.lg_read_fast_to_slow_data") use "LG_READ_FAST_TO_SLOW_DATA_SEP.ADA";
786 785      for Body ("lg_data_managers_pkg.lg_read_fast_interface_data") use "LG_READ_FAST_INTERFACE_DATA_SEP.ADA";
787 786      for Body ("lg_path_def_pkg.lg_populate_active_leg_record") use "LG_POPULATE_ACTIVE_LEG_RECORD_SEP.ADA";
788 787      for Body ("lg_fast_progress_data_pkg.lg_path_leg_prog") use "LG_PATH_LEG_PROG_SEP.ADA";
789 788      for Spec ("lg_path_def_pkg") use "LG_PATH_DEF_PKG.ADA";
790 789      for Body ("lg_path_def_pkg") use "LG_PATH_DEF_PKG.ADA";
791 790      for Spec ("lg_mode_control_pkg") use "LG_MODE_CONTROL_PKG.ADA";
792 791      for Body ("lg_mode_control_pkg") use "LG_MODE_CONTROL_PKG.ADA";
793 792      for Spec ("lg_lt_wrapper_pkg") use "LG_LT_WRAPPER_PKG.ADA";
794 793      for Body ("lg_lt_wrapper_pkg") use "LG_LT_WRAPPER_PKG.ADA";
795 794      for Spec ("lg_lt_segment_list_wrapper_pkg") use "LG_LT_SEGMENT_LIST_WRAPPER_PKG.ADA";
796 795      for Body ("lg_lt_segment_list_wrapper_pkg") use "LG_LT_SEGMENT_LIST_WRAPPER_PKG.ADA";
797 796      for Body ("lg_path_def_pkg.populate_lt_aircraft_state") use "LG_LT_POPULATE_AC_STATE_SEP.ADA";
798 797      for Spec ("lg_lt_interface_tpkg") use "LG_LT_INTERFACE_TPKG.ADA";
799 798      for Spec ("lg_lt_ifdata") use "LG_LT_IFDATA.ADA";
800 799      for Body ("lg_seq_legs_pkg.lg_line_segment_intercept") use "LG_LINE_SEGMENT_INTERCEPT_SEP.ADA";
801 800      for Spec ("lg_lftypes") use "LG_LFTYPES.ADA";
802 801      for Body ("lg_mode_control_pkg.lg_lat_path_cap") use "LG_LAT_PATH_CAP_SEP.ADA";
803 802      for Body ("lg_fast_progress_data_pkg.lg_latpatherror") use "LG_LATPATHERROR_SEP.ADA";
804 803      for Body ("lg_executives_pkg.lg_init") use "LG_INIT_SEP.ADA";
805 804      for Body ("lg_seq_legs_pkg.lg_initiate_trans") use "LG_INITIATE_TRANS_SEP.ADA";
806 805      for Body ("lg_path_def_pkg.lg_hx_size_limit") use "LG_HX_SIZE_LIMIT_SEP.ADA";
807 806      for Body ("lg_fast_progress_data_pkg.lg_hx_pi_leg_prog") use "LG_HX_PI_LEG_PROG_SEP.ADA";
808 807      for Body ("lg_fast_progress_data_pkg.lg_heading_leg_prog") use "LG_HEADING_LEG_PROG_SEP.ADA";
809 808      for Body ("lg_executives_pkg.lg_guidcontrol") use "LG_GUIDCONTROL_SEP.ADA";
810 809      for Spec ("lg_fast_to_slow_lfdata") use "LG_FAST_TO_SLOW_LFDATA.ADA";
811 810      for Spec ("lg_fast_task_lfdata") use "LG_FAST_TASK_LFDATA.ADA";
812 811      for Spec ("lg_fast_progress_data_pkg") use "LG_FAST_PROGRESS_DATA_PKG.ADA";
813 812      for Body ("lg_fast_progress_data_pkg") use "LG_FAST_PROGRESS_DATA_PKG.ADA";
814 813      for Spec ("lg_fast_in_lfdata") use "LG_FAST_IN_LFDATA.ADA";
815 814      for Body ("lg_executives_pkg.lg_fastguid") use "LG_FASTGUID_SEP.ADA";
816 815      for Spec ("lg_executives_pkg") use "LG_EXECUTIVES_PKG.ADA";
817 816      for Body ("lg_executives_pkg") use "LG_EXECUTIVES_PKG.ADA";
818 817      for Spec ("lg_efis_data_mgr_pkg") use "LG_EFIS_DATA_MGR_PKG.ADA";

```

File: CTP_B787_PERF_CRZINITE.ZIP\stubs.gpr (continued)

```

819 818      for Body ("lg_efis_data_mgr_pkg") use "LG_EFIS_DATA_MGR_PKG.ADA";
820 819      for Body ("lg_seq_legs_pkg.lg_det_hx_seg_ontrack_to") use "LG_DET_HX_SEG_ONTRACK_TO_SEP.ADA";
821 820      for Body ("lg_seq_legs_pkg.lg_determine_hxleg_progress") use "LG_DET_HX_PI_PROGRESS_SEP.ADA";
822 821      for Body ("lg_seq_legs_pkg.lg_det_hx_active_leg_segment") use "LG_DET_HX_ACTIVE_LEG_SEGMENT_SEP.ADA";
823 822      for Body ("lg_seq_legs_pkg.lg_determine_pathleg_progress") use "LG_DETERMINE_PATHLEG_PROGRESS_SEP.ADA";
824 823      for Body ("lg_seq_legs_pkg.lg_determine_ifleg_progress") use "LG_DETERMINE_IFLEG_PROGRESS_SEP.ADA";
825 824      for Spec ("lg_data_managers_pkg") use "LG_DATA_MANAGERS_PKG_.ADA";
826 825      for Body ("lg_data_managers_pkg") use "LG_DATA_MANAGERS_PKG.ADA";
827 826      for Body ("lg_slow_progress_data_pkg.lg_cptrans_data_comp") use "LG_CPTRANS_DATA_COMP_SEP.ADA";
828 827      for Body ("lg_slow_progress_data_pkg.lg_cpexit_comp") use "LG_CPEXIT_COMP_SEP.ADA";
829 828      for Body ("lg_slow_progress_data_pkg.lg_cpentry_comp") use "LG_CPENTRY_COMP_SEP.ADA";
830 829      for Spec ("lg_control_laws_pkg") use "LG_CONTROL_LAWS_PKG_.ADA";
831 830      for Body ("lg_control_laws_pkg") use "LG_CONTROL_LAWS_PKG.ADA";
832 831      for Body ("lg_path_def_pkg.lg_compute_ian_path") use "LG_COMPUTE_IAN_PATH_SEP.ADA";
833 832      for Body ("lg_path_def_pkg.lg_compute_ian_lrp") use "LG_COMPUTE_IAN_LRP_SEP.ADA";
834 833      for Body ("lg_path_def_pkg.lg_compute_ian_discretes") use "LG_COMPUTE_IAN_DISCRETES_SEP.ADA";
835 834      for Body ("lg_fast_progress_data_pkg.lg_compute_ian_deviations") use "LG_COMPUTE_IAN_DEVIATIONS_SEP.ADA";
836 835      for Body ("lg_executives_pkg.lg_compute_distances") use "LG_COMPUTE_DISTANCES_SEP.ADA";
837 836      for Body ("lg_seq_legs_pkg.lg_cntrllatran") use "LG_CNTRLLATRAN_SEP.ADA";
838 837      for Body ("lg_slow_progress_data_pkg.lg_cmpt_psuedo_df_pts") use "LG_CMPT_PSUEDO_DF_PTS_SEP.ADA";
839 838      for Body ("lg_path_def_pkg.lg_cmpt_hx_entry_type") use "LG_CMPT_HX_ENTRY_TYPE_SEP.ADA";
840 839      for Body ("lg_path_def_pkg.lg_cmpt_df_crs_change") use "LG_CMPT_DF_CRS_CHANGE_SEP.ADA";
841 840      for Body ("lg_slow_progress_data_pkg.lg_cmptvnavdist") use "LG_CMPTVNAVDIST_SEP.ADA";
842 841      for Body ("lg_slow_progress_data_pkg.lg_cmptrollim") use "LG_CMPTROLLIM_SEP.ADA";
843 842      for Body ("lg_path_def_pkg.lg_cmpttransit") use "LG_CMPTTRANSIT_SEP.ADA";
844 843      for Body ("lg_fast_progress_data_pkg.lg_cmptdistocrp") use "LG_CMPTDISTOCR_PSEP.ADA";
845 844      for Body ("lg_mode_control_pkg.lg_capture_limit") use "LG_CAPTURE_LIMIT_SEP.ADA";
846 845      for Body ("lg_seq_legs_pkg.lg_caplimgdist") use "LG_CAPLIMDIST_SEP.ADA";
847 846      for Spec ("lg_bank_angle_limited_message") use "LG_BANK_ANGLE_LIMITED_MESSAGE_.ADA";
848 847      for Body ("lg_bank_angle_limited_message") use "LG_BANK_ANGLE_LIMITED_MESSAGE.ADA";
849 848      for Body ("lg_seq_legs_pkg.lg_arc_segment_intercept") use "LG_ARC_SEGMENT_INTERCEPT_SEP.ADA";
850 849      for Body ("lg_fast_progress_data_pkg.lg_arcdistance") use "LG_ARCDISTANCE_SEP.ADA";
851 850      for Body ("lg_path_def_pkg.lg_activatefpln") use "LG_ACTIVATEFPLN_SEP.ADA";
852 851      for Spec ("lgb_error_code_dpkg") use "LGB_ERROR_CODE_DPKG_.ADA";
853 852      for Body ("lgb_directories_pkg.bld_rte_directory") use "LGB_DIRECTORIES_PKG__BLD_RTE_DIRECTORY.ADA";
854 853      for Body ("lgb_directories_pkg.bld_gb_directory") use "LGB_DIRECTORIES_PKG__BLD_GB_DIRECTORY.ADA";
855 854      for Spec ("lgb_directories_pkg") use "LGB_DIRECTORIES_PKG_.ADA";
856 855      for Body ("lgb_directories_pkg") use "LGB_DIRECTORIES_PKG.ADA";
857 856      for Spec ("lateral_offset_segment_type_tpkg") use "LATERAL_OFFSET_SEGMENT_TYPE_TPKG_.ADA";
858 857      for Spec ("fuel_weight_ltypes") use "FUEL_WEIGHT_LTYPES_.ADA";
859 858      for Spec ("fprequestrec_types") use "FPREQUESTREC_TYPES_.ADA";
860 859      for Spec ("fpp_wrap_segment_pkg") use "FPP_WRAP_SEGMENT_PKG_.ADA";

```

File: CTP_B787_PERF_CRZINITE.ZIP\stubs.gpr (continued)

```

861 860      for Body ("fpp_wrap_segment_pkg") use "FPP_WRAP_SEGMENT_PKG.ADA";
862 861      for Spec ("fpp_wrap_request_process_pkg") use "FPP_WRAP_REQUEST_PROCESS_PKG_.ADA";
863 862      for Body ("fpp_wrap_request_process_pkg") use "FPP_WRAP_REQUEST_PROCESS_PKG.ADA";
864 863      for Spec ("fpp_wrap_point_pkg") use "FPP_WRAP_POINT_PKG_.ADA";
865 864      for Body ("fpp_wrap_point_pkg") use "FPP_WRAP_POINT_PKG.ADA";
866 865      for Spec ("fpp_wrap_controller_pkg") use "FPP_WRAP_CONTROLLER_PKG_.ADA";
867 866      for Body ("fpp_wrap_controller_pkg") use "FPP_WRAP_CONTROLLER_PKG.ADA";
868 867      for Spec ("fpp_wrap_config_data_pkg") use "FPP_WRAP_CONFIG_DATA_PKG_.ADA";
869 868      for Body ("fpp_wrap_config_data_pkg") use "FPP_WRAP_CONFIG_DATA_PKG.ADA";
870 869      for Spec ("fpp_status_type_tpkg") use "FPP_STATUS_TYPE_TPKG_.ADA";
871 870      for Spec ("fpp_proj_utils_pkg") use "FPP_PROJ_UTILS_PKG_.ADA";
872 871      for Body ("fpp_proj_utils_pkg") use "FPP_PROJ_UTILS_PKG.ADA";
873 872      for Spec ("fpp_interface_type") use "FPP_INTERFACE_TYPE_.ADA";
874 873      for Spec ("fpp_handle_wrap_pkg") use "FPP_HANDLE_WRAP_PKG_.ADA";
875 874      for Body ("fpp_handle_wrap_pkg") use "FPP_HANDLE_WRAP_PKG.ADA";
876 875      for Spec ("fpp_fpa_wrap_pkg") use "FPP_FPA_WRAP_PKG_.ADA";
877 876      for Body ("fpp_fpa_wrap_pkg") use "FPP_FPA_WRAP_PKG.ADA";
878 877      for Spec ("fpp_common_lgb_wrap_pkg") use "FPP_COMMON_LGB_WRAP_PKG_.ADA";
879 878      for Body ("fpp_common_lgb_wrap_pkg") use "FPP_COMMON_LGB_WRAP_PKG.ADA";
880 879      for Spec ("fpp_away_utils_wrap_pkg") use "FPP_AWAY_UTILS_WRAP_PKG_.ADA";
881 880      for Body ("fpp_away_utils_wrap_pkg") use "FPP_AWAY_UTILS_WRAP_PKG.ADA";
882 881      for Spec ("ops_io_hotspare_ifpkg") use "FM_OPS_IO_HOTSPARE_IFPKG_.ADA";
883 882      for Body ("ops_io_hotspare_ifpkg") use "FM_OPS_IO_HOTSPARE_IFPKG.ADA";
884 883      for Spec ("fm_nav_toga_update") use "FM_NAV_TOGA_UPDATE_.ADA";
885 884      for Body ("fm_nav_toga_update") use "FM_NAV_TOGA_UPDATE.ADA";
886 885      for Body ("fm_nav_support.fm_onenav") use "FM_NAV_SUPPORT_FM_ONENAV.ADA";
887 886      for Body ("fm_nav_support.fm_fastnav") use "FM_NAV_SUPPORT_FM_FASTNAV.ADA";
888 887      for Spec ("fm_nav_support") use "FM_NAV_SUPPORT_.ADA";
889 888      for Body ("fm_nav_support") use "FM_NAV_SUPPORT.ADA";
890 889      for Body ("fm_nav_simsoft_interface.onehz_update") use "FM_NAV_SIMSOFT_INTERFACE_ONEHZ_UPDATE.ADA";
891 890      for Body ("fm_nav_simsoft_interface.fast_update") use "FM_NAV_SIMSOFT_INTERFACE_FAST_UPDATE.ADA";
892 891      for Spec ("fm_nav_simsoft_interface") use "FM_NAV_SIMSOFT_INTERFACE_.ADA";
893 892      for Body ("fm_nav_simsoft_interface") use "FM_NAV_SIMSOFT_INTERFACE.ADA";
894 893      for Spec ("fm_nav_rnp_interface") use "FM_NAV_RNP_INTERFACE_.ADA";
895 894      for Body ("fm_nav_rnp_interface") use "FM_NAV_RNP_INTERFACE.ADA";
896 895      for Body ("fm_nav_rnp.compute_rnp") use "FM_NAV_RNP_COMPUTE_RNP.ADA";
897 896      for Spec ("fm_nav_rnp") use "FM_NAV_RNP_.ADA";
898 897      for Body ("fm_nav_rnp") use "FM_NAV_RNP.ADA";
899 898      for Spec ("radio_sample") use "FM_NAV_RADIO_SAMPLE_.ADA";
900 899      for Body ("radio_sample") use "FM_NAV_RADIO_SAMPLE.ADA";
901 900      for Spec ("radio_navaid") use "FM_NAV_RADIO_NAVAID_.ADA";
902 901      for Body ("radio_navaid") use "FM_NAV_RADIO_NAVAID.ADA";

```

File: CTP_B787_PERF_CRZINITE.ZIP\stubs.gpr (continued)

```

903 902      for Body ("radio_management.navaid_in_line_of_sight") use "FM_NAV_RADIO_MANAGEMENT_NAVID_IN_LOS.ADA";
904 903      for Body ("radio_management.in_vor_zone_of_confusion") use "FM_NAV_RADIO_MANAGEMENT_IN_VOR_ZOC.ADA";
905 904      for Body ("radio_management.approximate_distance") use "FM_NAV_RADIO_MANAGEMENT_APPROX_DIST.ADA";
906 905      for Spec ("radio_management") use "FM_NAV_RADIO_MANAGEMENT_.ADA";
907 906      for Body ("radio_management") use "FM_NAV_RADIO_MANAGEMENT.ADA";
908 907      for Body ("fm_nav_pos_update.select_update_type") use "FM_NAV_POS_UPDATE_SELECT_UPDATE_TYPE.ADA";
909 908      for Spec ("fm_nav_pos_update") use "FM_NAV_POS_UPDATE_.ADA";
910 909      for Body ("fm_nav_pos_update") use "FM_NAV_POS_UPDATE.ADA";
911 910      for Body ("fm_nav_partition_interface.onehz_update") use "FM_NAV_PARTITION_INTERFACE_ONEHZ_UPD.ADA";
912 911      for Body ("fm_nav_partition_interface.fast_update") use "FM_NAV_PARTITION_INTERFACE_FAST_UPDATE.ADA";
913 912      for Spec ("fm_nav_partition_interface") use "FM_NAV_PARTITION_INTERFACE_.ADA";
914 913      for Body ("fm_nav_partition_interface") use "FM_NAV_PARTITION_INTERFACE.ADA";
915 914      for Body ("fm_nav_ops_interface.onehz_update") use "FM_NAV_OPS_INTERFACE_ONEHZ_UPDATE.ADA";
916 915      for Spec ("fm_nav_ops_interface") use "FM_NAV_OPS_INTERFACE_.ADA";
917 916      for Body ("fm_nav_ops_interface") use "FM_NAV_OPS_INTERFACE.ADA";
918 917      for Body ("fm_onenav_io_interface.put_fm_onenav_data") use "FM_NAV_ONENAV_IO_INTFACE_PUT_1NAV_DATA.ADA";
919 918      for Body ("fm_onenav_io_interface.get_fm_onenav_external_data") use "FM_NAV_ONENAV_IO_INTFACE_EXTERN_DATA.ADA";
920 919      for Spec ("fm_onenav_io_interface") use "FM_NAV_ONENAV_IO_INTFACE_.ADA";
921 920      for Body ("fm_onenav_io_interface") use "FM_NAV_ONENAV_IO_INTFACE.ADA";
922 921      for Body ("fm_nav_messages.vrnp_message") use "FM_NAV_MESSAGES_VRNP_MESSAGE.ADA";
923 922      for Body ("fm_nav_messages.vp_message") use "FM_NAV_MESSAGES_VP_MESSAGE.ADA";
924 923      for Body ("fm_nav_messages.vert_rnp_message") use "FM_NAV_MESSAGES_VERT_RNP_MESSAGES.ADA";
925 924      for Body ("fm_nav_messages.rw_ils_freq_message") use "FM_NAV_MESSAGES_RW_ILS_FREQ_MESSAGE.ADA";
926 925      for Body ("fm_nav_messages.rw_ils_crs_message") use "FM_NAV_MESSAGES_RW_ILS_CRS_MESSAGE.ADA";
927 926      for Body ("fm_nav_messages.runway_monitor") use "FM_NAV_MESSAGES_RUNWAY_MONITOR.ADA";
928 927      for Body ("fm_nav_messages.pre_nav_messages") use "FM_NAV_MESSAGES_PRE_NAV_MESSAGES.ADA";
929 928      for Body ("fm_nav_messages.post_nav_messages") use "FM_NAV_MESSAGES_POST_NAV_MESSAGES.ADA";
930 929      for Body ("fm_nav_messages.nit_message") use "FM_NAV_MESSAGES_NIT_MESSAGE.ADA";
931 930      for Body ("fm_nav_messages.messages") use "FM_NAV_MESSAGES_MESSAGES.ADA";
932 931      for Body ("fm_nav_messages.itl_message") use "FM_NAV_MESSAGES_ITI_MESSAGE.ADA";
933 932      for Body ("fm_nav_messages.initialize") use "FM_NAV_MESSAGES_INITIALIZE.ADA";
934 933      for Spec ("fm_nav_messages") use "FM_NAV_MESSAGES_.ADA";
935 934      for Body ("fm_nav_messages") use "FM_NAV_MESSAGES.ADA";
936 935      for Body ("fm_nav_inertial_interface.onehz_update") use "FM_NAV_INERTIAL_INTERFACE_ONEHZ_UPDATE.ADA";
937 936      for Body ("fm_nav_inertial_interface.fast_update") use "FM_NAV_INERTIAL_INTERFACE_FAST_UPDATE.ADA";
938 937      for Spec ("fm_nav_inertial_interface") use "FM_NAV_INERTIAL_INTERFACE_.ADA";
939 938      for Body ("fm_nav_inertial_interface") use "FM_NAV_INERTIAL_INTERFACE.ADA";
940 939      for Spec ("ops_fm_nav_hs_mgr_pkg") use "FM_NAV_HS_MGR_PKG_.ADA";
941 940      for Body ("ops_fm_nav_hs_mgr_pkg") use "FM_NAV_HS_MGR_PKG.ADA";
942 941      for Spec ("guidance_interface") use "FM_NAV_GUIDANCE_INTERFACE_.ADA";
943 942      for Body ("guidance_interface") use "FM_NAV_GUIDANCE_INTERFACE.ADA";
944 943      for Body ("fm_nav_flight_phase.compute_data") use "FM_NAV_FLIGHT_PHASE_COMPUTE_DATA.ADA";

```

File: CTP_B787_PERF_CRZINITE.ZIP\stubs.gpr (continued)

```

945 944      for Spec ("fm_nav_flight_phase") use "FM_NAV_FLIGHT_PHASE_.ADA";
946 945      for Body ("fm_nav_flight_phase") use "FM_NAV_FLIGHT_PHASE.ADA";
947 946      for Body ("fm_fastnav_io_interface.get_external_fm_fastnav_data") use "FM_NAV_FASTNAV_IO_INTFACE_EXTERN_DATA.ADA";
948 947      for Spec ("fm_fastnav_io_interface") use "FM_NAV_FASTNAV_IO_INTFACE_.ADA";
949 948      for Body ("fm_fastnav_io_interface") use "FM_NAV_FASTNAV_IO_INTFACE.ADA";
950 949      for Body ("fm_nav_cockpit_interface.onehz_update") use "FM_NAV_COCKPIT_INTERFACE_ONEHZ_UPDATE.ADA";
951 950      for Body ("fm_nav_cockpit_interface.fast_update") use "FM_NAV_COCKPIT_INTERFACE_FAST_UPDATE.ADA";
952 951      for Spec ("fm_nav_cockpit_interface") use "FM_NAV_COCKPIT_INTERFACE_.ADA";
953 952      for Body ("fm_nav_cockpit_interface") use "FM_NAV_COCKPIT_INTERFACE.ADA";
954 953      for Spec ("fm_nav_checkpoint") use "FM_NAV_CHECKPOINT_.ADA";
955 954      for Body ("fm_nav_checkpoint") use "FM_NAV_CHECKPOINT.ADA";
956 955      for Body ("fm_nav_cfp_interface.reciprocal_runway") use "FM_NAV_CFP_INTERFACE_RECIPROCAL_RWY.ADA";
957 956      for Body ("fm_nav_cfp_interface.onehz_update") use "FM_NAV_CFP_INTERFACE_ONEHZ_UPDATE.ADA";
958 957      for Body ("fm_nav_cfp_interface.get_proc_vor_crs") use "FM_NAV_CFP_INTERFACE_GET_PROC_VOR_CRS.ADA";
959 958      for Body ("fm_nav_cfp_interface.get_leg_data") use "FM_NAV_CFP_INTERFACE_GET_LEG_DATA.ADA";
960 959      for Body ("fm_nav_cfp_interface.get_header_data") use "FM_NAV_CFP_INTERFACE_GET_HEADER_DATA.ADA";
961 960      for Body ("fm_nav_cfp_interface.find_downpath_vor") use "FM_NAV_CFP_INTERFACE_FIND_DOWNPATH_VOR.ADA";
962 961      for Spec ("fm_nav_cfp_interface") use "FM_NAV_CFP_INTERFACE_.ADA";
963 962      for Body ("fm_nav_cfp_interface") use "FM_NAV_CFP_INTERFACE.ADA";
964 963      for Body ("fm_nav_cdk_interface.onehz_update") use "FM_NAV_CDK_INTERFACE_ONEHZ_UPDATE.ADA";
965 964      for Body ("fm_nav_cdk_interface.fast_update") use "FM_NAV_CDK_INTERFACE_FAST_UPDATE.ADA";
966 965      for Spec ("fm_nav_cdk_interface") use "FM_NAV_CDK_INTERFACE_.ADA";
967 966      for Body ("fm_nav_cdk_interface") use "FM_NAV_CDK_INTERFACE.ADA";
968 967      for Body ("fm_nav_anp_monitor.compute_vert_anp") use "FM_NAV_ANP_MONITOR_COMPUTE_VERT_ANP.ADA";
969 968      for Body ("fm_nav_anp_monitor.compute_unable_rnp") use "FM_NAV_ANP_MONITOR_COMPUTE_UNABLE_RNP.ADA";
970 969      for Spec ("fm_nav_anp_monitor") use "FM_NAV_ANP_MONITOR_.ADA";
971 970      for Body ("fm_nav_anp_monitor") use "FM_NAV_ANP_MONITOR.ADA";
972 971      for Spec ("navigation_data") use "FM_NAVIGATION_DATA_.ADA";
973 972      for Body ("navigation_data") use "FM_NAVIGATION_DATA.ADA";
974 973      for Body ("fm_navaid_selector.select_route_vor") use "FM_NAVAID_SELECTOR_SELECT_ROUTE_VOR.ADA";
975 974      for Body ("fm_navaid_selector.select_proc_vor") use "FM_NAVAID_SELECTOR_SELECT_PROC_VOR.ADA";
976 975      for Body ("fm_navaid_selector.select_localizer") use "FM_NAVAID_SELECTOR_SELECT_LOCALIZER.ADA";
977 976      for Body ("fm_navaid_selector.localizer_navaid") use "FM_NAVAID_SELECTOR_LOCALIZER_NAVAID.ADA";
978 977      for Body ("fm_navaid_selector.display_navaids") use "FM_NAVAID_SELECTOR_DISPLAY_NAVAIDS.ADA";
979 978      for Spec ("fm_navaid_selector") use "FM_NAVAID_SELECTOR_.ADA";
980 979      for Body ("fm_navaid_selector") use "FM_NAVAID_SELECTOR.ADA";
981 980      for Spec ("flx_semaphore_pkg") use "FM_FLX_SEMAPHORE_PKG_.ADA";
982 981      for Body ("flx_semaphore_pkg") use "FM_FLX_SEMAPHORE_PKG.ADA";
983 982      for Spec ("vg_slow_to_fast_lfdata") use "FMF_VG_SLOW_TO_FAST_LFDATA_.ADA";
984 983      for Body ("guidance_executive.vg_init") use "FMF_VG_INIT_SEP.ADA";
985 984      for Spec ("vg_hot_spare_lfdata") use "FMF_VG_HOT_SPARE_LFDATA_.ADA";
986 985      for Spec ("vg_fast_to_slow_lfdata") use "FMF_VG_FAST_TO_SLOW_LFDATA_.ADA";

```

File: CTP_B787_PERF_CRZINITE.ZIP\stubs.gpr (continued)

```

987 986      for Spec ("vg_executives") use "FMF_VG_EXECUTIVES_PKG.ADA";
988 987      for Body ("vg_executives") use "FMF_VG_EXECUTIVES_PKG.ADA";
989 988      for Spec ("vg_data_managers") use "FMF_VG_DATA_MANAGERS_PKG.ADA";
990 989      for Body ("vg_data_managers") use "FMF_VG_DATA_MANAGERS_PKG.ADA";
991 990      for Spec ("vg_control_laws") use "FMF_VG_CONTROL_LAWS_PKG.ADA";
992 991      for Body ("vg_control_laws") use "FMF_VG_CONTROL_LAWS_PKG.ADA";
993 992      for Spec ("vg_common_lftypes") use "FMF_VG_COMMON_LFTYPES.ADA";
994 993      for Spec ("vg_checkpoint_lfdata") use "FMF_VG_CHECKPOINT_LFDATA.ADA";
995 994      for Body ("vg_data_managers.vgs_write_data") use "FMF_VGS_WRITE_DATA_SEP.ADA";
996 995      for Body ("vgs_leg_setup.vgs_updatelegs") use "FMF_VGS_UPDATELEGS_SEP.ADA";
997 996      for Body ("vgs_speed_tgt_sel.vgs_tospdtgtssel") use "FMF_VGS_TOSPDGTGTSEL_SEP.ADA";
998 997      for Body ("vgs_leg_setup.vgs_tgtalt") use "FMF_VGS_TGTALT_SEP.ADA";
999 998      for Spec ("vgs_speed_tgt_sel") use "FMF_VGS_SPEED_TGT_SEL_PKG.ADA";
1000 999      for Body ("vgs_speed_tgt_sel") use "FMF_VGS_SPEED_TGT_SEL_PKG.ADA";
1001 1000      for Body ("vgs_speed_tgt_sel.vgs_spdrevers") use "FMF_VGS_SPEEDREVERS_SEP.ADA";
1002 1001      for Spec ("vgs_snapshot_lfdata") use "FMF_VGS_SNAPSHOT_LFDATA.ADA";
1003 1002      for Spec ("vgs_slow_logic") use "FMF_VGS_SLOW_LOGIC_PKG.ADA";
1004 1003      for Body ("vgs_slow_logic") use "FMF_VGS_SLOW_LOGIC_PKG.ADA";
1005 1004      for Body ("vgs_leg_setup.vgs_setvleg") use "FMF_VGS_SETVLEG_SEP.ADA";
1006 1005      for Body ("vgs_leg_setup.vgs_setupvnav") use "FMF_VGS_SETUPVNAV_SEP.ADA";
1007 1006      for Body ("vgs_leg_setup.vgs_setpleg") use "FMF_VGS_SETPLEG_SEP.ADA";
1008 1007      for Body ("vgs_leg_setup.vgs_sequencelegs") use "FMF_VGS_SEQUENCELEGS_SEP.ADA";
1009 1008      for Body ("vgs_slow_logic.vgs_selectttlclim") use "FMF_VGS_SELECTTTLCLIM_SEP.ADA";
1010 1009      for Body ("vgs_speed_tgt_sel.vgs_selcasmach") use "FMF_VGS_SELCAISMACH_SEP.ADA";
1011 1010      for Body ("vg_data_managers.vgs_read_data") use "FMF_VGS_READ_DATA_SEP.ADA";
1012 1011      for Body ("vg_executives.vgs_misc_vars") use "FMF_VGS_MISC_VARS_SEP.ADA";
1013 1012      for Body ("vgs_leg_setup.vgs_lvloffalt") use "FMF_VGS_LVLOFFALT_SEP.ADA";
1014 1013      for Body ("vgs_speed_tgt_sel.vgs_limitspeeds") use "FMF_VGS_LIMITSPEEDS_SEP.ADA";
1015 1014      for Spec ("vgs_leg_setup") use "FMF_VGS_LEG_SETUP_PKG.ADA";
1016 1015      for Body ("vgs_leg_setup") use "FMF_VGS_LEG_SETUP_PKG.ADA";
1017 1016      for Body ("vg_executives.vg_guidcontrol") use "FMF_VGS_GUIDCONTROL_SEP.ADA";
1018 1017      for Body ("vgs_leg_setup.vgs_get_access_id") use "FMF_VGS_GET_ACCESS_ID_SEP.ADA";
1019 1018      for Body ("vgs_leg_setup.vgs_descentcntrl") use "FMF_VGS_DESCENTCNTRL_SEP.ADA";
1020 1019      for Body ("vgs_leg_setup.vgs_cruisecontrol") use "FMF_VGS_CRUISECNTRL_SEP.ADA";
1021 1020      for Body ("vgs_leg_setup.vgs_cntrlvertran") use "FMF_VGS_CNTRLVERTRAN_SEP.ADA";
1022 1021      for Body ("vgs_leg_setup.vgs_cnstralt") use "FMF_VGS_CNSTRALT_SEP.ADA";
1023 1022      for Body ("vgs_leg_setup.vgs_cmptvpath") use "FMF_VGS_CMPTVPATH_SEP.ADA";
1024 1023      for Body ("vgs_speed_tgt_sel.vgs_cmptspeeds") use "FMF_VGS_CMPTSPEEDS_SEP.ADA";
1025 1024      for Body ("vgs_leg_setup.vgs_cmptindexes") use "FMF_VGS_CMPTINDEXES_SEP.ADA";
1026 1025      for Body ("vgs_leg_setup.vgs_climbcntrl") use "FMF_VGS_CLIMBCNTRL_SEP.ADA";
1027 1026      for Body ("vgs_leg_setup.vgs_activatefpln") use "FMF_VGS_ACTIVATEFPLN_SEP.ADA";
1028 1027      for Body ("vg_data_managers.vgf_write_data") use "FMF_VGF_WRITE_DATA_SEP.ADA";

```

File: CTP_B787_PERF_CRZINITE.ZIP\stubs.gpr (continued)

```

1029 1028     for Body ("vgf_engagement.vgf_vnav_engout") use "FMF_VGF_VNAV_ENGOUT_SEP.ADA";
1030 1029     for Body ("vgf_logic.vgf_transitions_executive") use "FMF_VGF_TRANSITIONS_EXECUTIVE_SEP.ADA";
1031 1030     for Body ("vg_control_laws.vgf_thrust_control") use "FMF_VGF_THRUST_CONTROL_SEP.ADA";
1032 1031     for Body ("vg_control_laws.vgf_tas_filter") use "FMF_VGF_TAS_FILTER_SEP.ADA";
1033 1032     for Body ("vgf_logic.vgf_submode_logic") use "FMF_VGF_SUBMODE_LOGIC_SEP.ADA";
1034 1033     for Body ("vg_control_laws.vgf_spd_tgt_rate_lim") use "FMF_VGF_SPD_TGT_RATE_LIM_SEP.ADA";
1035 1034     for Body ("vg_control_laws.vgf_spd_tgt_asf_bias") use "FMF_VGF_SPD_TGT_ASF_BIAS_SEP.ADA";
1036 1035     for Body ("vg_control_laws.vgf_spd_rev_cntrl_sel") use "FMF_VGF_SPD_REV_CNTRL_SEL_SEP.ADA";
1037 1036     for Body ("vg_control_laws.vgf_soe_control") use "FMF_VGF_SOE_CONTROL_SEP.ADA";
1038 1037     for Spec ("vgf_snapshot_lfdata") use "FMF_VGF_SNAPSHOT_LFDATA_.ADA";
1039 1038     for Body ("vg_data_managers.vgf_read_data") use "FMF_VGF_READ_DATA_SEP.ADA";
1040 1039     for Body ("vg_control_laws.vgf_poe_control") use "FMF_VGF_POE_CONTROL_SEP.ADA";
1041 1040     for Body ("vg_control_laws.vgf_pitch_command") use "FMF_VGF_PITCH_COMMAND_SEP.ADA";
1042 1041     for Body ("vgf_logic.vgf_path_error") use "FMF_VGF_PATH_ERROR_SEP.ADA";
1043 1042     for Body ("vgf_logic.vgf_path_error_filter") use "FMF_VGF_PATH_ERROR_FILTER_SEP.ADA";
1044 1043     for Spec ("vgf_logic") use "FMF_VGF_LOGIC_PKG_.ADA";
1045 1044     for Body ("vgf_logic") use "FMF_VGF_LOGIC_PKG.ADA";
1046 1045     for Body ("vgf_engagement.vgf_init_engagement") use "FMF_VGF_INIT_ENGAGEMENT_SEP.ADA";
1047 1046     for Body ("vg_executives.vg_fastguid") use "FMF_VGF_FASTGUID_SEP.ADA";
1048 1047     for Body ("vgf_logic.vgf_epr_adjust_mode") use "FMF_VGF_EPR_ADJUST_MODE_SEP.ADA";
1049 1048     for Body ("vg_control_laws.vgf_enhanced_speed_reversion") use "FMF_VGF_ENHANCED_SPEED_REVERSION_SEP.ADA";
1050 1049     for Body ("vgf_engagement.vgf_engagement") use "FMF_VGF_ENGAGEMENT_SEP.ADA";
1051 1050     for Spec ("vgf_engagement") use "FMF_VGF_ENGAGEMENT_PKG_.ADA";
1052 1051     for Body ("vgf_engagement") use "FMF_VGF_ENGAGEMENT_PKG.ADA";
1053 1052     for Body ("vgf_logic.vgf_descent_transitions") use "FMF_VGF_DESCENT_TRANSITIONS_SEP.ADA";
1054 1053     for Body ("vgf_logic.vgf_descent_path_smooth") use "FMF_VGF_DESCENT_PATH_SMOOTH_SEP.ADA";
1055 1054     for Body ("vgf_logic.vgf_cruise_transitions") use "FMF_VGF_CRUISE_TRANSITIONS_SEP.ADA";
1056 1055     for Body ("vg_control_laws.vgf_cmpt_wind_acc") use "FMF_VGF_CMPT_WIND_ACC_SEP.ADA";
1057 1056     for Body ("vg_control_laws.vgf_cmpt_potential_fpa") use "FMF_VGF_CMPT_POTENTIAL_FPA_SEP.ADA";
1058 1057     for Body ("vg_control_laws.vgf_cl_spd_select") use "FMF_VGF_CL_SPD_SELECT_SEP.ADA";
1059 1058     for Body ("vgf_logic.vgf_climb_transitions") use "FMF_VGF_CLIMB_TRANSITIONS_SEP.ADA";
1060 1059     for Body ("vgf_logic.vgf_autothrottle_mode") use "FMF_VGF_AUTOTHROTTLE_MODE_SEP.ADA";
1061 1060     for Body ("vg_control_laws.vgf_asf_target") use "FMF_VGF_ASFTARGET_SEP.ADA";
1062 1061     for Body ("vg_control_laws.vgf_asf_control") use "FMF_VGF_ASFCONTROL_SEP.ADA";
1063 1062     for Body ("vg_control_laws.vgf_asf_activate") use "FMF_VGF_ASFACTIVATE_SEP.ADA";
1064 1063     for Body ("vg_control_laws.vgf_asc_target") use "FMF_VGF_ASC_TARGET_SEP.ADA";
1065 1064     for Body ("vg_control_laws.vgf_asc_control") use "FMF_VGF_ASC_CONTROL_SEP.ADA";
1066 1065     for Body ("vg_control_laws.vgf_asc_activate") use "FMF_VGF_ASC_ACTIVATE_SEP.ADA";
1067 1066     for Body ("vgf_logic.vgf_altitude_target") use "FMF_VGF_ALTITUDE_TARGET_SEP.ADA";
1068 1067     for Body ("vgf_logic.vgf_above_path") use "FMF_VGF_ABOVE_PATH_SEP.ADA";
1069 1068     for Body ("fmf_simsoft_io_pkg.put_outputs") use "FMF_SIMSOFT_IO_PKG_PUT_OUTPUTS.ADA";
1070 1069     for Body ("fmf_simsoft_io_pkg.get_inputs") use "FMF_SIMSOFT_IO_PKG_GET_INPUTS.ADA";

```

File: CTP_B787_PERF_CRZINITE.ZIP\stubs.gpr (continued)

```

1071 1070      for Spec ("fmf_simsoft_io_pkg") use "FMF_SIMSOFT_IO_PKG.ADA";
1072 1071      for Body ("fmf_simsoft_io_pkg") use "FMF_SIMSOFT_IO_PKG.ADA";
1073 1072      for Body ("fmf_simsoft_exec_pkg.rte_rsng_edg_sgnls") use "FMF_SIMSOFT_EXEC_PKG_RTE_RSNG_EDG.ADA";
1074 1073      for Body ("fmf_simsoft_exec_pkg.rte_flgng_edg_sgnls") use "FMF_SIMSOFT_EXEC_PKG_RTE_FLNG_EDG.ADA";
1075 1074      for Body ("fmf_simsoft_exec_pkg.exec") use "FMF_SIMSOFT_EXEC_PKG_EXEC.ADA";
1076 1075      for Body ("fmf_simsoft_exec_pkg.cr8_fts_response") use "FMF_SIMSOFT_EXEC_PKG_CR8_FTS_RESPONSE.ADA";
1077 1076      for Spec ("fmf_simsoft_exec_pkg") use "FMF_SIMSOFT_EXEC_PKG.ADA";
1078 1077      for Body ("fmf_simsoft_exec_pkg") use "FMF_SIMSOFT_EXEC_PKG.ADA";
1079 1078      for Body ("fmf_simsoft_cnfig_pkg.rsng_edg_init") use "FMF_SIMSOFT_CNFIG_PKG_RSNG_EDG_INIT.ADA";
1080 1079      for Body ("fmf_simsoft_cnfig_pkg.init") use "FMF_SIMSOFT_CNFIG_PKG_INIT.ADA";
1081 1080      for Body ("fmf_simsoft_cnfig_pkg.flng_edg_init") use "FMF_SIMSOFT_CNFIG_PKG_FLNG_EDG_INIT.ADA";
1082 1081      for Spec ("fmf_simsoft_cnfig_pkg") use "FMF_SIMSOFT_CNFIG_PKG.ADA";
1083 1082      for Body ("fmf_simsoft_cnfig_pkg") use "FMF_SIMSOFT_CNFIG_PKG.ADA";
1084 1083      for Spec ("fmf_secondary_throttle_hold_pkg") use "FMF_SECONDARY_THROTTLE_HOLD_PKG.ADA";
1085 1084      for Body ("fmf_secondary_throttle_hold_pkg") use "FMF_SECONDARY_THROTTLE_HOLD_PKG.ADA";
1086 1085      for Body ("fmf_printer_snapshot_data_mgr_pkg.snapshot_data") use "FMF_PRINTER_SNAPSHOT_DATA_SEP.ADA";
1087 1086      for Spec ("fmf_printer_snapshot_data_mgr_pkg") use "FMF_PRINTER_SNAPSHOT_DATA_MGR_PKG.ADA";
1088 1087      for Body ("fmf_printer_snapshot_data_mgr_pkg") use "FMF_PRINTER_SNAPSHOT_DATA_MGR_PKG.ADA";
1089 1088      for Spec ("fmf_printer_lftypes_pkg") use "FMF_PRINTER_LFTYPES_PKG.ADA";
1090 1089      for Spec ("fmf_perf_takeoff_bump_pkg") use "FMF_PERF_TAKEOFF_BUMP_PKG.ADA";
1091 1090      for Body ("fmf_perf_takeoff_bump_pkg") use "FMF_PERF_TAKEOFF_BUMP_PKG.ADA";
1092 1091      for Spec ("perf_su_spd_utils_pkg") use "FMF_PERF_SU_SPD_UTILS_PKG.ADA";
1093 1092      for Body ("perf_su_spd_utils_pkg") use "FMF_PERF_SU_SPD_UTILS_PKG.ADA";
1094 1093      for Spec ("perf_st_spdtape_pkg") use "FMF_PERF_ST_SPDTAPE_PKG.ADA";
1095 1094      for Body ("perf_st_spdtape_pkg") use "FMF_PERF_ST_SPDTAPE_PKG.ADA";
1096 1095      for Body ("fmf_perf_epm_pkg.epm_trm_psp") use "FMF_PERF_EPM_TRM_PSP_SEP.ADA";
1097 1096      for Body ("fmf_perf_epm_pkg.epm_trip_pred") use "FMF_PERF_EPM_TRIP_PRED_SEP.ADA";
1098 1097      for Body ("fmf_perf_epm_pkg.epm_tko_spd") use "FMF_PERF_EPM_TKO_SPD_SEP.ADA";
1099 1098      for Body ("fmf_perf_epm_pkg.epm_reverse_thrust") use "FMF_PERF_EPM_REVERSE_THRUST_SEP.ADA";
1100 1099      for Body ("fmf_perf_epm_pkg.epm_rating_thrust_model_2") use "FMF_PERF_EPM_RATING_THRUST_2_SEP.ADA";
1101 1100      for Body ("fmf_perf_epm_pkg.epm_rating") use "FMF_PERF_EPM_RATING_SEP.ADA";
1102 1101      for Spec ("fmf_perf_epm_pkg") use "FMF_PERF_EPM_PKG.ADA";
1103 1102      for Body ("fmf_perf_epm_pkg") use "FMF_PERF_EPM_PKG.ADA";
1104 1103      for Body ("fmf_perf_epm_pkg.epm_n1_to_epr") use "FMF_PERF_EPM_N1_TO_EPR_SEP.ADA";
1105 1104      for Body ("fmf_perf_epm_pkg.epm_min_tko_spd") use "FMF_PERF_EPM_MIN_TKO_SPD_SEP.ADA";
1106 1105      for Body ("fmf_perf_epm_pkg.epm_idle") use "FMF_PERF_EPM_IDLE_SEP.ADA";
1107 1106      for Body ("fmf_perf_epm_pkg.epm_fuel_flow_model_2") use "FMF_PERF_EPM_FUEL_FLOW_2_SEP.ADA";
1108 1107      for Body ("fmf_perf_epm_pkg.epm_epr_to_n1") use "FMF_PERF_EPM_EPR_TO_N1_SEP.ADA";
1109 1108      for Body ("fmf_perf_epm_pkg.epm_bleed_flag") use "FMF_PERF_EPM_BLEED_FLAG_SEP.ADA";
1110 1109      for Spec ("fmf_perf_aer_acdrag_pkg") use "FMF_PERF_AER_ACDRAG_PKG.ADA";
1111 1110      for Body ("fmf_perf_aer_acdrag_pkg") use "FMF_PERF_AER_ACDRAG_PKG.ADA";
1112 1111      for Body ("ops_hs_cross_cab_pkg.set_onside_part_numbers") use "FMF_OPS_HS_CROSS_CAB_PKG_SET_FM_PNUMS.ADA";

```


File: CTP_B787_PERF_CRZINITE.ZIP\stubs.gpr (continued)

```

1113 1112     for Spec ("ops_hs_cross_cab_pkg") use "FMF_OPS_HS_CROSS_CAB_PKG.ADA";
1114 1113     for Body ("ops_hs_cross_cab_pkg") use "FMF_OPS_HS_CROSS_CAB_PKG.ADA";
1115 1114     for Spec ("fmf_hsp_types_pkg") use "FMF_HSP_TYPES_PKG.ADA";
1116 1115     for Spec ("fmf_hsp_sync_state_pkg") use "FMF_HSP_SYNC_STATE_PKG.ADA";
1117 1116     for Body ("fmf_hsp_sync_state_pkg") use "FMF_HSP_SYNC_STATE_PKG.ADA";
1118 1117     for Body ("fmf_hsp_sync_msg_pkg.send_seq_no_block") use "FMF_HSP_SYNC_MSG_PKG_SEND_SEQ_NO_BLOCK.ADA";
1119 1118     for Body ("fmf_hsp_sync_msg_pkg.send_init_reqst") use "FMF_HSP_SYNC_MSG_PKG_SEND_INIT_REQST.ADA";
1120 1119     for Body ("fmf_hsp_sync_msg_pkg.send_init_ack") use "FMF_HSP_SYNC_MSG_PKG_SEND_INIT_ACK.ADA";
1121 1120     for Body ("fmf_hsp_sync_msg_pkg.send_data_block") use "FMF_HSP_SYNC_MSG_PKG_SEND_DATA_BLOCK.ADA";
1122 1121     for Body ("fmf_hsp_sync_msg_pkg.send_data_ack") use "FMF_HSP_SYNC_MSG_PKG_SEND_DATA_ACK.ADA";
1123 1122     for Spec ("fmf_hsp_sync_msg_pkg") use "FMF_HSP_SYNC_MSG_PKG.ADA";
1124 1123     for Body ("fmf_hsp_sync_msg_pkg") use "FMF_HSP_SYNC_MSG_PKG.ADA";
1125 1124     for Body ("fmf_hsp_control_pkg.tx_data") use "FMF_HSP_CONTROL_PKG_TX_DATA.ADA";
1126 1125     for Body ("fmf_hsp_control_pkg.state_reset") use "FMF_HSP_CONTROL_PKG_STATE_RESET.ADA";
1127 1126     for Body ("fmf_hsp_control_pkg.send_init_reqst") use "FMF_HSP_CONTROL_PKG_SEND_INIT_REQST.ADA";
1128 1127     for Body ("fmf_hsp_control_pkg.send_init_ack") use "FMF_HSP_CONTROL_PKG_SEND_INIT_ACK.ADA";
1129 1128     for Body ("fmf_hsp_control_pkg.send_ack_nak") use "FMF_HSP_CONTROL_PKG_SEND_ACK_NAK.ADA";
1130 1129     for Body ("fmf_hsp_control_pkg.rx_data") use "FMF_HSP_CONTROL_PKG_RX_DATA.ADA";
1131 1130     for Body ("fmf_hsp_control_pkg.proc_input_frm") use "FMF_HSP_CONTROL_PKG_PROC_INPUT_FRM.ADA";
1132 1131     for Body ("fmf_hsp_control_pkg.proc_ack_nak") use "FMF_HSP_CONTROL_PKG_PROC_ACK_NAK.ADA";
1133 1132     for Body ("fmf_hsp_control_pkg.master_idle_proc") use "FMF_HSP_CONTROL_PKG_MASTER_IDLE_PROC.ADA";
1134 1133     for Body ("fmf_hsp_control_pkg.log_protocol_error") use "FMF_HSP_CONTROL_PKG_LOG_PROTOCOL_ERROR.ADA";
1135 1134     for Body ("fmf_hsp_control_pkg.init_reqst_pending") use "FMF_HSP_CONTROL_PKG_INIT_REQST_PENDING.ADA";
1136 1135     for Body ("fmf_hsp_control_pkg.init_ack_pending") use "FMF_HSP_CONTROL_PKG_INIT_ACK_PENDING.ADA";
1137 1136     for Body ("fmf_hsp_control_pkg.initialize") use "FMF_HSP_CONTROL_PKG_INITIALIZE.ADA";
1138 1137     for Body ("fmf_hsp_control_pkg.hs_exec") use "FMF_HSP_CONTROL_PKG_HS_EXEC.ADA";
1139 1138     for Body ("fmf_hsp_control_pkg.frame_control") use "FMF_HSP_CONTROL_PKG_FRAME_CONTROL.ADA";
1140 1139     for Spec ("fmf_hsp_control_pkg") use "FMF_HSP_CONTROL_PKG.ADA";
1141 1140     for Body ("fmf_hsp_control_pkg") use "FMF_HSP_CONTROL_PKG.ADA";
1142 1141     for Body ("fmf_hsp_blocking_pkg.rollback") use "FMF_HSP_BLOCKING_PKG_ROLLBACK.ADA";
1143 1142     for Body ("fmf_hsp_blocking_pkg.put_block") use "FMF_HSP_BLOCKING_PKG_PUT_BLOCK.ADA";
1144 1143     for Body ("fmf_hsp_blocking_pkg.init") use "FMF_HSP_BLOCKING_PKG_INIT.ADA";
1145 1144     for Body ("fmf_hsp_blocking_pkg.get_block") use "FMF_HSP_BLOCKING_PKG_GET_BLOCK.ADA";
1146 1145     for Spec ("fmf_hsp_blocking_pkg") use "FMF_HSP_BLOCKING_PKG.ADA";
1147 1146     for Body ("fmf_hsp_blocking_pkg") use "FMF_HSP_BLOCKING_PKG.ADA";
1148 1147     for Spec ("fmf_hsp_access_pkg") use "FMF_HSP_ACCESS_PKG.ADA";
1149 1148     for Body ("fmf_hsp_access_pkg") use "FMF_HSP_ACCESS_PKG.ADA";
1150 1149     for Body ("guidance_executive.guid_setup") use "FMF_GUID_SETUP_SEP.ADA";
1151 1150     for Body ("guidance_executive.guid_onehertz_exec") use "FMF_GUID_ONEHERTZ_EXEC_SEP.ADA";
1152 1151     for Body ("guidance_executive.guid_foregnd_exec") use "FMF_GUID_FOREGND_EXEC_SEP.ADA";
1153 1152     for Spec ("guidance_spare_buffered_ifdata") use "FMF_GUIDANCE_SPARE_BUFFERED_IFDATA.ADA";
1154 1153     for Spec ("guidance_ifdata") use "FMF_GUIDANCE_IFDATA.ADA";

```

File: CTP_B787_PERF_CRZINITE.ZIP\stubs.gpr (continued)

```

1155 1154      for Body ("guidance_ifdata") use "FMF_GUIDANCE_IFDATA.ADA";
1156 1155      for Spec ("guidance_executive") use "FMF_GUIDANCE_EXECUTIVE_PKG_.ADA";
1157 1156      for Body ("guidance_executive") use "FMF_GUIDANCE_EXECUTIVE_PKG.ADA";
1158 1157      for Spec ("guidance_data_managers_pkg") use "FMF_GUIDANCE_DATA_MANAGERS_PKG_.ADA";
1159 1158      for Body ("guidance_data_managers_pkg") use "FMF_GUIDANCE_DATA_MANAGERS_PKG.ADA";
1160 1159      for Spec ("fmf_dual_partition_ifdata") use "FMF_DUAL_PARTITION_IFDATA_.ADA";
1161 1160      for Body ("fmf_dual_partition_ifdata") use "FMF_DUAL_PARTITION_IFDATA.ADA";
1162 1161      for Spec ("fmf_chronometer") use "FMF_CHRONOMETER_.ADA";
1163 1162      for Body ("fmf_chronometer") use "FMF_CHRONOMETER.ADA";
1164 1163      for Spec ("fmf_cam_timer_pkg") use "FMF_CAM_TIMER_PKG_.ADA";
1165 1164      for Body ("fmf_cam_timer_pkg") use "FMF_CAM_TIMER_PKG.ADA";
1166 1165      for Spec ("fmf_cam_process_data") use "FMF_CAM_PROCESS_DATA_.ADA";
1167 1166      for Body ("fmf_cam_process_data") use "FMF_CAM_PROCESS_DATA.ADA";
1168 1167      for Spec ("fmf_cam_msg_pkg") use "FMF_CAM_MSG_PKG_.ADA";
1169 1168      for Body ("fmf_cam_msg_pkg") use "FMF_CAM_MSG_PKG.ADA";
1170 1169      for Spec ("fmf_cam_init_for_startup") use "FMF_CAM_INIT_FOR_STARTUP_.ADA";
1171 1170      for Body ("fmf_cam_init_for_startup") use "FMF_CAM_INIT_FOR_STARTUP.ADA";
1172 1171      for Spec ("fmf_cam_activity_pkg") use "FMF_CAM_ACTIVITY_PKG_.ADA";
1173 1172      for Body ("fmf_cam_activity_pkg") use "FMF_CAM_ACTIVITY_PKG.ADA";
1174 1173      for Spec ("bite_periodic_data_update_pkg") use "FMF_BITE_PERIODIC_DATA_UPDATE_PKG_.ADA";
1175 1174      for Body ("bite_periodic_data_update_pkg") use "FMF_BITE_PERIODIC_DATA_UPDATE_PKG.ADA";
1176 1175      for Spec ("bite_exec_pkg") use "FMF_BITE_EXEC_PKG_.ADA";
1177 1176      for Body ("bite_exec_pkg") use "FMF_BITE_EXEC_PKG.ADA";
1178 1177      for Spec ("bite_cmcfc_pkg") use "FMF_BITE_CMCF_PKG_.ADA";
1179 1178      for Body ("bite_cmcfc_pkg") use "FMF_BITE_CMCF_PKG.ADA";
1180 1179      for Spec ("fmf_atc_uplink_preloading_pkg") use "FMF_ATC_UPLINK_PRELOADING_PKG_.ADA";
1181 1180      for Body ("fmf_atc_uplink_preloading_pkg") use "FMF_ATC_UPLINK_PRELOADING_PKG.ADA";
1182 1181      for Spec ("fmf_atc_uplink_obj_mgr") use "FMF_ATC_UPLINK_OBJ_MGR_.ADA";
1183 1182      for Body ("fmf_atc_uplink_obj_mgr") use "FMF_ATC_UPLINK_OBJ_MGR.ADA";
1184 1183      for Spec ("fmf_atc_uplink_loading_pkg") use "FMF_ATC_UPLINK_LOADING_PKG_.ADA";
1185 1184      for Body ("fmf_atc_uplink_loading_pkg") use "FMF_ATC_UPLINK_LOADING_PKG.ADA";
1186 1185      for Spec ("fmf_atc_uplink_decoding_pkg") use "FMF_ATC_UPLINK_DECODING_PKG_.ADA";
1187 1186      for Body ("fmf_atc_uplink_decoding_pkg") use "FMF_ATC_UPLINK_DECODING_PKG.ADA";
1188 1187      for Spec ("fmf_atc_types") use "FMF_ATC_TYPES_.ADA";
1189 1188      for Body ("fmf_atc_route_request_pkg.bld_rte_clr") use "FMF_ATC_ROUTE_REQUEST_PKG__BLD_RTE_CLR.ADA";
1190 1189      for Spec ("fmf_atc_route_request_pkg") use "FMF_ATC_ROUTE_REQUEST_PKG_.ADA";
1191 1190      for Body ("fmf_atc_route_request_pkg") use "FMF_ATC_ROUTE_REQUEST_PKG.ADA";
1192 1191      for Spec ("fmf_atc_route_clearance_pkg") use "FMF_ATC_ROUTE_CLEARANCE_PKG_.ADA";
1193 1192      for Body ("fmf_atc_route_clearance_pkg") use "FMF_ATC_ROUTE_CLEARANCE_PKG.ADA";
1194 1193      for Spec ("fmf_atc_report_crossing_req_pkg") use "FMF_ATC_REPORT_CROSSING_REQ_PKG_.ADA";
1195 1194      for Body ("fmf_atc_report_crossing_req_pkg") use "FMF_ATC_REPORT_CROSSING_REQ_PKG.ADA";
1196 1195      for Spec ("fmf_atc_rc_additional_pkg") use "FMF_ATC_RC_ADDITIONAL_PKG_.ADA";

```

File: CTP_B787_PERF_CRZINITE.ZIP\stubs.gpr (continued)

```

1197 1196      for Body ("fmf_atc_rc_additional_pkg") use "FMF_ATC_RC_ADDITIONAL_PKG.ADA";
1198 1197      for Spec ("fmf_atc_procedure_request_pkg") use "FMF_ATC_PROCEDURE_REQUEST_PKG_.ADA";
1199 1198      for Body ("fmf_atc_procedure_request_pkg") use "FMF_ATC_PROCEDURE_REQUEST_PKG.ADA";
1200 1199      for Spec ("fmf_atc_output_parameters_pkg") use "FMF_ATC_OUTPUT_PARAMETERS_PKG_.ADA";
1201 1200      for Body ("fmf_atc_output_parameters_pkg") use "FMF_ATC_OUTPUT_PARAMETERS_PKG.ADA";
1202 1201      for Spec ("fmf_atc_manager_pkg") use "FMF_ATC_MANAGER_PKG_.ADA";
1203 1202      for Body ("fmf_atc_manager_pkg") use "FMF_ATC_MANAGER_PKG.ADA";
1204 1203      for Spec ("fmf_atc_internal_data_obj_mgr") use "FMF_ATC_INTERNAL_DATA_OBJ_MGR_.ADA";
1205 1204      for Body ("fmf_atc_internal_data_obj_mgr") use "FMF_ATC_INTERNAL_DATA_OBJ_MGR.ADA";
1206 1205      for Spec ("fmf_atc_flight_number_pkg") use "FMF_ATC_FLIGHT_NUMBER_PKG_.ADA";
1207 1206      for Body ("fmf_atc_flight_number_pkg") use "FMF_ATC_FLIGHT_NUMBER_PKG.ADA";
1208 1207      for Spec ("fmf_atc_downlink_obj_mgr") use "FMF_ATC_DOWNLINK_OBJ_MGR_.ADA";
1209 1208      for Body ("fmf_atc_downlink_obj_mgr") use "FMF_ATC_DOWNLINK_OBJ_MGR.ADA";
1210 1209      for Spec ("fmf_atc_divert_pos_request_pkg") use "FMF_ATC_DIVERT_POS_REQUEST_PKG_.ADA";
1211 1210      for Body ("fmf_atc_divert_pos_request_pkg") use "FMF_ATC_DIVERT_POS_REQUEST_PKG.ADA";
1212 1211      for Spec ("fmf_atc_dist_to_wpt_req_pkg") use "FMF_ATC_DIST_TO_WPT_REQ_PKG_.ADA";
1213 1212      for Body ("fmf_atc_dist_to_wpt_req_pkg") use "FMF_ATC_DIST_TO_WPT_REQ_PKG.ADA";
1214 1213      for Body ("fmf_secondary_throttle_hold_pkg.fmf_at2ndhold_sep") use "FMF_AT2NDHOLD_SEP.ADA";
1215 1214      for Spec ("fmf_ami_access_pkg") use "FMF_AMI_ACCESS_PKG_.ADA";
1216 1215      for Body ("fmf_ami_access_pkg") use "FMF_AMI_ACCESS_PKG.ADA";
1217 1216      for Spec ("fmcs_partition_itypes_pkg") use "FMCS_FM_PARTITION_ITYPES_PKG_.ADA";
1218 1217      for Spec ("fmci_widget_event_tpkg") use "FMCI_WIDGET_EVENT_TPKG_.ADA";
1219 1218      for Spec ("fmci_validate_address_pkg") use "FMCI_VALIDATE_ADDRESS_PKG_.ADA";
1220 1219      for Body ("fmci_validate_address_pkg") use "FMCI_VALIDATE_ADDRESS_PKG.ADA";
1221 1220      for Spec ("fmci_utility_tpkg") use "FMCI_UTILITY_TPKG_.ADA";
1222 1221      for Body ("fmci_utility_pkg.process_nd_spad_entry") use "FMCI_UTILITY_PKG_PROC_ND_SPAD_ENT.ADA";
1223 1222      for Spec ("fmci_utility_pkg") use "FMCI_UTILITY_PKG_.ADA";
1224 1223      for Body ("fmci_utility_pkg") use "FMCI_UTILITY_PKG.ADA";
1225 1224      for Spec ("fmci_spad_util_pkg") use "FMCI_SPAD_UTIL_PKG_.ADA";
1226 1225      for Body ("fmci_spad_util_pkg") use "FMCI_SPAD_UTIL_PKG.ADA";
1227 1226      for Spec ("fmci_spad_manager_pkg") use "FMCI_SPAD_MANAGER_PKG_.ADA";
1228 1227      for Body ("fmci_spad_manager_pkg") use "FMCI_SPAD_MANAGER_PKG.ADA";
1229 1228      for Spec ("fmci_pos_ref4_dpkg") use "FMCI_POS_REF4_DPKG_.ADA";
1230 1229      for Body ("fmci_pos_ref4_dpkg") use "FMCI_POS_REF4_DPKG.ADA";
1231 1230      for Spec ("fmci_message_util_pkg") use "FMCI_MESSAGE_UTIL_PKG_.ADA";
1232 1231      for Body ("fmci_message_util_pkg") use "FMCI_MESSAGE_UTIL_PKG.ADA";
1233 1232      for Spec ("fmci_message_iface_pkg") use "FMCI_MESSAGE_IFACE_PKG_.ADA";
1234 1233      for Body ("fmci_message_iface_pkg") use "FMCI_MESSAGE_IFACE_PKG.ADA";
1235 1234      for Spec ("fmci_memory_page_pkg") use "FMCI_MEMORY_PAGE_PKG_.ADA";
1236 1235      for Body ("fmci_memory_page_pkg") use "FMCI_MEMORY_PAGE_PKG.ADA";
1237 1236      for Body ("fmci_memory_page_pkg.dump_perf_data") use "FMCI_MEMORY_PAGE_DUMP_PERF_DATA.ADA";
1238 1237      for Body ("fmci_memory_page_pkg.dietm") use "FMCI_MEMORY_PAGE_DIE_TM.ADA";

```

File: CTP_B787_PERF_CRZINITE.ZIP\stubs.gpr (continued)

```

1239 1238      for Body ("fmci_memory_page_pkg.dienav") use "FMCI_MEMORY_PAGE_DIE_NAV.ADA";
1240 1239      for Body ("fmci_memory_page_pkg.diefm") use "FMCI_MEMORY_PAGE_DIE_FM.ADA";
1241 1240      for Body ("fmci_memory_page_pkg.build_save") use "FMCI_MEMORY_PAGE_BUILD_SAVE.ADA";
1242 1241      for Body ("fmci_memory_page_pkg.build_recall") use "FMCI_MEMORY_PAGE_BUILD_RECALL.ADA";
1243 1242      for Body ("fmci_memory_page_pkg.build_part_number") use "FMCI_MEMORY_PAGE_BUILD_PART_NUMBER.ADA";
1244 1243      for Body ("fmci_memory_page_pkg.build_mro_partition") use "FMCI_MEMORY_PAGE_BUILD_MRO_PARTITION.ADA";
1245 1244      for Body ("fmci_memory_page_pkg.build_line") use "FMCI_MEMORY_PAGE_BUILD_LINE.ADA";
1246 1245      for Body ("fmci_memory_page_pkg.build_clear_page") use "FMCI_MEMORY_PAGE_BUILD_CLEAR_PAGE.ADA";
1247 1246      for Body ("fmci_memory_page_pkg.build_clear") use "FMCI_MEMORY_PAGE_BUILD_CLEAR_ALL.ADA";
1248 1247      for Spec ("fmci_fm_memory_mem_dpkg") use "FMCI_FM_MEMORY_MEM_DPKG.ADA";
1249 1248      for Body ("fmci_fm_memory_mem_dpkg") use "FMCI_FM_MEMORY_MEM_DPKG.ADA";
1250 1249      for Spec ("fmci_event_tpkg") use "FMCI_EVENT_TPKG.ADA";
1251 1250      for Body ("fmci_event_pkg.process_fm_event") use "FMCI_EVENT_PROCESS_FM_EVENT.ADA";
1252 1251      for Body ("fmci_event_pkg.process_efis_event") use "FMCI_EVENT_PROCESS_EFIS_EVENT.ADA";
1253 1252      for Body ("fmci_event_pkg.process_ci_event") use "FMCI_EVENT_PROCESS_CI_EVENT.ADA";
1254 1253      for Spec ("fmci_event_pkg") use "FMCI_EVENT_PKG.ADA";
1255 1254      for Body ("fmci_event_pkg") use "FMCI_EVENT_PKG.ADA";
1256 1255      for Spec ("fmci_event_in_pkg") use "FMCI_EVENT_IN_PKG.ADA";
1257 1256      for Body ("fmci_event_in_pkg") use "FMCI_EVENT_IN_PKG.ADA";
1258 1257      for Spec ("fmci_event_in_dpkg") use "FMCI_EVENT_IN_DPKG.ADA";
1259 1258      for Body ("fmci_event_in_pkg.decoder_exception_processing") use "FMCI_EVENT_IN_DECODE_EXCEPTION.ADA";
1260 1259      for Body ("fmci_event_in_pkg.decoder_event_processing") use "FMCI_EVENT_IN_DECODE_EVENT.ADA";
1261 1260      for Spec ("fmci_error_code_dpkg") use "FMCI_ERROR_CODE_DPKG.ADA";
1262 1261      for Spec ("fmci_efis_center_spad_pkg") use "FMCI_EFIS_CENTER_SPAD_PKG.ADA";
1263 1262      for Body ("fmci_efis_center_spad_pkg") use "FMCI_EFIS_CENTER_SPAD_PKG.ADA";
1264 1263      for Spec ("fmci_display_tpkg") use "FMCI_DISPLAY_TPKG.ADA";
1265 1264      for Spec ("fmci_display_pkg") use "FMCI_DISPLAY_PKG.ADA";
1266 1265      for Body ("fmci_display_pkg") use "FMCI_DISPLAY_PKG.ADA";
1267 1266      for Spec ("fmci_display_dpkg") use "FMCI_DISPLAY_DPKG.ADA";
1268 1267      for Body ("fmci_display_dpkg") use "FMCI_DISPLAY_DPKG.ADA";
1269 1268      for Body ("fmci_display_pkg.convert_page_data") use "FMCI_DISPLAY_CONVERT_PAGE_DATA.ADA";
1270 1269      for Body ("fmci_display_pkg.build_page_data") use "FMCI_DISPLAY_BUILD_PAGE_DATA.ADA";
1271 1270      for Spec ("fmci_cit_key_pkg") use "FMCI_CIT_KEY_PKG.ADA";
1272 1271      for Body ("fmci_cit_key_pkg") use "FMCI_CIT_KEY_PKG.ADA";
1273 1272      for Spec ("fmci_bp_req_que_manager_pkg") use "FMCI_BP_REQ_QUE_MANAGER_PKG.ADA";
1274 1273      for Body ("fmci_bp_req_que_manager_pkg") use "FMCI_BP_REQ_QUE_MANAGER_PKG.ADA";
1275 1274      for Body ("fmci_bp_req_que_manager_pkg.convert_widget_2_event") use "FMCI_BP_REQ_MANAGER_WIDGET_2_EVENT.ADA";
1276 1275      for Body ("fmci_bp_req_que_manager_pkg.build_bp_event") use "FMCI_BP_REQ_MANAGER_BUILD_BP_EVENT.ADA";
1277 1276      for Spec ("fmci_bite_dpkg") use "FMCI_BITE_TPKG.ADA";
1278 1277      for Spec ("flx_fm_register_navdb_pkg") use "FLX_FM_REGISTER_NAVDB_PKG.ADA";
1279 1278      for Body ("flx_fm_register_navdb_pkg") use "FLX_FM_REGISTER_NAVDB_PKG.ADA";
1280 1279      for Spec ("eosid_types") use "EOSID_TYPES.ADA";

```

File: CTP_B787_PERF_CRZINITE.ZIP\stubs.gpr (continued)

```

1281 1280      for Spec ("efis_waypoint_pkg") use "EFIS_WAYPOINT_PKG.ADA";
1282 1281      for Body ("efis_waypoint_pkg") use "EFIS_WAYPOINT_PKG.ADA";
1283 1282      for Body ("efis_vsd_utilities_pkg.object_within_footprint") use "EFIS_VSD_UTILITIES_OBJ_FOOTPRINT_SEP.ADA";
1284 1283      for Spec ("efis_vsd_utilities_pkg") use "EFIS_VSD_UTILITIES_PKG.ADA";
1285 1284      for Body ("efis_vsd_utilities_pkg") use "EFIS_VSD_UTILITIES_PKG.ADA";
1286 1285      for Body ("efis_vsd_utilities_pkg.determine_path_or_track_mode") use "EFIS_VSD_UTILITIES_PATH_OR_TRACK_SEP.ADA";
1287 1286      for Body ("efis_vsd_utilities_pkg.object_within_display_range") use "EFIS_VSD_UTILITIES_DISPLAY_RANGE_SEP.ADA";
1288 1287      for Body ("efis_vsd_utilities_pkg.calculate_swath") use "EFIS_VSD_UTILITIES_CALC_SWATH_SEP.ADA";
1289 1288      for Body ("efis_vsd_utilities_pkg.calculate_footprint") use "EFIS_VSD_UTILITIES_CALC_FOOTPRINT_SEP.ADA";
1290 1289      for Body ("efis_vsd_pkg.process_waypoints") use "EFIS_VSD_PROCESS_WAYPOINTS_SEP.ADA";
1291 1290      for Body ("efis_vsd_pkg.process_vnav_path") use "EFIS_VSD_PROCESS_VNAV_PATH_SEP.ADA";
1292 1291      for Body ("efis_vsd_pkg.process_runway") use "EFIS_VSD_PROCESS_RUNWAY_SEP.ADA";
1293 1292      for Body ("efis_vsd_pkg.process_missed_approach_point") use "EFIS_VSD_PROCESS_MISS_APP_POINT_SEP.ADA";
1294 1293      for Body ("efis_vsd_pkg.process_reference_approach_vector_and_decision_gates") use
      » "EFIS_VSD_PROCESS_DECISION_GATES_SEP.ADA";
1295 1294      for Spec ("efis_vsd_pkg") use "EFIS_VSD_PKG.ADA";
1296 1295      for Body ("efis_vsd_pkg") use "EFIS_VSD_PKG.ADA";
1297 1296      for Body ("efis_vsd_io_pkg.send_1hz_data") use "EFIS_VSD_IO_SEND_1HZ_DATA_SEP.ADA";
1298 1297      for Body ("efis_vsd_io_pkg.send_10hz_data") use "EFIS_VSD_IO_SEND_10HZ_DATA_SEP.ADA";
1299 1298      for Spec ("efis_vsd_io_pkg") use "EFIS_VSD_IO_PKG.ADA";
1300 1299      for Body ("efis_vsd_io_pkg") use "EFIS_VSD_IO_PKG.ADA";
1301 1300      for Body ("efis_vsd_io_pkg.get_external_data") use "EFIS_VSD_IO_GET_DATA_SEP.ADA";
1302 1301      for Spec ("efis_vsd_io_dpkg") use "EFIS_VSD_IO_DPKG.ADA";
1303 1302      for Body ("efis_vsd_pkg.get_waypoint_data_from_buffer") use "EFIS_VSD_GET_WPOINT_DATA_SEP.ADA";
1304 1303      for Body ("efis_vsd_pkg.get_runway_data_from_header") use "EFIS_VSD_GET_RUNWAY_DATA_SEP.ADA";
1305 1304      for Body ("efis_vsd_pkg.get_reference_approach_data_from_buffer") use "EFIS_VSD_GET_REF_APPROACH_DATA_SEP.ADA";
1306 1305      for Body ("efis_vsd_pkg.get_missed_approach_point_data_from_buffer") use "EFIS_VSD_GET_MP_DATA_SEP.ADA";
1307 1306      for Body ("efis_vsd_pkg.get_lgb_data") use "EFIS_VSD_GET_LGB_DATA_SEP.ADA";
1308 1307      for Body ("efis_vsd_pkg.get_lgb_data_for_iss") use "EFIS_VSD_GET_LGB_DATA_FOR_ISS_SEP.ADA";
1309 1308      for Spec ("efis_viewable_window_pkg") use "EFIS_VIEWABLE_WINDOW_PKG.ADA";
1310 1309      for Body ("efis_viewable_window_pkg") use "EFIS_VIEWABLE_WINDOW_PKG.ADA";
1311 1310      for Spec ("efis_text_utilities_pkg") use "EFIS_TEXT_UTILITIES_PKG.ADA";
1312 1311      for Body ("efis_text_utilities_pkg") use "EFIS_TEXT_UTILITIES_PKG.ADA";
1313 1312      for Spec ("efis_symbols_lines_tpkg") use "EFIS_SYMBOLS_LINES_TPKG.ADA";
1314 1313      for Spec ("efis_store_utilities_pkg") use "EFIS_STORE_UTILITIES_PKG.ADA";
1315 1314      for Body ("efis_store_utilities_pkg") use "EFIS_STORE_UTILITIES_PKG.ADA";
1316 1315      for Spec ("efis_store_buff_pkg") use "EFIS_STORE_BUF.ADA";
1317 1316      for Body ("efis_store_buff_pkg") use "EFIS_STORE_BUF.ADA";
1318 1317      for Body ("efis_special_data_pkg.process_tuned_nav aids") use "EFIS_SPECIAL_TUNED_NAV AID S_SEP.ADA";
1319 1318      for Body ("efis_special_data_pkg.process_srp_points") use "EFIS_SPECIAL_SRP_POINTS_SEP.ADA";
1320 1319      for Body ("efis_special_data_pkg.convert_saf32_to_char_degrees") use "EFIS_SPECIAL_SAF32_TO_CHAR_SEP.ADA";
1321 1320      for Body ("efis_special_data_pkg.process_radial_text") use "EFIS_SPECIAL_PROCESS_RADIAL_TXT.ADA";

```

File: CTP_B787_PERF_CRZINITE.ZIP\stubs.gpr (continued)

```

1322 1321      for Body ("efis_special_data_pkg.process_special_data") use "EFIS_SPECIAL_PROCESS_DATA_SEP.ADA";
1323 1322      for Body ("efis_special_data_pkg.process_orig_dest_airports") use "EFIS_SPECIAL_ORIG_DEST_AIRPORTS_SEP.ADA";
1324 1323      for Body ("efis_special_data_pkg.process_offpath_descent") use "EFIS_SPECIAL_OFFPATH_DESCENT_SEP.ADA";
1325 1324      for Body ("efis_special_data_pkg.get_global_data") use "EFIS_SPECIAL_GLOBAL_DATA_SEP.ADA";
1326 1325      for Spec ("efis_special_data_pkg") use "EFIS_SPECIAL_DATA_PKG_.ADA";
1327 1326      for Body ("efis_special_data_pkg") use "EFIS_SPECIAL_DATA_PKG.ADA";
1328 1327      for Body ("efis_special_data_pkg.update_alt_airports_org_des") use "EFIS_SPECIAL_ALT_ORG_DES_SEP.ADA";
1329 1328      for Body ("efis_special_data_pkg.update_alt_aprt_waypoints") use "EFIS_SPECIAL_ALT_APRT_WAYPOINTS_SEP.ADA";
1330 1329      for Body ("efis_special_data_pkg.update_alt_aprt_list") use "EFIS_SPECIAL_ALT_APRT_LIST_SEP.ADA";
1331 1330      for Body ("efis_special_data_pkg.process_alternate_airports") use "EFIS_SPECIAL_ALTERNATE_AIRPORTS_SEP.ADA";
1332 1331      for Spec ("efis_search_pkg") use "EFIS_SEARCH_PKG_.ADA";
1333 1332      for Body ("efis_search_pkg") use "EFIS_SEARCH_PKG.ADA";
1334 1333      for Spec ("efis_search_lfdata") use "EFIS_SEARCH_LFDATA_.ADA";
1335 1334      for Spec ("efis_rwy_def_pkg") use "EFIS_RWY_DEF_PKG_.ADA";
1336 1335      for Body ("efis_rwy_def_pkg") use "EFIS_RWY_DEF_PKG.ADA";
1337 1336      for Spec ("efis_profile_point_pkg") use "EFIS_PROFILE_POINT_PKG_.ADA";
1338 1337      for Body ("efis_profile_point_pkg") use "EFIS_PROFILE_POINT_PKG.ADA";
1339 1338      for Body ("efis_profile_point_pkg.format_altitude_profile_points") use "EFIS_PROFILE_POINT_FORMAT_POINT_SEP.ADA";
1340 1339      for Body ("efis_profile_point_pkg.efis_alt_prof_point_location") use "EFIS_PROFILE_POINT_EFIAPPLOC_SEP.ADA";
1341 1340      for Body ("efis_profile_point_pkg.calculate_altitude_locations") use "EFIS_PROFILE_POINT_CAL_ALT_LOC_SEP.ADA";
1342 1341      for Spec ("efis_proc_offscale_alternates_pkg") use "EFIS_PROC_OFFSCALE_ALTERNATES_PKG_.ADA";
1343 1342      for Body ("efis_proc_offscale_alternates_pkg") use "EFIS_PROC_OFFSCALE_ALTERNATES_PKG.ADA";
1344 1343      for Spec ("efis_path_protected_ifdata_pkg") use "EFIS_PATH_PROTECTED_IFDATA_.ADA";
1345 1344      for Spec ("efis_path_perf_data_pkg") use "EFIS_PATH_PERF_DATA_PKG_.ADA";
1346 1345      for Body ("efis_path_perf_data_pkg") use "EFIS_PATH_PERF_DATA_PKG.ADA";
1347 1346      for Spec ("efis_path_lg_manager") use "EFIS_PATH_LG_MANAGER_.ADA";
1348 1347      for Body ("efis_path_lg_manager") use "EFIS_PATH_LG_MANAGER.ADA";
1349 1348      for Spec ("efis_path_lftypes") use "EFIS_PATH_LFTYPES_.ADA";
1350 1349      for Body ("efis_path_exec_lt_pkg.efis_store_segments_into_lgb") use "EFIS_PATH_EXEC_STORE_SEP.ADA";
1351 1350      for Body ("efis_path_exec_lt_pkg.efis_path_smooth_plan") use "EFIS_PATH_EXEC_SMOOTH_SEP.ADA";
1352 1351      for Body ("efis_path_exec_lt_pkg.efis_path_setup") use "EFIS_PATH_EXEC_SETUP_SEP.ADA";
1353 1352      for Body ("efis_path_exec_lt_pkg.efis_path_prov_setup") use "EFIS_PATH_EXEC_PROV_SEP.ADA";
1354 1353      for Body ("efis_path_exec_lt_pkg.efis_path_ppos_hold_push_ahead") use "EFIS_PATH_EXEC_PPOS_SEP.ADA";
1355 1354      for Body ("efis_path_exec_lt_pkg.efis_path_plan_select") use "EFIS_PATH_EXEC_PLN_SEL_SEP.ADA";
1356 1355      for Spec ("efis_path_exec_lt_pkg") use "EFIS_PATH_EXEC_LT_PKG_.ADA";
1357 1356      for Body ("efis_path_exec_lt_pkg") use "EFIS_PATH_EXEC_LT_PKG.ADA";
1358 1357      for Body ("efis_path_exec_lt_pkg.efis_path_distance") use "EFIS_PATH_EXEC_DIST_SEP.ADA";
1359 1358      for Body ("efis_path_exec_lt_pkg.efis_path_l4k_update_holds") use "EFIS_PATH_EXEC_L4K_UPDATE_SEP.ADA";
1360 1359      for Spec ("efis_nd_10hz_sublayer_manager_pkg") use "EFIS_ND_10HZ_SUBLAYER_MANAGER_PKG_.ADA";
1361 1360      for Body ("efis_nd_10hz_sublayer_manager_pkg") use "EFIS_ND_10HZ_SUBLAYER_MANAGER_PKG.ADA";
1362 1361      for Spec ("efis_ndbdata_pkg") use "EFIS_NDBDATA_PKG_.ADA";
1363 1362      for Body ("efis_ndbdata_pkg") use "EFIS_NDBDATA_PKG.ADA";

```

File: CTP_B787_PERF_CRZINITE.ZIP\stubs.gpr (continued)

```

1364 1363      for Body ("efis_ndbdata_pkg.ndb_wpt_or_cwpt_process") use "EFIS_NDBDATA_NDB_WPT_OR_CWP_SEP.ADA";
1365 1364      for Body ("efis_ndbdata_pkg.ndb_ndrb_process") use "EFIS_NDBDATA_NDB_NDRB_SEP.ADA";
1366 1365      for Body ("efis_ndbdata_pkg.ndb_nav_process") use "EFIS_NDBDATA_NDB_NAV_SEP.ADA";
1367 1366      for Body ("efis_ndbdata_pkg.ndb_buffer_access_manager") use "EFIS_NDBDATA_NDB_BUFF_ACCESS_MGR_SEP.ADA";
1368 1367      for Body ("efis_ndbdata_pkg.ndb_apt_process") use "EFIS_NDBDATA_NDB_APT_SEP.ADA";
1369 1368      for Spec ("efis_map_req_pkg") use "EFIS_MAP_REQ_PKG.ADA";
1370 1369      for Body ("efis_map_req_pkg") use "EFIS_MAP_REQ_PKG.ADA";
1371 1370      for Spec ("efis_map_parameter_lfdata") use "EFIS_MAP_PARAMETER_LFDATA.ADA";
1372 1371      for Spec ("efis_map_background_lfdata") use "EFIS_MAP_BACKGROUND_LFDATA.ADA";
1373 1372      for Spec ("efis_lt_wrapper_pkg") use "EFIS_LT_WRAPPER_PKG.ADA";
1374 1373      for Body ("efis_lt_wrapper_pkg") use "EFIS_LT_WRAPPER_PKG.ADA";
1375 1374      for Spec ("efis_lt_segment_list_wrapper_pkg") use "EFIS_LT_SEGMENT_LIST_WRAPPER_PKG.ADA";
1376 1375      for Body ("efis_lt_segment_list_wrapper_pkg") use "EFIS_LT_SEGMENT_LIST_WRAPPER_PKG.ADA";
1377 1376      for Spec ("efis_lt_leg_list_wrapper_pkg") use "EFIS_LT_LEG_LIST_WRAPPER_PKG.ADA";
1378 1377      for Body ("efis_lt_leg_list_wrapper_pkg") use "EFIS_LT_LEG_LIST_WRAPPER_PKG.ADA";
1379 1378      for Spec ("efis_leg_seg_proc_pkg") use "EFIS_LEG_SEG_PROC_PKG.ADA";
1380 1379      for Body ("efis_leg_seg_proc_pkg") use "EFIS_LEG_SEG_PROC_PKG.ADA";
1381 1380      for Body ("efis_leg_seg_proc_pkg.insert_hold_proc_symbols") use "EFIS_LEG_SEG_HOLD_PROC_SYMBOLS_SEP.ADA";
1382 1381      for Body ("efis_leg_seg_proc_pkg.hold_proc_display_as_symbol") use "EFIS_LEG_SEG_DISPLAY_AS_SYMBOL_SEP.ADA";
1383 1382      for Spec ("efis_iss_utilty_pkg") use "EFIS_ISS_UTILITY_PKG.ADA";
1384 1383      for Body ("efis_iss_utilty_pkg") use "EFIS_ISS_UTILITY_PKG.ADA";
1385 1384      for Body ("efis_iss_traj_manager_pkg.refresh_events") use "EFIS_ISS_TRAJ_MG_PKG_REFRESH_EVENTS.ADA";
1386 1385      for Body ("efis_iss_traj_manager_pkg.determine_traj_intent") use "EFIS_ISS_TRAJ_MG_PKG_DETERMINE_TRAJ.ADA";
1387 1386      for Body ("efis_iss_traj_manager_pkg.process_iss_trajectory") use "EFIS_ISS_TRAJ_MANAGER_PKG_PRO_ISS_TRAJ.ADA";
1388 1387      for Spec ("efis_iss_traj_manager_pkg") use "EFIS_ISS_TRAJ_MANAGER_PKG.ADA";
1389 1388      for Body ("efis_iss_traj_manager_pkg") use "EFIS_ISS_TRAJ_MANAGER_PKG.ADA";
1390 1389      for Spec ("efis_iss_size_pkg") use "EFIS_ISS_SIZE_PKG.ADA";
1391 1390      for Spec ("efis_iss_output_tpkg") use "EFIS_ISS_OUTPUT_TPKG.ADA";
1392 1391      for Body ("efis_iss_output_pkg.determine_endpoint_type") use "EFIS_ISS_OUTPUT_PKG_DETER_ENDPT.ADA";
1393 1392      for Spec ("efis_iss_output_pkg") use "EFIS_ISS_OUTPUT_PKG.ADA";
1394 1393      for Body ("efis_iss_output_pkg") use "EFIS_ISS_OUTPUT_PKG.ADA";
1395 1394      for Spec ("efis_iss_output_dpkg") use "EFIS_ISS_OUTPUT_DPKG.ADA";
1396 1395      for Body ("efis_iss_output_dpkg") use "EFIS_ISS_OUTPUT_DPKG.ADA";
1397 1396      for Spec ("efis_iss_iftypes") use "EFIS_ISS_IFTYPES.ADA";
1398 1397      for Spec ("efis_iss_dpkg") use "EFIS_ISS_DPKG.ADA";
1399 1398      for Spec ("efis_io_iftypes") use "EFIS_IO_IFTYPES.ADA";
1400 1399      for Spec ("efis_init_pkg") use "EFIS_INIT_PKG.ADA";
1401 1400      for Body ("efis_init_pkg") use "EFIS_INIT_PKG.ADA";
1402 1401      for Spec ("efis_get_acstate_data_pkg") use "EFIS_GET_ACSTATE_DATA_PKG.ADA";
1403 1402      for Body ("efis_get_acstate_data_pkg") use "EFIS_GET_ACSTATE_DATA_PKG.ADA";
1404 1403      for Spec ("efis_generic_pkg") use "EFIS_GENERIC_PKG.ADA";
1405 1404      for Body ("efis_flight_data_pkg.find_provisional_legs") use "EFIS_FLIGHT_DATA_PROV_SEP.ADA";

```

File: CTP_B787_PERF_CRZINITE.ZIP\stubs.gpr (continued)

```

1406 1405      for Spec ("efis_flight_data_pkg") use "EFIS_FLIGHT_DATA_PKG.ADA";
1407 1406      for Body ("efis_flight_data_pkg") use "EFIS_FLIGHT_DATA_PKG.ADA";
1408 1407      for Body ("efis_flight_data_pkg.find_inactive_legs") use "EFIS_FLIGHT_DATA_INACTIVE_SEP.ADA";
1409 1408      for Body ("efis_flight_data_pkg.find_active_legs") use "EFIS_FLIGHT_DATA_ACTIVE_SEP.ADA";
1410 1409      for Body ("efis_flight_data_pkg.find_active_legs.draw_swath") use "EFIS_FLIGHT_DATA_ACTIVE_DRAW_SWATH_SEP.ADA";
1411 1410      for Spec ("efis_ext_interfaces_pkg") use "EFIS_EXT_INTERFACES_PKG.ADA";
1412 1411      for Body ("efis_ext_interfaces_pkg") use "EFIS_EXT_INTERFACES_PKG.ADA";
1413 1412      for Body ("efis_edit_pkg.determine_edit_area") use "EFIS_EDIT_PKG_DET_EDIT_AREA.ADA";
1414 1413      for Body ("efis_edit_pkg.determine_corner_points") use "EFIS_EDIT_PKG_DET_CORNER_POINTS.ADA";
1415 1414      for Body ("efis_edit_pkg.define_window") use "EFIS_EDIT_PKG_DEFINE_WINDOW.ADA";
1416 1415      for Spec ("efis_edit_pkg") use "EFIS_EDIT_PKG.ADA";
1417 1416      for Body ("efis_edit_pkg") use "EFIS_EDIT_PKG.ADA";
1418 1417      for Spec ("efis_dynamic_data_pkg") use "EFIS_DYNAMIC_DATA_PKG.ADA";
1419 1418      for Body ("efis_dynamic_data_pkg") use "EFIS_DYNAMIC_DATA_PKG.ADA";
1420 1419      for Spec ("efis_duplicate_utility_pkg") use "EFIS_DUPLICATE_UTILITY_PKG.ADA";
1421 1420      for Body ("efis_duplicate_utility_pkg") use "EFIS_DUPLICATE_UTILITY_PKG.ADA";
1422 1421      for Spec ("efis_debug_control_pkg") use "EFIS_DEBUG_CONTROL_PKG.ADA";
1423 1422      for Spec ("efis_convert_pkg") use "EFIS_CONVERT_PKG.ADA";
1424 1423      for Body ("efis_convert_pkg") use "EFIS_CONVERT_PKG.ADA";
1425 1424      for Spec ("efis_ci_interface_mgr_pkg") use "EFIS_CI_INTERFACE_MGR_PKG.ADA";
1426 1425      for Body ("efis_ci_interface_mgr_pkg") use "EFIS_CI_INTERFACE_MGR_PKG.ADA";
1427 1426      for Body ("efis_bkgnd_buff_cntrl_pkg.process_nd_optional_layer") use "EFIS_BKGN_BUFF_CNTRL_PROC_OPT_SEP.ADA";
1428 1427      for Body ("efis_bkgnd_buff_cntrl_pkg.process_minimap") use "EFIS_BKGN_BUFF_CNTRL_PROC_MINI_SEP.ADA";
1429 1428      for Body ("efis_bkgnd_buff_cntrl_pkg.process_build_map") use "EFIS_BKGN_BUFF_CNTRL_PROC_MAP_SEP.ADA";
1430 1429      for Spec ("efis_bkgnd_buff_cntrl_pkg") use "EFIS_BKGN_BUFF_CNTRL_PKG.ADA";
1431 1430      for Body ("efis_bkgnd_buff_cntrl_pkg") use "EFIS_BKGN_BUFF_CNTRL_PKG.ADA";
1432 1431      for Body ("efis_bkgnd_buff_cntrl_pkg.map_edit_area") use "EFIS_BKGN_BUFF_CNTRL_MAP_AREA_SEP.ADA";
1433 1432      for Body ("efis_bkgnd_buff_cntrl_pkg.build_background_buffer") use "EFIS_BKGN_BUFF_CNTRL_BUILD_MAP_SEP.ADA";
1434 1433      for Spec ("efis_base_lftypes") use "EFIS_BASE_LFTYPES.ADA";
1435 1434      for Spec ("efis_661_vsd_layer_pkg") use "EFIS_661_VSD_LAYER_PKG.ADA";
1436 1435      for Body ("efis_661_vsd_layer_pkg") use "EFIS_661_VSD_LAYER_PKG.ADA";
1437 1436      for Spec ("efis_661_nd_opt_layer_pkg") use "EFIS_661_ND_OPT_LAYER_PKG.ADA";
1438 1437      for Body ("efis_661_nd_opt_layer_pkg") use "EFIS_661_ND_OPT_LAYER_PKG.ADA";
1439 1438      for Spec ("efis_661_nd_layer_pkg") use "EFIS_661_ND_LAYER_PKG.ADA";
1440 1439      for Body ("efis_661_nd_layer_pkg") use "EFIS_661_ND_LAYER_PKG.ADA";
1441 1440      for Spec ("efis_661_nd_10hz_sublayer_pkg") use "EFIS_661_ND_10HZ_SUBLAYER_PKG.ADA";
1442 1441      for Body ("efis_661_nd_10hz_sublayer_pkg") use "EFIS_661_ND_10HZ_SUBLAYER_PKG.ADA";
1443 1442      for Spec ("efis_661_minimap_layer_pkg") use "EFIS_661_MINIMAP_LAYER_PKG.ADA";
1444 1443      for Body ("efis_661_minimap_layer_pkg") use "EFIS_661_MINIMAP_LAYER_PKG.ADA";
1445 1444      for Spec ("efis_661_iftypes") use "EFIS_661_IFTYPES.ADA";
1446 1445      for Spec ("efis_661_ifdata") use "EFIS_661_IFDATA.ADA";
1447 1446      for Body ("earth_center_cood_math_pkg.convert_to_lat_lon") use "EARTH_CENTER_COOD_MATH_PKG_TO_LATLON.ADA";

```


File: CTP_B787_PERF_CRZINITE.ZIP\stubs.gpr (continued)

```

1448 1447      for Body ("earth_center_cood_math_pkg.convert_to_earth_center") use "EARTH_CENTER_COOD_MATH_PKG_TO_EAR_CTR.ADA";
1449 1448      for Body ("earth_center_cood_math_pkg.size_to_earth_radius") use "EARTH_CENTER_COOD_MATH_PKG_SIZE_TO.ADA";
1450 1449      for Body ("earth_center_cood_math_pkg.rotate_and_shift") use "EARTH_CENTER_COOD_MATH_PKG_ROT_SHIFT.ADA";
1451 1450      for Body ("earth_center_cood_math_pkg.nm_ft") use "EARTH_CENTER_COOD_MATH_PKG_NM_FT.ADA";
1452 1451      for Body ("earth_center_cood_math_pkg.magnitude") use "EARTH_CENTER_COOD_MATH_PKG_MAGNITUDE.ADA";
1453 1452      for Body ("earth_center_cood_math_pkg.intersection_point_of_two_lines") use "EARTH_CENTER_COOD_MATH_PKG_INTRSCT.ADA";
1454 1453      for Body ("earth_center_cood_math_pkg.convert_ft_nm") use "EARTH_CENTER_COOD_MATH_PKG_FT_TO_NM.ADA";
1455 1454      for Body ("earth_center_cood_math_pkg.local_earth_radius") use "EARTH_CENTER_COOD_MATH_PKG_EARTH_RAD.ADA";
1456 1455      for Body ("earth_center_cood_math_pkg.dot_product") use "EARTH_CENTER_COOD_MATH_PKG_DOT_PRODUCT.ADA";
1457 1456      for Body ("earth_center_cood_math_pkg.center_point_of_curve") use "EARTH_CENTER_COOD_MATH_PKG_CURVE_CTR.ADA";
1458 1457      for Body ("earth_center_cood_math_pkg.cross_product") use "EARTH_CENTER_COOD_MATH_PKG_CROSS_PRD.ADA";
1459 1458      for Spec ("earth_center_cood_math_pkg") use "EARTH_CENTER_COOD_MATH_PKG_.ADA";
1460 1459      for Body ("earth_center_cood_math_pkg") use "EARTH_CENTER_COOD_MATH_PKG.ADA";
1461 1460      for Spec ("descent_path_mgr") use "DESCENT_PATH_MGR_.ADA";
1462 1461      for Body ("descent_path_mgr") use "DESCENT_PATH_MGR.ADA";
1463 1462      for Spec ("cut_takeoff_vs speeds_check_pkg") use "CUT TAKEOFF_VSPEEDS_CHECK_PKG_.ADA";
1464 1463      for Body ("cut_takeoff_vs speeds_check_pkg") use "CUT TAKEOFF_VSPEEDS_CHECK_PKG.ADA";
1465 1464      for Spec ("cut_takeoff_ref_states_pkg") use "CUT TAKEOFF_REF_STATES_PKG_.ADA";
1466 1465      for Body ("cut_takeoff_ref_states_pkg") use "CUT TAKEOFF_REF_STATES_PKG.ADA";
1467 1466      for Spec ("cut_string_pkg") use "CUT_STRING_PKG_.ADA";
1468 1467      for Body ("cut_string_pkg") use "CUT_STRING_PKG.ADA";
1469 1468      for Spec ("cut_string_conversion_pkg") use "CUT_STRING_CONVERSION_PKG_.ADA";
1470 1469      for Body ("cut_string_conversion_pkg") use "CUT_STRING_CONVERSION_PKG.ADA";
1471 1470      for Spec ("cut_std_inclusion_pkg") use "CUT_STD_INCLUSION_PKG_.ADA";
1472 1471      for Spec ("cut_rte_util_pkg") use "CUT RTE_UTIL_PKG_.ADA";
1473 1472      for Body ("cut_rte_util_pkg") use "CUT RTE_UTIL_PKG.ADA";
1474 1473      for Spec ("cut_key_conversion_pkg") use "CUT_KEY_CONVERSION_PKG_.ADA";
1475 1474      for Body ("cut_key_conversion_pkg") use "CUT_KEY_CONVERSION_PKG.ADA";
1476 1475      for Spec ("cut_gen_inclusion_pkg") use "CUT_GEN_INCLUSION_PKG_.ADA";
1477 1476      for Body ("cut_gen_inclusion_pkg") use "CUT_GEN_INCLUSION_PKG.ADA";
1478 1477      for Spec ("cut_datalink_pkg") use "CUT_DATALINK_PKG_.ADA";
1479 1478      for Body ("cut_datalink_pkg") use "CUT_DATALINK_PKG.ADA";
1480 1479      for Spec ("cut_cpi_operation_pkg") use "CUT_CPI_OPERATION_PKG_.ADA";
1481 1480      for Body ("cut_cpi_operation_pkg") use "CUT_CPI_OPERATION_PKG.ADA";
1482 1481      for Body ("common_lgb.putlgbleg") use "COMMON_LGB_PUTLGBLEG.ADA";
1483 1482      for Body ("common_lgb.putlgbhdr") use "COMMON_LGB_PUTLGBHDR.ADA";
1484 1483      for Spec ("common_lgb_int_nonresync_dpkg") use "COMMON_LGB_INT_NONRESYNC_DPKG_.ADA";
1485 1484      for Body ("common_lgb.initlgb") use "COMMON_LGB_INITLGB.ADA";
1486 1485      for Body ("common_lgb.getlgbleg") use "COMMON_LGB_GETLGBLEG.ADA";
1487 1486      for Body ("common_lgb.getlgbhdr") use "COMMON_LGB_GETLGBHDR.ADA";
1488 1487      for Body ("common_lgb.convert_leg_data") use "COMMON_LGB_CONVERT_LEG_DATA.ADA";
1489 1488      for Spec ("common_lgb") use "COMMON_LGB_.ADA";

```

File: CTP_B787_PERF_CRZINITE.ZIP\stubs.gpr (continued)

```

1490 1489      for Body ("common_lgb") use "COMMON_LGB.ADA";
1491 1490      for Spec ("cnd_dbgettext_pkg") use "CND_DBGETTYPE_PKG.ADA";
1492 1491      for Body ("cnd_dbgettext_pkg") use "CND_DBGETTYPE_PKG.ADA";
1493 1492      for Spec ("cky_wind_page_pkg") use "CKY_WIND_PAGE_PKG.ADA";
1494 1493      for Body ("cky_wind_page_pkg") use "CKY_WIND_PAGE_PKG.ADA";
1495 1494      for Spec ("cky_vnav_key_util_pkg") use "CKY_VNAV_KEY_UTIL_PKG.ADA";
1496 1495      for Body ("cky_vnav_key_util_pkg") use "CKY_VNAV_KEY_UTIL_PKG.ADA";
1497 1496      for Spec ("cky_time_date_init_page_pkg") use "CKY_TIME_DATE_INIT_PAGE_PKG.ADA";
1498 1497      for Body ("cky_time_date_init_page_pkg") use "CKY_TIME_DATE_INIT_PAGE_PKG.ADA";
1499 1498      for Spec ("cky_thrust_lim_page_pkg") use "CKY_THRUST_LIM_PAGE_PKG.ADA";
1500 1499      for Body ("cky_thrust_lim_page_pkg") use "CKY_THRUST_LIM_PAGE_PKG.ADA";
1501 1500      for Spec ("cky_takeoff_ref_page_pkg") use "CKY_TAKEOFF_REF_PAGE_PKG.ADA";
1502 1501      for Body ("cky_takeoff_ref_page_pkg") use "CKY_TAKEOFF_REF_PAGE_PKG.ADA";
1503 1502      for Spec ("cky_takeoff_ref_page_2_pkg") use "CKY_TAKEOFF_REF_PAGE_2_PKG.ADA";
1504 1503      for Body ("cky_takeoff_ref_page_2_pkg") use "CKY_TAKEOFF_REF_PAGE_2_PKG.ADA";
1505 1504      for Spec ("cky_takeoff_ref_page_1_pkg") use "CKY_TAKEOFF_REF_PAGE_1_PKG.ADA";
1506 1505      for Body ("cky_takeoff_ref_page_1_pkg") use "CKY_TAKEOFF_REF_PAGE_1_PKG.ADA";
1507 1506      for Spec ("cky_takeoff_entry_util_pkg") use "CKY_TAKEOFF_ENTRY_UTIL_PKG.ADA";
1508 1507      for Body ("cky_takeoff_entry_util_pkg") use "CKY_TAKEOFF_ENTRY_UTIL_PKG.ADA";
1509 1508      for Spec ("cky_std_entries_pkg") use "CKY_STD_ENTRIES_PKG.ADA";
1510 1509      for Body ("cky_std_entries_pkg") use "CKY_STD_ENTRIES_PKG.ADA";
1511 1510      for Spec ("cky_std_downselect_pkg") use "CKY_STD_DOWNSELECT_PKG.ADA";
1512 1511      for Body ("cky_std_downselect_pkg") use "CKY_STD_DOWNSELECT_PKG.ADA";
1513 1512      for Spec ("cky_select_wpt_page_pkg") use "CKY_SELECT_WPT_PAGE_PKG.ADA";
1514 1513      for Body ("cky_select_wpt_page_pkg") use "CKY_SELECT_WPT_PAGE_PKG.ADA";
1515 1514      for Spec ("cky_rte_page_via_pkg") use "CKY_RTE_PAGE_VIA_PKG.ADA";
1516 1515      for Body ("cky_rte_page_via_pkg") use "CKY_RTE_PAGE_VIA_PKG.ADA";
1517 1516      for Spec ("cky_rte_page_to_pkg") use "CKY_RTE_PAGE_TO_PKG.ADA";
1518 1517      for Body ("cky_rte_page_to_pkg") use "CKY_RTE_PAGE_TO_PKG.ADA";
1519 1518      for Spec ("cky_rte_page_access_pkg") use "CKY_RTE_PAGE_ACCESS_PKG.ADA";
1520 1519      for Body ("cky_rte_page_access_pkg") use "CKY_RTE_PAGE_ACCESS_PKG.ADA";
1521 1520      for Spec ("cky_rte_page_6r_pkg") use "CKY_RTE_PAGE_6R_PKG.ADA";
1522 1521      for Body ("cky_rte_page_6r_pkg") use "CKY_RTE_PAGE_6R_PKG.ADA";
1523 1522      for Spec ("cky_rte_page_2_x_pkg") use "CKY_RTE_PAGE_2_X_PKG.ADA";
1524 1523      for Body ("cky_rte_page_2_x_pkg") use "CKY_RTE_PAGE_2_X_PKG.ADA";
1525 1524      for Spec ("cky_rte_page_1_x_pkg") use "CKY_RTE_PAGE_1_X_PKG.ADA";
1526 1525      for Body ("cky_rte_page_1_x_pkg") use "CKY_RTE_PAGE_1_X_PKG.ADA";
1527 1526      for Spec ("cky_rte_legs_page_pkg") use "CKY_RTE_LEGS_PAGE_PKG.ADA";
1528 1527      for Body ("cky_rte_legs_page_pkg") use "CKY_RTE_LEGS_PAGE_PKG.ADA";
1529 1528      for Spec ("cky_rte_legs_dirto_page_pkg") use "CKY_RTE_LEGS_DIRTO_PAGE_PKG.ADA";
1530 1529      for Body ("cky_rte_legs_dirto_page_pkg") use "CKY_RTE_LEGS_DIRTO_PAGE_PKG.ADA";
1531 1530      for Body ("cky_rte_legs_common_pkg.process_waypoint") use "CKY_RTE_LEGS_COMMON_PKG_PROC_WPT.ADA";

```

File: CTP_B787_PERF_CRZINITE.ZIP\stubs.gpr (continued)

```

1532 1531      for Body ("cky_rte_legs_common_pkg.process_speed_altitude") use "CKY_RTE_LEGS_COMMON_PKG_PROC_SPALT.ADA";
1533 1532      for Spec ("cky_rte_legs_common_pkg") use "CKY_RTE_LEGS_COMMON_PKG.ADA";
1534 1533      for Body ("cky_rte_legs_common_pkg") use "CKY_RTE_LEGS_COMMON_PKG.ADA";
1535 1534      for Spec ("cky_rte_data_page_pkg") use "CKY_RTE_DATA_PAGE_PKG.ADA";
1536 1535      for Body ("cky_rte_data_page_pkg") use "CKY_RTE_DATA_PAGE_PKG.ADA";
1537 1536      for Spec ("cky_rte_common_pkg") use "CKY_RTE_COMMON_PKG.ADA";
1538 1537      for Body ("cky_rte_common_pkg") use "CKY_RTE_COMMON_PKG.ADA";
1539 1538      for Spec ("cky_ref_nav_data_page_pkg") use "CKY_REF_NAV_DATA_PAGE_PKG.ADA";
1540 1539      for Body ("cky_ref_nav_data_page_pkg") use "CKY_REF_NAV_DATA_PAGE_PKG.ADA";
1541 1540      for Spec ("cky_prov_request_pkg") use "CKY_PROV_REQUEST_PKG.ADA";
1542 1541      for Body ("cky_prov_request_pkg") use "CKY_PROV_REQUEST_PKG.ADA";
1543 1542      for Spec ("cky_progress_page_4_pkg") use "CKY_PROGRESS_PAGE_4_PKG.ADA";
1544 1543      for Body ("cky_progress_page_4_pkg") use "CKY_PROGRESS_PAGE_4_PKG.ADA";
1545 1544      for Spec ("cky_progress_page_3_pkg") use "CKY_PROGRESS_PAGE_3_PKG.ADA";
1546 1545      for Body ("cky_progress_page_3_pkg") use "CKY_PROGRESS_PAGE_3_PKG.ADA";
1547 1546      for Spec ("cky_progress_page_2_pkg") use "CKY_PROGRESS_PAGE_2_PKG.ADA";
1548 1547      for Body ("cky_progress_page_2_pkg") use "CKY_PROGRESS_PAGE_2_PKG.ADA";
1549 1548      for Spec ("cky_progress_page_1_pkg") use "CKY_PROGRESS_PAGE_1_PKG.ADA";
1550 1549      for Body ("cky_progress_page_1_pkg") use "CKY_PROGRESS_PAGE_1_PKG.ADA";
1551 1550      for Spec ("cky_pos_report_page_pkg") use "CKY_POS_REPORT_PAGE_PKG.ADA";
1552 1551      for Body ("cky_pos_report_page_pkg") use "CKY_POS_REPORT_PAGE_PKG.ADA";
1553 1552      for Spec ("cky_pos_ref_page_4_4_pkg") use "CKY_POS_REF_PAGE_4_4_PKG.ADA";
1554 1553      for Body ("cky_pos_ref_page_4_4_pkg") use "CKY_POS_REF_PAGE_4_4_PKG.ADA";
1555 1554      for Spec ("cky_pos_ref_page_3_4_pkg") use "CKY_POS_REF_PAGE_3_4_PKG.ADA";
1556 1555      for Body ("cky_pos_ref_page_3_4_pkg") use "CKY_POS_REF_PAGE_3_4_PKG.ADA";
1557 1556      for Spec ("cky_pos_ref_page_2_4_pkg") use "CKY_POS_REF_PAGE_2_4_PKG.ADA";
1558 1557      for Body ("cky_pos_ref_page_2_4_pkg") use "CKY_POS_REF_PAGE_2_4_PKG.ADA";
1559 1558      for Spec ("cky_pos_init_page_pkg") use "CKY_POS_INIT_PAGE_PKG.ADA";
1560 1559      for Body ("cky_pos_init_page_pkg") use "CKY_POS_INIT_PAGE_PKG.ADA";
1561 1560      for Spec ("cky_post_flight_plan_pkg") use "CKY_POST_FLIGHT_PLAN_PKG.ADA";
1562 1561      for Body ("cky_post_flight_plan_pkg") use "CKY_POST_FLIGHT_PLAN_PKG.ADA";
1563 1562      for Spec ("cky_perf_init_page_pkg") use "CKY_PERF_INIT_PAGE_PKG.ADA";
1564 1563      for Body ("cky_perf_init_page_pkg") use "CKY_PERF_INIT_PAGE_PKG.ADA";
1565 1564      for Spec ("cky_offpath_des_page_pkg") use "CKY_OFFPATH_DES_PAGE_PKG.ADA";
1566 1565      for Body ("cky_offpath_des_page_pkg") use "CKY_OFFPATH_DES_PAGE_PKG.ADA";
1567 1566      for Spec ("cky_nav_radio_page_pkg") use "CKY_NAV_RADIO_PAGE_PKG.ADA";
1568 1567      for Body ("cky_nav_radio_page_pkg") use "CKY_NAV_RADIO_PAGE_PKG.ADA";
1569 1568      for Spec ("cky_mode_keys_pkg") use "CKY_MODE_KEYS_PKG.ADA";
1570 1569      for Body ("cky_mode_keys_pkg") use "CKY_MODE_KEYS_PKG.ADA";
1571 1570      for Spec ("cky_maint_index_page_pkg") use "CKY_MAINT_INDEX_PAGE_PKG.ADA";
1572 1571      for Body ("cky_maint_index_page_pkg") use "CKY_MAINT_INDEX_PAGE_PKG.ADA";
1573 1572      for Spec ("cky_key_pkg") use "CKY_KEY_PKG.ADA";

```

File: CTP_B787_PERF_CRZINITE.ZIP\stubs.gpr (continued)

```

1574 1573      for Body ("cky_key_pkg") use "CKY_KEY_PKG.ADA";
1575 1574      for Spec ("cky_irs_mon_page_pkg") use "CKY_IRS_MON_PAGE_PKG.ADA";
1576 1575      for Body ("cky_irs_mon_page_pkg") use "CKY_IRS_MON_PAGE_PKG.ADA";
1577 1576      for Spec ("cky_init_ref_index_page_pkg") use "CKY_INIT_REF_INDEX_PAGE_PKG.ADA";
1578 1577      for Body ("cky_init_ref_index_page_pkg") use "CKY_INIT_REF_INDEX_PAGE_PKG.ADA";
1579 1578      for Spec ("cky_ident_page_pkg") use "CKY_IDENT_PAGE_PKG.ADA";
1580 1579      for Body ("cky_ident_page_pkg") use "CKY_IDENT_PAGE_PKG.ADA";
1581 1580      for Spec ("cky_hold_page_pkg") use "CKY_HOLD_PAGE_PKG.ADA";
1582 1581      for Body ("cky_hold_page_pkg") use "CKY_HOLD_PAGE_PKG.ADA";
1583 1582      for Spec ("cky_hold_at_page_pkg") use "CKY_HOLD_AT_PAGE_PKG.ADA";
1584 1583      for Body ("cky_hold_at_page_pkg") use "CKY_HOLD_AT_PAGE_PKG.ADA";
1585 1584      for Spec ("cky_fuel_weight_entry_pkg") use "CKY_FUEL_WEIGHT_ENTRY_PKG.ADA";
1586 1585      for Body ("cky_fuel_weight_entry_pkg") use "CKY_FUEL_WEIGHT_ENTRY_PKG.ADA";
1587 1586      for Spec ("cky_fmc_comm_page_pkg") use "CKY_FMC_COMM_PAGE_PKG.ADA";
1588 1587      for Body ("cky_fmc_comm_page_pkg") use "CKY_FMC_COMM_PAGE_PKG.ADA";
1589 1588      for Body ("cky_fix_info_page_pkg.process_ll_entry") use "CKY_FIX_INFO_PAGE_PKG_PROC_LL_ENT.ADA";
1590 1589      for Spec ("cky_fix_info_page_pkg") use "CKY_FIX_INFO_PAGE_PKG.ADA";
1591 1590      for Body ("cky_fix_info_page_pkg") use "CKY_FIX_INFO_PAGE_PKG.ADA";
1592 1591      for Body ("cky_fix_entry_pkg.verify_runway") use "CKY_FIX_ENTRY_PKG__VERIFY_RUNWAY.ADA";
1593 1592      for Body ("cky_fix_entry_pkg.verify_pb") use "CKY_FIX_ENTRY_PKG__VERIFY_PB.ADA";
1594 1593      for Body ("cky_fix_entry_pkg.search_ndb") use "CKY_FIX_ENTRY_PKG__SEARCH_NDB.ADA";
1595 1594      for Body ("cky_fix_entry_pkg.request_pb_pb") use "CKY_FIX_ENTRY_PKG__REQUEST_PB_PB.ADA";
1596 1595      for Body ("cky_fix_entry_pkg.request_pbd") use "CKY_FIX_ENTRY_PKG__REQUEST_PBD.ADA";
1597 1596      for Body ("cky_fix_entry_pkg.request_ndb") use "CKY_FIX_ENTRY_PKG__REQUEST_NDB.ADA";
1598 1597      for Body ("cky_fix_entry_pkg.request_leg") use "CKY_FIX_ENTRY_PKG__REQUEST_LEG.ADA";
1599 1598      for Body ("cky_fix_entry_pkg.get_ndb") use "CKY_FIX_ENTRY_PKG__GET_NDB.ADA";
1600 1599      for Body ("cky_fix_entry_pkg.get_bearings") use "CKY_FIX_ENTRY_PKG__GET_BEARINGS.ADA";
1601 1600      for Body ("cky_fix_entry_pkg.check_xing_pt") use "CKY_FIX_ENTRY_PKG__CHECK_XING_PT.ADA";
1602 1601      for Body ("cky_fix_entry_pkg.check_runway") use "CKY_FIX_ENTRY_PKG__CHECK_RUNWAY.ADA";
1603 1602      for Body ("cky_fix_entry_pkg.check_pb_term") use "CKY_FIX_ENTRY_PKG__CHECK_PB_TERM.ADA";
1604 1603      for Body ("cky_fix_entry_pkg.check_pb_pb") use "CKY_FIX_ENTRY_PKG__CHECK_PB_PB.ADA";
1605 1604      for Body ("cky_fix_entry_pkg.check_pbd") use "CKY_FIX_ENTRY_PKG__CHECK_PBD.ADA";
1606 1605      for Body ("cky_fix_entry_pkg.check_pb") use "CKY_FIX_ENTRY_PKG__CHECK_PB.ADA";
1607 1606      for Body ("cky_fix_entry_pkg.check_ndb") use "CKY_FIX_ENTRY_PKG__CHECK_NDB.ADA";
1608 1607      for Body ("cky_fix_entry_pkg.check_fp") use "CKY_FIX_ENTRY_PKG__CHECK_FP.ADA";
1609 1608      for Body ("cky_fix_entry_pkg.check_fix") use "CKY_FIX_ENTRY_PKG__CHECK_FIX.ADA";
1610 1609      for Body ("cky_fix_entry_pkg.check_atw") use "CKY_FIX_ENTRY_PKG__CHECK_ATW.ADA";
1611 1610      for Body ("cky_fix_entry_pkg.check_airway_term") use "CKY_FIX_ENTRY_PKG__CHECK_AIRWAY_TERM.ADA";
1612 1611      for Spec ("cky_fix_entry_pkg") use "CKY_FIX_ENTRY_PKG.ADA";
1613 1612      for Body ("cky_fix_entry_pkg") use "CKY_FIX_ENTRY_PKG.ADA";
1614 1613      for Spec ("cky_field_util_pkg") use "CKY_FIELD_UTIL_PKG.ADA";
1615 1614      for Body ("cky_field_util_pkg") use "CKY_FIELD_UTIL_PKG.ADA";

```

File: CTP_B787_PERF_CRZINITE.ZIP\stubs.gpr (continued)

```

1616 1615      for Spec ("cky_determine_rte_type_pkg") use "CKY_DETERMINE_RTE_TYPE_PKG.ADA";
1617 1616      for Body ("cky_determine_rte_type_pkg") use "CKY_DETERMINE_RTE_TYPE_PKG.ADA";
1618 1617      for Body ("cky_des_page_pkg.process_3r") use "CKY_DES_PAGE_PKG_PROCESS_3R.ADA";
1619 1618      for Spec ("cky_des_page_pkg") use "CKY_DES_PAGE_PKG.ADA";
1620 1619      for Body ("cky_des_page_pkg") use "CKY_DES_PAGE_PKG.ADA";
1621 1620      for Spec ("cky_des_forecast_page_pkg") use "CKY_DES_FORECAST_PAGE_PKG.ADA";
1622 1621      for Body ("cky_des_forecast_page_pkg") use "CKY_DES_FORECAST_PAGE_PKG.ADA";
1623 1622      for Spec ("cky_dep_arr_index_page_pkg") use "CKY_DEP_ARR_INDEX_PAGE_PKG.ADA";
1624 1623      for Body ("cky_dep_arr_index_page_pkg") use "CKY_DEP_ARR_INDEX_PAGE_PKG.ADA";
1625 1624      for Spec ("cky_dep_arr_common_pkg") use "CKY_DEP_ARR_COMMON_PKG.ADA";
1626 1625      for Body ("cky_dep_arr_common_pkg") use "CKY_DEP_ARR_COMMON_PKG.ADA";
1627 1626      for Spec ("cky_departures_page_pkg") use "CKY_DEPARTURES_PAGE_PKG.ADA";
1628 1627      for Body ("cky_departures_page_pkg") use "CKY_DEPARTURES_PAGE_PKG.ADA";
1629 1628      for Spec ("cky_crz_page_pkg") use "CKY_CRZ_PAGE_PKG.ADA";
1630 1629      for Body ("cky_crz_page_pkg") use "CKY_CRZ_PAGE_PKG.ADA";
1631 1630      for Spec ("cky_constants_pkg") use "CKY_CONSTANTS_PKG.ADA";
1632 1631      for Spec ("cky_clb_page_pkg") use "CKY_CLB_PAGE_PKG.ADA";
1633 1632      for Body ("cky_clb_page_pkg") use "CKY_CLB_PAGE_PKG.ADA";
1634 1633      for Spec ("cky_arrivals_page_pkg") use "CKY_ARRIVALS_PAGE_PKG.ADA";
1635 1634      for Body ("cky_arrivals_page_pkg") use "CKY_ARRIVALS_PAGE_PKG.ADA";
1636 1635      for Spec ("cky_approach_ref_page_pkg") use "CKY_APPROACH_REF_PAGE_PKG.ADA";
1637 1636      for Body ("cky_approach_ref_page_pkg") use "CKY_APPROACH_REF_PAGE_PKG.ADA";
1638 1637      for Spec ("cky_apf_page_2_pkg") use "CKY_APF_PAGE_2_PKG.ADA";
1639 1638      for Body ("cky_apf_page_2_pkg") use "CKY_APF_PAGE_2_PKG.ADA";
1640 1639      for Spec ("cky_apf_page_1_pkg") use "CKY_APF_PAGE_1_PKG.ADA";
1641 1640      for Body ("cky_apf_page_1_pkg") use "CKY_APF_PAGE_1_PKG.ADA";
1642 1641      for Spec ("cky_altn_plan_page_pkg") use "CKY_ALTN_PLAN_PAGE_PKG.ADA";
1643 1642      for Body ("cky_altn_plan_page_pkg") use "CKY_ALTN_PLAN_PAGE_PKG.ADA";
1644 1643      for Spec ("cky_altn_page_pkg") use "CKY_ALTN_PAGE_PKG.ADA";
1645 1644      for Body ("cky_altn_page_pkg") use "CKY_ALTN_PAGE_PKG.ADA";
1646 1645      for Spec ("cky_altn_page_access_pkg") use "CKY_ALTN_PAGE_ACCESS_PKG.ADA";
1647 1646      for Body ("cky_altn_page_access_pkg") use "CKY_ALTN_PAGE_ACCESS_PKG.ADA";
1648 1647      for Spec ("cky_altn_list_page_pkg") use "CKY_ALTN_LIST_PAGE_PKG.ADA";
1649 1648      for Body ("cky_altn_list_page_pkg") use "CKY_ALTN_LIST_PAGE_PKG.ADA";
1650 1649      for Spec ("cfp_vert_ltypes") use "CFP_VERT_LTYPES.ADA";
1651 1650      for Spec ("cfp_vert_ldata") use "CFP_VERT_LDATA.ADA";
1652 1651      for Spec ("cfp_utils") use "CFP_UTILS.ADA";
1653 1652      for Body ("cfp_utils") use "CFP_UTILS.ADA";
1654 1653      for Spec ("deactivate_fpln") use "CFP_STDBY_DEACTIVATE.ADA";
1655 1654      for Body ("deactivate_fpln") use "CFP_STDBY_DEACTIVATE.ADA";
1656 1655      for Spec ("clear_provisional") use "CFP_STDBY_CLEARPROV.ADA";
1657 1656      for Body ("clear_provisional") use "CFP_STDBY_CLEARPROV.ADA";

```

File: CTP_B787_PERF_CRZINITE.ZIP\stubs.gpr (continued)

```

1658 1657      for Spec ("cfp_route_distance_pkg") use "CFP_ROUTE_DISTANCE_PKG_.ADA";
1659 1658      for Body ("cfp_route_distance_pkg") use "CFP_ROUTE_DISTANCE_PKG.ADA";
1660 1659      for Spec ("cfp_put_hud_data_pkg") use "CFP_PUT_HUD_DATA_PKG_.ADA";
1661 1660      for Body ("cfp_put_hud_data_pkg") use "CFP_PUT_HUD_DATA_PKG.ADA";
1662 1661      for Spec ("cfp_perf_rta_ifotypes") use "CFP_PERF_RTA_IFTYPES_.ADA";
1663 1662      for Spec ("cfp_legsequence_pkg") use "CFP_LEGSEQUENCE_PKG_.ADA";
1664 1663      for Body ("cfp_legsequence_pkg") use "CFP_LEGSEQUENCE_PKG.ADA";
1665 1664      for Spec ("cfp_ldata") use "CFP_LDATA_.ADA";
1666 1665      for Spec ("cfp_key_eng_out_sid") use "CFP_KEY_ENG_OUT_SID_.ADA";
1667 1666      for Body ("cfp_key_eng_out_sid") use "CFP_KEY_ENG_OUT_SID.ADA";
1668 1667      for Spec ("cfp_keyexecprov_pkg") use "CFP_KEYEXECPROV_PKG_.ADA";
1669 1668      for Body ("cfp_keyexecprov_pkg") use "CFP_KEYEXECPROV_PKG.ADA";
1670 1669      for Spec ("cfp_intc_crs") use "CFP_INTC_CRS_PKG_.ADA";
1671 1670      for Body ("cfp_intc_crs") use "CFP_INTC_CRS_PKG.ADA";
1672 1671      for Spec ("cfp_hold_pkg") use "CFP_HOLD_PKG_.ADA";
1673 1672      for Body ("cfp_hold_pkg") use "CFP_HOLD_PKG.ADA";
1674 1673      for Spec ("cfp_halfway_rule_pkg") use "CFP_HALFWAY_RULE_PKG_.ADA";
1675 1674      for Body ("cfp_halfway_rule_pkg") use "CFP_HALFWAY_RULE_PKG.ADA";
1676 1675      for Spec ("cfp_fp_io_iface_pkg") use "CFP_FP_IO_IFACE_PKG_.ADA";
1677 1676      for Body ("cfp_fp_io_iface_pkg") use "CFP_FP_IO_IFACE_PKG.ADA";
1678 1677      for Spec ("cfp_fpwinds") use "CFP_FPWINDS_.ADA";
1679 1678      for Body ("cfp_fpwinds") use "CFP_FPWINDS.ADA";
1680 1679      for Spec ("cfp_fpviallxing") use "CFP_FPVIALLXING_.ADA";
1681 1680      for Body ("cfp_fpviallxing") use "CFP_FPVIALLXING.ADA";
1682 1681      for Spec ("cfp_fpviadirect_pkg") use "CFP_FPVIADIRECT_PKG_.ADA";
1683 1682      for Body ("cfp_fpviadirect_pkg") use "CFP_FPVIADIRECT_PKG.ADA";
1684 1683      for Spec ("cfp_fpviaawy") use "CFP_FPVIAAWY_.ADA";
1685 1684      for Body ("cfp_fpviaawy") use "CFP_FPVIAAWY.ADA";
1686 1685      for Spec ("cfp_fpvert_pkg") use "CFP_FPVERT_PKG_.ADA";
1687 1686      for Body ("cfp_fpvert_pkg") use "CFP_FPVERT_PKG.ADA";
1688 1687      for Spec ("cfp_fpsrchiaf") use "CFP_FPSRCHIAF_.ADA";
1689 1688      for Body ("cfp_fpsrchiaf") use "CFP_FPSRCHIAF.ADA";
1690 1689      for Spec ("cfp_fpspdtrans") use "CFP_FPSPDTRANS_.ADA";
1691 1690      for Body ("cfp_fpspdtrans") use "CFP_FPSPDTRANS.ADA";
1692 1691      for Spec ("cfp_fpsmthcrzalt_pkg") use "CFP_FPSMTHCRZALT_PKG_.ADA";
1693 1692      for Body ("cfp_fpsmthcrzalt_pkg") use "CFP_FPSMTHCRZALT_PKG.ADA";
1694 1693      for Spec ("cfp_fprtecopy_pkg") use "CFP_FPRTECOPY_PKG_.ADA";
1695 1694      for Body ("cfp_fprtecopy_pkg") use "CFP_FPRTECOPY_PKG.ADA";
1696 1695      for Spec ("cfp_fprta_pkg") use "CFP_FPRTA_PKG_.ADA";
1697 1696      for Body ("cfp_fprta_pkg") use "CFP_FPRTA_PKG.ADA";
1698 1697      for Spec ("cfp_fpperfmode_pkg") use "CFP_FPPERFMODE_PKG_.ADA";
1699 1698      for Body ("cfp_fpperfmode_pkg") use "CFP_FPPERFMODE_PKG.ADA";

```

File: CTP_B787_PERF_CRZINITE.ZIP\stubs.gpr (continued)

1700	1699	for Spec ("cfp_fpoktodelete") use "CFP_FPOKTODELETE_.ADA";
1701	1700	for Body ("cfp_fpoktodelete") use "CFP_FPOKTODELETE.ADA";
1702	1701	for Spec ("cfp_fpoffset_pkg") use "CFP_FPOFFSET_PKG_.ADA";
1703	1702	for Body ("cfp_fpoffset_pkg") use "CFP_FPOFFSET_PKG.ADA";
1704	1703	for Spec ("cfp_fpnewalt_pkg") use "CFP_FPNEWALT_PKG_.ADA";
1705	1704	for Body ("cfp_fpnewalt_pkg") use "CFP_FPNEWALT_PKG.ADA";
1706	1705	for Spec ("cfp_fpholdtime") use "CFP_FPHOLDTIME_.ADA";
1707	1706	for Body ("cfp_fpholdtime") use "CFP_FPHOLDTIME.ADA";
1708	1707	for Spec ("cfp_fpexec_vert") use "CFP_FPEXEC_VERT_.ADA";
1709	1708	for Body ("cfp_fpexec_vert") use "CFP_FPEXEC_VERT.ADA";
1710	1709	for Spec ("cfp_fpexecute") use "CFP_FPEXECUTE_.ADA";
1711	1710	for Body ("cfp_fpexecute") use "CFP_FPEXECUTE.ADA";
1712	1711	for Spec ("cfp_fpefis") use "CFP_FPEFIS_.ADA";
1713	1712	for Body ("cfp_fpefis") use "CFP_FPEFIS.ADA";
1714	1713	for Spec ("cfp_fpdicon") use "CFP_FPDICON_.ADA";
1715	1714	for Body ("cfp_fpdicon") use "CFP_FPDICON.ADA";
1716	1715	for Spec ("cfp_fpdeparr_pkg") use "CFP_FPDEPARR_PKG_.ADA";
1717	1716	for Body ("cfp_fpdeparr_pkg") use "CFP_FPDEPARR_PKG.ADA";
1718	1717	for Spec ("cfp_fpdelvialeg") use "CFP_FPDELVIALEG_.ADA";
1719	1718	for Body ("cfp_fpdelvialeg") use "CFP_FPDELVIALEG.ADA";
1720	1719	for Spec ("cfp_fpdelete") use "CFP_FPDELETE_.ADA";
1721	1720	for Body ("cfp_fpdelete") use "CFP_FPDELETE.ADA";
1722	1721	for Spec ("cfp_fpckvprofile") use "CFP_FPCKVPROFILE_.ADA";
1723	1722	for Body ("cfp_fpckvprofile") use "CFP_FPCKVPROFILE.ADA";
1724	1723	for Spec ("cfp_fpblavia") use "CFP_FPBLDIA_.ADA";
1725	1724	for Body ("cfp_fpblavia") use "CFP_FPBLDIA.ADA";
1726	1725	for Spec ("cfp_fpblvfrleg") use "CFP_FPBLDVFRLEG_.ADA";
1727	1726	for Body ("cfp_fpblvfrleg") use "CFP_FPBLDVFRLEG.ADA";
1728	1727	for Spec ("cfp_fpblvfrleg") use "CFP_FPBLDVFRLEG_.ADA";
1729	1728	for Body ("cfp_fpblvfrleg") use "CFP_FPBLDVFRLEG.ADA";
1730	1729	for Spec ("cfp_fpblvfrleg") use "CFP_FPBLDVFRLEG_.ADA";
1731	1730	for Body ("cfp_fpblvfrleg") use "CFP_FPBLDVFRLEG.ADA";
1732	1731	for Spec ("cfp_fpblvfrleg") use "CFP_FPBLDVFRLEG_.ADA";
1733	1732	for Body ("cfp_fpblvfrleg") use "CFP_FPBLDVFRLEG.ADA";
1734	1733	for Spec ("cfp_fpblvfrleg") use "CFP_FPBLDVFRLEG_.ADA";
1735	1734	for Body ("cfp_fpblvfrleg") use "CFP_FPBLDVFRLEG.ADA";
1736	1735	for Spec ("cfp_fpawysrch_pkg") use "CFP_FPAWYSRCH_PKG_.ADA";
1737	1736	for Body ("cfp_fpawysrch_pkg") use "CFP_FPAWYSRCH_PKG.ADA";
1738	1737	for Spec ("cfp_fpawysrch_pkg") use "CFP_FPAWYSRCH_PKG_.ADA";
1739	1738	for Body ("cfp_fpawysrch_pkg") use "CFP_FPAWYSRCH_PKG.ADA";
1740	1739	for Spec ("cfp_fpaltdelete") use "CFP_FPALTDELETE_.ADA";
1741	1740	for Body ("cfp_fpaltdelete") use "CFP_FPALTDELETE.ADA";

File: CTP_B787_PERF_CRZINITE.ZIP\stubs.gpr (continued)

```

1742 1741      for Spec ("cfp_fpaltcnstr") use "CFP_FPALTCNSTR.ADA";
1743 1742      for Body ("cfp_fpaltcnstr") use "CFP_FPALTCNSTR.ADA";
1744 1743      for Spec ("cfp_fpactivvert") use "CFP_FPACTVVERT.ADA";
1745 1744      for Body ("cfp_fpactivvert") use "CFP_FPACTVVERT.ADA";
1746 1745      for Spec ("cfp_fpactivate") use "CFP_FPACTIVATE.ADA";
1747 1746      for Body ("cfp_fpactivate") use "CFP_FPACTIVATE.ADA";
1748 1747      for Spec ("cfp_fpabeampts") use "CFP_FPABEAMPTS.ADA";
1749 1748      for Body ("cfp_fpabeampts") use "CFP_FPABEAMPTS.ADA";
1750 1749      for Spec ("cfp_flight_complete_pkg") use "CFP_FLIGHT_COMPLETE_PKG.ADA";
1751 1750      for Body ("cfp_flight_complete_pkg") use "CFP_FLIGHT_COMPLETE_PKG.ADA";
1752 1751      for Spec ("cfp_findllxingtpt_pkg") use "CFP_FINDLLXINGPT_PKG.ADA";
1753 1752      for Body ("cfp_findllxingtpt_pkg") use "CFP_FINDLLXINGPT_PKG.ADA";
1754 1753      for Spec ("cfp_delete_act_fpln") use "CFP_DELETE_ACT_FPLN.ADA";
1755 1754      for Body ("cfp_delete_act_fpln") use "CFP_DELETE_ACT_FPLN.ADA";
1756 1755      for Spec ("cfp_createprov") use "CFP_CREATEPROV.ADA";
1757 1756      for Body ("cfp_createprov") use "CFP_CREATEPROV.ADA";
1758 1757      for Spec ("cfp_co_route_pkg") use "CFP_CO_ROUTE_PKG.ADA";
1759 1758      for Body ("cfp_co_route_pkg") use "CFP_CO_ROUTE_PKG.ADA";
1760 1759      for Spec ("cfp_computeabm") use "CFP_COMPUTEABM.ADA";
1761 1760      for Body ("cfp_computeabm") use "CFP_COMPUTEABM.ADA";
1762 1761      for Spec ("cfp_clearrttes") use "CFP_CLEARRTTES.ADA";
1763 1762      for Body ("cfp_clearrttes") use "CFP_CLEARRTTES.ADA";
1764 1763      for Spec ("cfp_clearprov") use "CFP_CLEARPROV.ADA";
1765 1764      for Body ("cfp_clearprov") use "CFP_CLEARPROV.ADA";
1766 1765      for Spec ("cfp_chkoffdes") use "CFP_CHKOFFDES.ADA";
1767 1766      for Body ("cfp_chkoffdes") use "CFP_CHKOFFDES.ADA";
1768 1767      for Body ("cfp_cdurtedist") use "CFP_CDURTEDIST.ADA";
1769 1768      for Spec ("cfp_cdufp_pkg") use "CFP_CDUFP_PKG.ADA";
1770 1769      for Body ("cfp_cdufp_pkg") use "CFP_CDUFP_PKG.ADA";
1771 1770      for Spec ("cfp_cdualtintv_pkg") use "CFP_CDUALTINTV_PKG.ADA";
1772 1771      for Body ("cfp_cdualtintv_pkg") use "CFP_CDUALTINTV_PKG.ADA";
1773 1772      for Spec ("cfp_calcintrsect_pkg") use "CFP_CALCINTRSECT_PKG.ADA";
1774 1773      for Body ("cfp_calcintrsect_pkg") use "CFP_CALCINTRSECT_PKG.ADA";
1775 1774      for Spec ("cfp_cabin_pressure") use "CFP_CABIN_PRESSURE.ADA";
1776 1775      for Body ("cfp_cabin_pressure") use "CFP_CABIN_PRESSURE.ADA";
1777 1776      for Spec ("cfp_awytoawyintc_pkg") use "CFP_AWYTOAWYINTC_PKG.ADA";
1778 1777      for Body ("cfp_awytoawyintc_pkg") use "CFP_AWYTOAWYINTC_PKG.ADA";
1779 1778      for Spec ("cfp_alternates_pkg") use "CFP_ALTERNATES_PKG.ADA";
1780 1779      for Body ("cfp_alternates_pkg") use "CFP_ALTERNATES_PKG.ADA";
1781 1780      for Spec ("cex_transition_pkg") use "CEX_TRANSITION_PKG.ADA";
1782 1781      for Body ("cex_transition_pkg") use "CEX_TRANSITION_PKG.ADA";
1783 1782      for Spec ("cex_lgb_utils_pkg") use "CEX_LGB_UTILS_PKG.ADA";

```


File: CTP_B787_PERF_CRZINITE.ZIP\stubs.gpr (continued)

```

1784 1783      for Body ("cex_lgb_utils_pkg") use "CEX_LGB_UTILS_PKG.ADA";
1785 1784      for Spec ("cex_exec_ldata") use "CEX_EXEC_LDATA_.ADA";
1786 1785      for Spec ("cex_cinit_pkg") use "CEX_CINIT_PKG.ADA";
1787 1786      for Body ("cex_cinit_pkg") use "CEX_CINIT_PKG.ADA";
1788 1787      for Spec ("cex_cdu_util_pkg") use "CEX_CDU_UTIL_PKG_.ADA";
1789 1788      for Body ("cex_cdu_util_pkg") use "CEX_CDU_UTIL_PKG.ADA";
1790 1789      for Spec ("cex_cdu_pkg") use "CEX_CDU_PKG_.ADA";
1791 1790      for Body ("cex_cdu_pkg") use "CEX_CDU_PKG.ADA";
1792 1791      for Spec ("cex_cdu_enq_event_pkg") use "CEX_CDU_ENQ_EVENT_PKG_.ADA";
1793 1792      for Body ("cex_cdu_enq_event_pkg") use "CEX_CDU_ENQ_EVENT_PKG.ADA";
1794 1793      for Spec ("cex_cdu_cyclic_pkg") use "CEX_CDU_CYCLIC_PKG_.ADA";
1795 1794      for Body ("cex_cdu_cyclic_pkg") use "CEX_CDU_CYCLIC_PKG.ADA";
1796 1795      for Spec ("cex_cdusrv") use "CEX_CDUSRV_.ADA";
1797 1796      for Body ("cex_cdusrv") use "CEX_CDUSRV.ADA";
1798 1797      for Spec ("cex_cdurtdnld") use "CEX_CDURTDNLD_.ADA";
1799 1798      for Body ("cex_cdurtdnld") use "CEX_CDURTDNLD.ADA";
1800 1799      for Spec ("cex_cduefimap_proc") use "CEX_CDUEFIMAP_PROC_.ADA";
1801 1800      for Body ("cex_cduefimap_proc") use "CEX_CDUEFIMAP_PROC.ADA";
1802 1801      for Spec ("cex_bite_interface_pkg") use "CEX_BITE_INTERFACE_PKG_.ADA";
1803 1802      for Body ("cex_bite_interface_pkg") use "CEX_BITE_INTERFACE_PKG.ADA";
1804 1803      for Spec ("cdm_disp_buffer_mon_pkg") use "CDM_DISP_BUFFER_MON_PKG_.ADA";
1805 1804      for Body ("cdm_disp_buffer_mon_pkg") use "CDM_DISP_BUFFER_MON_PKG.ADA";
1806 1805      for Spec ("cdl_wind_uplink_pkg") use "CDL_WIND_UPLINK_PKG_.ADA";
1807 1806      for Body ("cdl_wind_uplink_pkg") use "CDL_WIND_UPLINK_PKG.ADA";
1808 1807      for Spec ("cdl_wind_request_downlink_pkg") use "CDL_WIND_REQUEST_DOWNLINK_PKG_.ADA";
1809 1808      for Body ("cdl_wind_request_downlink_pkg") use "CDL_WIND_REQUEST_DOWNLINK_PKG.ADA";
1810 1809      for Body ("cdl_wind_uplink_pkg.uplink_wd_wm") use "CDL_UPLINK_WD_WM_SEP.ADA";
1811 1810      for Body ("cdl_takeoff_uplink_pkg.uplink_rw") use "CDL_UPLINK_RW_SEP.ADA";
1812 1811      for Body ("cdl_request_uplink_pkg.uplink_req") use "CDL_UPLINK_REQ_SEP.ADA";
1813 1812      for Body ("cdl_wind_uplink_pkg.uplink_pwx") use "CDL_UPLINK_PWX_SEP.ADA";
1814 1813      for Body ("cdl_position_trigger_uplink_pkg.uplink_pos_trigger") use "CDL_UPLINK_POS_TRIGGER_SEP.ADA";
1815 1814      for Body ("cdl_takeoff_uplink_pkg.uplink_ldi") use "CDL_UPLINK_LDI_SEP.ADA";
1816 1815      for Body ("cdl_wind_uplink_pkg.uplink_dd") use "CDL_UPLINK_DD_SEP.ADA";
1817 1816      for Body ("cdl_takeoff_uplink_pkg.uplink_cg") use "CDL_UPLINK_CG_SEP.ADA";
1818 1817      for Body ("cdl_alternate_uplink_pkg.uplink_alt") use "CDL_UPLINK_ALT_SEP.ADA";
1819 1818      for Body ("cdl_alternate_uplink_pkg.uplink_alt_inhibit") use "CDL_UPLINK_ALT_INHIBIT_SEP.ADA";
1820 1819      for Body ("cdl_alternate_uplink_pkg.uplink_alt_flight_list") use "CDL_UPLINK_ALT_FLIGHT_LIST_SEP.ADA";
1821 1820      for Body ("cdl_alternate_uplink_pkg.uplink_alt_company_preferred") use "CDL_UPLINK_ALT_COMPANY_PREFERRED_SEP.ADA";
1822 1821      for Spec ("cdl_takeoff_uplink_pkg") use "CDL_TAKEOFF_UPLINK_PKG_.ADA";
1823 1822      for Body ("cdl_takeoff_uplink_pkg") use "CDL_TAKEOFF_UPLINK_PKG.ADA";
1824 1823      for Spec ("cdl_takeoff_downlink_pkg") use "CDL_TAKEOFF_DOWNLINK_PKG_.ADA";
1825 1824      for Body ("cdl_takeoff_downlink_pkg") use "CDL_TAKEOFF_DOWNLINK_PKG.ADA";

```

File: CTP_B787_PERF_CRZINITE.ZIP\stubs.gpr (continued)

```

1826 1825      for Body ("cdl_flight_plan_uplink_pkg.step") use "CDL_STEP_SEP.ADA";
1827 1826      for Body ("cdl_flight_plan_uplink_pkg.speed_altitude") use "CDL_SPEED_ALTITUDE_SEP.ADA";
1828 1827      for Body ("cdl_takeoff_uplink_pkg.search_ldi_rwy") use "CDL_SEARCH_LDI_RWY_SEP.ADA";
1829 1828      for Body ("cdl_fix_utilities_pkg.search_fpln_fix") use "CDL_SEARCH_FPLN_FIX_SEP.ADA";
1830 1829      for Spec ("cdl_search_fix_list_pkg") use "CDL_SEARCH_FIX_LIST_PKG.ADA";
1831 1830      for Body ("cdl_search_fix_list_pkg") use "CDL_SEARCH_FIX_LIST_PKG.ADA";
1832 1831      for Body ("cdl_flight_plan_uplink_pkg.runway") use "CDL_RUNWAY_SEP.ADA";
1833 1832      for Spec ("cdl_response_downlink_pkg") use "CDL_RESPONSE_DOWNLINK_PKG.ADA";
1834 1833      for Body ("cdl_response_downlink_pkg") use "CDL_RESPONSE_DOWNLINK_PKG.ADA";
1835 1834      for Spec ("cdl_request_uplink_pkg") use "CDL_REQUEST_UPLINK_PKG.ADA";
1836 1835      for Body ("cdl_request_uplink_pkg") use "CDL_REQUEST_UPLINK_PKG.ADA";
1837 1836      for Body ("cdl_position_trigger_uplink_pkg.reporting_points") use "CDL_REPORTING_POINTS_SEP.ADA";
1838 1837      for Spec ("cdl_rejection_downlink_pkg") use "CDL_REJECTION_DOWNLINK_PKG.ADA";
1839 1838      for Body ("cdl_rejection_downlink_pkg") use "CDL_REJECTION_DOWNLINK_PKG.ADA";
1840 1839      for Spec ("cdl_position_trigger_uplink_pkg") use "CDL_POSITION_TRIGGER_UPLINK_PKG.ADA";
1841 1840      for Body ("cdl_position_trigger_uplink_pkg") use "CDL_POSITION_TRIGGER_UPLINK_PKG.ADA";
1842 1841      for Spec ("cdl_position_downlink_pkg") use "CDL_POSITION_DOWNLINK_PKG.ADA";
1843 1842      for Body ("cdl_position_downlink_pkg") use "CDL_POSITION_DOWNLINK_PKG.ADA";
1844 1843      for Spec ("cdl_periodic_pkg") use "CDL_PERIODIC_PKG.ADA";
1845 1844      for Body ("cdl_periodic_pkg") use "CDL_PERIODIC_PKG.ADA";
1846 1845      for Spec ("cdl_perf_init_uplink_pkg") use "CDL_PERF_INIT_UPLINK_PKG.ADA";
1847 1846      for Body ("cdl_perf_init_uplink_pkg") use "CDL_PERF_INIT_UPLINK_PKG.ADA";
1848 1847      for Spec ("cdl_perf_init_downlink_pkg") use "CDL_PERF_INIT_DOWNLINK_PKG.ADA";
1849 1848      for Body ("cdl_perf_init_downlink_pkg") use "CDL_PERF_INIT_DOWNLINK_PKG.ADA";
1850 1849      for Body ("cdl_flight_plan_uplink_pkg.origin_destination") use "CDL_ORIGIN_DESTINATION_SEP.ADA";
1851 1850      for Body ("cdl_flight_plan_uplink_pkg.offset") use "CDL_OFFSET_SEP.ADA";
1852 1851      for Body ("cdl_flight_plan_uplink_pkg.new_fpn") use "CDL_NEW_FPN_SEP.ADA";
1853 1852      for Spec ("cdl_loc_buffer_data") use "CDL_LOC_BUFFER_DATA.ADA";
1854 1853      for Body ("cdl_flight_plan_uplink_pkg.lat_lon_crossing") use "CDL_LAT_LON_CROSSING_SEP.ADA";
1855 1854      for Body ("cdl_flight_plan_uplink_pkg.intercept_course_from") use "CDL_INTERCEPT_COURSE_FROM_SEP.ADA";
1856 1855      for Spec ("cdl_init_token_rules_pkg") use "CDL_INIT_TOKEN_RULES_PKG.ADA";
1857 1856      for Body ("cdl_init_token_rules_pkg") use "CDL_INIT_TOKEN_RULES_PKG.ADA";
1858 1857      for Body ("cdl_takeoff_uplink_pkg.initialize_standard_data") use "CDL_INITIALIZE_STANDARD_DATA_SEP.ADA";
1859 1858      for Body ("cdl_flight_plan_uplink_pkg.hold") use "CDL_HOLD_SEP.ADA";
1860 1859      for Spec ("cdl_global_data") use "CDL_GLOBAL_DATA.ADA";
1861 1860      for Spec ("cdl_get_element_token_pkg") use "CDL_GET_ELEMENT_TOKEN_PKG.ADA";
1862 1861      for Body ("cdl_get_element_token_pkg") use "CDL_GET_ELEMENT_TOKEN_PKG.ADA";
1863 1862      for Spec ("cdl_fpx_init_pkg") use "CDL_FPX_INIT_PKG.ADA";
1864 1863      for Body ("cdl_fpx_init_pkg") use "CDL_FPX_INIT_PKG.ADA";
1865 1864      for Spec ("cdl_fpxrectype_data") use "CDL_FPXRECTYPE_DATA.ADA";
1866 1865      for Spec ("cdl_fpn_uplink_transition_pkg") use "CDL_FPN_UPLINK_TRANSITION_PKG.ADA";
1867 1866      for Body ("cdl_fpn_uplink_transition_pkg") use "CDL_FPN_UPLINK_TRANSITION_PKG.ADA";

```

File: CTP_B787_PERF_CRZINITE.ZIP\stubs.gpr (continued)

```

1868 1867      for Body ("cdl_flight_plan_uplink_pkg.fpn_uplink") use "CDL_FPN_UPLINK_SEP.ADA";
1869 1868      for Spec ("cdl_flight_plan_uplink_pkg") use "CDL_FLIGHT_PLAN_UPLINK_PKG_.ADA";
1870 1869      for Body ("cdl_flight_plan_uplink_pkg") use "CDL_FLIGHT_PLAN_UPLINK_PKG.ADA";
1871 1870      for Body ("cdl_flight_plan_uplink_pkg.flight_plan_element") use "CDL_FLIGHT_PLAN_ELEMENT_SEP.ADA";
1872 1871      for Spec ("cdl_flight_plan_downlink_pkg") use "CDL_FLIGHT_PLAN_DOWNLINK_PKG_.ADA";
1873 1872      for Body ("cdl_flight_plan_downlink_pkg") use "CDL_FLIGHT_PLAN_DOWNLINK_PKG.ADA";
1874 1873      for Body ("cdl_flight_plan_uplink_pkg.flight_number") use "CDL_FLIGHT_NUMBER_SEP.ADA";
1875 1874      for Spec ("cdl_fix_utilities_pkg") use "CDL_FIX_UTILITIES_PKG_.ADA";
1876 1875      for Body ("cdl_fix_utilities_pkg") use "CDL_FIX_UTILITIES_PKG.ADA";
1877 1876      for Spec ("cdl_executive_pkg") use "CDL_EXECUTIVE_PKG_.ADA";
1878 1877      for Body ("cdl_executive_pkg") use "CDL_EXECUTIVE_PKG.ADA";
1879 1878      for Spec ("cdl_error_codes_data") use "CDL_ERROR_CODES_DATA_.ADA";
1880 1879      for Body ("cdl_wind_request_downlink_pkg.downlink_req_wind") use "CDL_DOWNLINK_REQ_WIND_SEP.ADA";
1881 1880      for Body ("cdl_request_uplink_pkg.downlink_req") use "CDL_DOWNLINK_REQ_SEP.ADA";
1882 1881      for Body ("cdl_alternates_request_downlink_pkg.downlink_req_alternates") use "CDL_DOWNLINK_REQ_ALTERNATES_SEP.ADA";
1883 1882      for Spec ("cdl_downlink_buffer_data") use "CDL_DOWNLINK_BUFFER_DATA_.ADA";
1884 1883      for Spec ("cdl_dlupendawy_data") use "CDL_DLUPENDAWY_DATA_.ADA";
1885 1884      for Spec ("cdl_dltermtype_types") use "CDL_DLTERMTYPE_TYPES_.ADA";
1886 1885      for Spec ("cdl_dlsrchtypes_types") use "CDL_DLSRCHTYPES_TYPES_.ADA";
1887 1886      for Body ("cdl_flight_plan_uplink_pkg.direct_fix") use "CDL_DIRECT_FIX_SEP.ADA";
1888 1887      for Body ("cdl_flight_plan_uplink_pkg.departure_arrival") use "CDL_DEPARTURE_ARRIVAL_SEP.ADA";
1889 1888      for Spec ("cdl_demand_pkg") use "CDL_DEMAND_PKG_.ADA";
1890 1889      for Body ("cdl_demand_pkg") use "CDL_DEMAND_PKG.ADA";
1891 1890      for Spec ("cdl_data_link_init_pkg") use "CDL_DATA_LINK_INIT_PKG_.ADA";
1892 1891      for Body ("cdl_data_link_init_pkg") use "CDL_DATA_LINK_INIT_PKG.ADA";
1893 1892      for Spec ("cdl_datalinerec_types") use "CDL_DATALINEREC_TYPES_.ADA";
1894 1893      for Body ("cdl_flight_plan_uplink_pkg.cruise_speed_segment") use "CDL_CRUISE_SPEED_SEGMENT_SEP.ADA";
1895 1894      for Spec ("cdl_constants_data") use "CDL_CONSTANTS_DATA_.ADA";
1896 1895      for Body ("cdl_flight_plan_uplink_pkg.company_route") use "CDL_COMPANY_ROUTE_SEP.ADA";
1897 1896      for Spec ("cdl_check_sequence_number_pkg") use "CDL_CHECK_SEQUENCE_NUMBER_PKG_.ADA";
1898 1897      for Body ("cdl_check_sequence_number_pkg") use "CDL_CHECK_SEQUENCE_NUMBER_PKG.ADA";
1899 1898      for Body ("cdl_fix_utilities_pkg.check_pbd") use "CDL_CHECK_PBD_SEP.ADA";
1900 1899      for Body ("cdl_fix_utilities_pkg.check_lat_lon") use "CDL_CHECK_LAT_LON_SEP.ADA";
1901 1900      for Spec ("cdl_check_fpn_uplink_status_pkg") use "CDL_CHECK_FPN_UPLINK_STATUS_PKG_.ADA";
1902 1901      for Body ("cdl_check_fpn_uplink_status_pkg") use "CDL_CHECK_FPN_UPLINK_STATUS_PKG.ADA";
1903 1902      for Body ("cdl_fix_utilities_pkg.check_fix_format") use "CDL_CHECK_FIX_FORMAT_SEP.ADA";
1904 1903      for Body ("cdl_fix_utilities_pkg.check_crs_int") use "CDL_CHECK_CRS_INT_SEP.ADA";
1905 1904      for Body ("cdl_flight_plan_uplink_pkg.check_altitude") use "CDL_CHECK_ALTITUDE_SEP.ADA";
1906 1905      for Body ("cdl_flight_plan_uplink_pkg.approach") use "CDL_APPROACH_SEP.ADA";
1907 1906      for Body ("cdl_wind_request_downlink_pkg.append_xq") use "CDL_APPEND_XQ_SEP.ADA";
1908 1907      for Spec ("cdl_append_ts_ga_ca_pkg") use "CDL_APPEND_TS_GA_CA_PKG_.ADA";
1909 1908      for Body ("cdl_append_ts_ga_ca_pkg") use "CDL_APPEND_TS_GA_CA_PKG.ADA";

```

File: CTP_B787_PERF_CRZINITE.ZIP\stubs.gpr (continued)

```

1910 1909      for Spec ("cdl_append_sp_pkg") use "CDL_APPEND_SP_PKG.ADA";
1911 1910      for Body ("cdl_append_sp_pkg") use "CDL_APPEND_SP_PKG.ADA";
1912 1911      for Spec ("cdl_append_rx_pkg") use "CDL_APPEND_RX_PKG.ADA";
1913 1912      for Body ("cdl_append_rx_pkg") use "CDL_APPEND_RX_PKG.ADA";
1914 1913      for Spec ("cdl_append_rp_pkg") use "CDL_APPEND_RP_PKG.ADA";
1915 1914      for Body ("cdl_append_rp_pkg") use "CDL_APPEND_RP_PKG.ADA";
1916 1915      for Spec ("cdl_append_px_pkg") use "CDL_APPEND_PX_PKG.ADA";
1917 1916      for Body ("cdl_append_px_pkg") use "CDL_APPEND_PX_PKG.ADA";
1918 1917      for Spec ("cdl_append_fn_pkg") use "CDL_APPEND_FN_PKG.ADA";
1919 1918      for Body ("cdl_append_fn_pkg") use "CDL_APPEND_FN_PKG.ADA";
1920 1919      for Spec ("cdl_append_error_pkg") use "CDL_APPEND_ERROR_PKG.ADA";
1921 1920      for Body ("cdl_append_error_pkg") use "CDL_APPEND_ERROR_PKG.ADA";
1922 1921      for Spec ("cdl_append_dt_pkg") use "CDL_APPEND_DT_PKG.ADA";
1923 1922      for Body ("cdl_append_dt_pkg") use "CDL_APPEND_DT_PKG.ADA";
1924 1923      for Body ("cdl_wind_request_downlink_pkg.append_dq") use "CDL_APPEND_DQ_SEP.ADA";
1925 1924      for Body ("cdl_alternates_request_downlink_pkg.append_ax") use "CDL_APPEND_AX_SEP.ADA";
1926 1925      for Body ("cdl_alternates_request_downlink_pkg.append_aq") use "CDL_APPEND_AQ_SEP.ADA";
1927 1926      for Spec ("cdl_aoc_message_recording_pkg") use "CDL_AOC_MESSAGE_RECORDING_PKG.ADA";
1928 1927      for Body ("cdl_aoc_message_recording_pkg") use "CDL_AOC_MESSAGE_RECORDING_PKG.ADA";
1929 1928      for Spec ("cdl_alternate_uplink_pkg") use "CDL_ALTERNATES_UPLINK_PKG.ADA";
1930 1929      for Body ("cdl_alternate_uplink_pkg") use "CDL_ALTERNATES_UPLINK_PKG.ADA";
1931 1930      for Spec ("cdl_alternates_request_downlink_pkg") use "CDL_ALTERNATES_REQUEST_DOWNLINK_PKG.ADA";
1932 1931      for Body ("cdl_alternates_request_downlink_pkg") use "CDL_ALTERNATES_REQUEST_DOWNLINK_PKG.ADA";
1933 1932      for Body ("cdl_flight_plan_uplink_pkg.along_track_waypoint") use "CDL_ALONG_TRACK_WAYPOINT_SEP.ADA";
1934 1933      for Body ("cdl_flight_plan_uplink_pkg.airway") use "CDL_AIRWAY_SEP.ADA";
1935 1934      for Spec ("cdk_takeoff_ref_page_lftypes") use "CDK TAKEOFF_REF_PAGE_LFTYPES.ADA";
1936 1935      for Spec ("cdk_takeoff_ref_page_2_lftypes") use "CDK TAKEOFF_REF_PAGE_2_LFTYPES.ADA";
1937 1936      for Spec ("cdk_takeoff_ref_page_1_lftypes") use "CDK TAKEOFF_REF_PAGE_1_LFTYPES.ADA";
1938 1937      for Spec ("cdk_takeoff_lftypes") use "CDK TAKEOFF_LFTYPES.ADA";
1939 1938      for Spec ("cdk_refnav_lftypes") use "CDK_REFNAV_LFTYPES.ADA";
1940 1939      for Spec ("cdk_progress_lftypes") use "CDK_PROGRESS_LFTYPES.ADA";
1941 1940      for Spec ("cdk_offset_valid_pkg") use "CDK_OFFSET_VALID_PKG.ADA";
1942 1941      for Body ("cdk_offset_valid_pkg") use "CDK_OFFSET_VALID_PKG.ADA";
1943 1942      for Spec ("cdk_key_reaction_lfdata") use "CDK_KEY_REACTION_LFDATA.ADA";
1944 1943      for Spec ("cdk_entry_lftypes") use "CDK_ENTRY_LFTYPES.ADA";
1945 1944      for Spec ("cdk_display_format_lftypes") use "CDK_DISPLAY_FORMAT_LFTYPES.ADA";
1946 1945      for Spec ("cdk_des_page_lftypes") use "CDK_DES_PAGE_LFTYPES.ADA";
1947 1946      for Spec ("cdk_dep_arr_support_pkg") use "CDK_DEP_ARR_SUPPORT_PKG.ADA";
1948 1947      for Body ("cdk_dep_arr_support_pkg") use "CDK_DEP_ARR_SUPPORT_PKG.ADA";
1949 1948      for Spec ("data_conversion_pkg") use "CDK_DATA_CONVERSION_PKG.ADA";
1950 1949      for Body ("data_conversion_pkg") use "CDK_DATA_CONVERSION_PKG.ADA";
1951 1950      for Spec ("cdk_datalink_lftypes") use "CDK_DATA_LINK_LFTYPES.ADA";

```

File: CTP_B787_PERF_CRZINITE.ZIP\stubs.gpr (continued)

```

1952 1951      for Spec ("cdk_constants_pkg") use "CDK_CONSTANTS_PKG.ADA";
1953 1952      for Spec ("cdk_altn_airports_candidates_pkg") use "CDK_ALTN_AIRPORTS_CANDIDATES_PKG.ADA";
1954 1953      for Body ("cdk_altn_airports_candidates_pkg") use "CDK_ALTN_AIRPORTS_CANDIDATES_PKG.ADA";
1955 1954      for Spec ("cdkm_thrust_lim_mon_pkg") use "CDKM_THRUST_LIM_MON_PKG.ADA";
1956 1955      for Body ("cdkm_thrust_lim_mon_pkg") use "CDKM_THRUST_LIM_MON_PKG.ADA";
1957 1956      for Spec ("cdkm_progress_page_2_mon_pkg") use "CDKM_PROGRESS_PAGE_2_MON_PKG.ADA";
1958 1957      for Body ("cdkm_progress_page_2_mon_pkg") use "CDKM_PROGRESS_PAGE_2_MON_PKG.ADA";
1959 1958      for Spec ("cdkm_progress_page_1_mon_pkg") use "CDKM_PROGRESS_PAGE_1_MON_PKG.ADA";
1960 1959      for Body ("cdkm_progress_page_1_mon_pkg") use "CDKM_PROGRESS_PAGE_1_MON_PKG.ADA";
1961 1960      for Spec ("cdkm_pos_ref_page_3_4_pkg") use "CDKM_POS_REF_PAGE_3_4_PKG.ADA";
1962 1961      for Body ("cdkm_pos_ref_page_3_4_pkg") use "CDKM_POS_REF_PAGE_3_4_PKG.ADA";
1963 1962      for Spec ("cdkm_pos_ref_page_2_4_pkg") use "CDKM_POS_REF_PAGE_2_4_PKG.ADA";
1964 1963      for Body ("cdkm_pos_ref_page_2_4_pkg") use "CDKM_POS_REF_PAGE_2_4_PKG.ADA";
1965 1964      for Spec ("cdkm_pos_init_page_pkg") use "CDKM_POS_INIT_PAGE_PKG.ADA";
1966 1965      for Body ("cdkm_pos_init_page_pkg") use "CDKM_POS_INIT_PAGE_PKG.ADA";
1967 1966      for Spec ("cdkm_nav_radio_page_pkg") use "CDKM_NAV_RADIO_PAGE_PKG.ADA";
1968 1967      for Body ("cdkm_nav_radio_page_pkg") use "CDKM_NAV_RADIO_PAGE_PKG.ADA";
1969 1968      for Spec ("cdi_wind_page_pkg") use "CDI_WIND_PAGE_PKG.ADA";
1970 1969      for Body ("cdi_wind_page_pkg") use "CDI_WIND_PAGE_PKG.ADA";
1971 1970      for Spec ("cdi_time_date_init_page_pkg") use "CDI_TIME_DATE_INIT_PAGE_PKG.ADA";
1972 1971      for Body ("cdi_time_date_init_page_pkg") use "CDI_TIME_DATE_INIT_PAGE_PKG.ADA";
1973 1972      for Spec ("cdi_thrust_lim_page_pkg") use "CDI_THRUST_LIM_PAGE_PKG.ADA";
1974 1973      for Body ("cdi_thrust_lim_page_pkg") use "CDI_THRUST_LIM_PAGE_PKG.ADA";
1975 1974      for Spec ("cdi_takeoff_ref_states_pkg") use "CDI TAKEOFF_REF_STATES_PKG.ADA";
1976 1975      for Body ("cdi_takeoff_ref_states_pkg") use "CDI TAKEOFF_REF_STATES_PKG.ADA";
1977 1976      for Body ("cdi_takeoff_ref_page_pkg.get_data") use "CDI TAKEOFF_REF_PAGE_PKG.GET_DATA.ADA";
1978 1977      for Spec ("cdi_takeoff_ref_page_pkg") use "CDI TAKEOFF_REF_PAGE_PKG.ADA";
1979 1978      for Body ("cdi_takeoff_ref_page_pkg") use "CDI TAKEOFF_REF_PAGE_PKG.ADA";
1980 1979      for Spec ("cdi_takeoff_ref_page_2_pkg") use "CDI TAKEOFF_REF_PAGE_2_PKG.ADA";
1981 1980      for Body ("cdi_takeoff_ref_page_2_pkg") use "CDI TAKEOFF_REF_PAGE_2_PKG.ADA";
1982 1981      for Spec ("cdi_takeoff_ref_page_1_pkg") use "CDI TAKEOFF_REF_PAGE_1_PKG.ADA";
1983 1982      for Body ("cdi_takeoff_ref_page_1_pkg") use "CDI TAKEOFF_REF_PAGE_1_PKG.ADA";
1984 1983      for Spec ("cdi_takeoff_ref_2_states_pkg") use "CDI TAKEOFF_REF_2_STATES_PKG.ADA";
1985 1984      for Body ("cdi_takeoff_ref_2_states_pkg") use "CDI TAKEOFF_REF_2_STATES_PKG.ADA";
1986 1985      for Spec ("cdi_takeoff_ref_1_states_pkg") use "CDI TAKEOFF_REF_1_STATES_PKG.ADA";
1987 1986      for Body ("cdi_takeoff_ref_1_states_pkg") use "CDI TAKEOFF_REF_1_STATES_PKG.ADA";
1988 1987      for Spec ("cdi_std_disp_pkg") use "CDI_STD_DISP_PKG.ADA";
1989 1988      for Body ("cdi_std_disp_pkg") use "CDI_STD_DISP_PKG.ADA";
1990 1989      for Spec ("cdi_select_wpt_page_pkg") use "CDI_SELECT_WPT_PAGE_PKG.ADA";
1991 1990      for Body ("cdi_select_wpt_page_pkg") use "CDI_SELECT_WPT_PAGE_PKG.ADA";
1992 1991      for Spec ("cdi_rte_page_6r_pkg") use "CDI_RTE_PAGE_6R_PKG.ADA";
1993 1992      for Body ("cdi_rte_page_6r_pkg") use "CDI_RTE_PAGE_6R_PKG.ADA";

```

File: CTP_B787_PERF_CRZINITE.ZIP\stubs.gpr (continued)

1994	1993	for Spec ("cdi_rte_page_2_x_pkg") use "CDI_RTE_PAGE_2_X_PKG.ADA";
1995	1994	for Body ("cdi_rte_page_2_x_pkg") use "CDI_RTE_PAGE_2_X_PKG.ADA";
1996	1995	for Spec ("cdi_rte_page_1_x_pkg") use "CDI_RTE_PAGE_1_X_PKG.ADA";
1997	1996	for Body ("cdi_rte_page_1_x_pkg") use "CDI_RTE_PAGE_1_X_PKG.ADA";
1998	1997	for Spec ("cdi_rte_legs_page_pkg") use "CDI_RTE_LEGS_PAGE_PKG.ADA";
1999	1998	for Body ("cdi_rte_legs_page_pkg") use "CDI_RTE_LEGS_PAGE_PKG.ADA";
2000	1999	for Spec ("cdi_rte_legs_dirto_page_pkg") use "CDI_RTE_LEGS_DIRTO_PAGE_PKG.ADA";
2001	2000	for Body ("cdi_rte_legs_dirto_page_pkg") use "CDI_RTE_LEGS_DIRTO_PAGE_PKG.ADA";
2002	2001	for Spec ("cdi_rte_legs_common_pkg") use "CDI_RTE_LEGS_COMMON_PKG.ADA";
2003	2002	for Body ("cdi_rte_legs_common_pkg") use "CDI_RTE_LEGS_COMMON_PKG.ADA";
2004	2003	for Spec ("cdi_rte_dir_dep_arr_pkg") use "CDI_RTE_DIR_DEP_ARR_PKG.ADA";
2005	2004	for Body ("cdi_rte_dir_dep_arr_pkg") use "CDI_RTE_DIR_DEP_ARR_PKG.ADA";
2006	2005	for Body ("cdi_rte_directory_indices_pkg.via_trans") use "CDI_RTE_DIRECTORY_INDICES_PKG_VIATRANS.ADA";
2007	2006	for Body ("cdi_rte_directory_indices_pkg.rte_page_index") use "CDI_RTE_DIRECTORY_INDICES_PKG_RTE_PAGE.ADA";
2008	2007	for Spec ("cdi_rte_directory_indices_pkg") use "CDI_RTE_DIRECTORY_INDICES_PKG.ADA";
2009	2008	for Body ("cdi_rte_directory_indices_pkg") use "CDI_RTE_DIRECTORY_INDICES_PKG.ADA";
2010	2009	for Spec ("cdi_rte_data_page_pkg") use "CDI_RTE_DATA_PAGE_PKG.ADA";
2011	2010	for Body ("cdi_rte_data_page_pkg") use "CDI_RTE_DATA_PAGE_PKG.ADA";
2012	2011	for Spec ("cdi_ref_nav_data_page_pkg") use "CDI_REF_NAV_DATA_PAGE_PKG.ADA";
2013	2012	for Body ("cdi_ref_nav_data_page_pkg") use "CDI_REF_NAV_DATA_PAGE_PKG.ADA";
2014	2013	for Spec ("cdi_progress_page_4_pkg") use "CDI_PROGRESS_PAGE_4_PKG.ADA";
2015	2014	for Body ("cdi_progress_page_4_pkg") use "CDI_PROGRESS_PAGE_4_PKG.ADA";
2016	2015	for Spec ("cdi_progress_page_3_pkg") use "CDI_PROGRESS_PAGE_3_PKG.ADA";
2017	2016	for Body ("cdi_progress_page_3_pkg") use "CDI_PROGRESS_PAGE_3_PKG.ADA";
2018	2017	for Spec ("cdi_progress_page_2_pkg") use "CDI_PROGRESS_PAGE_2_PKG.ADA";
2019	2018	for Body ("cdi_progress_page_2_pkg") use "CDI_PROGRESS_PAGE_2_PKG.ADA";
2020	2019	for Spec ("cdi_progress_page_1_pkg") use "CDI_PROGRESS_PAGE_1_PKG.ADA";
2021	2020	for Body ("cdi_progress_page_1_pkg") use "CDI_PROGRESS_PAGE_1_PKG.ADA";
2022	2021	for Spec ("cdi_pos_report_page_pkg") use "CDI_POS_REPORT_PAGE_PKG.ADA";
2023	2022	for Body ("cdi_pos_report_page_pkg") use "CDI_POS_REPORT_PAGE_PKG.ADA";
2024	2023	for Spec ("cdi_pos_ref_page_4_4_pkg") use "CDI_POS_REF_PAGE_4_4_PKG.ADA";
2025	2024	for Body ("cdi_pos_ref_page_4_4_pkg") use "CDI_POS_REF_PAGE_4_4_PKG.ADA";
2026	2025	for Spec ("cdi_pos_ref_page_3_4_pkg") use "CDI_POS_REF_PAGE_3_4_PKG.ADA";
2027	2026	for Body ("cdi_pos_ref_page_3_4_pkg") use "CDI_POS_REF_PAGE_3_4_PKG.ADA";
2028	2027	for Spec ("cdi_pos_ref_page_2_4_pkg") use "CDI_POS_REF_PAGE_2_4_PKG.ADA";
2029	2028	for Body ("cdi_pos_ref_page_2_4_pkg") use "CDI_POS_REF_PAGE_2_4_PKG.ADA";
2030	2029	for Spec ("cdi_pos_init_page_pkg") use "CDI_POS_INIT_PAGE_PKG.ADA";
2031	2030	for Body ("cdi_pos_init_page_pkg") use "CDI_POS_INIT_PAGE_PKG.ADA";
2032	2031	for Spec ("cdi_perf_init_page_pkg") use "CDI_PERF_INIT_PAGE_PKG.ADA";
2033	2032	for Body ("cdi_perf_init_page_pkg") use "CDI_PERF_INIT_PAGE_PKG.ADA";
2034	2033	for Spec ("cdi_offpath_des_page_pkg") use "CDI_OFFPATH_DES_PAGE_PKG.ADA";
2035	2034	for Body ("cdi_offpath_des_page_pkg") use "CDI_OFFPATH_DES_PAGE_PKG.ADA";

File: CTP_B787_PERF_CRZINITE.ZIP\stubs.gpr (continued)

2036	2035	for Spec ("cdi_nav_radio_page_pkg") use "CDI_NAV_RADIO_PAGE_PKG.ADA";
2037	2036	for Body ("cdi_nav_radio_page_pkg") use "CDI_NAV_RADIO_PAGE_PKG.ADA";
2038	2037	for Spec ("cdi_maint_index_page_pkg") use "CDI_MAINT_INDEX_PAGE_PKG.ADA";
2039	2038	for Body ("cdi_maint_index_page_pkg") use "CDI_MAINT_INDEX_PAGE_PKG.ADA";
2040	2039	for Spec ("cdi_irs_mon_page_pkg") use "CDI_IRS_MON_PAGE_PKG.ADA";
2041	2040	for Body ("cdi_irs_mon_page_pkg") use "CDI_IRS_MON_PAGE_PKG.ADA";
2042	2041	for Spec ("cdi_init_ref_index_page_pkg") use "CDI_INIT_REF_INDEX_PAGE_PKG.ADA";
2043	2042	for Body ("cdi_init_ref_index_page_pkg") use "CDI_INIT_REF_INDEX_PAGE_PKG.ADA";
2044	2043	for Spec ("cdi_ident_page_pkg") use "CDI_IDENT_PAGE_PKG.ADA";
2045	2044	for Body ("cdi_ident_page_pkg") use "CDI_IDENT_PAGE_PKG.ADA";
2046	2045	for Spec ("cdi_hold_page_pkg") use "CDI_HOLD_PAGE_PKG.ADA";
2047	2046	for Body ("cdi_hold_page_pkg") use "CDI_HOLD_PAGE_PKG.ADA";
2048	2047	for Spec ("cdi_hold_at_page_pkg") use "CDI_HOLD_AT_PAGE_PKG.ADA";
2049	2048	for Body ("cdi_hold_at_page_pkg") use "CDI_HOLD_AT_PAGE_PKG.ADA";
2050	2049	for Spec ("cdi_fmc_comm_page_pkg") use "CDI_FMC_COMM_PAGE_PKG.ADA";
2051	2050	for Body ("cdi_fmc_comm_page_pkg") use "CDI_FMC_COMM_PAGE_PKG.ADA";
2052	2051	for Spec ("cdi_fix_info_page_pkg") use "CDI_FIX_INFO_PAGE_PKG.ADA";
2053	2052	for Body ("cdi_fix_info_page_pkg") use "CDI_FIX_INFO_PAGE_PKG.ADA";
2054	2053	for Spec ("cdi_display_pkg") use "CDI_DISPLAY_PKG.ADA";
2055	2054	for Body ("cdi_display_pkg") use "CDI_DISPLAY_PKG.ADA";
2056	2055	for Spec ("cdi_des_page_pkg") use "CDI_DES_PAGE_PKG.ADA";
2057	2056	for Body ("cdi_des_page_pkg") use "CDI_DES_PAGE_PKG.ADA";
2058	2057	for Spec ("cdi_des_forecast_page_pkg") use "CDI_DES_FORECAST_PAGE_PKG.ADA";
2059	2058	for Body ("cdi_des_forecast_page_pkg") use "CDI_DES_FORECAST_PAGE_PKG.ADA";
2060	2059	for Spec ("cdi_dep_arr_index_page_pkg") use "CDI_DEP_ARR_INDEX_PAGE_PKG.ADA";
2061	2060	for Body ("cdi_dep_arr_index_page_pkg") use "CDI_DEP_ARR_INDEX_PAGE_PKG.ADA";
2062	2061	for Spec ("cdi_dep_arr_common_pkg") use "CDI_DEP_ARR_COMMON_PKG.ADA";
2063	2062	for Body ("cdi_dep_arr_common_pkg") use "CDI_DEP_ARR_COMMON_PKG.ADA";
2064	2063	for Spec ("cdi_departures_page_pkg") use "CDI_DEPARTURES_PAGE_PKG.ADA";
2065	2064	for Body ("cdi_departures_page_pkg") use "CDI_DEPARTURES_PAGE_PKG.ADA";
2066	2065	for Spec ("cdi_crz_page_pkg") use "CDI_CRZ_PAGE_PKG.ADA";
2067	2066	for Body ("cdi_crz_page_pkg") use "CDI_CRZ_PAGE_PKG.ADA";
2068	2067	for Spec ("cdi_constants_pkg") use "CDI_CONSTANTS_PKG.ADA";
2069	2068	for Spec ("cdi_clb_page_pkg") use "CDI_CLB_PAGE_PKG.ADA";
2070	2069	for Body ("cdi_clb_page_pkg") use "CDI_CLB_PAGE_PKG.ADA";
2071	2070	for Spec ("cdi_arrivals_page_pkg") use "CDI_ARRIVALS_PAGE_PKG.ADA";
2072	2071	for Body ("cdi_arrivals_page_pkg") use "CDI_ARRIVALS_PAGE_PKG.ADA";
2073	2072	for Spec ("cdi_approach_ref_page_pkg") use "CDI_APPROACH_REF_PAGE_PKG.ADA";
2074	2073	for Body ("cdi_approach_ref_page_pkg") use "CDI_APPROACH_REF_PAGE_PKG.ADA";
2075	2074	for Spec ("cdi_apf_page_2_pkg") use "CDI_APF_PAGE_2_PKG.ADA";
2076	2075	for Body ("cdi_apf_page_2_pkg") use "CDI_APF_PAGE_2_PKG.ADA";
2077	2076	for Spec ("cdi_apf_page_1_pkg") use "CDI_APF_PAGE_1_PKG.ADA";

File: CTP_B787_PERF_CRZINITE.ZIP\stubs.gpr (continued)

```

2078 2077      for Body ("cdi_apf_page_1_pkg") use "CDI_APF_PAGE_1_PKG.ADA";
2079 2078      for Spec ("cdi_altn_plan_page_pkg") use "CDI_ALTN_PLAN_PAGE_PKG.ADA";
2080 2079      for Body ("cdi_altn_plan_page_pkg") use "CDI_ALTN_PLAN_PAGE_PKG.ADA";
2081 2080      for Spec ("cdi_altn_page_pkg") use "CDI_ALTN_PAGE_PKG.ADA";
2082 2081      for Body ("cdi_altn_page_pkg") use "CDI_ALTN_PAGE_PKG.ADA";
2083 2082      for Spec ("cdi_altn_list_page_pkg") use "CDI_ALTN_LIST_PAGE_PKG.ADA";
2084 2083      for Body ("cdi_altn_list_page_pkg") use "CDI_ALTN_LIST_PAGE_PKG.ADA";
2085 2084      for Spec ("cdck_fpln_request_status_tpkg") use "CDCK_FPLN_REQUEST_STATUS_TPKG.ADA";
2086 2085      for Spec ("brown_lgbm") use "BROWN_LGBM.ADA";
2087 2086      for Body ("brown_lgbm") use "BROWN_LGBM.ADA";
2088 2087      for Spec ("bite_large_bp_service_pkg") use "BITE_LARGE_BP_SERVICE_PKG.ADA";
2089 2088      for Body ("bite_large_bp_service_pkg") use "BITE_LARGE_BP_SERVICE_PKG.ADA";
2090 2089      for Spec ("bite_fmf_ac_state_data_pkg") use "BITE_FMF_AC_STATE_DATA_PKG.ADA";
2091 2090      for Body ("bite_fmf_ac_state_data_pkg") use "BITE_FMF_AC_STATE_DATA_PKG.ADA";
2092 2091      for Spec ("atc_msg_encoder_pkg") use "ATC_MSG_ENCODER_PKG.ADA";
2093 2092      for Body ("atc_msg_encoder_pkg") use "ATC_MSG_ENCODER_PKG.ADA";
2094 2093      for Spec ("atc_msg_decoder_pkg") use "ATC_MSG_DECODER_PKG.ADA";
2095 2094      for Body ("atc_msg_decoder_pkg") use "ATC_MSG_DECODER_PKG.ADA";
2096 2095      for Spec ("atc_msg_common_types_pkg") use "ATC_MSG_COMMON_TYPES_PKG.ADA";
2097 2096      for Spec ("atc_fmf_cmf_interface_types") use "ATC_FMF_CMF_INTERFACE_TYPES.ADA";
2098 2097      for Spec ("atc_encode_components_pkg") use "ATC_ENCODE_COMPONENTS_PKG.ADA";
2099 2098      for Body ("atc_encode_components_pkg") use "ATC_ENCODE_COMPONENTS_PKG.ADA";
2100 2099      for Spec ("atc_encoder_utility_pkg") use "ATC_ENCODER_UTILITY_PKG.ADA";
2101 2100      for Body ("atc_encoder_utility_pkg") use "ATC_ENCODER_UTILITY_PKG.ADA";
2102 2101      for Spec ("atc_decode_components_pkg") use "ATC_DECODE_COMPONENTS_PKG.ADA";
2103 2102      for Body ("atc_decode_components_pkg") use "ATC_DECODE_COMPONENTS_PKG.ADA";
2104 2103      for Spec ("atc_decode_clearance_pkg") use "ATC_DECODE_CLEARANCE_PKG.ADA";
2105 2104      for Body ("atc_decode_clearance_pkg") use "ATC_DECODE_CLEARANCE_PKG.ADA";
2106 2105      for Spec ("atc_decoder_utility_pkg") use "ATC_DECODER_UTILITY_PKG.ADA";
2107 2106      for Body ("atc_decoder_utility_pkg") use "ATC_DECODER_UTILITY_PKG.ADA";
2108 2107      for Spec ("atc_decoder_rte_info_pkg") use "ATC_DECODER_RTE_INFO_PKG.ADA";
2109 2108      for Body ("atc_decoder_rte_info_pkg") use "ATC_DECODER_RTE_INFO_PKG.ADA";
2110 2109      for Spec ("atc_datalink_io_pkg") use "ATC_DATA_LINK_IO_PKG.ADA";
2111 2110      for Body ("atc_datalink_io_pkg") use "ATC_DATA_LINK_IO_PKG.ADA";
2112 2111      for Spec ("atc_common_interface_types") use "ATC_COMMON_INTERFACE_TYPES.ADA";
2113 2112      for Spec ("atc_cmf_fmf_interface_types") use "ATC_CMF_FMF_INTERFACE_TYPES.ADA";
2114 2113      for Spec ("assert_fm_codes_pkg") use "ASSERT_FM_CODES_PKG.ADA";
2115 2114      for Spec ("arr_dep_ldata_obj_mgr") use "ARR_DEP_LDATA_OBJ_MGR.ADA";
2116 2115      for Body ("arr_dep_ldata_obj_mgr") use "ARR_DEP_LDATA_OBJ_MGR.ADA";
2117 2116      for Spec ("arr_dep_ldata") use "ARR_DEP_LDATA.ADA";
2118 2117      for Spec ("aoc_datalink_io_pkg") use "AOC_DATA_LINK_IO_PKG.ADA";
2119 2118      for Body ("aoc_datalink_io_pkg") use "AOC_DATA_LINK_IO_PKG.ADA";

```


File: CTP_B787_PERF_CRZINITE.ZIP\stubs.gpr (continued)

```

2120 2119     for Spec ("angle_const_pkg") use "ANGLE_CONST_PKG_.ADA";
2121 2120     for Spec ("airway_types") use "AIRWAY_TYPES_.ADA";
2122 2121     for Spec ("ads_manager_pkg") use "ADS_MANAGER_PKG_.ADA";
2123 2122     for Body ("ads_manager_pkg") use "ADS_MANAGER_PKG.ADA";
2124 2123     for Spec ("ads_lftypes") use "ADS_LFTYPES_.ADA";
2125 2124     for Spec ("ads_iftypes") use "ADS_IFTYPES_.ADA";
2126 2125     for Spec ("ads_fm_interface_mgr_pkg") use "ADS_FM_INTERFACE_MGR_PKG_.ADA";
2127 2126     for Body ("ads_fm_interface_mgr_pkg") use "ADS_FM_INTERFACE_MGR_PKG.ADA";
2128 2127     for Spec ("act_prov_index_manager") use "ACT_PROV_INDEX_MANAGER_.ADA";
2129 2128     for Body ("act_prov_index_manager") use "ACT_PROV_INDEX_MANAGER.ADA";
2130 2129     for Spec ("acars_periodic_pkg") use "ACARS_PERIODIC_PKG_.ADA";
2131 2130     for Body ("acars_periodic_pkg") use "ACARS_PERIODIC_PKG.ADA";
2132 2131     for Spec ("acars_incoming_buffer_pkg") use "ACARS_INCOMING_BUFFER_PKG_.ADA";
2133 2132     for Spec ("acars_crc_pkg") use "ACARS_CRC_PKG_.ADA";
2134 2133     for Body ("acars_crc_pkg") use "ACARS_CRC_PKG.ADA";
2135 2134     for Spec ("acars_buffer_ltypes") use "ACARS_BUFFER_LTYPES_.ADA";
2136 2135     for Spec ("acars_buffer_interface_mgr_pkg") use "ACARS_BUFFER_INTERFACE_MGR_PKG_.ADA";
2137 2136     for Body ("acars_buffer_interface_mgr_pkg") use "ACARS_BUFFER_INTERFACE_MGR_PKG.ADA";
2138 2137
2139 2138     for Specification_Suffix ("ada") use "__.ada";
2140 2139     for Implementation_Suffix ("ada") use ".ada";
2141 2140     for Separate_Suffix use ".ada";
2142 2141 end Naming;
2143 2142
2144 2143 Tornado := external ("WIND_BASE");
2145 2144 Hi_Scoe := external ("SCOE_BASE");
2146 2145 Hi_Platform := "wrSbc750gx_scoe";
2147 2146 for Languages use ("Ada");
2148 2147 for Object_Dir use "..\OBJ";
2149 2148 for Main use ("CTP_B787_PERF_ALTNESTDES.ADA");
2150 2149 for Exec_Dir use "..";
2151 2150
2152 2151 package Pretty_Printer is
2153 2152 end Pretty_Printer;
2154 2153
2155 2154 package Builder is
2156 2155     for Default_Switches ("ada") use ("-x", "--RTS=cert", "-j2", "-m", "-I" & Hi_Scoe & "\platforms\" & Hi_Platform &
» "\include");
2157 2156     for Executable ("fm_startup.ada") use "fm_ada_startup";
2158 2157 end Builder;
2159 2158
2160 2159 package Binder is

```

File: CTP_B787_PERF_CRZINITE.ZIP\stubs.gpr (continued)

```

2161 2160     for Default_Switches ("ada") use ("-E");
2162 2161 end Binder;
2163 2162
2164 2163 package Linker is
2165 2164     for Default_Switches ("ada") use ("--LINK=ldppc",
2166 2165         "-nostdlib",
2167 2166         "-r",
2168 2167         "-d",
2169 2168         Hi_Scoe & "\platforms\" & Hi_Platform & "\lib\adaLCH.PPC604gnu.cert.o",
2170 2169         Hi_Scoe & "\platforms\" & Hi_Platform & "\lib\tftp.PPC604gnu.cert.o",
2171 2170         "-L..\..\LIB",
2172 2171         "--start-group",
2173 2172         "-l_io_c_fmf",
2174 2173         "-l_bite_c_fmf",
2175 2174         "-l_bsvc_c_fmf",
2176 2175         "-l_flxcore_c_fmf",
2177 2176         "-l_flxprj_c_fmf",
2178 2177         "-l_ci_c",
2179 2178         "-l_hmi_c",
2180 2179         "-l_fpprj_c",
2181 2180         "-l_ltc_core_c",
2182 2181         "-l_fpcore_c",
2183 2182         "-l_psvc_c",
2184 2183         "-l_dbam_c",
2185 2184         "--end-group");
2186 2185 end Linker;
2187 2186
2188 2187 package Compiler is
2189 2188     for Default_Switches ("ada") use (
2190 2189         "-gdwarf-2",
2191 2190         "-ansi",
2192 2191         "-gnatf",
2193 2192         "-gnatn",
2194 2193         "-gnato",
2195 2194         "-fno-common",
2196 2195         "-mstrict-align",
2197 2196         "-fno-crossjumping",
2198 2197         "-fno-strict-aliasing",
2199 2198         "-fstack-check");
2200 2199 end Compiler;
2201 2200
2202 2201 package Ide is

```

File: CTP_B787_PERF_CRZINITE.ZIP\stubs.gpr (continued)

2203	2202	for Compiler_Command ("ada") use "powerpc-wrs-vxworksae-gnatmake";
2204	2203	for Gnatlist use "powerpc-wrs-vxworksae-gnatls";
2205	2204	for Debugger_Command use "powerpc-wrs-vxworksae-gdb";
2206	2205	for Program_Host use "SBC-session";
2207	2206	for Communication_Protocol use "wtx";
2208	2207	end Ide;
2209	2208	
2210	2209	end stubs;

Mode: All Lines

Left File: D:\B787_Download\CTP_B787_PERF_CRZINITE\OLD\CTP_B787_PERF_CRZINITE.TRT Right File: D:\B787_Download\CTP_B787_PERF_CRZINITE\NEW\CTP_B787_PERF_C

1	1	!*****			
2	2	!Trace Filename: CTP_B787_PERF_CRZINITE.TRT			
3	3	!			
4	4	!			
5	5	<Last modified BY> : <Author> <SCR:XXXXX> <mm/dd/yyyy> <Description>			
6	6	Last modified BY : TcSE SCR:14655.00 09/16/2011 Rolled over from TcSE			
7	7	!			
	8	! Chen Yongbing 15655.04 7/2/2014 Update for B787 BP3 LD3 on Build SBC2415_93C.			
	9	! 1. Added anchor PERF_SRD_B_00413 as per SCR 15655.01.			
	10	!			
8	11	!*****			
9	12	B787	SRD	PERF_TEST_00014	FMCS_19_20006099
10	13	B787	SRD	PERF_TEST_00014	FMCS_19_20006100
11	14	B787	SRD	PERF_TEST_00014	FMCS_19_20006026
12	15	B787	SRD	PERF_TEST_00014	FMCS_19_20006028
13	16	B787	SRD	PERF_TEST_00014	FMCS_19_20006102
14	17	B787	SRD	PERF_TEST_00014	FMCS_19_20006098
15	18	B787	SRD	PERF_TEST_00014	FMCS_19_20006097
16	19	B787	SRD	PERF_TEST_00014	FMCS_19_20006025
17	20	B787	SRD	PERF_TEST_00014	FMCS_19_20006029
18	21	B787	SDD	PERF_TEST_00014	FMCS_19_21027005
19	22	B787	SRD	PERF_TEST_00014	FMCS_19_20006027
	23	B787	SRD	PERF_TEST_00014	PERF_SRD_B_00413