FMS E-Review 2005 Cover Sheet

07/16/12

INSPECTION COMPLETE

Load: BP3_TST_REV_X03

Review ID:	15655.04	Life Cycle:	Low-Level Test Procedures	FAREA:	TEST	DO-178 Level:	B
Review Type:	Standard Meeting	ACM Project:	B7E7			Rework Effort (hours):	0
Produced:	HTS - Shanghai	ACM Subproject:	B7E7FMS			Closure Effort (hours):	0.5

Review: Date | Time	Jul 3, 2014 | 15:30	Meeting Duration:	0.5	Moderator Closure →	✔ APPROVED By Liu Mingyang at 11:58 am, Jul 06, 2014
Review Location:	Moderator Cubicle	# Review Participants:	4		
Conference Room:	N/A	Date Complete:	7/6/14	Audit:	Stamp Here
Telephone | Participant Code:	N/A | N/A	Review Status: (result of review)	Accepted As Is		

Some Work Deferred: ☐ Version(s) Unchanged: ☐

Work Product Type(s): B787

Trace Data, Component Test

Supporting Material(s) / Comments:

FMF_PERF_CRZ_PHASE.SDD ; 12
FMF_PERF_PREDS_CRZ_PHASE_SRD.DOCX ; 23

Producer Employer:	Shanghai (Avionics excl EDS	Oversight Eligible:	✔
# of Producer Technical Defects:	0	# of Oversight Focal Technical Defects:	0
# of Producer Non-technical Defects:	0	# of Oversight Focal Non-technical Defects:	0
# of Producer Process Defects:	0	# of Oversight Focal Process Defects:	0

Work Products Under Review

Reuse Scope: N/A

Problem Report	File Name	File Version	Review Size	Size Units	Approved Version
15655.04	CTP_B787_PERF_CRZINITE.TDF	6	12	test case	6
15655.04	CTP_B787_PERF_CRZINITE.ZIP	8	20	undefined	8
15655.04	CTP_B787_PERF_CRZINITE.TRT	3	12	anchors	3

Participants

Name	Function (discipline)/ Responsibility	Review Time (hours)	Role in review	Attend	Will Close	Signature check complete	
Chen Yongbing	S/W Test	0.5	Producer	☒		✔	REVIEWED By Chen Yongbing at 1:31 pm, Jul 02, 2014
Liu, Mingyang	S/W Test	1.5	Moderator	☒	☒	✔	REVIEWED By Liu Mingyang at 11:56 am, Jul 06, 2014
			Oversight	☐	☐	☐	
Zou Xing	S/W Test	0.5	Reviewer	☒	☐	✔	REVIEWED By Zou Xing at 11:55 am, Jul 06, 2014
Zhou, Qiong	Process	1.0	Auditor	☒	☐	✔	REVIEWED By Zhou, Qiong at 4:08 pm, Jul 03, 2014

Assignee 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.
Oversight Reviewer's signature (stamp) indicates that Supplier Oversight was conducted.
Safety Focal's signature (stamp) indicates partiicpation of a Safety Focal.

FIELD LEGEND:	Pre-Populated/Producer	Producer	Moderator	Participants	Oversight

Coversheet Continued

Name	Function (discipline)/ Responsibility	Review Time (hours)	Role in review	Attend	Will Close	Signature check complete
Barnes, Debra	PDQE		Auditor			

FIELD LEGEND:	Pre-Populated/Producer	Producer	Moderator	Participants	Oversight

Component Test Procedure (CTP) Checklist (CTP_CHECKLIST_WORD.doc 10/24/07)	**ACM Project:**	**B7E7**
	ACM Sub-Project:	**B7E7FMS**
	SCR Number:	**15655.04**
	Affected Area:	**TEST**

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?

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

Does the TDF header include the following fields:

[✓ Yes] No N/A — 12. Does the TDF header include the following fields:

- Filename
- Title
- Author
- Creation Date
- Modification History
- Source
- Description of TDF

[✓ Yes] No N/A — 13. Is the SCR number and description updated for this SCR?

[✓ Yes] No N/A — 14. Does the TDF header include a unique ANCHOR name for this CTP?

[✓ Yes] No N/A — 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)**

[✓ Yes] No N/A — 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?

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?

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.

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

Explanations for N/A
For item 22 :- there is no bad trace.
For item 29,30 :- B787 FMS software is a Level B software.
For item 33,34 :- there are no "complex algorithms" or "complex requirements" for this CTP.
For item 40 :- there are no failures in this CTP.
For item 41-43 :- there is no C++ code or C++ stub code.
For item 45: there is no defect found during review.

Trace Checklist **(Trace_Check_Word.doc 8/5/09)**	**ACM Project:**	B7E7
	ACM Sub-Project:	B7E7FMS
	SCR Number:	15655.04
	Affected Area:	TEST

Overview: Use this checklist to verify tracing data is correct, complete, and complies to standards.

Administrative

Y	N	N/A		
			1.	Are the following artifacts available at the work product review?
●	○	○	a	A copy of the applicable SCR(s)?
●	○	○	b	For non-TcSE, a copy of the trace file under review? For TcSE, a copy of the trace artifact or trace report under review?
●	○	○	c	A copy of the anchored requirement, design, test case/procedures, and/or source listings addressed by the trace file under review? (applicable pages only) (check Cover Sheet "Reference/Supporting Material" for element/gen.)

Trace File Standard – Applies to non-TcSE Only

Y	N	N/A		
			2.	Does the trace file header comply with the standard?
●	○	○	a	Header contains File Name and Revision History?
●	○	○	b	Is the Revision History description consistent with the SCR analysis?

Y	N	N/A		
			3.	Does the trace file trace information comply with the standard?
●	○	○	a	Are there 4 columns (Program, Relationship, Source_Anchor, Destination_Anchor) for each row?
●	○	○	b	Are the values for Relationship valid for the project?

Trace Data

Y	N	N/A		
			4.	Are anchors linked to appropriate higher-level document anchor(s)?
●	○	○		

Explanation for any "N" answer(s):

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.

SRB Reviewed By: Colaiacovo, John   Date: 13-AUG-2013

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_PREDS_CRZ_PHASE.SRD; 22 ->
   FMF_PERF_PREDS_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

Mode: All Lines

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

```
FILE                    :  CTP_B787_PERF_CRZINITE.TDF

SOURCE CONFIGURATION    :  ISS (Instruction Set Simulator)

DESCRIPTION             :  B787 Crz_InitStepTerms initializes the step climb terminations.

MODIFICATION HISTORY    :
                           DATE             SCR #           AUTHOR              DESCRIPTION
                           =====            =====           ======              ===========
                          11-May-2006       1134.00        Henson Zhao     Initial Development for B787 cycle 1 phase 1 Build
» ML134.

                          25-Aug-2006       1134.00         Alex Xie            Execution for B787 cycle 1 phase 1 Build SBC127.
                                                                                1.Format changed from HDB to GDB.
                                                                                2.Updated SUT_VARS.

                          17-Sep-2007       4845.00         He Wang         Update for B787 Load 4.5 Build SBC425.
                                                                             1. Updated SRD/SDD generations:
                                                                              FMF_PERF_CRZ_PHASE.SDD; 6 --> 12
                                                                              FMF_PERF_PREDS_CRZ_PHASE.SRD; 13 --> 20
                                                                             2. Removed FMF_PERF_PREDS_PHASES.SRD;7
                                                                             3. Removed SRD FMCS_19_3067 and FMCS_19_20006096.
                                                                             4. Added some SUTs and removed extra SUTs.
                                                                             5. Updated all breakpoints as per code changing.
                                                                             6. Added TC 12 for robust test.

                          24-Jun-2008        6880.00        Xinghua Liu      Updated for B787 Load 7.0 Build SBC617_8F2.
                                                                             1. Updated SRD generation:
                                                                               FMF_PERF_PREDS_CRZ_PHASE.SRD ; 20 -->22
                                                                             2. Modified all TC for remove stub.
                          May-24-2010        13550.00       Sumei Li         Updated for B787 RFS Build SBC922_811B2
                                                                             1. Updated the break points.

                          18-Mar-2013        15875.00       Lu Shubo         Update for B787 BP2 LD5 on Build ACMBLD_070_SBC.
                                                                             1. Updated breakpoints in TC 1~12.

                          2-Jul-2014         15655.04       Chen Yongbing  Update for B787 BP3 LD3 on Build SBC2415_93C.
                                                                             1. Updated SRD generation:
                                                                                FMF_PERF_PREDS_CRZ_PHASE.SRD; 22 ->
                                                                                FMF_PERF_PREDS_CRZ_PHASE_SRD.DOCX; 23.
```

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_C

```
       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_CI

```
  72    80 Perf_Integrators_Lfdata.TermBuf.TermArray().Value
  73    81 Perf_Integrators_Lfdata.IntProgBuf.Xprog
  74    82 Idx_Profile_Ifdata.Ialtprofptrec()().Dtd.Data
  75    83 Idx_Profile_Ifdata.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_C

```
 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       ~~                               FMCS_19_20006097, FMCS_19_20006098, FMCS_19_20006099, FMCS_19_20006102~~
       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_CI

```
 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_Ifdata.Ialtprofptrec(0)(1).Dtd.Data                   := 100.0~~
 171      ~~Idx_Profile_Ifdata.Ialtprofptrec(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_CI

```

~~# break perf_crz_predexec_sep.ada:657~~
# break perf_crz_predexec_sep.ada:659
# continue
# return
# return

!end_test()
--OUTPUTS:
Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Valid  = boolean'(false)
Perf_Integrators_Lfdata.TermBuf.TermArray(11).Value                = 100.0
------------------------------------------------------------------------------------------------------------------------
TESTID: 2
This verify When If the step type is SpecStep, the StepAlt termination value is set to the  Specified step
altitude.If the aircraft is in climb, and a step was moved to the cruise phase from the climb phase the StepAlt
value is set to the climb specified step altitude, and the StepDist is activated with a distance value indicating
the step should begin immediately.

REQUIREMENTS UNDER EVALUATION: FMCS_19_20006025, FMCS_19_20006026, FMCS_19_20006027, FMCS_19_20006028, FMCS_19_20006029
                               FMCS_19_20006097, FMCS_19_20006098, FMCS_19_20006099, FMCS_19_20006102, FMCS_19_20006100
SUPPORTING REQUIREMENTS :      FMCS_19_21027005


--INPUTS:
-- SETLANGMODE = ADA
Test_Firstpass                                                    := boolean'(true)
Test_Steptype                                                     := CFP_PERF_STEP_IFTYPES.specstep
Test_LGB_Search                                                   := 0
-- Test_ICAO_Low                                                     := 6000.0
-- Test_ICAO_High                                                    := 10000.0

------------------------------------------------------------------------
-- set variable for enter SUT
Perf_Preds_Lfdata.VTPlogic.Firstpass            := boolean'(true)
Perf_Crzalt_Lfdata.LastCrzAlt.Valid             := boolean'(false)
Perf_Preds_Lfdata.Vgbptr                        := CLB2L

Perf_RTA_Lfdata.Pred_Pastrta    := boolean'(true)
Perf_WTS_Lfdata.Always_Compute_Max_Speed    := boolean'(false)

Perf_Integrators_Lfdata.IntProgBuf.Hprog    := 27004.0
Perf_Crzalt_Lfdata.Crzalt           := 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_CI

```
 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_CI

```
!end_test()
--OUTPUTS:
Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Valid = boolean'(false)

------------------------------------------------------------------------------------------------------------------------
TESTID: 3
This verify When If the step type is SpecStep, the StepAlt termination value is set to the Specified step
altitude.If the aircraft is in climb, and a step was moved to the cruise phase from the climb phase the StepAlt
value is set to the climb specified step altitude, and the StepDist is activated with a distance value indicating
the step should begin immediately.

REQUIREMENTS UNDER EVALUATION: FMCS_19_20006025, FMCS_19_20006026, FMCS_19_20006027, FMCS_19_20006028, FMCS_19_20006029
                               FMCS_19_20006097, FMCS_19_20006098, FMCS_19_20006099, FMCS_19_20006102

SUPPORTING REQUIREMENTS :      FMCS_19_21027005


--INPUTS:
-- SETLANGMODE = ADA
Test_Firstpass                                                      := boolean'(true)
Test_Steptype                                                       := CFP_PERF_STEP_IFTYPES.specstep
Test_LGB_Search                                                     := 0
-- Test_ICAO_Low                                                       := 6000.0
-- Test_ICAO_High                                                      := 10000.0

-----------------------------------------------------------------------
-- set variable for enter SUT
Perf_Preds_Lfdata.VTPlogic.Firstpass             := boolean'(true)
Perf_Crzalt_Lfdata.LastCrzAlt.Valid              := boolean'(false)
Perf_Preds_Lfdata.Vgbptr                         := CLB2L

Perf_RTA_Lfdata.Pred_Pastrta    := boolean'(true)
Perf_WTS_Lfdata.Always_Compute_Max_Speed    := boolean'(false)

Perf_Integrators_Lfdata.IntProgBuf.Hprog    := 27004.0
Perf_Crzalt_Lfdata.Crzalt           := 27000.0
-----------------------------------------------------------------------

-- Initialize the variable
perf_Preds_Lfdata.Fltphase                                          := FMCS_Base_Types.climb
Perf_Preds_Lfdata.NavPtr                                            := 1
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_C

```
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_CI

```
altitude.If the aircraft is not in climb, and the StepDist is invalid.

REQUIREMENTS UNDER EVALUATION: FMCS_19_20006025, FMCS_19_20006026, FMCS_19_20006027, FMCS_19_20006028, FMCS_19_20006029
                               FMCS_19_20006097, FMCS_19_20006098, FMCS_19_20006099, FMCS_19_20006102

SUPPORTING REQUIREMENTS :      FMCS_19_21027005


--INPUTS:
-- SETLANGMODE = ADA
Test_Firstpass                                                     := boolean'(false)
Test_Steptype                                                      := CFP_PERF_STEP_IFTYPES.specstep
Test_LGB_Search                                                    := 0
-- Test_ICAO_Low                                                      := 6000.0
-- Test_ICAO_High                                                     := 10000.0
------------------------------------------------------------------------
-- set variable for enter SUT
Perf_Preds_Lfdata.VTPlogic.Firstpass            := boolean'(true)
Perf_Crzalt_Lfdata.LastCrzAlt.Valid             := boolean'(false)
Perf_Preds_Lfdata.Vgbptr                        := CLB2L

Perf_RTA_Lfdata.Pred_Pastrta    := boolean'(true)
Perf_WTS_Lfdata.Always_Compute_Max_Speed    := boolean'(false)

Perf_Integrators_Lfdata.IntProgBuf.Hprog    := 27004.0
Perf_Crzalt_Lfdata.Crzalt           := 27000.0
------------------------------------------------------------------------

-- Initialize the variable
Perf_Preds_Lfdata.Fltphase                                         := FMCS_Base_Types.Descent
Perf_Preds_Lfdata.NavPtr                                           := 1
Perf_Preds_Lfdata.PrevNavPtr                                       := 1

# break Perf_Su_Spd_Utils_Pkg.Su_Frmtgtspdrec
# continue
# return

-- enter SUT
# break Crz_InitStepTerms
# continue
#define Loc_Clb_Step_Exec                                          := boolean'(false)
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_CI

```
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_CI

```
 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
```

```
Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Valid = boolean'(false)
-------------------------------------------------------------------------------------------------------------------------
TESTID: 6
This verify When If the Step type is PEopt, a check is made to see if the aircraft is past
the current step distance to destination.

REQUIREMENTS UNDER EVALUATION: FMCS_19_20006025, FMCS_19_20006026, FMCS_19_20006027, FMCS_19_20006028, FMCS_19_20006029
~~                               FMCS_19_20006097, FMCS_19_20006098, FMCS_19_20006099, FMCS_19_20006102~~
                               FMCS_19_20006097, FMCS_19_20006098, FMCS_19_20006099, FMCS_19_20006102, PERF_SRD_B_00413

SUPPORTING REQUIREMENTS :      FMCS_19_21027005


--INPUTS:
-- SETLANGMODE = ADA
Test_Firstpass                                                     := boolean'(false)
Test_Steptype                                                      := CFP_PERF_STEP_IFTYPES.PEopt
Test_LGB_Search                                                    := 0
-- Test_ICAO_Low                                                      := 6000.0
-- Test_ICAO_High                                                     := 10000.0
-------------------------------------------------------------------------
-- set variable for enter SUT
Perf_Preds_Lfdata.VTPlogic.Firstpass            := boolean'(true)
Perf_Crzalt_Lfdata.LastCrzAlt.Valid             := boolean'(false)
Perf_Preds_Lfdata.Vgbptr                        := CLB2L

Perf_RTA_Lfdata.Pred_Pastrta    := boolean'(true)
Perf_WTS_Lfdata.Always_Compute_Max_Speed    := boolean'(false)

Perf_Integrators_Lfdata.IntProgBuf.Hprog    := 27004.0
Perf_Crzalt_Lfdata.Crzalt           := 27000.0
-------------------------------------------------------------------------

-- Initialize the variable
# break Perf_Su_Spd_Utils_Pkg.Su_Frmtgtspdrec
# continue
# return

-- enter SUT
# break Crz_InitStepTerms
# continue

```

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_C

```
 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_Ifdata.Ialtprofptrec(0)(1).Dtd.Valid                   := boolean'(true)~~
 517      ~~Idx_Profile_Ifdata.Ialtprofptrec(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_C

```
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_INITSTEPTERMS_SEP.ADA:325~~
     597 # break PERF_CRZ_INITSTEPTERMS_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

```
# continue
# return
# return

!end_test()
--OUTPUTS:
Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Stepptr           = 0
Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Valid = boolean'(false)
Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Opt_Stepalt      = 1000.0
Perf_Integrators_Lfdata.TermBuf.TermArray(11).Active               = boolean'(true)

------------------------------------------------------------------------------------------------------------------------
TESTID: 8
This verify When If the step type is Opt, it is determined whether:  both a new step altitude
and gross weight need to be computed.

REQUIREMENTS UNDER EVALUATION: FMCS_19_20006025, FMCS_19_20006026, FMCS_19_20006027, FMCS_19_20006028, FMCS_19_20006029,
                               FMCS_19_20006097, FMCS_19_20006098, FMCS_19_20006099, FMCS_19_20006102

SUPPORTING REQUIREMENTS :      FMCS_19_21027005


--INPUTS:
-- SETLANGMODE = ADA
Test_Firstpass                                                     := boolean'(false)
Test_Steptype                                                      := CFP_PERF_STEP_IFTYPES.Opt
Test_LGB_Search                                                    := 0
-- Test_ICAO_Low                                                      := 6000.0
-- Test_ICAO_High                                                     := 60000.0
------------------------------------------------------------------------
-- set variable for enter SUT
Perf_Preds_Lfdata.VTPlogic.Firstpass            := boolean'(true)
Perf_Crzalt_Lfdata.LastCrzAlt.Valid             := boolean'(false)
Perf_Preds_Lfdata.Vgbptr                        := CLB2L

Perf_RTA_Lfdata.Pred_Pastrta    := boolean'(true)
Perf_WTS_Lfdata.Always_Compute_Max_Speed    := boolean'(false)

Perf_Integrators_Lfdata.IntProgBuf.Hprog    := 80006.0
Perf_Crzalt_Lfdata.Crzalt           := 80000.0
------------------------------------------------------------------------

```

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_CI

```
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_CI

```
 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()
```

```
--OUTPUTS:
Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Stepptr           = 1
Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Valid = boolean'(false)

------------------------------------------------------------------------------------------------------------------------
TESTID: 10
This verify When If the Step type is PastSpecStep, the StepAlt termination value is set to
the current specified step altitude.

REQUIREMENTS UNDER EVALUATION: FMCS_19_20006025, FMCS_19_20006026, FMCS_19_20006027, FMCS_19_20006028, FMCS_19_20006029
                               FMCS_19_20006097, FMCS_19_20006098, FMCS_19_20006099, FMCS_19_20006102

SUPPORTING REQUIREMENTS :      FMCS_19_21027005


--INPUTS:
-- SETLANGMODE = ADA
Test_Firstpass                                                      := boolean'(false)
Test_Steptype                                                       := CFP_PERF_STEP_IFTYPES.PastSpecStep
Test_LGB_Search                                                     := 0
-- Test_ICAO_Low                                                      := 6000.0
-- Test_ICAO_High                                                     := 10000.0
------------------------------------------------------------------------
-- set variable for enter SUT

Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Stepptr            := 0
Perf_Preds_Lfdata.Navptr                                            := 1
Perf_Preds_Lfdata.Fltphase                                          := FMCS_Base_Types.Descent

# break Perf_Su_Spd_Utils_Pkg.Su_Frmtgtspdrec
# continue
# return

-- enter SUT
# break Crz_InitStepTerms
# continue
Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Valid  := boolean'(true)
Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Data   := 500.0
Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Fixdistodest       := 1000.0
Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Valid  := boolean'(true)
Perf_Integrators_Lfdata.TermBuf.TermArray(6).Active                 := boolean'(false)
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_CI

```
 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_CI

```
 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_CI

```
Test_LGB_Search                                                    := 0
-- Test_ICAO_Low                                                      := 6000.0
-- Test_ICAO_High                                                     := 10000.0
------------------------------------------------------------------------
-- set variable for enter SUT
Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Stepptr           := 0
Perf_Preds_Lfdata.Navptr                                           := 1
Perf_Preds_Lfdata.Fltphase                                         := FMCS_Base_Types.Descent
# break Perf_Su_Spd_Utils_Pkg.Su_Frmtgtspdrec
# continue
# return

-- enter SUT
# break Crz_InitStepTerms
# continue
Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Valid := boolean'(true)
Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Data  := 1000.0
Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Fixdistodest      := 500.0
Perf_Integrators_Lfdata.TermBuf.TermArray(6).Active                := boolean'(false)
Perf_Integrators_Lfdata.TermBuf.TermArray(6).Value                 := 0.0
Perf_Integrators_Lfdata.IntProgBuf.Xprog                           := 999.0

~~# break perf_crz_predexec_sep.ada:657~~
# break perf_crz_predexec_sep.ada:659
# continue
# return
# return

!end_test()
--OUTPUTS:
Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Stepptr           = 1
Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Valid = boolean'(false)
Perf_Integrators_Lfdata.TermBuf.TermArray(6).Active                = boolean'(true)
Perf_Integrators_Lfdata.TermBuf.TermArray(6).Value                 = 999.0
------------------------------------------------------------------------------------------------------------------------
```

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.ada

```
with CTP_B787_PERF_CRZINITE_DRV;
use CTP_B787_PERF_CRZINITE_DRV;

with Scoe_Amio_Enable_Itf;
with Scoe_Iolib_Api;
use Scoe_Iolib_Api;

with Gnat.Io; use Gnat.Io;

procedure CTP_B787_PERF_CRZINITE is

begin

   If Scoe_Amio_Enable_Itf.Scoe_Amio_Enable /=  Scoe_Iolib_Api.Scoe_Status_Ok then
        Gnat.IO.put_line("$$$$");
   End if;

   Gnat.IO.Put_line ("Entry point for PDB!");

   loop
      CTP_B787_PERF_CRZINITE_D;
   end loop;

end CTP_B787_PERF_CRZINITE;
```

File: CTP_B787_PERF_CRZINITE.ZIP\CTP_B787_PERF_CRZINITE.bat

```
@echo off
REM
REM /*+
REM  ***************************************************************************
REM  *
REM  *          HONEYWELL PROPRIETARY, CONFIDENTIAL, AND/OR TRADE SECRET
REM  *               Copyright (c) 2007 Honeywell International, Inc.
REM  *                  Unpublished Work -- All Rights Reserved
REM  *
REM  *   NAME:   CTP_B787_PERF_CRZINITE.BAT
REM  *
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)

```
REM  *
REM  *   COMMAND LINE:  CTP_B787_PERF_CRZINITE.BAT
REM  *
REM  *   INPUT:  None
REM  *
REM  *   OUTPUT:
REM  *       The B787 CTP test will be built and executed with no human interaction.
REM  *       All results files will be stored in appropriate directories.
REM  *
REM  *   INSTRUCTIONS:
REM  *       There are three items that you need to change in this B787 CTP batch template file.
REM  *       They are:
REM  *
REM  *       1) Make a copy of the B787 CTP batch template file and rename it for the real CTP.
REM  *          Substitute the "TEMPLATE" in this file name with the name of the real CTP.
REM  *
REM  *          For example:  COPY  CTP_B787_TEMPLATE.BAT  CTP_B787_PERF_CRZINITE.BAT  /V
REM  *
REM  *       2) Edit the new B787 CTP batch file and just before the ":step_1" line, replace the
REM  *          string to change "CTP_B787_XXXX" to use the name of the real CTP.
REM  *
REM  *          For example:  Change from: SET CTP_INPUT_NAME=CTP_B787_XXXX
REM  *                        Change to:   SET CTP_INPUT_NAME=CTP_B787_PERF_CRZINITE
REM  *
REM  *          Set the string INPUT_LANG to C or A depending on the language being used for CTP
REM  *
REM  *          For example:  If the CTP is in ADA and C or only ADA
REM  *                        Change from: SET INPUT_LANG=C/A
REM  *                        Change to:   SET INPUT_LANG=A
REM  *
REM  *
REM  *                        If the CTP is in ADA and C++
REM  *                        Change from: SET INPUT_LANG=C/A
REM  *                        Change to:   SET INPUT_LANG=C
REM  *
REM  *       3) In Step 3 of this new B787 CTP batch file, follow the instructions there
REM  *          for all C source code files that are a part of this test.
REM  *
REM  *       4) In Step 4 of this new B787 CTP batch file, follow the instructions there
REM  *          for all CPP source code files that are a part of this test.
REM  *
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)

```
REM  *          for all ADA source code files that are a part of this test.
REM  *
REM  *   NOTES:
REM  *       Do not make any other changes to this B787 CTP batch file other that those
REM  *       instructions shown above.
REM  *
REM  *   HISTORY:
REM  *       Feb. 2007  Rev. 1.0  Jayalakshmi Kadiwal  Author
REM  *
REM  ******************************************************************************
REM -*/
REM
ECHO.
ECHO CTP_B787_PERF_CRZINITE.BAT, Version Number 1.0
ECHO Copyright (c) 2007 Honeywell International, Inc.  All rights reserved.
ECHO Batch file to build and execute a B787 CTP test.
ECHO.
REM
ECHO.
REM
SET CTP_INPUT_NAME=CTP_B787_PERF_CRZINITE
SET INPUT_LANG=C/A
REM
SET CTP_START_DATE=%DATE%
SET CTP_START_TIME=%TIME%
REM
:step_1
REM
ECHO.
ECHO **************************************************************************
ECHO * CTP_B787_PERF_CRZINITE.BAT-I-Step 1 of 11                                    *
ECHO * Check the tool installations for Tornado and GPS.                      *
ECHO **************************************************************************
ECHO.
REM
%chk_init% %CTP_INPUT_NAME%
if %temp_err% EQU %BATCH_ERROR% goto exit_fail
:step_2
REM
ECHO.
ECHO **************************************************************************
ECHO * CTP_B787_PERF_CRZINITE.BAT-I-Step 2 of 11                                    *
```

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

```
ECHO * Generate Makefile in Test_Partition                                  *
ECHO * and Netboot directories                                              *
ECHO ***********************************************************************
ECHO.
REM
%make_file% %CTP_INPUT_NAME% %INPUT_LANG%
REM
:step_3
REM
ECHO.
ECHO ***********************************************************************
ECHO * CTP_B787_PERF_CRZINITE.BAT-I-Step 3 of 11                             *
ECHO * Compile all C files (if any).                                        *
ECHO ***********************************************************************
ECHO.
REM
REM **************************************************************************************
REM * If there are no C files for this CTP test, then comment out the "make_c" line below.         *
REM *                                                                                      *
REM * If there is only one C file for this CTP test, then edit the "make_c" line below to          *
REM *     replace the "YYYY" with the B787 CTP GPR file name and                           *
REM *     replace the "ZZZZ" with the name of the C source code file to be compiled.       *
REM *                                                                                      *
REM * If there is more than one C file for this CTP test, then make copies of the "make_c" line,   *
REM *     below and edit each line as follows:                                             *
REM *     replace the "YYYY" with the B787 CTP GPR file name and                           *
REM *     replace the "ZZZZ" with the name of the C source code file to be compiled.       *
REM *                                                                                      *
REM * Notes:                                                                               *
REM * 1) If the GPR file does not does follow the B787 CTP file name conventions, you will need to *
REM *    rename the GPR file to match B787 CTP file name conventions with "GPR" as the file type.  *
REM * 2) If the GPR file contains hardcoded pathname references, these will have to be edited to   *
REM *    follow the guidelines contained in the README file for this B787 CTP batch file.  *
REM * 3) Include the line "if EXIST..." and "md..." only if the CTP requires creation of "obj_c"   *
REM *    folder. Else comment these 2 lines using "REM" command                            *
REM **************************************************************************************
REM
REM
if EXIST %BUILD_BASE%\CFG\%CTP_INPUT_NAME%\obj_c goto sub_step_3
md %BUILD_BASE%\CFG\%CTP_INPUT_NAME%\obj_c
goto sub_step_3

```

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

```
:sub_step_3
REM %make_c% CTP_B787_PERF_CRZINITE_c.gpr CTP_B787_PERF_CRZINITE_drv_c.C
REM
:step_4
REM
ECHO.
ECHO *************************************************************************
ECHO * CTP_B787_PERF_CRZINITE.BAT-I-Step 4 of 11                                  *
ECHO * Compile all CPP files (if any).                                       *
ECHO *************************************************************************
ECHO.
REM
REM ************************************************************************************************
REM * If there are no CPP files for this CTP test, then comment out the "make_cpp" line below.     *
REM *                                                                                              *
REM * If there is only one CPP file for this CTP test, then edit the "make_cpp" line below to      *
REM *     replace the "YYYY" with the B787 CTP GPR file name and                                   *
REM *     replace the "ZZZZ" with the name of the CPP source code file to be compiled.             *
REM *                                                                                              *
REM * If there is more than one CPP file for this CTP test, then make copies of the "make_cpp"     *
REM *     line below and edit each line as follows:                                                *
REM *     replace the "YYYY" with the B787 CTP GPR file name and                                   *
REM *     replace the "ZZZZ" with the name of the CPP source code file to be compiled.             *
REM *                                                                                              *
REM * Notes:                                                                                       *
REM * 1) If the GPR file does not does follow the B787 CTP file name conventions, you will need to *
REM *    rename the GPR file to match B787 CTP file name conventions with "GPR" as the file type.  *
REM * 2) If the GPR file contains hardcoded pathname references, these will have to be edited to   *
REM *    follow the guidelines contained in the README file for this B787 CTP batch file.          *
REM ************************************************************************************************
REM %make_cpp% CTP_B787_YYYY_cpp.gpr CTP_B787_ZZZZ.CC
REM
:step_5
REM
ECHO.
ECHO *************************************************************************
ECHO * CTP_B787_PERF_CRZINITE.BAT-I-Step 5 of 11                                  *
ECHO * Compile all ADA files.                                                *
ECHO * Link the test executable.                                             *
ECHO *************************************************************************
ECHO.
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 does 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 th1s 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      ~~set FMS_NAV_DB=HEF0509031.SM~~
     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)

```
ECHO ************************************************************************
ECHO * CTP_B787_PERF_CRZINITE.BAT-I-Step 7 of 11                                  *
ECHO * Invoke the ftp server, simics and target server.                       *
ECHO ************************************************************************
ECHO.
REM
%run_tools%
if %temp_err% EQU %BATCH_ERROR% goto exit_fail
REM
:step_8
REM
ECHO.
ECHO ************************************************************************
ECHO * CTP_B787_PERF_CRZINITE.BAT-I-Step 8 of 11                                  *
ECHO * Run the Test Generation System (TGS) tool.                             *
ECHO * CTPs with multiple TDF should comment out this step-8                  *
ECHO ************************************************************************
ECHO.
REM
%runtgs% %CTP_INPUT_NAME%
if %temp_err% EQU %BATCH_ERROR% goto exit_fail
REM
REM Copy the TGS results files to the B787 "save" directory.
REM
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
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
REM


:step_9
REM
ECHO.
ECHO ************************************************************************
ECHO * CTP_B787_PERF_CRZINITE.BAT-I-Step 9 of 11                                  *
ECHO * Run the Test Generation System (TGS) tool.                             *
ECHO * ONLY CTPs with multiple TDFS should include this step                  *
ECHO * If there is more than 1 TDF, copy the %runtgs_multiple%                *
ECHO * line below and replace "DDDD" with the TDF name                        *
ECHO ************************************************************************
ECHO.
```

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

```
REM
REM %runtgs_multiple% %CTP_INPUT_NAME% CTP_B787_DDDDn

:step_10
REM
ECHO.
ECHO *************************************************************************
ECHO * CTP_B787_PERF_CRZINITE.BAT-I-Step 10 of 11                                 *
ECHO * Run the Test Coverage Analyzer (TCA) tool.                            *
ECHO * CTPs with multiple TDF should comment out this step-10                *
ECHO *************************************************************************
ECHO.
REM
%runtca% %CTP_INPUT_NAME%
if %temp_err% EQU %BATCH_ERROR% goto exit_fail
REM
REM Copy the TCA results files to the B787 "save" directory.
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
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
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
REM
REM
:step_11
REM
ECHO.
ECHO *************************************************************************
ECHO * CTP_B787_PERF_CRZINITE.BAT-I-Step 11 of 11                                 *
ECHO * Exit.                                                                 *
ECHO *************************************************************************
ECHO.
REM
:exit_success
Set temp_err=0
ECHO CTP_B787_PERF_CRZINITE.BAT-S-NORMAL, Normal termination.
ECHO.
goto exit
REM
:exit_fail
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_INITSTEPTERMS_SEP.ADA.PERF_CRZ_PKG.CRZ_INITSTEPTERMS
   7     7 PERF_CRZ_INITSTEPTERMS_SEP.ADA.PERF_CRZ_PKG.CRZ_INITSTEPTERMS.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_INITSTEPTERMS_SEP.ADA.PERF_CRZ_PKG.CRZ_INITSTEPTERMS
	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		~~<b> 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)

```

XInfo file(s)   Test Date   Test Platform:

  (P01) CTP_B787_PERF_CRZINITE.pth
~~        (X01) CTP_B787_PERF_CRZINITE.xin   Mon Mar 25 09:46:20 2013   Simics PowerPC TCA XInfo, Platform V3.00.09~~
        (X01) CTP_B787_PERF_CRZINITE.xin   Wed Jul 02 12:53:59 2014   Simics PowerPC TCA XInfo, Platform V3.00.09

------------------------------------------------------------------------------
       Compilation                     Test Coverage Statistics       Warnings
        Unit Name                Total Decision Cond Statemnt Block  Mixed Bool
------------------------------- ----------------------------------- ----------
PERF_CRZ_INITSTEPTERMS_SEP -
  .ADA.PERF_CRZ_PKG.CRZ_INITS -
  TEPTERMS.LOC_CLB_STEP           100.0   100.0   n/a   100.0  100.0     1   0
                                         8/8     n/a   16/16  11/11

PERF_CRZ_INITSTEPTERMS_SEP -
  .ADA.PERF_CRZ_PKG.CRZ_INITS -
  TEPTERMS                        100.0   100.0   n/a   100.0  100.0     1   0
                                         16/16   n/a   32/32  32/32

------------------------------------------------------------------------------
Total Percentages                        100.0   n/a   100.0  100.0
Totals                                   24/24   n/a   48/48  43/43
Total Coverage                   100.0
------------------------------------------------------------------------------
□
************************************************************************

        Test Coverage Analyzer (TCA)  Version 6.7 CLASS A

************************************************************************

Coverage Type: 3

Date of report / Report name :

~~        Mon Mar 25 09:56:03 2013 CTP_B787_PERF_CRZINITE.RPT~~
        Wed Jul 02 13:02:59 2014 CTP_B787_PERF_CRZINITE.RPT

Current Directory:

```

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)

```
     Note that no prefix indicates source line was not executed


************************************************************************

Compilation Unit / Source file :

        PERF_CRZ_INITSTEPTERMS_SEP.ADA.PERF_CRZ_PKG.CRZ_INITSTEPTERMS.LOC_CL -
           B_STEP
~~        C:\B787\Builds\ACMBLD_070_SBC\SRC\FM\perf_crz_initstepterms_sep.ada~~
        C:\B787\Builds\SBC2415_93C\SRC\fm\perf_crz_initstepterms_sep.ada

Coverage statistics :

        TYPE 3, 100.0%

                         Executed         Total
        Decision Paths         8              8
        Condition Paths      n/a            n/a
        Statements            16             16
        Blocks                11             11



************************************************************************

Compilation Unit / Source file :

        PERF_CRZ_INITSTEPTERMS_SEP.ADA.PERF_CRZ_PKG.CRZ_INITSTEPTERMS
~~        C:\B787\Builds\ACMBLD_070_SBC\SRC\FM\perf_crz_initstepterms_sep.ada~~
        C:\B787\Builds\SBC2415_93C\SRC\fm\perf_crz_initstepterms_sep.ada

Coverage statistics :

        TYPE 3, 100.0%

                         Executed         Total
        Decision Paths        16             16
        Condition Paths      n/a            n/a
        Statements            32             32
        Blocks                32             32

```

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_Crzalt_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_Crzalt_Lfdata.Crzalt                                                                               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_Ifdata.Ialtprofptrec(0)(1).Dtd.Data                                                         100.0~~
107      ~~Idx_Profile_Ifdata.Ialtprofptrec(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)

```
Perf_WTS_Lfdata.Previous_Pass_Info_Rec.Step(1).Distance.Valid                 boolean'(true)
Perf_LGB_Lfdata.LGB(2).Fpln_Data.SpAlt1Val                                   boolean'(false)
Perf_LGB_Lfdata.LGB(2).Fpln_Data.SpAlt1Pos                               AC_Position_Types.At_Alt
Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Stepptr                                     99
Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Valid            boolean'(true)
Perf_Integrators_Lfdata.TermBuf.TermArray(6).Active                          boolean'(false)
Perf_Integrators_Lfdata.TermBuf.TermArray(6).Value                                          1.0
Step_Ptr                                                                                     99


OUTPUT                                                          EXPECTED              TOLERANCE             ACTUAL
» P/F
----------------------------------------------------  -----------------------------  -------------  -------------------------
» ----
Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Stepptr                      99          (N/A)                          99
» P
Perf_Integrators_Lfdata.TermBuf.TermArray(6).Active                boolean'(false)          (N/A)                       false
» P
Perf_Integrators_Lfdata.TermBuf.TermArray(6).Value                           1.0         0.0001                         1.0
» P
Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Valid  boolean'(false)          (N/A)                       false
» P
Perf_Integrators_Lfdata.TermBuf.TermArray(11).Value                        100.0         0.0001                       100.0
» P


====> All 5 Comparisons Passed <====


TESTID: 2
This verify When If the step type is SpecStep, the StepAlt termination value is set to the  Specified step
altitude.If the aircraft is in climb, and a step was moved to the cruise phase from the climb phase the StepAlt
value is set to the climb specified step altitude, and the StepDist is activated with a distance value indicating
the step should begin immediately.

REQUIREMENTS UNDER EVALUATION: FMCS_19_20006025, FMCS_19_20006026, FMCS_19_20006027, FMCS_19_20006028, FMCS_19_20006029
                               FMCS_19_20006097, FMCS_19_20006098, FMCS_19_20006099, FMCS_19_20006102, FMCS_19_20006100
SUPPORTING REQUIREMENTS :      FMCS_19_21027005


INPUT                                                                                                  VALUE
----------------------------------------------------------------------------------------------  -------------------------
```

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

```
Test_Firstpass                                                                                              boolean'(true)
Test_Steptype                                                                                  CFP_PERF_STEP_IFTYPES.specstep
Test_LGB_Search                                                                                                           0
Perf_Preds_Lfdata.VTPlogic.Firstpass                                                                        boolean'(true)
Perf_Crzalt_Lfdata.LastCrzAlt.Valid                                                                        boolean'(false)
Perf_Preds_Lfdata.Vgbptr                                                                                             CLB2L
Perf_RTA_Lfdata.Pred_Pastrta                                                                                boolean'(true)
Perf_WTS_Lfdata.Always_Compute_Max_Speed                                                                   boolean'(false)
Perf_Integrators_Lfdata.IntProgBuf.Hprog                                                                           27004.0
Perf_Crzalt_Lfdata.Crzalt                                                                                          27000.0
Perf_Preds_Lfdata.Fltphase                                                                           FMCS_Base_Types.climb
Perf_Preds_Lfdata.NavPtr                                                                                                 1
Perf_Preds_Lfdata.PrevNavPtr                                                                                             2
Perf_LGB_Lfdata.LGB(2).Fpln_Data.SpAlt1Val                                                                  boolean'(true)
Perf_LGB_Lfdata.LGB(2).Fpln_Data.SpAlt1Pos                                                           AC_Position_Types.SC
Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Stepptr                                                                  0
Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Valid                                          boolean'(true)
Perf_Integrators_Lfdata.TermBuf.TermArray(6).Active                                                        boolean'(false)
Perf_Integrators_Lfdata.TermBuf.TermArray(6).Value                                                                     1.0
Perf_Integrators_Lfdata.IntProgBuf.Xprog                                                                            1000.0
Step_Ptr                                                                                                                 99
Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Steptype                                 CFP_PERF_STEP_IFTYPES.PastSpecStep


OUTPUT                                                          EXPECTED                TOLERANCE              ACTUAL
» P/F
----------------------------------------------------  -----------------------------  -------------  --------------------------
» ----
Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Stepptr                               2          (N/A)                         2
» P
Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Steptype
                                                     CFP_PERF_STEP_IFTYPES.specstep          (N/A)                  specstep
» P
Perf_Integrators_Lfdata.TermBuf.TermArray(6).Active                       boolean'(true)          (N/A)                      true
» P
Perf_Integrators_Lfdata.TermBuf.TermArray(6).Value                                1000.0         0.0001                    1000.0
» P
Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Valid  boolean'(false)          (N/A)                     false
» P


====> 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)

```
----------------------------------------------------  -----------------------------  -------------  -------------------------
» ----
Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Stepptr                               0          (N/A)                         0
» P
Perf_Integrators_Lfdata.TermBuf.TermArray(6).Active                  boolean'(false)          (N/A)                     false
» P
Perf_Integrators_Lfdata.TermBuf.TermArray(6).Value                                   1.0         0.0001                       1.0
» P
Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Valid  boolean'(false)          (N/A)                     false
» P


====> All 4 Comparisons Passed <====


TESTID: 4
This verify When If the step type is SpecStep, the StepAlt termination value is set to the  Specified step
altitude.If the aircraft is not in climb, and the StepDist is invalid.

REQUIREMENTS UNDER EVALUATION: FMCS_19_20006025, FMCS_19_20006026, FMCS_19_20006027, FMCS_19_20006028, FMCS_19_20006029
                               FMCS_19_20006097, FMCS_19_20006098, FMCS_19_20006099, FMCS_19_20006102

SUPPORTING REQUIREMENTS :      FMCS_19_21027005


INPUT                                                                                                               VALUE
----------------------------------------------------------------------------------------------------------  -------------------------
Test_Firstpass                                                                                              boolean'(false)
Test_Steptype                                                                                  CFP_PERF_STEP_IFTYPES.specstep
Test_LGB_Search                                                                                                             0
Perf_Preds_Lfdata.VTPlogic.Firstpass                                                                         boolean'(true)
Perf_Crzalt_Lfdata.LastCrzAlt.Valid                                                                         boolean'(false)
Perf_Preds_Lfdata.Vgbptr                                                                                              CLB2L
Perf_RTA_Lfdata.Pred_Pastrta                                                                                 boolean'(true)
Perf_WTS_Lfdata.Always_Compute_Max_Speed                                                                    boolean'(false)
Perf_Integrators_Lfdata.IntProgBuf.Hprog                                                                            27004.0
Perf_Crzalt_Lfdata.Crzalt                                                                                           27000.0
Perf_Preds_Lfdata.Fltphase                                                                         FMCS_Base_Types.Descent
Perf_Preds_Lfdata.NavPtr                                                                                                  1
Perf_Preds_Lfdata.PrevNavPtr                                                                                              1
false
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)

```


OUTPUT                                                          EXPECTED                TOLERANCE               ACTUAL
» P/F
----------------------------------------------------  -----------------------------  -------------  -------------------------
» ----
Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Valid  boolean'(false)          (N/A)                      false
» P


====> All 1 Comparisons Passed <====


TESTID: 5
This verify When If the Step type is PastSpecStep, the StepAlt termination value is set to the current specified step
altitude, and the StepDist termination is activated with a distance value indicating the step should begin immediately
unless the step is moved due to maximum altitude restrictions,in which case the StepDist termination is set to the
estimated step point distance.  If the aircraft is in climb, and no previous step point exists,the StepAlt termination
value is set to the current specified step altitude, and the StepDist termination is activated with a distance value
indicating the step should begin immediately.

REQUIREMENTS UNDER EVALUATION: FMCS_19_20006025, FMCS_19_20006026, FMCS_19_20006027, FMCS_19_20006028, FMCS_19_20006029
                               FMCS_19_20006097, FMCS_19_20006098, FMCS_19_20006099, FMCS_19_20006102, FMCS_19_20006100

SUPPORTING REQUIREMENTS :      FMCS_19_21027005


INPUT                                                                                                         VALUE
--------------------------------------------------------------------------------------------------  -------------------------
Test_Firstpass                                                                                             boolean'(false)
Test_Steptype                                                                                  CFP_PERF_STEP_IFTYPES.PastSpecStep
Test_LGB_Search                                                                                                          0
Perf_Preds_Lfdata.VTPlogic.Firstpass                                                                        boolean'(true)
Perf_Crzalt_Lfdata.LastCrzAlt.Valid                                                                        boolean'(false)
Perf_Preds_Lfdata.Vgbptr                                                                                             CLB2L
Perf_RTA_Lfdata.Pred_Pastrta                                                                                boolean'(true)
Perf_WTS_Lfdata.Always_Compute_Max_Speed                                                                   boolean'(false)
Perf_Integrators_Lfdata.IntProgBuf.Hprog                                                                           27004.0
Perf_Crzalt_Lfdata.Crzalt                                                                                          27000.0
Perf_Preds_Lfdata.Fltphase                                                                            FMCS_Base_Types.climb
Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Stepptr                                                                 0
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)

```
Perf_Crzalt_Lfdata.LastCrzAlt.Valid                                                     boolean'(false)
Perf_Preds_Lfdata.Vgbptr                                                                          CLB2L
Perf_RTA_Lfdata.Pred_Pastrta                                                             boolean'(true)
Perf_WTS_Lfdata.Always_Compute_Max_Speed                                                boolean'(false)
Perf_Integrators_Lfdata.IntProgBuf.Hprog                                                        27004.0
Perf_Crzalt_Lfdata.Crzalt                                                                       27000.0
Perf_Integrators_Lfdata.TermBuf.TermArray(6).Active                                     boolean'(false)
Perf_Integrators_Lfdata.TermBuf.TermArray(6).Value                                               9999.9
Perf_Integrators_Lfdata.IntProgBuf.Xprog                                                        10000.0
~~Idx_Profile_Ifdata.Ialtprofptrec(0)(1).Dtd.Valid                                         boolean'(true)~~
~~Idx_Profile_Ifdata.Ialtprofptrec(0)(1).Dtd.Data                                                  1000.0~~
Perf_Crz_Pkg.Opt_Step_PND_Ptr                                                                         1
Perf_WTS_Lfdata.Previous_Pass_Info_Rec.Step(1).Distance.Data                                     1000.0
Perf_WTS_Lfdata.Previous_Pass_Info_Rec.Step(1).Distance.Valid                            boolean'(true)
Perf_Integrators_Lfdata.TermBuf.TermArray(11).Value                                                 1.0
Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Valid                       boolean'(true)
Perf_Integrators_Lfdata.TermBuf.TermArray(11).Active                                    boolean'(false)


OUTPUT                                                            EXPECTED             TOLERANCE             ACTUAL
» P/F
---------------------------------------------------- ----------------------------- ------------- -------------------------
» ----
Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Valid  boolean'(false)          (N/A)                    false
» P
Perf_Integrators_Lfdata.TermBuf.TermArray(11).Active                 boolean'(true)          (N/A)                     true
» P
Perf_Integrators_Lfdata.TermBuf.TermArray(11).Value                          1000.0         0.0001                   1000.0
» P


====> All 3 Comparisons Passed <====


TESTID: 7
This verify When If the step type is Opt, it is determined whether:  both a new step altitude
and gross weight need to be computed.

REQUIREMENTS UNDER EVALUATION: FMCS_19_20006025, FMCS_19_20006026, FMCS_19_20006027, FMCS_19_20006028, FMCS_19_20006029
                               FMCS_19_20006097, FMCS_19_20006098, FMCS_19_20006099, FMCS_19_20006102

SUPPORTING REQUIREMENTS :      FMCS_19_21027005
```

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

```


INPUT                                                                                                     VALUE
-----------------------------------------------------------------------------------------------  -------------------------
Test_Firstpass                                                                                        boolean'(false)
Test_Steptype                                                                                     CFP_PERF_STEP_IFTYPES.Opt
Test_LGB_Search                                                                                                         0
Perf_Preds_Lfdata.VTPlogic.Firstpass                                                                   boolean'(true)
Perf_Crzalt_Lfdata.LastCrzAlt.Valid                                                                   boolean'(false)
Perf_Preds_Lfdata.Vgbptr                                                                                        CLB2L
Perf_RTA_Lfdata.Pred_Pastrta                                                                           boolean'(true)
Perf_WTS_Lfdata.Always_Compute_Max_Speed                                                              boolean'(false)
Perf_Integrators_Lfdata.IntProgBuf.Hprog                                                                        907.0
Perf_Crzalt_Lfdata.Crzalt                                                                                       900.0
Perf_Crz_Pkg.Step_Size.Valid                                                                           boolean'(true)
Perf_Crz_Pkg.Step_Size.Data                                                                                     100.0
Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Stepptr                                                           99
Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Valid                                     boolean'(true)
Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Opt_Stepalt                                                     0.0
Perf_Integrators_Lfdata.TermBuf.TermArray(11).Active                                                  boolean'(false)


OUTPUT                                                               EXPECTED              TOLERANCE             ACTUAL
» P/F
----------------------------------------------------  ----------------------------  -------------  -------------------------
» ----
Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Stepptr                        0          (N/A)                          0
» P
Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Valid  boolean'(false)          (N/A)                      false
» P
Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Opt_Stepalt              1000.0         0.0001                     1000.0
» P
Perf_Integrators_Lfdata.TermBuf.TermArray(11).Active                boolean'(true)          (N/A)                       true
» P


====> All 4 Comparisons Passed <====


TESTID: 8
This verify When If the step type is Opt, it is determined whether:  both a new step altitude
and gross weight need to be computed.
```

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

```

REQUIREMENTS UNDER EVALUATION: FMCS_19_20006025, FMCS_19_20006026, FMCS_19_20006027, FMCS_19_20006028, FMCS_19_20006029,
                               FMCS_19_20006097, FMCS_19_20006098, FMCS_19_20006099, FMCS_19_20006102

SUPPORTING REQUIREMENTS :      FMCS_19_21027005


INPUT                                                                                                         VALUE
---------------------------------------------------------------------------------------------------  -------------------------
Test_Firstpass                                                                                            boolean'(false)
Test_Steptype                                                                                   CFP_PERF_STEP_IFTYPES.Opt
Test_LGB_Search                                                                                                         0
Perf_Preds_Lfdata.VTPlogic.Firstpass                                                                       boolean'(true)
Perf_Crzalt_Lfdata.LastCrzAlt.Valid                                                                       boolean'(false)
Perf_Preds_Lfdata.Vgbptr                                                                                            CLB2L
Perf_RTA_Lfdata.Pred_Pastrta                                                                               boolean'(true)
Perf_WTS_Lfdata.Always_Compute_Max_Speed                                                                  boolean'(false)
Perf_Integrators_Lfdata.IntProgBuf.Hprog                                                                          80006.0
Perf_Crzalt_Lfdata.Crzalt                                                                                         80000.0
Perf_Crz_Pkg.Step_Size.Valid                                                                              boolean'(false)
Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Stepptr                                                               99
Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Valid                                         boolean'(true)
Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Opt_Stepalt                                                         0.0


OUTPUT                                                          EXPECTED                TOLERANCE               ACTUAL
» P/F
------------------------------------------------------  ----------------------------  -------------  -------------------------
» ----
Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Stepptr                                 0          (N/A)                         0
» P
Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Valid  boolean'(false)          (N/A)                     false
» P
Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Opt_Stepalt              81000.0         0.0001                   81000.0
» P
Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Steptype
                                                       CFP_PERF_STEP_IFTYPES.Nostep          (N/A)                    nostep
» P


====> All 4 Comparisons Passed <====

```

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

```

TESTID: 9
This verify When If the Step type is PastSpecStep, the StepAlt termination value is set to
the current specified step altitude.

REQUIREMENTS UNDER EVALUATION: FMCS_19_20006025, FMCS_19_20006026, FMCS_19_20006027, FMCS_19_20006028, FMCS_19_20006029
                               FMCS_19_20006097, FMCS_19_20006098, FMCS_19_20006099, FMCS_19_20006102

SUPPORTING REQUIREMENTS :      FMCS_19_21027005


INPUT                                                                                                          VALUE
--------------------------------------------------------------------------------------------------  --------------------------
Test_Firstpass                                                                                                 boolean'(false)
Test_Steptype                                                                                       CFP_PERF_STEP_IFTYPES.PastSpecStep
Test_LGB_Search                                                                                                              0
Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Stepptr                                                                     0
Perf_Preds_Lfdata.Navptr                                                                                                     1
Perf_Preds_Lfdata.Fltphase                                                                                FMCS_Base_Types.climb
Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Valid                                              boolean'(true)
Perf_Preds_Lfdata.PrevNavPtr                                                                                                 0
Perf_Integrators_Lfdata.TermBuf.TermArray(6).Active                                                            boolean'(false)
Perf_Integrators_Lfdata.TermBuf.TermArray(6).Value                                                                         0.0
Perf_Integrators_Lfdata.IntProgBuf.Xprog                                                                               10000.0
true
Step_Ptr                                                                                                                    99


OUTPUT                                                          EXPECTED               TOLERANCE              ACTUAL
» P/F
----------------------------------------------------  -----------------------------  --------------  --------------------------
» ----
Perf_Integrators_Lfdata.TermBuf.TermArray(6).Active                   boolean'(true)          (N/A)                        true
» P
Perf_Integrators_Lfdata.TermBuf.TermArray(6).Value                           10000.0         0.0001                     10000.0
» P
Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Stepptr                           1          (N/A)                           1
» P
Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Valid  boolean'(false)          (N/A)                       false
» P


```

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

```
====> All 4 Comparisons Passed <====


TESTID: 10
This verify When If the Step type is PastSpecStep, the StepAlt termination value is set to
the current specified step altitude.

REQUIREMENTS UNDER EVALUATION: FMCS_19_20006025, FMCS_19_20006026, FMCS_19_20006027, FMCS_19_20006028, FMCS_19_20006029
                               FMCS_19_20006097, FMCS_19_20006098, FMCS_19_20006099, FMCS_19_20006102

SUPPORTING REQUIREMENTS :      FMCS_19_21027005


INPUT                                                                                                          VALUE
--------------------------------------------------------------------------------------------------  --------------------------
Test_Firstpass                                                                                                 boolean'(false)
Test_Steptype                                                                                   CFP_PERF_STEP_IFTYPES.PastSpecStep
Test_LGB_Search                                                                                                              0
Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Stepptr                                                                     0
Perf_Preds_Lfdata.Navptr                                                                                                     1
Perf_Preds_Lfdata.Fltphase                                                                             FMCS_Base_Types.Descent
Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Valid                                              boolean'(true)
Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Data                                                        500.0
Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Fixdistodest                                                           1000.0
Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Valid                                              boolean'(true)
Perf_Integrators_Lfdata.TermBuf.TermArray(6).Active                                                            boolean'(false)
Perf_Integrators_Lfdata.TermBuf.TermArray(6).Value                                                                         0.0


OUTPUT                                                          EXPECTED               TOLERANCE              ACTUAL
» P/F
----------------------------------------------------  -----------------------------  -------------  --------------------------
» ----
Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Stepptr                          1          (N/A)                           1
» P
Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Valid  boolean'(false)          (N/A)                       false
» P
Perf_Integrators_Lfdata.TermBuf.TermArray(6).Active                  boolean'(true)          (N/A)                        true
» P
Perf_Integrators_Lfdata.TermBuf.TermArray(6).Value                            500.0         0.0001                       500.0
» P

```

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

```

====> All 4 Comparisons Passed <====


TESTID: 11
This verify When If the Step type is PastSpecStep, the StepAlt termination value is set to
the current specified step altitude.

REQUIREMENTS UNDER EVALUATION: FMCS_19_20006025, FMCS_19_20006026, FMCS_19_20006027, FMCS_19_20006028, FMCS_19_20006029
                               FMCS_19_20006097, FMCS_19_20006098, FMCS_19_20006099, FMCS_19_20006102

SUPPORTING REQUIREMENTS :      FMCS_19_21027005


INPUT                                                                                                         VALUE
--------------------------------------------------------------------------------------------------  --------------------------
Test_Firstpass                                                                                                 boolean'(false)
Test_Steptype                                                                               CFP_PERF_STEP_IFTYPES.PastSpecStep
Test_LGB_Search                                                                                                              0
Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Stepptr                                                                     0
Perf_Preds_Lfdata.Navptr                                                                                                     1
Perf_Preds_Lfdata.Fltphase                                                                             FMCS_Base_Types.Descent
Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Valid                                             boolean'(false)
Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Data                                                        500.0
Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Fixdistodest                                                           1000.0
Perf_Integrators_Lfdata.TermBuf.TermArray(6).Active                                                            boolean'(false)
Perf_Integrators_Lfdata.TermBuf.TermArray(6).Value                                                                         0.0
Perf_Integrators_Lfdata.IntProgBuf.Xprog                                                                                 999.0


OUTPUT                                                          EXPECTED               TOLERANCE              ACTUAL
» P/F
----------------------------------------------------  -----------------------------  -------------  --------------------------
» ----
Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Stepptr                          1          (N/A)                           1
» P
Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Valid  boolean'(false)          (N/A)                       false
» P
Perf_Integrators_Lfdata.TermBuf.TermArray(6).Active                  boolean'(true)          (N/A)                        true
» P
Perf_Integrators_Lfdata.TermBuf.TermArray(6).Value                            999.0         0.0001                       999.0
» P
```

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

```


====> All 4 Comparisons Passed <====


TESTID: 12
This verify When If the Step type is PastSpecStep, the StepAlt termination value is set to
the current specified step altitude.

Robust test follows:
Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Data >
                                     Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Fixdistodest.
Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Valid is true.

REQUIREMENTS UNDER EVALUATION: FMCS_19_20006025, FMCS_19_20006026, FMCS_19_20006027, FMCS_19_20006028, FMCS_19_20006029
                               FMCS_19_20006097, FMCS_19_20006098, FMCS_19_20006099, FMCS_19_20006102, FMCS_19_20006100

SUPPORTING REQUIREMENTS :      FMCS_19_21027005


INPUT                                                                                                 VALUE
---------------------------------------------------------------------------------------------------  -------------------------
Test_Firstpass                                                                                        boolean'(false)
Test_Steptype                                                                                    CFP_PERF_STEP_IFTYPES.PastSpecStep
Test_LGB_Search                                                                                                             0
Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Stepptr                                                                    0
Perf_Preds_Lfdata.Navptr                                                                                                    1
Perf_Preds_Lfdata.Fltphase                                                                            FMCS_Base_Types.Descent
Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Valid                                            boolean'(true)
Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Data                                                      1000.0
Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Fixdistodest                                                           500.0
Perf_Integrators_Lfdata.TermBuf.TermArray(6).Active                                                          boolean'(false)
Perf_Integrators_Lfdata.TermBuf.TermArray(6).Value                                                                        0.0
Perf_Integrators_Lfdata.IntProgBuf.Xprog                                                                                999.0


OUTPUT                                                          EXPECTED                TOLERANCE             ACTUAL
» P/F
----------------------------------------------------  -----------------------------  -------------  -------------------------
» ----
Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Stepptr                          1          (N/A)                            1
» 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="DO178B_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)

```
            MemorySizeRoData="0x00161000"
            MemorySizeData="0x00614000"
            MemorySizeBss="0x0144b000"
            MemorySizePersistentData="0x00001000"
            MemorySizePersistentBss="0x00426000"/>
    <Ports>
    </Ports>
</ApplicationDescription>

```

File: CTP_B787_PERF_CRZINITE.ZIP\CTP_B787_PERF_CRZINITE_ADA.gpr

```
with "..\..\..\GPS\fm\fm.gpr";
project CTP_B787_PERF_CRZINITE_ADA is

   for Languages use ("Ada");
   for Object_Dir use "..\OBJ";
   for Exec_Dir use "..";
   for Source_Dirs use ("..\SRC\ADA_SRC");

   for Main use ("CTP_B787_PERF_CRZINITE.ADA");
   Tornado := external ("WIND_BASE");
   Hi_Scoe := external ("SCOE_BASE");
   Build_Use := external ("Build_Path");
   Hi_Platform := "wrSbc750gx_scoe";

   package Ide is
      for Compiler_Command ("ada") use "powerpc-wrs-vxworksae-gnatmake";
      for Gnatlist use "powerpc-wrs-vxworksae-gnatls";
      for Debugger_Command use "powerpc-wrs-vxworksae-gdb";
      for Program_Host use "ISS-session";
      for Communication_Protocol use "wtx";
   end Ide;

   package Builder is
      for Default_Switches ("ada") use ("--RTS=cert", "-j2", "-m",
      "-I" & Hi_Scoe & "\platforms\" & Hi_Platform & "\include",
      "-I"& Build_Use & "\CFG\CTP_B787_PERF_CRZINITE\SRC\Ada_src\StubSRC");
      for Executable ("CTP_B787_PERF_CRZINITE.ada") use "CTP_B787_PERF_CRZINITE";
   end Builder;

   package Binder is
      for Default_Switches ("ada") use ("-E", "-t",
```

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

```
                                        "-aO" & Build_Use & "\PLATFORM\CSW\LIB",
                                        "-aO" & Build_Use & "\PLATFORM\SCOE\LIB",
     "-aO" & Build_Use & "\CFG\CTP_B787_PERF_CRZINITE\OBJ");
   end Binder;

   package Compiler is
        for Default_Switches ("ada") use (
        "-gdwarf-2",
        "-ansi",
        "-gnatf",
        "-gnatn",
        "-gnato",
        "-fno-common",
        "-mstrict-align",
        "-fno-crossjumping",
        "-fno-strict-aliasing",
        "-fstack-check",
        "-I"& Build_Use & "\CFG\CTP_B787_PERF_CRZINITE\SRC\Ada_src\StubSRC");
   end Compiler;

   package Linker is
   for Default_Switches ("ada") use ("--LINK=ldppc",
      "-nostdlib",
      "-r",
      "-d",
                 Hi_Scoe & "\platforms\" & Hi_Platform & "\lib\adaLCH.PPC604gnu.cert.o",
                 Hi_Scoe & "\platforms\" & Hi_Platform & "\lib\tftp.PPC604gnu.cert.o",

         "-L" & Build_Use & "\LIB",
         "-L" & Build_Use & "\LIB\IO",
         "-L" & Build_Use & "\LIB\BSVC",
         "-L" & Build_Use & "\LIB\COM",
         "-L" & Build_Use & "\LIB\FM",
         "-L..\..\..\PLATFORM\CSW\LIB",
         "--start-group",
         "-l_bite_c_fmf",
         "-l_bsvc_c_fmf",
         "-l_ci_c",
         "-l_dbam_c",
         "-l_flxcore_c_fmf",
         "-l_flxprj_c_fmf",
         "-l_fpcore_c",
```

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

```
         "-l_fpprj_c",
         "-l_hmi_c",
         "-l_io_c_fmf",
         "-l_ltcore_c",
         "-l_psvc_c",
         --"-l_io",
         "-l_io_tmf",
         "-l_io_nav",
         "-l_io_fmf",
         "-l_com",
         "-l_bsvc",
         "-l_fm",
         "-lCSW_V1_0",
         "--end-group");
   end Linker;

   package Naming is
      for Specification_Suffix ("ada") use "_.ada";
      for Implementation_Suffix ("ada") use ".ada";
      for Separate_Suffix use ".ada";
   end Naming;

end CTP_B787_PERF_CRZINITE_ADA;

```

File: CTP_B787_PERF_CRZINITE.ZIP\CTP_B787_PERF_CRZINITE_DRV.ada

```
with Perf_Preds_Lfdata;
with Perf_Max_Opt_Lfdata;
with Perf_Integrators_Lfdata;
with Perf_Profile_Lfdata;
with Perf_Crz_Pkg;
with Perf_Crzalt_Lfdata;

with FMF_IO_FMF_OUT_DPKG;
with Efis_661_Ifdata;
with Fmf_Dual_Partition_Ifdata;
with Fmci_Event_In_Dpkg;
with Nam_Waypoint_Ifdata;
with Nam_Runway_Ifdata;
with Nam_Corte_Ifdata;
with Nam_Navaid_Ifdata;
with OPS_CDK_Page_Data_Mgr_Pkg;
```

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

```
with Options_And_Data_Pkg;
with Fmci_Memory_Page_Pkg;
with CKY_Key_Pkg;
with Fmci_Display_Pkg;
with Flx_Semaphore_Pkg;

with Ops_Aedb_Ifdata;
with FMCS_AEDB_INIT;
with Fmcs_Partition_Data_Pkg;
with OPS_Data_Retained_Pkg;

Package Body CTP_B787_PERF_CRZINITE_DRV is

procedure end_dummy is
begin
null;
end  end_dummy;

procedure  CTP_B787_PERF_CRZINITE_D is
begin
--  execute SUT
    if ( Start_SUT = 1 )  then

        if ( load_ge_config ) then

            Fmcs_Partition_Data_Pkg.Ops_Engine_Manufacturer          := Fmcs_Partition_Data_Pkg.ge;
            Fmcs_Partition_Data_Pkg.Ops_Minor_Airframe_Model         := Fmcs_Partition_Data_Pkg.minor_airframe_dash_8 ;
            Ops_Data_Retained_Pkg.Ops_Internal_EEC_Data.BOM_L_Data   := OPS_DATA_RETAINED_PKG.Engine_BOM_1;
            Ops_Data_Retained_Pkg.Ops_Internal_EEC_Data.BOM_R_Data   := OPS_DATA_RETAINED_PKG.Engine_BOM_1;
            Ops_Data_Retained_Pkg.OPS_Internal_EEC_Data.Rating_Data  := OPS_Data_Retained_Pkg.GEnx_1B64;

        else

            Fmcs_Partition_Data_Pkg.Ops_Engine_Manufacturer          := Fmcs_Partition_Data_Pkg.rr;
            Fmcs_Partition_Data_Pkg.Ops_Minor_Airframe_Model         := Fmcs_Partition_Data_Pkg.minor_airframe_dash_9;
            Ops_Data_Retained_Pkg.Ops_Internal_EEC_Data.BOM_L_Data   := OPS_DATA_RETAINED_PKG.Engine_BOM_1;
            Ops_Data_Retained_Pkg.Ops_Internal_EEC_Data.BOM_R_Data   := OPS_DATA_RETAINED_PKG.Engine_BOM_1;
            Ops_Data_Retained_Pkg.OPS_Internal_EEC_Data.Rating_Data  := OPS_Data_Retained_Pkg.Trent_1000_C;

        end if;

        Ops_Aedb_Ifdata.Aedb_Load_Signature := "FMCSAEDB";
```

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

```

        FMCS_AEDB_INIT.INITIALIZE (config_found   => CTP_CFG_Found,
                                   no_aedb_loaded => CTP_AEDB_not_Loaded,
                                   signature_fail => CTP_Sig_Fail,
                                   version_compat => CTP_Ver_Compat);

        Perf_Preds_Lfdata.VTPlogic.Firstpass := Test_Firstpass ;
        Perf_Crzalt_Lfdata.LastCrzAlt.Valid := False ;
        Perf_Preds_Lfdata.Vtpfplnindex := 0 ;
        Perf_Max_Opt_Lfdata.Optalt_Is_LTOA(0) := True ;
        Perf_Integrators_Lfdata.TermBuf.TermArray(5).Detected := False ;
        Perf_Profile_Lfdata.Step_Climb_Rec.CFP_Step_Data.Steptype := Test_Steptype ;

        Perf_Crz_Pkg.Crz_Predexec;

        end_dummy;

    end if;

end  CTP_B787_PERF_CRZINITE_D;

end CTP_B787_PERF_CRZINITE_DRV;
```

File: CTP_B787_PERF_CRZINITE.ZIP\CTP_B787_PERF_CRZINITE_DRV_.ada

```
with Cfp_Perf_Step_IFtypes;
with fmcs_base_types;
with AC_Position_Types;
with Portable_Types_Pkg;

package CTP_B787_PERF_CRZINITE_DRV is

Start_SUT                :   integer;
CTP_CFG_Found            :   boolean;
CTP_AEDB_not_Loaded      :   boolean;
CTP_Sig_Fail             :   boolean;
CTP_Ver_Compat           :   boolean;
load_ge_config           :   boolean;

-- Global test variables go here
--
Test_Steptype            : Cfp_Perf_Step_IFtypes.Step_Types ;
Test_Firstpass           : Boolean ;
```

File: CTP_B787_PERF_CRZINITE.ZIP\CTP_B787_PERF_CRZINITE_DRV_.ada (continued)

```
Test_LGB_Search          : Portable_Types_Pkg.Integer_32;
--
procedure  CTP_B787_PERF_CRZINITE_D;
procedure end_dummy;

end  CTP_B787_PERF_CRZINITE_DRV;
```

File: CTP_B787_PERF_CRZINITE.ZIP\module.xml

```
<!-- edited with XMLSpy v2005 rel. 3 U (http://www.altova.com) by Jerry (Honeywell) -->
<Module xmlns="ARINC653" xmlns:xi="http://www.w3.org/2001/XInclude"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                         xsi:schemaLocation="ARINC653 ./schema/VXCR_Module.xsd" Name="vxWorks">
    <CoreOS>
        <xi:include href="$(SCOE_BASE)/Platforms/$(PLATFORM)/coreos.xml"/>
    </CoreOS>
    <Applications>

        <Application Name="CTP_B787_PERF_CRZINITE">
            <xi:include href="CTP_B787_PERF_CRZINITE.xml"/>
        </Application>

    </Applications>
  <SharedDataRegions>
    <SharedData Name="APP-AEDB_DB">
      <SharedDataDescription Size="0x005DC000" CachePolicy="COPY_BACK" DataType="DATABASE" />
    </SharedData>

    <SharedData Name="APP-NAV_DB">
      <SharedDataDescription Size="0x02800000" CachePolicy="COPY_BACK" DataType="DATABASE" />
    </SharedData>

    <SharedData Name="APP-FMS_AMI">
      <SharedDataDescription Size="0x00001000" CachePolicy="COPY_BACK"  DataType="DATABASE" />
    </SharedData>

    <SharedData Name="APP-FMS_OSS">
      <SharedDataDescription Size="0x00001000" CachePolicy="COPY_BACK"  DataType="DATABASE" />
    </SharedData>

    </SharedDataRegions>
    <SharedLibraryRegions>
        <SharedLibrary Name="ssl">
```

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

```
            <xi:include href="$(SCOE_BASE)/Platforms/$(PLATFORM)/ssl.xml"/>
        </SharedLibrary>
    </SharedLibraryRegions>
    <Partitions>
        <Partition Id="1" Name="Test_Partition" Type="APP_PARTITION">
            <PartitionDescription>
                <Application NameRef="CTP_B787_PERF_CRZINITE"/>
                    <SharedDataRegion NameRef="APP-AEDB_DB" UserAccess="READ_ONLY" />
                    <SharedDataRegion NameRef="APP-NAV_DB" UserAccess="READ_ONLY" />
                    <SharedDataRegion NameRef="APP-FMS_AMI" UserAccess="READ_ONLY" />
                    <SharedDataRegion NameRef="APP-FMS_OSS" UserAccess="READ_ONLY" />
                            <SharedLibraryRegion NameRef="ssl"/>
                <Settings
        allocDisable="true" appsIdleRelinquishEnabled="0"
        appsPriority="-1" isrStackSize="0xffffffff"
        maxEventQStallDuration="INFINITE_TIME" maxGlobalFDs="10"
        numDrivers="0xffffffff" numFiles="0xffffffff"
        numLogMsgs="0xffffffff" numStackGuardPages="0xffffffff"
        numWorkerTasks="0" PartitionHMTable="DefaultPartitionHM"
        selSvrQSize="0xffffffff" syscallPermissions="0x0007cff0"
        watchDogDuration="0" RequiredMemorySize="0x04000000"
        fpExcEnable="1" />
            </PartitionDescription>
        </Partition>

    </Partitions>
    <Schedules>
         <Schedule Id="0" Name="schedule0" MajorFrame="0.0500" MinorFrame="0.000250">
                        <PartitionWindow PartitionNameRef="Test_Partition" Duration="0.030"  />
                        <PartitionWindow PartitionNameRef="SPARE" Duration="0.020"  />
        </Schedule>
    </Schedules>
      <Connections>
   </Connections>
   <!--  Health Monitor Settings  -->

      <xi:include href="$(SCOE_BASE)/Platforms/$(PLATFORM)/HealthMonitorConfig.xml" />

      <xi:include href="$(SCOE_BASE)/Platforms/$(PLATFORM)/ace.xml" />

</Module>

```

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_CREATE_POINT_SEP.ADA

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

with Perf_Point_Termination_Types; use Perf_Point_Termination_Types;
with standard_angle_pkg;           use standard_angle_pkg;
with fpp_interface_type;           use fpp_interface_type;
with Fmcs_Base_Types;              use Fmcs_Base_Types;
with Perf_Profile_Lfdata;          use Perf_Profile_Lfdata;
with Dst_Brg_Utilities_Pkg;

separate ( Perf_Lgb_Pkg )

procedure Create_Point
  (
   Event : in Perf_Point_Termination_Types.Termination_Type;
   Display_Suppressed : boolean := false
  ) is
--!
-- ============================================================================================
-- PURPOSE:    Create an LGB Point Layer Object for The Current Predicted State
-- ANCHOR:     PERF_CODE_00021
-- SOURCE:     FMFSDD; PERF_SDD_00001 |
--
-- DESCRIPTION: This procedure creates a Core Flight Planning point layer object record for the
--              current predicted aircraft state and stores it for later output by
--              OUTPUT_PREDS.
--
-- SPECIAL_CONSIDERATIONS:
--   None.
--
-- REVISION_HISTORY (787):
--  Date       SCR     Engineer
--  12/07/05   519.00  Pat Caulfield
--                     Initial creation.
```

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

```
--
--  01/19/06   787.01  Pat Caulfield
--                     Updated based on point record changes; added lat/lon output.
--
--  06/04/07  1359.20  Pat Caulfield
--                     Modified to set new segment_data fields and calculate position.
--
--  07/20/07  4003.00  Ravish
--                     Added Logic for storing Latitude and Longitude of Top Of Descent
--                     for outputting to IO
--
--  04/28/08  6951.00  Added setting of Display_Suppressed.
--
-- ===========================================================================================
--!


  Point_Data : Fpp_Interface_Type.Point_Type := Fpp_Interface_Type.Init_Point_Type;


  procedure Calculate_Position (Point : in out Fpp_Interface_Type.Point_Type;
                                Minileg : in Perf_Lgb_Minileg_Types.Minileg_Rec_Type) is

    Turn_Direction : constant array (Fpp_Interface_Type.Turn_Direction_Type) of
    Fmcs_Base_Types.Turn_Direction_Type :=
        ( Fpp_Interface_Type.Noturn => Fmcs_Base_Types.No_Turn,
          Fpp_Interface_Type.Right => Fmcs_Base_Types.Right_Turn,
          Fpp_Interface_Type.Left => Fmcs_Base_Types.Left_Turn,
          Fpp_Interface_Type.Either => Fmcs_Base_Types.Either);

    -- define PI
    PI : constant Portable_Types_Pkg.Float_32 := 3.14159265359;

    -- Variables for Sodano calls
    Dummy_Distance  : Portable_Types_Pkg.Float_32;
    Bearing         : Standard_Angle_Pkg.SAF_32;
    Temp_Bearing    : Standard_Angle_Pkg.SAF_32;

    -- Bearing from end to start of straight segments
    Reverse_Bearing : Standard_Angle_Pkg.SAF_32;

    Prof_Point_To_Seg_End_Dist : Portable_Types_Pkg.Float_32;
```

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

```

    --required for arc calculations
    Temp_Angle    : Standard_Angle_Pkg.SAF_32;

  begin
    Prof_Point_To_Seg_End_Dist := point.aircraft_state.distance_to_destination - Minileg.Common_data.fixdistodest;
    -- Lower limit Prof_Point_To_Seg_End_Dist to 0.0
    if Prof_Point_To_Seg_End_Dist <= 0.0 then
       Prof_Point_To_Seg_End_Dist := 0.0;
    end if;

    if (Minileg.Segment_Data.Segment_Path = Fpp_Interface_Type.Arc) then

      -- Compute the bearing from turn center to segment end.
      Dst_Brg_Utilities_Pkg.Sodanoinv
          ( Lat1  => Minileg.Segment_Data.Segment_Arc_Center.Lat,
            Lon1  => Minileg.Segment_Data.Segment_Arc_Center.Lon,
            Lat2  => Minileg.Efis_Data.Tolatlon.Lat,
            Lon2  => Minileg.Efis_Data.Tolatlon.Lon,
            Dist  => Dummy_Distance,
            Brg12 => Bearing,
            Brg21 => Temp_Bearing );

      -- Calculate and normalize the course change
      Course_Change := Standard_Angle_Pkg.SAF_32 (
         ( Prof_Point_To_Seg_End_Dist / Minileg.Segment_Data.Segment_Arc_Radius ) * 180.0 / PI );
      -- Limit course change; anything at or over 360 or at or under -360 is zero.
      if (Course_Change >= 360.0) or else (Course_Change <= -360.0) then
         Course_Change := 0.0;
      end if;

      if (Turn_Direction(Minileg.Segment_Data.Segment_Turn_Direction) = Fmcs_Base_Types.Right_Turn) then
        Temp_Angle := Standard_Angle_Pkg.Normalize( Bearing - Course_Change );
      else
        Temp_Angle := Standard_Angle_Pkg.Normalize( Bearing + Course_Change );
      end if;

      -- Compute the coordinates of the point.
      Dst_Brg_Utilities_Pkg.Sodanodir
          ( Lat1  => Minileg.Segment_Data.Segment_Arc_Center.Lat,
            Lon1  => Minileg.Segment_Data.Segment_Arc_Center.Lon,
            Brg12 => Temp_Angle,
```

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

```
            Dist  => abs (Minileg.Segment_Data.Segment_Arc_Radius),
            Lat2  => Point.Aircraft_State.Position.Lat,
            Lon2  => Point.Aircraft_State.Position.Lon,
            Brg21 => Temp_Bearing );

      Point.Point_Ac_State_Position_Valid := true;

    elsif (Minileg.Segment_Data.Segment_Path = Fpp_Interface_Type.Straight) then

      if Minileg.Efis_Data.Magnetic_North_Bearing then
        Reverse_Bearing := Standard_Angle_Pkg.Normalize( 180.0 + Minileg.Efis_Data.Outcourse +
            Minileg.Efis_Data.Magvar );
      else
        Reverse_Bearing := Standard_Angle_Pkg.Normalize
            ( 180.0 + Minileg.Efis_Data.Incourse );
      end if;

      -- Compute the coordinates of the point.
      Dst_Brg_Utilities_Pkg.Sodanodir
          ( Lat1  => Minileg.Efis_Data.ToLatLon.Lat,
            Lon1  => Minileg.Efis_Data.ToLatLon.Lon,
            Brg12 => Reverse_Bearing,
            Dist  => Prof_Point_To_Seg_End_Dist,
            Lat2  => Point.Aircraft_State.Position.Lat,
            Lon2  => Point.Aircraft_State.Position.Lon,
            Brg21 => Temp_Bearing );

      Point.Point_Ac_State_Position_Valid := true;

    else
      Point.Point_Ac_State_Position_Valid := false;
    end if;

  end Calculate_Position;

begin
  if not Perf_Preds_Lfdata.Vtplogic.Hold_Multiple_Laps then
    -- Populate a point record with current trajectory predictions data.

    Point_Data.Aircraft_State.Predictions_Stable := not Perf_Preds_Lfdata.Vtplogic.Firstpass;
    Point_Data.Aircraft_State.Cas := Perf_Integrators_Lfdata.Intprogbuf.Cas2;
    Point_Data.Aircraft_State.Tas := Perf_Integrators_Lfdata.Intprogbuf.Tas2;
```

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

```
    Point_Data.Aircraft_State.Mach := Perf_Integrators_Lfdata.Intprogbuf.Mach2;
    Point_Data.Aircraft_State.Wind := (Direction => Perf_Wind_Lfdata.Predwind.Dir, --
                                       Speed => Perf_Wind_Lfdata.Predwind.Mag);
    Point_Data.Aircraft_State.Isa_Deviation := Perf_Preds_Lfdata.Isadelta;
    Point_Data.Aircraft_State.Flight_Phase := Perf_Preds_Lfdata.Desiredphase;
    Point_Data.Aircraft_State.Ete := Perf_Integrators_Lfdata.Intprogbuf.Tprog;
    Point_Data.Aircraft_State.Gross_Weight := Perf_Integrators_Lfdata.Intprogbuf.Gwprog;
    Point_Data.Aircraft_State.Groundspeed := Perf_Integrators_Lfdata.Intprogbuf.Gndspd2;
    Point_Data.Aircraft_State.Fuel_Weight :=
      Perf_Integrators_Lfdata.Intprogbuf.Gwprog - Perf_Preds_Lfdata.Aircraft_State.Zfw.Data;
    Point_Data.Aircraft_State.Distance_To_Destination := Perf_Integrators_Lfdata.Intprogbuf.Xprog;
    Point_Data.Aircraft_State.Pressure_Altitude := Perf_Integrators_Lfdata.Intprogbuf.Hprog;
    Point_Data.Aircraft_State.Pressure_Altitude_Rate := Perf_Integrators_Lfdata.Intprogbuf.Rateofclb2;
    Point_Data.Aircraft_State.True_Track := Perf_Preds_Lfdata.Gsdata.Fplntrack;
    Point_Data.Aircraft_State.Fuel_Flow := Perf_Integrators_Lfdata.Intprogbuf.Fuelflow2;
    Point_Data.Aircraft_State.Gamma_Airmass := Perf_Integrators_Lfdata.Intprogbuf.Gamaair2;
    Point_Data.Aircraft_State.Acceleration := Perf_Integrators_Lfdata.Intprogbuf.Accel2;

    -- set the display suppressed flag true if:
    -- 1.  We're predicting an early descent.
    -- 2.  The display_suppressed input parameter is true.
    -- 3.  The predicted flight phase is descent or approach and there are no descent constraints.
    Point_Data.Display_Suppressed := Perf_Preds_Lfdata.Early_Descent or else Display_Suppressed or else
     (Perf_Preds_Lfdata.DesiredPhase >= Descent and then Perf_LGB_Lfdata.Last_Constraint_Index = 0);

    Point_Data.Event := Event;
    Point_Data.Priority := 1;

    -- Output segment data to the point for VSD and Separation Assurance to use.
    if Perf_Preds_Lfdata.Navptr in 1..Perf_Lgb_Minileg_Types.Max_Number_Minilegs then
      Point_Data.Segment_Index := Perf_Lgb_Lfdata.Lgb(Perf_Preds_Lfdata.Navptr).Segment_Data.Segment_Index;
      Point_Data.Segment_Arc_Center := Perf_Lgb_Lfdata.Lgb(Perf_Preds_Lfdata.Navptr).Segment_Data.Segment_Arc_Center;
      Point_Data.Segment_Arc_Radius := Perf_Lgb_Lfdata.Lgb(Perf_Preds_Lfdata.Navptr).Segment_Data.Segment_Arc_Radius;
      Point_Data.Segment_Path := Perf_Lgb_Lfdata.Lgb(Perf_Preds_Lfdata.Navptr).Segment_Data.Segment_Path;
      Point_Data.Segment_Discon_Follows := Perf_Lgb_Lfdata.Lgb(Perf_Preds_Lfdata.Navptr).Segment_Data.Segment_Discon_Follows;
      Point_Data.Segment_Turn_Direction := Perf_Lgb_Lfdata.Lgb(Perf_Preds_Lfdata.Navptr).Segment_Data.Segment_Turn_Direction;
      Point_Data.Segment_Not_Computed_Trajectory :=
» Perf_Lgb_Lfdata.Lgb(Perf_Preds_Lfdata.Navptr).Segment_Data.Segment_Not_Computed_Trajectory;
    end if;

    -- if the termination is a waypoint or segment endpoint, fill in the position with the segment's
    -- endpoint position from the current mini-leg.
```

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

```

    if (Event = Waypoint) or else (Event = Segment_Endpoint) or else (Event = Constrained_Des_Waypoint) then
      Point_Data.Aircraft_State.Position.Lat := Perf_Lgb_Lfdata.Lgb(Perf_Preds_Lfdata.Navptr).Efis_Data.Tolatlon.Lat;
      Point_Data.Aircraft_State.Position.Lon := Perf_Lgb_Lfdata.Lgb(Perf_Preds_Lfdata.Navptr).Efis_Data.Tolatlon.Lon;
      Point_Data.Point_AC_State_Position_Valid := true;

    elsif (Event = Start_Of_Predictions) then
      Point_Data.Aircraft_State.Position.Lat := Perf_Preds_Lfdata.Aircraft_State.Lat_lon.Data.Lat;
      Point_Data.Aircraft_State.Position.Lon := Perf_Preds_Lfdata.Aircraft_State.Lat_lon.Data.Lon;
      Point_Data.Point_AC_State_Position_Valid := Perf_Preds_Lfdata.Aircraft_State.Lat_lon.Valid;

      -- set the flight phase to the current aircraft phase.
      Point_Data.Aircraft_State.Flight_Phase := Perf_Preds_Lfdata.Fltphase;

    elsif not Point_Data.Point_AC_State_Position_Valid then

      if Perf_Preds_Lfdata.Navptr in 1..Perf_Lgb_Minileg_Types.Max_Number_Minilegs then

        -- need to calculate the point location directly
        -- using distance to destination and segment data.
        Calculate_Position (point_data, perf_lgb_lfdata.lgb(perf_preds_lfdata.navptr));
      end if;
        -- if the event is Top of Descent store Lat/Lon of the Position
        -- to output it to IO.
        -- SCR 4003.00
        if (Event= Top_Of_Descent) and then
           (Perf_LGB_Lfdata.Last_Constraint_Index >0) then

           Perf_Profile_Lfdata.Todperfdata.Position.Lat := Point_Data.Aircraft_State.Position.Lat;
            Perf_Profile_Lfdata.Todperfdata.Position.Lon := Point_Data.Aircraft_State.Position.Lon;
           Perf_Profile_Lfdata.Todperfdata.Position_Valid := true;

        end if;

    end if;

    -- Determine if the predicted speed should be tagged as a mach or a CAS value.

    if Perf_Preds_Lfdata.Tgtspdrec.Tgtspdtag = Casonly or else ( Perf_Preds_Lfdata.Tgtspdrec.Tgtspdtag = Casmach and then
        ( Point_Data.Aircraft_State.Pressure_Altitude < Perf_Preds_Lfdata.Tgtspdrec.Cmxalt ) ) then

      Point_Data.Aircraft_State.Speed_Command := Perf_Preds_Lfdata.Tgtspdrec.Cas;
```

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

```
    else
      Point_Data.Aircraft_State.Speed_Command := Perf_Preds_Lfdata.Tgtspdrec.Mach;
    end if;

    -- Copy the predictions data sequence counter from the snapshot of the flight plan header
    -- currently being used by predictions into the point record.  The header and point record
    -- counters matching indicates the predicted data in general is valid.

    Point_Data.Aircraft_State.Predictions_Data_Seq_Counter := Perf_Lgb_Lfdata.Lgb_Header.Prddataseq;

    -- If this is the initial condition point, start at one.  If not, increment the point count.

    if Event = Start_Of_Predictions then
      Point_Count := 1;
    else
      if Point_Count = Perf_Lgb_Minileg_Types.Point_Index_Type'Last then
        Point_Count := 1;
      else
        Point_Count := Point_Count + 1;
      end if;
    end if;

    -- Store the point record within this package; it will be output when we're done predicting the route.

    Points( Point_Count ) := Point_Data;

  end if;
end Create_Point;
```

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_OUTPUT_PREDS_SEP.ADA

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

-- 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_Ifdata;
  22    22 with Idx_Profile_Ifdata;
  23    23 -- perf data objects
  24    24 with Perf_Crzalt_Lfdata;
  25    25 with Perf_Idx_Crzalt_Lfdata;
  26    26 with Perf_Idx_Msg_Flags_Lfdata;
  27    27 with Perf_Idx_Top_Of_Des_Lfdata;
  28    28 with Perf_Lgb_Lfdata;
  29    29 with Perf_Msg_Flags_Lfdata;
  30    30 with Perf_Offpath_Des_Lfdata;
  31    31 with Perf_Offpath_Descent_Ifdata;
  32    32 with Perf_Profile_Lfdata;
  33    33 with Perf_Rta_Lfdata;
  34    34 with Perf_Task_Control_Lfdata;
  35    35 with Perf_Top_Of_Des_Lfdata;
  36    36 with Perf_Vdu_Lfdata;
  37    37 with Perf_Vtp_Lfdata;
  38    38 -- global packages
  39    39 with Apex_Processes;
  40    40 with Dst_Brg_Utilities_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_Lfdata;
        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)

```
--  03/11/08  5839.00  Pat Caulfield
--  Output the inssufficient fuel messages (Help window and Eicas) based on the message flags.
--  This used to be done in DST_ESTIMATES.
--
-- ==========================================================================================
--!

   New_Lock_Level : Apex_Processes.Lock_Level_Type;
   Lock_Status : Apex_Types.Return_Code_Type;
   Stepclbrec : Cfp_Perf_Step_Iftypes.Step_Clb_Rec_Type;

begin
~~   -- lock preemption so that we can get this out quickly.~~
  --VDU dump for window pass using PERF copy of LGB buffer.
  if (Perf_Preds_Lfdata.Dump_Window_Preds /= 0) then
     Perf_Vdu_Utils.Dump_Window_predictions;
  end if;

~~   Apex_Processes.Lock_Preemption ( New_Lock_Level, Lock_Status );~~

  if (Perf_Rta_Lfdata.Rta_Window_Task) then

    if(Perf_Preds_Lfdata.Perf_Pass = Perf_preds_lftypes.Early) then

» 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);

    else

» 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);

    end if;

  else
    -- lock preemption so that we can get this out quickly.
    Apex_Processes.Lock_Preemption ( New_Lock_Level, Lock_Status );

~~   if Perf_Lgb_Lfdata.Minileg_Fpln_Count_Snapshot = Perf_Efis_Lgb_Mgr_Pkg.Minileg_Fpln_Count (Perf_Lgb_Lfdata.Act_Prov_Index) ~~
» ~~then~~
    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)

```

      Perf_Lgb_Lfdata.Pass_Thrown_Out := false;

      -- transfer the minileg flight plan from our copy in Perf_Lgb_Lfdata to the Perf_Efis_Lgb_Mgr_Pkg.

      for Mini_Index in 1..Perf_Lgb_Lfdata.Lgb_Header.Lastfplnptr loop
         Perf_Efis_Lgb_Mgr_Pkg.Put_Minileg ( Perf_Lgb_Lfdata.Act_Prov_Index, Perf_Lgb_Lfdata.Lgb( Mini_Index ), Mini_Index );
      end loop;

      -- Transfer the point layer data that was stored internally to this package by Create_Point calls to the
      -- Perf_Efis_Lgb_Mgr_Pkg.

      if Perf_Preds_Lfdata.Vtplogic.Haveactfpln then -- only for active flight plan
         for Point_Index in 1..Point_Count loop
            Perf_Efis_Lgb_Mgr_Pkg.Put_Point ( Points( Point_Index ), ( Point_Index = 1 ) );
         end loop;
      end if; -- active flight plan

      -- Assuming we have a flight plan, signal EFIS PATH that we're done predicting either
      -- the active or provisional route, and have output predictions to the PERF_EFIS_LGB_MGR_PKG
      -- (done up above via the output_preds call) for them to store into the LGB.

      if Perf_Preds_Lfdata.Vtplogic.Haveactfpln then
         Ops_Perf_Change_Flags_Mgr_Pkg.Put_Perf_Efis_Do_Act( True );
      else
         Ops_Perf_Change_Flags_Mgr_Pkg.Put_Perf_Efis_Do_Prov( True );
      end if;

      -- Un-lock preemption to free up processing

      Apex_Processes.Unlock_Preemption ( New_Lock_Level, Lock_Status );

      -- Clear the EICAS message when we're predicting the active fpln, and
      -- the most recent evaluation (up above) doesn't show insufficient fuel,
      -- but the message is displayed.  Clear the help window when the same is
      -- true for the provisional flight plan.

      if Perf_Preds_Lfdata.Vtplogic.Haveactfpln then

         -- if the insufficient fuel Eicas message needs to be displayed, do so.
         if Perf_Msg_Flags_Lfdata.Insufficient_Fuel_Eicas and then
            not Perf_Idx_Msg_Flags_Lfdata.Iinsuffuel_Eicas then
```

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

```
            Fmf_Io_Fmf_Out_Dpkg.Insufficient_Fuel_Discr.Put ( Data => True, Is_Valid => True );
         end if;

         -- if the insufficient fuel message Eicas message needs to be cleared, do so.
         if not Perf_Msg_Flags_Lfdata.Insufficient_Fuel_Eicas and then
            Perf_Idx_Msg_Flags_Lfdata.Iinsuffuel_Eicas then

            Fmf_Io_Fmf_Out_Dpkg.Insufficient_Fuel_Discr.Put ( Data => False, Is_Valid => True );
         end if;
      else  -- provisional fpln predictions

         -- if the insufficient fuel Help Window message needs to be displayed, do so.
         if Perf_Msg_Flags_Lfdata.Insufficient_Fuel_Help and then
            not Perf_Idx_Msg_Flags_Lfdata.Iinsuffuel_Help then

            Fmci_Spad_Manager_Pkg.Display_Message (Message_Id => Scratch_Pad_Iftypes.Insufficient_Fuel);
         end if;

         -- if the insufficient fuel Help Window message needs to be cleared, do so.
         if not Perf_Msg_Flags_Lfdata.Insufficient_Fuel_Help and then
            Perf_Idx_Msg_Flags_Lfdata.Iinsuffuel_Help then

            Fmci_Spad_Manager_Pkg.Clear_Message (Message_Id => Scratch_Pad_Iftypes.Insufficient_Fuel);
         end if;
      end if;

      -- For active flight plan, perform ATC conditional clearance processing.
      if Perf_Preds_Lfdata.Vtplogic.Haveactfpln then
         Perf_Atc_Cond_Pkg.Atc_Cond_Clearance_Exec;
      end if;

      -- mark the first-pass flag false

      Perf_Task_Control_Lfdata.Idofirstpass( Perf_Preds_Lfdata.Vtpfplnindex ) := False;

      --  Save local Perf data items into interfunctional IDX data packages

      Idx_Profile_Ifdata.Itocperfdata( Perf_Preds_Lfdata.Vtpfplnindex ) := Perf_Profile_Lfdata.Tocperfdata;
      Idx_Profile_Ifdata.Itodperfdata( Perf_Preds_Lfdata.Vtpfplnindex ) := Perf_Profile_Lfdata.Todperfdata;

```
~~Perf_Idx_Top_Of_Des_Lfdata.Itodcrzspeed(Perf_Preds_Lfdata.Vtpfplnindex) := Perf_Top_Of_Des_Lfdata.Todcrzspeed;~~

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

```

      Idx_Profile_Ifdata.Idestinrec( Perf_Preds_Lfdata.Vtpfplnindex ) := Perf_Profile_Lfdata.Destinrec;
      Idx_Profile_Ifdata.Ialtprofptrec( Perf_Preds_Lfdata.Vtpfplnindex ) := Perf_Profile_Lfdata.Altprofptrec;
      Idx_Profile_Ifdata.Ilvlataltrec( Perf_Preds_Lfdata.Vtpfplnindex ) := Perf_Profile_Lfdata.Lvlataltrec;
      Idx_Profile_Ifdata.Ieodperfdata( Perf_Preds_Lfdata.Vtpfplnindex ) := Perf_Profile_Lfdata.Eodperfdata;

      Perf_Idx_Crzalt_Lfdata.Ilastcrzalt( Perf_Preds_Lfdata.Vtpfplnindex ) := Perf_Crzalt_Lfdata.Lastcrzalt;
      Perf_Idx_Msg_Flags_Lfdata.Iunablecrzalt( Perf_Preds_Lfdata.Vtpfplnindex ) := Perf_Msg_Flags_Lfdata.Unablecrzalt;
      Perf_Idx_Msg_Flags_Lfdata.Iinsuffuel_Help := Perf_Msg_Flags_Lfdata.Insufficient_Fuel_Help;
      Perf_Idx_Msg_Flags_Lfdata.Iinsuffuel_Eicas := Perf_Msg_Flags_Lfdata.Insufficient_Fuel_Eicas;
      Perf_Idx_Msg_Flags_Lfdata.Iabovemaxalt( Perf_Preds_Lfdata.Vtpfplnindex ) := Perf_Msg_Flags_Lfdata.Abovemaxalt;
      Perf_Idx_Msg_Flags_Lfdata.Imax_Alt_Msg_Latch( Perf_Preds_Lfdata.Vtpfplnindex ) := Perf_Msg_Flags_Lfdata.Max_Alt_Msg_Latch;
      Perf_Idx_Msg_Flags_Lfdata.Imax_Alt_Msg_Leg_Indx( Perf_Preds_Lfdata.Vtpfplnindex ) :=
» Perf_Msg_Flags_Lfdata.Max_Alt_Msg_Leg_Indx;
      Perf_Idx_Top_Of_Des_Lfdata.Itoddata( Perf_Preds_Lfdata.Vtpfplnindex ) := Perf_Top_Of_Des_Lfdata.Toddata;
      Perf_Idx_Top_Of_Des_Lfdata.Itodcrzspeed( Perf_Preds_Lfdata.Vtpfplnindex ) := Perf_Top_Of_Des_Lfdata.Todcrzspeed;
      Perf_Rta_Lfdata.Idx_Data(Perf_Preds_Lfdata.Vtpfplnindex):= Perf_Rta_Lfdata.Perf_Rta_Data;
      Perf_WTS_Lfdata.Idx_Rta_CI(Perf_Preds_Lfdata.Vtpfplnindex) := Perf_WTS_Lfdata.Rta_CI;

      -- output predictions that only are done for the active flight plan

      if Perf_Preds_Lfdata.Vtplogic.Haveactfpln then
         -- Output fixinfo predictions if active flight plan.

         for I in Fix_Info_Iftypes.Fixinfo_Array_Type' range loop
            Ops_Cdk_Common_Mgr_Pkg.Put_Fix_Info_Pred_Data(Page_Number => I,
                                                          New_Data    => Fix_Info_Ifdata.Fix_Info(I));
         end loop;

         --  Supply efisrad with new top of climb, step climb,
         --  top of descent, & end of descent records if the
         --  active flight plan was processed.

         Idx_Profile_Ifdata.Inew_Profpts( Perf_Preds_Lfdata.Vtpfplnindex ) := True;

         --Stores Latitude/Longitude of TOD to IO from IDX
         --IO uses Float_64 so type conversion is needed from SAF_32 type to Float_64

         -- Lock to prevent the Float_64 from being corrupted during the write.
         Apex_Processes.Lock_Preemption ( New_Lock_Level, Lock_Status );

         Fmf_IO_Fmf_out_dpkg.TOD_Position_Latitude.Put(Portable_Types_Pkg.Float_64(Idx_Profile_Ifdata.Itodperfdata(
```

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

```
» Perf_Preds_Lfdata.Vtpfplnindex ).Position.Lat),
                                                       Idx_Profile_Ifdata.Itodperfdata( Perf_Preds_Lfdata.Vtpfplnindex
» ).Position_Valid);

         Fmf_IO_Fmf_out_dpkg.TOD_Position_Longitude.Put(Portable_Types_Pkg.Float_64(Idx_Profile_Ifdata.Itodperfdata(
» Perf_Preds_Lfdata.Vtpfplnindex ).Position.Lon),
                                                        Idx_Profile_Ifdata.Itodperfdata( Perf_Preds_Lfdata.Vtpfplnindex
» ).Position_Valid);

         --  Unlock to free it up
         Apex_Processes.Unlock_Preemption ( New_Lock_Level, Lock_Status );

      end if;

      -- write recommended takeoff time to manager

      Ops_Cdk_Perf_Pdb_Mgr_Pkg.Put_Perf_Takeoff_Time(Takeoff_Time =>
» Perf_Rta_Lfdata.Idx_Data(Perf_Preds_Lfdata.Vtpfplnindex).Rcmd_Takeoff,
                                                     Fpln_Index   => Perf_Preds_Lfdata.Vtpfplnindex);

      --  Fetch a copy of the step climh data record from the shared object manager.

      Ops_Cdk_Perf_Pdb_Mgr_Pkg.Get_Stepclbrec( Perf_Preds_Lfdata.Vtpfplnindex, Stepclbrec );

      -- if we've just sequenced a step climb point...

      if ( ( Perf_Profile_Lfdata.Step_Climb_Rec.Cfp_Step_Data.Steptype = Cfp_Perf_Step_Iftypes.Pastspecstep ) and then
          ( Perf_Profile_Lfdata.Step_Climb_Rec.Cfp_Step_Data.Spec_Stepalt /=
          Stepclbrec.Cfp_Step_Data.Spec_Stepalt ) ) then

        --  update s/c record with s/c point just sequenced

        Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Data := Stepclbrec.Cfp_Step_Data.Fixdistodest;

        Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Disttodest.Valid := False;

        -- allow preds to determine if it's maxalt limited

        Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data.Eta.Valid := False;

      end if;

```

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

```
      --  store computed s/c data
      Ops_Cdk_Perf_Pdb_Mgr_Pkg.Put_Stepclbrec_Perf( Perf_Preds_Lfdata.Vtpfplnindex,
» Perf_Profile_Lfdata.Step_Climb_Rec.Perf_Step_Data );
      Ops_Cdk_Perf_Pdb_Mgr_Pkg.Put_Stepclbrec_Perfdataval( Perf_Preds_Lfdata.Vtpfplnindex,
» Perf_Profile_Lfdata.Step_Climb_Rec.Perfdataval );

      -- Do not update the indexed Rta_Iter_Counter until the end of preds
      Perf_WTS_Lfdata.Rta_Iter_Counter(Perf_Preds_Lfdata.Vtpfplnindex) :=
        Perf_WTS_Lfdata.New_Rta_Iter_Counter(Perf_Preds_Lfdata.Vtpfplnindex);

      -- Save the WTS Flat_Bias_Factor to be used for the next pass
      -- of predictions of this flight plan
      Perf_WTS_Lfdata.Idx_Flat_Bias_Factor(Perf_Preds_Lfdata.Vtpfplnindex) :=
         Perf_WTS_Lfdata.New_Flat_Bias_Factor;

      -- 'Push down' the WTS and RTA info saved from previous passes of predictions
      -- note: We do not currently plan to use data from passes other than the most recent
      -- pass, however it is helpful to have this data for debugging and it used to be
      -- stored for Perf_WTS_Lfdata.Num_Values_Type passes so we will continue to save it
      -- in case it is needed for something.
      for Push_Index in reverse Perf_WTS_Lfdata.Num_Stored_Passes loop
         if Push_index < Perf_WTS_Lfdata.Num_Stored_Passes'last then
            Perf_WTS_Lfdata.Idx_Pass_Info_Rec (Perf_Preds_Lfdata.Vtpfplnindex , Push_Index + 1) :=
               Perf_WTS_Lfdata.Idx_Pass_Info_Rec (Perf_Preds_Lfdata.Vtpfplnindex , Push_Index);
         end if;
      end loop;

      -- Save important WTS and RTA info from this pass of predictions,
      -- to be used in future passes [to support the follwing requirements that
      -- need data from the prior trip prediction pass: PERF_SRD_B_00413,
      -- PERF_SRD_B_00414, PERF_SRD_B_00415, PERF_SRD_B_00416]
      Perf_WTS_Lfdata.Current_Pass_Info_Rec.Pass_Info_Valid := True;
      Perf_WTS_Lfdata.Idx_Pass_Info_Rec (Perf_Preds_Lfdata.Vtpfplnindex , Perf_WTS_Lfdata.Num_Values_Type'first) :=
         Perf_WTS_Lfdata.Current_Pass_Info_Rec;
   else -- the results of this pass of predictions is being thrown out (by not storing them out)

      -- Un-lock preemption to free up processing

      Apex_Processes.Unlock_Preemption ( New_Lock_Level, Lock_Status );

      Perf_Lgb_Lfdata.Pass_Thrown_Out := true;

```

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)

```
with Perf_Lgb_Minileg_Types;
with Perf_Preds_Lftypes;
use Perf_Preds_Lftypes;
with Standard_Angle_Pkg;
use Standard_Angle_Pkg;
use Fmcs_Base_Types;

package body Perf_LGB_Pkg is
  --!
  -- ANCHOR:      FMCS_19_21023511
  -- SOURCE:      FMFSDD; FMCS_19_21023000 |
  --| @DESCRIPTION: This package body contains the procedure bodies
  --|               (declared as separates) for manipulation of PERF
  --|               predictions copy of the LGB
  --|               (Perf_LGB_Lfdata.LGB)
  --
  -- SPECIAL_CONSIDERATIONS:
  --
  -- REVISION_HISTORY:
  --   DATE           SCR #        Programmer              DRCM#
  -- =====================================================================
  --   12/18/95       8011         B. O'Laughlin           M777B_FMF_00548
  -- Added Lgb_Seq_Rta_Leg for processing predicted RTA leg sequence.
  --
  -- ====================== 787 HISTORY STARTS HERE =======================
  --
  --   12/06/05       519.00       Pat Caulfield
  --   Added new procedures Create_Point and Output_Preds, as well as internal
  --   temporary (during flight plan predictions) storage of the points.
  --
  --   01/18/06       787.01       Pat Caulfield
  --   Cleaned up the point array declaration now that the event field is defined
  --   in the point record.  Also renamed Create_Point's parameter to Event.
  --
  --   07/28/08      7562.00       Pat Caulfield
  --   Moved Course_Change here from Lgb_Seq_Leg to aid debugging.
  --
  --   08/07/08      6676.00       Pat Caulfield
  --   Added new parameter Display_Suppressed to Create_Point.
  --
  --!

```

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

```
   -- V A R I A B L E   D E C L A R A T I O N S --

   Points : Perf_Lgb_Minileg_Types.Point_Array_Type;

   Point_Count : Perf_Lgb_Minileg_Types.Point_Index_Type;

   -- DESCRIPTION Difference between current and previous flight plan tracks
   Course_Change : Standard_Angle_Pkg.Saf_32;

   Hold_Idx        : Boolean := False;
   Hold_In_Descent : Boolean := False;
   Valid_Idx       : Boolean := False;
   Valid_Spd_Idx   : Boolean := False;
   -- P R O C E D U R E   D E C L A R A T I O N S --
  procedure Get_Num_Points (Num_Points : out Portable_Types_Pkg.Integer_32) is
   --!
   --| PURPOSE: This procedure get the total Number of Points in the Point Layer.
   --
   --
   -- PARAMETERS: Number of points in the point layer.
   -- RAISES: None
   --
   --!
   begin
     Num_Points := Point_Count;
   end Get_Num_Points;

   procedure Get_Point_Data(Point_Index: in Portable_Types_Pkg.Integer_32;
                            Point_Data : out Fpp_Interface_Type.Point_Type) is
   --!
   --| PURPOSE: This procedure get the point data in the Point Layer and copy into
   --|          the VDU buffer.
   --
   --
   -- PARAMETERS: Point data index.
   -- RAISES: None
   --
   --!
   begin
     Point_Data := Points(Point_Index);
   end Get_Point_Data;
   procedure LGB_Store_Data is separate;
```

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

```

    procedure LGB_Seq_Leg is separate;

~~    procedure LGB_Seq_Rta_Leg (Initial_Est : in boolean := False)~~
~~    is separate;~~

    function LGB_Search (Starting_Leg_Index : Portable_Types_Pkg.Integer_32;
                         --| DESCRIPTION Starting Leg Index for the search.
                         --
                         Search_Thing : Search_Thing_Type;
                         --| DESCRIPTION The thing being searched for.
                         --
                         Search_Direction :
                              FMCS_Base_Types.Horizontal_Direction_Type
                         --| DESCRIPTION The direction of the search.
                         --| Regular predictions search forward while
                         --| Descent Path Generation searches backward.
                         --
                        ) return Portable_Types_Pkg.Integer_32 is separate;


   procedure Lgb_Next_Hold (Starting_Leg_Index : in Portable_Types_Pkg.Integer_32;
                            Entry_Index : out Portable_Types_Pkg.Integer_32;
                            Exit_Index  : out Portable_Types_Pkg.Integer_32) is
    --!
    --| PURPOSE: This procedure will search forward through Perf's copy of the LGB
    --|          from the provided starting index for the first HA (hold to altitude)
    --|          or HM (hold to manual termination) holding pattern in the flight
    --|          plan.  It returns the "last_segment_of_leg" index for the holding
    --|          pattern as the exit index, and the waypoint prior to it as the
    --|          exit index.  If not found, zero is returned.
    --
    -- PARAMETERS:
    --    Starting_Leg_Index - The index of the starting segment in the minleg fpln.
    --    Entry_Index        - The index of the waypoint prior to the holding pattern.
    --    Exit_Index         - The index of the holding pattern waypoint.
    --
    -- RAISES: None
    --
    --!

      Idx : Portable_Types_Pkg.Integer_32 := Starting_Leg_Index;
```

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

```

begin
   -- Search forward from the starting index for the first HA or HM holding pattern.
   While (Idx /= 0) and then
     ((Perf_Lgb_Lfdata.Lgb (Idx).Fpln_Data.Pathterm /= Fmcs_Base_Types.hm) and then
      (Perf_Lgb_Lfdata.Lgb (Idx).Fpln_Data.Pathterm /= Fmcs_Base_Types.ha))
   loop
      Idx:= Lgb_Search (Starting_Leg_Index => Idx,
                        Search_Thing => Next_Waypoint,
                        Search_Direction   => Fmcs_Base_Types.Forward);
   end loop;

   -- Return the found hold as the exit index.  If not found, this is zero.
   Exit_Index := Idx;

   -- Now find the hold entry waypoint, which is the waypoint prior to the hold exit.
   Entry_Index := Lgb_Search (Starting_Leg_Index => Idx,
                              Search_Thing => Next_Waypoint,
                              Search_Direction   => Fmcs_Base_Types.Backward);

end Lgb_Next_Hold;

--|
--------------------------------------------------------------------------------
function Lgb_Next_Lowest_Spd_Cnstr (Search_Thing       : Search_Thing_Type;
                                    Starting_Leg_Index : Portable_Types_Pkg.Integer_32;
                                    CnstrSpd           : Portable_Types_Pkg.Float_32;
                                    Search_Direction   : FMCS_Base_Types.Horizontal_Direction_Type :=
» FMCS_Base_Types.Forward)
                                    return Portable_Types_Pkg.Integer_32 is

 --!
 --| PURPOSE: This procedure will search through Perf's copy of the LGB
 --|          from the provided starting index for the MOST RESTRICTIVE
 --|          speed constraint in the flight plan. Default direction is forward.
 --|          If no speed constraint is found in the specified direction,
 --|          zero is returned.
 --
 -- PARAMETERS:
 --    Search_Thing       - The type of speed constraint to be searched for
 --                         (Next_Clb_Spd_Cstr, Next_Des_Spd_Cstr).
 --    Starting_Leg_Index - The index of the starting segment in the minleg fpln.
```

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

```
   --    CnstrSpd           - The current constraint speed (for comparison against).
   --    Search_Direction   - The direction to search (default is forward).
   --
   -- RAISES: None
   --
   --!

     Idx    : Portable_Types_Pkg.Integer_32 := Starting_Leg_Index;
     SpdPtr : Portable_Types_Pkg.Integer_32 := 0;
     Spd    : Portable_Types_Pkg.Float_32   := CnstrSpd;

  begin
     -- Return zero when end of flight plan detected or the Search_Thing
     -- is not a constraint speed.
     if (Idx = 0) or else
          ((Search_Thing /= Next_Des_Spd_Cstr) and then
          (Search_Thing /= Next_Clb_Spd_Cstr))
     then
         return(0);
     end if;

     loop

        Valid_Idx := False;
        Hold_Idx  := False;

        --  move to the next speed constrained mini-leg
        Idx := Perf_LGB_Pkg.LGB_Search(Starting_Leg_Index => Idx,
                                       Search_Thing       => Search_Thing,
                                       Search_Direction   => Search_Direction);

        Valid_Idx := (Idx /= 0) and then (not Perf_LGB_LFData.LGB(Idx).Fpln_Data.MissedAppr);

        Hold_Idx := (Idx /= 0) and then Is_Hold_Leg(Idx);

        if (Valid_Idx) and then
           (not Hold_Idx) and then
           (Perf_LGB_LFData.LGB(Idx).Perf_Data.SpcSpd < Spd) then

            --  this is the lowest speed constraint found so far, save it
            SpdPtr := Idx;
            Spd := Perf_LGB_LFData.LGB(Idx).Perf_Data.SpcSpd;
```

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

```

         end if;

         exit when not Valid_Idx;

      end loop;
      return SpdPtr;

   end Lgb_Next_Lowest_Spd_Cnstr;

   --|
   --------------------------------------------------------------------------------
   procedure Lgb_Next_Lowest_Spd_Cnstr_For_Descent (Starting_Leg_Index : in Portable_Types_Pkg.Integer_32;
                                                    CnstrSpd           : in Portable_Types_Pkg.Float_32;
                                                    Leg_Index          : out Portable_Types_Pkg.Integer_32;
                                                    CSS_Found          : out Boolean) is

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

      Idx     : Portable_Types_Pkg.Integer_32 := Starting_Leg_Index;
      Spd     : Portable_Types_Pkg.Float_32    := CnstrSpd;
      Hold_Bridge_Dist : constant := 10.0;  -- NAUTICAL MILES

   begin
```

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

```
      Leg_Index := 0;
      loop

         Valid_Idx := False;
         Hold_In_Descent := False;
         Hold_Idx  := False;
         Valid_Spd_Idx := True;

         -- move to the next waypoint. We are using the waypoint instead
         -- of the speed constraint because we need to stop the search if we
         -- encounter a cruise speed segment
         Idx := Perf_LGB_Pkg.LGB_Search(Starting_Leg_Index => Idx,
                                        Search_Thing       => Next_Waypoint,
                                        Search_Direction   => FMCS_Base_Types.Backward);

         -- It is not a valid Index and we should exit the search if we reach the
         -- end of the flight plan, we are in missed approach, or if we reach a cruise
         -- speed segment
         Valid_Idx := (Idx /= 0) and then
                      (not Perf_LGB_LFData.LGB(Idx).Fpln_Data.MissedAppr) and then
                      (not Perf_LGB_LFData.LGB(Idx).Fpln_Data.Crzspdtgt_Val);

         -- Since we are searching on 'waypoint', ensure that this waypoint contains
         -- a speed constraint. This will include holds in descent, because the spcspd
         -- for holds in descent will always contain at least the best hold speed
         if (Valid_Idx) and then
            (Perf_LGB_LFData.LGB(Idx).Perf_Data.SpcSpdVal) and then
            (Perf_LGB_LFData.LGB(Idx).Perf_Data.FltMode = GbDescent) then

            -- For descent, a hold is defined as an HF or HM leg
            Hold_Idx := (Perf_LGB_LFData.LGB(Idx).FPln_Data.PathTerm = HF) or else
                        (Perf_LGB_LFData.LGB(Idx).FPln_Data.PathTerm = HM);

            -- If this hold is within 10nm of the top of descent, consider it when
            -- determining the most limiting descent-like speed constraint
            Hold_In_Descent :=
              (Perf_Top_Of_Des_Lfdata.TODdata.Valid) and then
              ((Perf_LGB_LFData.LGB(Idx).Common_Data.Fixdistodest + Hold_Bridge_Dist) <= Perf_Top_Of_Des_Lfdata.TODdata.Dist);

            --  a speed constraint has been found
            if (not Hold_Idx or else
               (Hold_Idx and then Hold_In_Descent)) and then
```

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

```
               (Perf_LGB_LFData.LGB(Idx).Perf_Data.SpcSpd <= Spd) then

               --  this is the lowest speed constraint found so far, save it
               Leg_Index := Idx;
               Spd := Perf_LGB_LFData.LGB(Idx).Perf_Data.SpcSpd;

            end if;
         end if;

         CSS_Found := Perf_LGB_LFData.LGB(Idx).Fpln_Data.Crzspdtgt_Val;

         exit when not Valid_Idx;

      end loop;

   end Lgb_Next_Lowest_Spd_Cnstr_For_Descent;

   --|
   --------------------------------------------------------------------------------
   function Is_Hold_Leg (Leg_Index : in Portable_Types_Pkg.Integer_32) return Boolean is
    --!
    --| PURPOSE: This function returns true if the specified mini-leg index is an
    --|          HM, HA, or HF leg.
    --|
    --| PARAMETERS:
    --|    Leg_Index - The index to evaluate.
    --|
    --| SPECIAL CONSIDERATIONS: Assumes a valid leg index is supplied.
    --|
    --| RAISES: None.
    --!

   begin
      return ((Perf_Lgb_Lfdata.Lgb(Leg_Index).Fpln_Data.Pathterm = HA) or else
              (Perf_Lgb_Lfdata.Lgb(Leg_Index).Fpln_Data.Pathterm = HM) or else
              (Perf_Lgb_Lfdata.Lgb(Leg_Index).Fpln_Data.Pathterm = HF));
   end Is_Hold_Leg;

   --|
   --------------------------------------------------------------------------------
    procedure Create_Point
      (
```

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

```
      Event : in Perf_Point_Termination_Types.Termination_Type;
      Display_Suppressed : boolean := false
      ) is separate;
    --!
    --| PURPOSE: This procedure will create an LGB point layer record using the
    --|          passed-in termination type and other external predictions data
    --|          objects to create a point layer record for the current predictions
    --|          state.  The point record will be kept internally to this package
    --|          until Output_Preds is called do just that.
    --
    -- PARAMETERS: Termination_Type - The type of integration termination that
    --               predictions stopped at, from Perf's point of view.  This roughly
    --               equates to the events checked for in the various PROCTERM routines.
    --
    -- RAISES: None
    --
    --!

    procedure Output_Preds is separate;
    --!
    --| PURPOSE: This procedure outputs the predicted flight plan in Perf's copy
    --|          of the route, as well as the point layer records generated during
    --|          flight plan predictions, out to the interface with EFIS, who will
    --|          then write them to the LGB.
    --
    -- PARAMETERS: None
    -- RAISES: None
    --
    --!

end Perf_LGB_Pkg;
```

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_PKG_.ADA

```
--|
--|DATA_RIGHTS: HONEYWELL CONFIDENTIAL & PROPRIETARY
--|             THIS WORK CONTAINS VALUABLE CONFIDENTIAL AND PROPRIETARY
--|             INFORMATION. DISCLOSURE, USE OR REPRODUCTION OUTSIDE OF
--|             HONEYWELL INTERNATIONAL, INC. IS PROHIBITED EXCEPT AS
--|             AUTHORIZED IN WRITING. THIS UNPUBLISHED WORK IS PROTECTED BY
--|             THE LAWS OF THE UNITED STATES AND OTHER COUNTRIES. IN THE
--|             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)

```
--|

with Portable_Types_Pkg;
with FMCS_Base_Types;
with Perf_Point_Termination_Types;
with Fpp_Interface_Type;

package Perf_LGB_Pkg is
    --!
    -- PURPOSE:     Performance Predictions LGB package specification.
    --
    -- ANCHOR:      FMCS_19_21023510
    --
    --| @DESCRIPTION: Makes visible the procedure declarations for the
    --| predictions LGB related procedures.
    --
    -- SPECIAL_CONSIDERATIONS: None
    --
    -- REVISION_HISTORY:
    --   DATE           SCR #        Programmer               DRCM#
    -- ======================================================================
    --   12/18/95       8011         B. O'Laughlin            M777B_FMF_00547
    -- Added Lgb_Seq_Rta_Leg for processing predicted RTA leg sequence.
    --
    -- ====================== 787 HISTORY STARTS HERE =======================
    --
    --   12/06/05       519.00       Pat Caulfield
    --   Added new procedures Create_Point and Output_Preds, and added Next_Wind
    --   and Next_Temp to Search_Thing_Type.
    --
    --   01/05/06       519.04       Pat Caulfield
    --   Replaced Next_Wind, Next_Temp with Next_Waypoint; Changed Next_Leg to
    --   Next_Segment.
    --
    --   01/18/06       787.01       Pat Caulfield
    --   Renamed Create_Point's parameter to Event.
    --
    --   08/07/08      6676.00       Pat Caulfield
    --   Added new parameter Display_Suppressed to Create_Point.
    --
    --   07/18/2012    8301.02       Pat Caulfield
    --   Added Lgb_Next_Hold.
```

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

```
    --!
--------------------------------------------------------------------------------
    type Search_Thing_Type is (Next_Segment,
                               Next_Alt_Cstr,
~~                               Next_Spd_Cstr,~~
                               Next_Clb_Spd_Cstr,
                               Next_Des_Spd_Cstr,
                               Next_Alt_And_Spd_Cstr,
                               Next_Alt_Or_Spd_Cstr,
                               Next_Step_Alt_Term,
                               Next_Waypoint);

~~  for Search_Thing_Type use (Next_Segment          => 0,~~
~~                             Next_Alt_Cstr         => 1,~~
~~                             Next_Spd_Cstr         => 2,~~
~~                             Next_Alt_And_Spd_Cstr => 3,~~
~~                             Next_Alt_Or_Spd_Cstr  => 4,~~
~~                             Next_Step_Alt_Term    => 5,~~
~~                             Next_Waypoint         => 6);~~
    for Search_Thing_Type use (Next_Segment          => 0,
                               Next_Alt_Cstr         => 1,
                               Next_Clb_Spd_Cstr     => 2,
                               Next_Des_Spd_Cstr     => 3,
                               Next_Alt_And_Spd_Cstr => 4,
                               Next_Alt_Or_Spd_Cstr  => 5,
                               Next_Step_Alt_Term    => 6,
                               Next_Waypoint         => 7);


    procedure LGB_Store_Data;
    --!
    --| @PURPOSE: This procedure stores predicted data from the integration
    --| progress buffer into Perf's copy of the LGB (PERF_LGB_LFDATA.LGB).
    --
    -- PARAMETERS: None
    -- RAISES: None
    --
    --!

    procedure LGB_Seq_Leg;
    --!
    --| @PURPOSE: This procedure changes the PERF LGB leg pointer
```

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

```
    --| (PERF_PREDS_LFDATA.NAVPTR) to point to the next leg in the flight
    --| plan (in PERF_LGB_PKG.LGB). It also initializes the flight plan track
    --| to be used for ground speed computations on the new leg, and some
    --| other small miscellaneous implementation requirements for processing
    --| the end of the flight plan (NAVPTR =0).
~~    --~~
~~    -- PARAMETERS: None~~
~~    -- RAISES: None~~
~~    --~~
~~    --!~~

~~    procedure LGB_Seq_Rta_Leg (Initial_Est : in boolean := False);~~
~~    --!~~
~~    --| @PURPOSE: This procedure does all of the necessary processing when~~
~~    --| performance predictions sequence a leg with a RTA time constraint.~~
    --
    -- PARAMETERS: None
    -- RAISES: None
    --
    --!

    function LGB_Search (Starting_Leg_Index : Portable_Types_Pkg.Integer_32;
                         --| @DESCRIPTION Starting Leg Index for the search.
                         --
                         Search_Thing : Search_Thing_Type;
                         --| @DESCRIPTION The thing being searched for.
                         --
                         Search_Direction :
                              FMCS_Base_Types.Horizontal_Direction_Type
                         --| @DESCRIPTION The direction of the search.
                         --| Regular predictions search forward while
                         --| Descent Path Generation searches backward.
                         --
                        ) return Portable_Types_Pkg.Integer_32;
    --!
    --| @PURPOSE: This function is a performance predictions utility that
    --| searches PERF_LGB_LFDATA.LGB for a desired leg index. The caller inputs
    --| what to search for and the function returns the desired index, or 0 if
    --| the index is not found.
    --
    -- PARAMETERS: None
    -- RAISES: None
```

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

```
   --
   --!

  procedure Lgb_Next_Hold (Starting_Leg_Index : in Portable_Types_Pkg.Integer_32;
                           --| @DESCRIPTION: The index of the starting segment in
                           --|               the minleg fpln.
                           Entry_Index : out Portable_Types_Pkg.Integer_32;
                           --| @DESCRIPTION: The index of the waypoint prior to
                           --|               the holding pattern.
                           Exit_Index         : out Portable_Types_Pkg.Integer_32);
                           --| @DESCRIPTION: The index of the holding pattern waypoint.

  --|
  --------------------------------------------------------------------------------
  function Lgb_Next_Lowest_Spd_Cnstr (Search_Thing       : Search_Thing_Type;
                                      Starting_Leg_Index : Portable_Types_Pkg.Integer_32;
                                      CnstrSpd           : Portable_Types_Pkg.Float_32;
                                      Search_Direction   : FMCS_Base_Types.Horizontal_Direction_Type :=
» FMCS_Base_Types.Forward)
                                      return Portable_Types_Pkg.Integer_32;
   --!
   --| @PURPOSE: This function is a performance predictions utility that searches
   --|           PERF_LGB_LFDATA.LGB for the next most restrictive waypoint
   --|           climb-like or descent-like speed constraint that is not on a
   --|           holding pattern.
   --
   -- PARAMETERS: Search_Thing (Next_Clb_Spd_Cstr or Next_Des_Spd_Cstr)
   --             Starting_Leg_Index
   --             CnstrSpd
   --             Search_Direction (defaults to Forward)
   --             Returns the index of the next most restrictive speed constraint (Clb or Des) or 0 if none found
   -- RAISES: None
   --
   --!

  --|
  --------------------------------------------------------------------------------
  procedure Lgb_Next_Lowest_Spd_Cnstr_For_Descent (Starting_Leg_Index : in Portable_Types_Pkg.Integer_32;
                                                   CnstrSpd           : in Portable_Types_Pkg.Float_32;
                                                   Leg_Index          : out Portable_Types_Pkg.Integer_32;
                                                   CSS_Found          : out Boolean);

```

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

```
 --!
 --| PURPOSE: This procedure will search through Perf's copy of the LGB
 --|          from the provided starting index for the MOST RESTRICTIVE
 --|          descent-like speed constraint in the flight plan, including
 --|          holds in the descent path. If no speed constraint is found,
 --|          zero is returned.
 --
 -- PARAMETERS:
 --    Starting_Leg_Index - The index of the starting segment in the minleg fpln.
 --    CnstrSpd           - The current constraint speed (for comparision against).
 --    Leg_Index          - The most restrictive descent-like speed constraint uppath
 --                         from Starting_Leg_Index. Will return 0 if no constraint
 --                         is found.
 --    CSS_Found          - Will return true if the search was halted due to an
 --                         encountered Cruise Speed Segment.
 --
 -- RAISES: None
 --

--|
--------------------------------------------------------------------------------
 function Is_Hold_Leg (Leg_Index : in Portable_Types_Pkg.Integer_32) return Boolean;
 --!
 --| PURPOSE: This function returns true if the specified mini-leg index is an
 --|          HM, HA, or HF leg.
 --|
 --| PARAMETERS:
 --|    Leg_Index - The index to evaluate.
 --|
 --| SPECIAL CONSIDERATIONS: Assumes a valid leg index is supplied.
 --|
 --| RAISES: None.
 --!

--|
--------------------------------------------------------------------------------
 procedure Create_Point
   (
    Event : in Perf_Point_Termination_Types.Termination_Type;
    Display_Suppressed : boolean := false
   );
 --!
```

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

```
   --| PURPOSE: This procedure will create an LGB point layer record using the
   --|          passed-in termination type and other external predictions data
   --|          objects to create a point layer record for the current predictions
   --|          state.  The point record will be kept internally to this package
   --|          until Output_Preds is called do just that.
   --
   -- PARAMETERS: Termination_Type - The type of integration termination that
   --               predictions stopped at, from Perf's point of view.  This roughly
   --               equates to the events checked for in the various PROCTERM routines.
   --
   -- RAISES: None
   --
   --!

   procedure Output_Preds;
   --!
   --| PURPOSE: This procedure outputs the predicted flight plan in Perf's copy
   --|          of the route, as well as the point layer records generated during
   --|          flight plan predictions, out to the interface with EFIS, who will
   --|          then write them to the LGB.
   --
   -- PARAMETERS: None
   -- RAISES: None
   --
   --!
  procedure Get_Num_Points (Num_Points : out Portable_Types_Pkg.Integer_32);
   --!
   --| PURPOSE: This procedure get the Total Number of Points in the Point Layer.
   --
   --
   -- PARAMETERS: Number of points in the point layer.
   -- RAISES: None
   --
   --!

   procedure Get_Point_Data(Point_Index: in Portable_Types_Pkg.Integer_32;
                            Point_Data : out Fpp_Interface_Type.Point_Type);
   --!
   --| PURPOSE: This procedure get the point data in the Point Layer and copy into
   --|          the VDU buffer.
   --
   --
```

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

```
    -- PARAMETERS: Point data index.
    -- RAISES: None
    --
    --!
end Perf_LGB_Pkg;
```

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_SEARCH_SEP.ADA

```
--
--     STUB File
--
--     CTP_B787_PERF_LGBSEARCH_STB.ada
--
--     REASONS FOR STUBBING :  The Procedure LGB_Search has been stubbed for return Starting Leg Index for the search.
--
--|
--|DATA_RIGHTS: HONEYWELL CONFIDENTIAL & PROPRIETARY
--|             THIS WORK CONTAINS VALUABLE CONFIDENTIAL AND PROPRIETARY
--|             INFORMATION. DISCLOSURE, USE OR REPRODUCTION OUTSIDE OF
--|             HONEYWELL INTERNATIONAL, INC. IS PROHIBITED EXCEPT AS
--|             AUTHORIZED IN WRITING. THIS UNPUBLISHED WORK IS PROTECTED BY
--|             THE LAWS OF THE UNITED STATES AND OTHER COUNTRIES. IN THE
--|             EVENT OF PUBLICATION, THE FOLLOWING NOTICE SHALL APPLY:
--|             COPR. 2005 HONEYWELL INTERNATIONAL, INC. ALL RIGHTS RESERVED.
--|
with Portable_Types_Pkg;
with FMCS_Base_Types;
with FMCS_FP_Guid_Btypes;
with AC_Position_Types;

with Perf_LGB_Lfdata;

use Portable_Types_Pkg;
use FMCS_Base_Types;
use FMCS_FP_Guid_Btypes;
use AC_Position_Types;
-----------------------------------
with CTP_B787_PERF_CRZINITE_DRV;
use CTP_B787_PERF_CRZINITE_DRV;

separate (Perf_LGB_Pkg)

    function LGB_Search (Starting_Leg_Index : Portable_Types_Pkg.Integer_32;
```

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

```
                         --| DESCRIPTION Starting Leg Index for the search.
                         --
                         Search_Thing : Search_Thing_Type;
                         --| DESCRIPTION The thing being searched for.
                         --
                         Search_Direction :
                              FMCS_Base_Types.Horizontal_Direction_Type
                         --| DESCRIPTION The direction of the search.
                         --| Regular predictions search forward while
                         --| Descent Path Generation searches backward.
                         --
                        ) return Portable_Types_Pkg.Integer_32 is
--!
-- ANCHOR:      FMCS_19_21023514
-- SOURCE:      FMFSDD; FMCS_19_21023003 |
--| @DESCRIPTION:
--| This procedure is a performance predictions utility used to search the
--| LGB (PERF_LGB_LFDATA.LGB) for the next leg index that matches a desired
--| input. The calling procedure inputs as a parameter what to search for
--| (For example, the next altitude constraint).
--|
--| Possible search criteria are as follows :
--|    o  Next segment
--|    o  Next specified speed constraint
--|    o  Next specified altitude constraint
--|    o  Next specified speed and altitude constraint
--|    o  Next specified speed or altitude constraint
--|    o  Next specified step altitude constraint
--|    o  Next waypoint (last segment of leg)
--|
--| The procedure checks the FLTMODE field of the leg to limit the search to
--| the current predicted flight phase, if applicable.
--|
--| The procedure also has a flag as an input parameter that tells it
--| whether to search forward or backward through the flight plan.
--
-- SPECIAL_CONSIDERATIONS:
--
--   SHARED_DATA_FOR:
--   IOBLK
--
-- REVISION_HISTORY:
```

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

```
  --   DATE           SCR #              Programmer              DRCM#
  --   01-11-94       1777               D. Groethe              10016
  --   Initial version - complete
  --
  --   DATE           SCR #              Programmer              DRCM#
  --   03-22-94       1777               D. Groethe              11844
  --   Wrong logic on step search was causing an erroneous index return of 0.
  --
  --   =============================== 787 HISTORY ==========================
  --
  --   DATE           SCR #              Programmer
  --   11-11-2005     519.00             Pat Caulfield
  --   Added case for wind/temp entry searches, which amounts to searching
  --   for the last lateral segment of the leg.
  --
  --   DATE           SCR #              Programmer
  --   01-05-2006     519.04             Pat Caulfield
  --   Changed Next_Wind and Next_Temp to be Next_Waypoint to simplify things
  --   and avoid confusion, and renamed Next_leg to Next_Segment.
  --
  --!


    -- L O C A L   V A R I A B L E S --

    Return_Index : Portable_Types_Pkg.Integer_32;
    --| DESCRIPTION Leg index returned to calling procedure

    Starting_Fltmode : FMCS_FP_Guid_Btypes.Fltphasetyp;
    --| DESCRIPTION Fltmode of the input mini-leg (or the first non-Nogbmode
    --| value found in the direction of the search).  This is used to limit
    --| the searches for constraints to the relevant flight phase.
begin  -- LGB_Search

-- protect against calls with a bad starting leg index

-- if Starting_Leg_Index = 0 then
--   return 0;
-- end if;
--
-- Return_Index := Starting_Leg_Index;
-- loop
```

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

```
--
--     if (Search_Direction = Forward) then
--         Return_Index := Perf_LGB_Lfdata.LGB(Return_Index).Nextfpn;
--     else
--         Return_Index := Perf_LGB_Lfdata.LGB(Return_Index).Prevfpn;
--     end if;
--
--     -- RETURN ZERO WHEN END OF FLIGHT PLAN DETECTED OR
--     -- MISSED APPROACH DETECTED OR
--     -- CHANGE IN PREDICTED GLEG FLIGHT PHASE
--     if (Return_Index = 0) or else
--
--        (Perf_LGB_Lfdata.LGB(Return_Index).Fpln_Data.MissedAppr) or else
--
--        ((Perf_LGB_Lfdata.LGB(Return_Index).Perf_Data.Fltmode) /=
--         (Perf_LGB_Lfdata.LGB(Starting_Leg_Index).Perf_Data.Fltmode) and
--         (Search_Thing /= Next_Step_Alt_Term) and
--         (Search_Thing /= Next_Segment) and
--         (Search_Thing /= Next_Waypoint))
--     then
--         return(0);
--     end if;
--
--     case Search_Thing is
--
--       when Next_Segment => -- already advanced the pointer up above, so just exit.
--           exit;
--
--       when Next_Alt_Cstr =>
--           exit when (Perf_LGB_Lfdata.LGB(Return_Index).Fpln_Data.HavePerf);
--
--       when Next_Spd_Cstr =>
--           exit when (Perf_LGB_Lfdata.LGB(Return_Index).Perf_Data.SpcSpdVal);
--
--       when Next_Alt_And_Spd_Cstr =>
--           exit when (Perf_LGB_Lfdata.LGB(Return_Index).Fpln_Data.HavePerf) and
--                     (Perf_LGB_Lfdata.LGB(Return_Index).Perf_Data.SpcSpdVal);
--
--       when Next_Alt_Or_Spd_Cstr =>
--           exit when (Perf_LGB_Lfdata.LGB(Return_Index).Fpln_Data.HavePerf) or
--                     (Perf_LGB_Lfdata.LGB(Return_Index).Perf_Data.SpcSpdVal);
--
```

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

```
--       when Next_Step_Alt_Term =>
--           exit when (Perf_LGB_Lfdata.LGB(Return_Index).Fpln_Data.SpAlt1Val) and
--                     (Perf_LGB_Lfdata.
--                        LGB(Return_Index).Fpln_Data.SpAlt1Pos = SC);
--
--       -- Search for the next segment that is marked the last segment of the leg;
--       -- it contains data for the next waypoint.
--
--       when Next_Waypoint =>
--           exit when (Perf_LGB_Lfdata.LGB(Return_Index).Last_Segment_Of_Leg);
--
--     end case;
--
-- end loop;

Return_Index := Test_LGB_Search;

return(Return_Index);

end LGB_Search;
```

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_SEQ_LEG_SEP.ADA

```
--|
--|DATA_RIGHTS: HONEYWELL CONFIDENTIAL & PROPRIETARY
--|             THIS WORK CONTAINS VALUABLE CONFIDENTIAL AND PROPRIETARY
--|             INFORMATION. DISCLOSURE, USE OR REPRODUCTION OUTSIDE OF
--|             HONEYWELL INTERNATIONAL, INC. IS PROHIBITED EXCEPT AS
--|             AUTHORIZED IN WRITING. THIS UNPUBLISHED WORK IS PROTECTED BY
--|             THE LAWS OF THE UNITED STATES AND OTHER COUNTRIES. IN THE
--|             EVENT OF PUBLICATION, THE FOLLOWING NOTICE SHALL APPLY:
--|             COPR. 2005 HONEYWELL INTERNATIONAL, INC. ALL RIGHTS RESERVED.
--|
with Portable_Types_Pkg;

with Perf_ADS_Intent_Pkg;
with Perf_Su_Spd_Utils_Pkg;
with Ops_Cdk_Perf_Pdb_Mgr_Pkg;
with Perf_Preds_Lfdata;
with Perf_LGB_Lfdata;
with Perf_Task_Control_Lfdata;
with Perf_Integrators_Lfdata;
with Perf_Wind_Lfdata;
```

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

```
with Perf_Rta_Lfdata;
with Perf_Rta_Pkg;

use Portable_Types_Pkg;

separate (Perf_LGB_Pkg)

procedure LGB_Seq_Leg is
  --!
  -- DATA_RIGHTS: Honeywell ATSD Proprietary
  -- ANCHOR:      FMCS_19_21023513
  -- SOURCE:      FMFSDD; FMCS_19_21023002, FMCS_19_21023004
  --              FMFSRD; FMCS_19_20006056, FMCS_19_20006062
  --              FMFSRD; FMCS_19_20012452, FMCS_19_20012450
  --              FMFSRD; FMCS_19_20006073, FMCS_19_20006075
  --              FMFSRD; FMCS_19_20010030
  --              FMFSRD; FMCS_19_20006316 |
  --| @DESCRIPTION:
  --| This procedure is a performance predictions utility. It is used to change
  --| the leg index (Perf_Preds_Lfdata.NavPtr) that points to the current
  --| predicted leg in Perf's copy of the LGB (PERF_LGB_LFDATA.LGB) to point
  --| to the next leg.
  --|
  --| It is also used to write out a set of waypoints to the real (FM Global)
  --| LGB, once a certain number (see 'LGB_Leg_Output_Constant' declared in
  --| package body) have been predicted.
  --|
  -- SPECIAL_CONSIDERATIONS:
  --
  --   Use of Perf_BG_Waiting_On_Semaphore to signal the Preds Restart
  --   function not to restart Perf_BG while Perf_BG is accessing
  --   the LGB. The semaphore associated with accessing the LGB needs to be
  --   restored (by releasing access to the LGB) before a process restart can
  --   be done.
  --
  --   If a restart was needed while the LGB was being accessed (as indicated
  --   by the Unsuccessful_Restart flag), PERF BG restarts its self after
  --   regaining processor control and signalling the semaphore (releasing LGB
  --   access).
  --
  --
  --   SHARED_DATA_FOR:
```

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

```
  --   IOBLK
  --
  -- TRANSLATION HISTORY:
  -- This module was translated from the 747 SEQNAVLEG.PAS, B7X7FMS, Gen. 2
  --
  -- REVISION_HISTORY:
  --   DATE           SCR #              Programmer              DRCM#
  --   01-11-94       1777               D. Groethe              10015
  --   Initial version - complete
  --
  --   01-11-94       1777, 2402         D. Groethe              10541
  --   Initial value of First_Output_Leg_Index was NavPtr. Should be
  --   PrevNavPtr because NavPtr is already updated to the next leg by
  --   the time First_Output_Leg_Index is set. Also made LGB object update
  --   (MagVar) because of a type change in LGB manager (2402).
  --
  --   11-21-94       5686               B. O'Laughlin           19484
  --   Update groundspeed when there is a turn at the waypoint being sequenced,
  --   because a turn represents a step change in groundspeed if there is a wind.
  --
  --   01/17/96       8011         B. O'Laughlin           M777B_FMF_00549
  -- Added call to Lgb_Seq_Rta_Leg for processing predicted RTA leg sequence.
  --
  --   04/14/96       8030.08      Karen Hegeman           M777B_FMF_01596
  -- Added call to Perf_ADS_Intent_Pkg.Calc_Intermediate_Point to store
  -- intermediate intent data at a flight plan track change, or due to
  -- sequencing the RTA fix.
  --
  -- =============================== 787 HISTORY ===============================
  --
  --   12/01/05        519.00      Pat Caulfield
  -- Reworked for segment predictions; the logic for writing predictions to the
  -- LGB has been removed - PERF_EFIS_LGB_MGR_PKG does all of that now.
  --
  --!
--------------------------------------------------------------------------------

    -- L O C A L   V A R I A B L E S --

    -- DESCRIPTION Local copy of Perf_Preds_Lfdata.PrevNavPtr used for
    -- efficiency to reduce global memory access. Initialized to the current
    -- 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)

```
    PrevNavPtr : Portable_Types_Pkg.Integer_32 := Perf_Preds_Lfdata.NavPtr;

    -- DESCRIPTION Local copy of Perf_Preds_Lfdata.NavPtr used for
    -- efficiency to reduce global memory access. Initialized to the Nextfpn
    -- field of the current leg, which advances its current value to point to
    -- the next leg.
    NavPtr     : Portable_Types_Pkg.Integer_32 :=
                   Perf_LGB_Lfdata.LGB(Perf_Preds_Lfdata.NavPtr).NextFpn;

    -- DESCRIPTION Leg track in.
    Track_In   : Standard_Angle_Pkg.Saf_32;

    -- DESCRIPTION Leg track out.
    Track_Out  : Standard_Angle_Pkg.Saf_32;

begin  -- LGB_Seq_Leg

  -- IF NOT END OF FLIGHT PLAN
  if (NavPtr > 0) then

      -- SET UP GROUND SPEED DATA (FLIGHT PLAN TRACK AND PATH TERM)
      Track_In  := Perf_LGB_Lfdata.LGB(NavPtr).Efis_Data.Incourse +
                   Perf_LGB_Lfdata.LGB(NavPtr).Efis_Data.MagVar;
      Track_Out := Perf_LGB_Lfdata.LGB(NavPtr).Efis_Data.Outcourse +
                   Perf_LGB_Lfdata.LGB(PrevNavPtr).Efis_Data.MagVar;
      Perf_Preds_Lfdata.Gsdata.Fplntrack :=
        Track_Out + (Track_In - Track_Out) / Portable_Types_Pkg.Float_32(2.0);
      Perf_Preds_Lfdata.Gsdata.Pathterm :=
        Perf_LGB_Lfdata.LGB(NavPtr).Fpln_Data.Pathterm;

      -- STOP PREDICTING UPON SEQUENCING MISSED APPROACH
      if Perf_LGB_Lfdata.LGB(NavPtr).Fpln_Data.MissedAppr
      then
        -- SET NAVPTR TO 0 TO INDICATE END OF FLIGHT PLAN
        NavPtr := 0;
      end if;

  end if; -- IF NOT END OF FLIGHT PLAN

  -- if we've reached the next waypoint, find the next one.  If there are no more,
  -- lgb_search will return zero.
  If PrevNavPtr >= Perf_Preds_Lfdata.Next_Waypoint_NavPtr then
```

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

```
     Perf_Preds_Lfdata.Next_Waypoint_Navptr := Lgb_Search (Starting_Leg_Index => PrevNavPtr,
                                                           Search_Thing => Next_Waypoint,
                                                           Search_Direction => Fmcs_Base_Types.Forward);
  end if;

  -- RESTORE GLOBAL VALUES OF NAVPTR AND PREVNAVPTR
  -- this also sequences the segment/leg (see above in local var declarations)
  Perf_Preds_Lfdata.NavPtr     := NavPtr;
  Perf_Preds_Lfdata.PrevNavPtr := PrevNavPtr;

  -- IF FLIGHT PLAN HAS TURNED, UPDATE GROUNDSPEED FOR NEW TRACK
  -- and determine if ADS intermediate intent data needs to be sent
  -- Note that PrevNavPtr points to the segment/leg we're sequencing.  Only do this
  -- at a waypoint sequence (last segment of leg), not every segment.
  if (PrevNavPtr > 0) and then Perf_LGB_Lfdata.LGB(PrevNavPtr).Last_Segment_Of_Leg then

    -- compute the course change at the waypoint being sequenced
    if Perf_Preds_Lfdata.Next_Waypoint_Navptr > 0 then
       -- course change = next leg's outcourse minus current leg's incourse
       Course_Change := abs(Perf_LGB_Lfdata.LGB(Perf_Preds_Lfdata.Next_Waypoint_Navptr).Efis_Data.Leg_Outcourse -
                            Perf_LGB_Lfdata.LGB(PrevNavPtr).Efis_Data.Leg_Incourse);
    else
       Course_Change := 0.0;
    end if;

    -- recompute the ground speed if the track has changed more than 5 degrees
    if (Course_Change > 5.0) then
      -- (may want to call atmosphere model here if the waypoint
      -- termination caused significant rollback in CHEKINTERMS)
      Perf_Su_Spd_Utils_Pkg.Su_Compgndspd
            (Perf_Preds_Lfdata.Gsdata,
             Perf_Integrators_Lfdata.Intprogbuf.Tas2,
             Perf_Wind_Lfdata.Predwind,
             Perf_Integrators_Lfdata.Intprogbuf.Gndspd2);
    end if;

    if (Course_Change > 1.5) then
      -- Store ADS intermediate intent data if the track has changed more than 1.5 degrees
      Perf_ADS_Intent_Pkg.Calc_Intermediate_Point;
    end if;
  end if;

```

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

```
~~  -- CHECK TO SEE IF LEG HAS A RTA TIME CONSTRAINT~~
~~  if ((Ops_Cdk_Perf_Pdb_Mgr_Pkg.RTA_Is_Active(Perf_Preds_Lfdata.Vtpfplnindex)) and then~~
~~      (PrevNavPtr = Perf_LGB_Lfdata.LGB_Header.RTA_Fix_Ptr) and then~~
~~      (PrevNavPtr > 0) and then~~
~~      (Perf_LGB_Lfdata.LGB(PrevNavPtr).Fpln_Data.RTA_Time_Val)) then~~
~~    ----------------------------------------------------------------------------------~~
~~    --  set ads intent point (speed change), see FMCS_19_20006075~~
~~    --  note Pred_Ksa will be 0.0 after call to Perf_Lgb_Pkg.Lgb_Seq_Rta_Leg so~~
~~    --  you'll need to use a different variable if this line is moved~~
~~    ----------------------------------------------------------------------------------~~
~~    if (Perf_Rta_Lfdata.Pred_Ksa /= 0.0) then~~
~~      Perf_Ads_Intent_Pkg.Calc_Intermediate_Point;~~
~~    end if;~~
```

```

   if((PrevNavPtr > 0) and then (PrevNavPtr = Perf_LGB_Lfdata.LGB_Header.RTA_Fix_Ptr) and then
      (Perf_Rta_Lfdata.Rta_Window_Task or else
      ((Ops_Cdk_Perf_Pdb_Mgr_Pkg.RTA_Is_Active(Perf_Preds_Lfdata.Vtpfplnindex)) and then
      (Perf_LGB_Lfdata.LGB(PrevNavPtr).Fpln_Data.RTA_Time_Val)))) then


      if Perf_Rta_Lfdata.Rta_Window_Task then
      --  Added a logic to stop ETA window predictions when it reaches at RTA waypoint.
      --  This logic will ensure the storing of the MAX/MIN data.
      --  At the end of logics set stop preds flag to TRUE.
      --  It ensure to come out of the VTP exec . PERF_SRD_B_00067
        if(Perf_Preds_Lfdata.Perf_Pass = Early) then
         Perf_Rta_Lfdata.Rta_Windows(Perf_Preds_Lfdata.Vtpfplnindex).Min_Time.valid :=True;
         Perf_Rta_Lfdata.Rta_Windows(Perf_Preds_Lfdata.Vtpfplnindex).Min_Time.Data :=
» Portable_Types_Pkg.Integer_32(Perf_Integrators_Lfdata.IntProgBuf.TProg) +
           Perf_Preds_Lfdata.Aircraft_State.GMT.Data;
        else
         Perf_Rta_Lfdata.Rta_Windows(Perf_Preds_Lfdata.Vtpfplnindex).Max_Time.valid :=True;
         Perf_Rta_Lfdata.Rta_Windows(Perf_Preds_Lfdata.Vtpfplnindex).Max_Time.Data :=
» Portable_Types_Pkg.Integer_32(Perf_Integrators_Lfdata.IntProgBuf.TProg) +
           Perf_Preds_Lfdata.Aircraft_State.GMT.Data;
        end if;

        Perf_Preds_Lfdata.Vtplogic.Stop_Preds :=True; -- PERF_SRD_B_00068

      else
      --------------------------------------------------------------------------------
      --  set ads intent point (speed change), see FMCS_19_20006075
```

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

```
      --  note Pred_Ksa will be 0.0 after call to Perf_Lgb_Pkg.Lgb_Seq_Rta_Leg so
      --  you'll need to use a different variable if this line is moved
      --------------------------------------------------------------------------------
         if (Perf_Rta_Lfdata.Pred_Ksa /= 0.0) then
           Perf_Ads_Intent_Pkg.Calc_Intermediate_Point;
         end if;

~~    if (Perf_Preds_Lfdata.Aircraft_State.GMT.Valid) then~~
~~      -- PROCESS THE RTA LEG SEQUENCE, COMPUTE NEW KSA, SET/CLEAR SPAD MSG~~
~~      Perf_Lgb_Pkg.Lgb_Seq_Rta_Leg;~~
~~    end if;~~
~~    Perf_Rta_Lfdata.Pred_Pastrta := True;~~
~~    Perf_Rta_Lfdata.Pred_Ksa := 0.0;~~
         if (Perf_Preds_Lfdata.Aircraft_State.GMT.Valid) then
          -- PROCESS THE RTA LEG SEQUENCE, COMPUTE NEW KSA, SET/CLEAR SPAD MSG
           Perf_Rta_Pkg.Rta_Seq_Rta_Leg;
         end if;
         Perf_Rta_Lfdata.Pred_Pastrta := True;
         Perf_Rta_Lfdata.Pred_Ksa := 0.0;
      end if;
  end if;

end LGB_Seq_Leg;
```

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_SEQ_RTA_LEG_SEP.ADA

```
--|
--|DATA_RIGHTS: HONEYWELL CONFIDENTIAL & PROPRIETARY
--|             THIS WORK CONTAINS VALUABLE CONFIDENTIAL AND PROPRIETARY
--|             INFORMATION. DISCLOSURE, USE OR REPRODUCTION OUTSIDE OF
--|             HONEYWELL INTERNATIONAL, INC. IS PROHIBITED EXCEPT AS
--|             AUTHORIZED IN WRITING. THIS UNPUBLISHED WORK IS PROTECTED BY
--|             THE LAWS OF THE UNITED STATES AND OTHER COUNTRIES. IN THE
--|             EVENT OF PUBLICATION, THE FOLLOWING NOTICE SHALL APPLY:
--|             COPR. 2012 HONEYWELL INTERNATIONAL, INC. ALL RIGHTS RESERVED.
--|
with Portable_Types_Pkg;       use Portable_Types_Pkg;
with Ac_Position_Types;        use Ac_Position_Types;
with Fmcs_Base_Types;          use Fmcs_Base_Types;
with Fmcs_Fp_Guid_Btypes;      use Fmcs_Fp_Guid_Btypes;
with Scratch_Pad_Iftypes;
with Cfp_Perf_Rta_Iftypes;
with Perf_Integrators_Lftypes;
```

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

```
with Flight_Pln_Leg_Types;
with Flight_Pln_Hdr_Types;
with Alt_Profile_Iftypes;
with VGB_Iftypes;
with Perf_Air_Data_Pkg;
with FMCS_AEDB_Constants_Ifdata;
with OPS_CDK_Common_Mgr_Pkg;
with Perf_Aero_Speed_Pkg;
with Perf_Crz_Pkg;     use Perf_Crz_Pkg;

with Perf_Rp_Guidprms_Ifdata;

with Perf_Preds_Lfdata;
with Perf_Cmd_Spd_Pkg;
with Perf_LGB_Lfdata;
with Perf_Integrators_Lfdata;
with Perf_Wind_Lfdata;
with Perf_Rta_Lfdata;
with Perf_Idx_Msg_Flags_Lfdata;
with Perf_Crzalt_Lfdata;
with Idx_Profile_Ifdata;
with Perf_Profile_Lfdata;
with Perf_WTS_Lfdata;

with Math_Pkg;  use Math_Pkg;
with Math_Rad_Pkg;  use Math_Rad_Pkg;
with Ops_Cdk_Perf_Pdb_Mgr_Pkg;
with Fmci_Spad_Manager_Pkg;
with Act_Prov_Index_Manager;
with Ops_Lateral_Guidance_Buffer_Manager;
with Fpp_Common_Lgb_Wrap_Pkg;
with Fpp_Status_Type_Tpkg; use Fpp_Status_Type_Tpkg;


separate (Perf_LGB_Pkg)

procedure LGB_Seq_Rta_Leg (Initial_Est : in boolean := False) is
  --!
  -- ANCHOR:      FMCS_19_21023515
  --
  --| @DESCRIPTION:
  --| 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)

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

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

```
--
--   02/26/96       8011.04      B. O'Laughlin           M777B_FMF_01165
-- Fixes to turn off the VG RTA speed filter when the speed target changes
-- by a large amount.
--
--   06/14/96       8544,8578,8579    B. O'Laughlin      M777B_FMF_02143
-- Extensive changes to module to improve RTA behavior near speed limits
-- and to improve readability.  Added Max/Min_Ksa.
-- Fixed case where recommended takeoff time was in the past (instead of NOW).
-- Check if PE T/O time is in past before using it to compute ETA.
--
--   08/04/96       8544         B. O'Laughlin           M777B_FMF_02637
-- Added Perf_Rta_Lfdata.k8 and k9 to make algorithim more controllable.
--
--   04/02/97       9204.08      D. Turner/A. Comaduran  M777B_FMF_04122
-- Extensive changes made to integrate required time of arrival and wind
-- trade step climb functions.  Alternate method of computing KSA based on
-- previous guesses for KSA and the resulting ATE added.  Computation of
-- "RTA" Cost Index - corresponding to the computed KSA - added.  Additional
-- limits on the change in KSA between subsequent calls to this module added.
--
--   04/15/97       9204.08      D. Turner/A. Comaduran  M777B_FMF_04293
-- Fix errors in the Compute_Ksa module.  Incorporated the pre-DRCM#4122
-- compute ksa logic into the logic added for Wind Trade Step Climbs (4122).
--
--   04/17/97       9204.08      D. Turner               M777B_FMF_04349
-- The CRZ Alt used to compute the Rta Cost Index was changed to
-- Perf_Preds_Lfdata.VGB(CRZ2L).CmdSpdAlt so that it would always match
-- the T/C Cruise altitude.
--
--   04/18/97       9204.08      D. Turner               M777B_FMF_04368
-- Fixed problem with Perf_WTS_Lfdata.Rta_Iter_Counter not being updated
-- until after the second (not first) pass through Predictions.  Problem
-- was in the call to Common_Init from Predicted_Sequence.  Also expanded
-- data sorting logic in Compute_Ksa so that "Bad_data" points will
-- automatically be moved to the lowest array location so that they can
-- be eliminated from consideration first.
--
--   04/21/97       9204.08      D. Turner               M777B_FMF_04431
-- The cost biasing logic has been changed so that the step point data
-- is always stored in array location Rta iteration counter + 1, except
-- for case of first time thru predictions with an active rta.
-- Added logic to not compute a new Rta_Ci when already on time.
--
--   04/23/97       9204.08      D. Turner               M777B_FMF_04460
-- Fixed a potential reset which would occur if tried to assign the local
-- Iter_Ctr a value of the rta iteration counter which is outside it's range.
-- Also fixed the ksa computation logic so that it would work properly
-- without sorting the ksa-ate data.
--
--   04/24/97       9204.08      D. Turner               M777B_FMF_04488
-- Added logic to copy the step-point storage and other data which is indexed
-- to a flight plan to the other flight plan (Act or Prov) when entering this
-- module in Initial_Est.
--
--   05/01/97       9204.08      D. Turner               M777B_FMF_04618
-- Added limiting logic to the KSA_Guess sub-unit to control the value of
-- Ksa_ATE_Slope.
--
--   05/04/97       9204.08      D. Turner               M777B_FMF_04642
-- Fixed a divide by zero reset in the Compute_Ksa module in the logic to sort
-- through the date to determine if a point of the opposite sign as current data
-- exists.
--
--   05/12/97       9313         D. Turner               M777B_FMF_04763
-- Added a Reset of the Max_Speed_Up and Max_Slow_Down booleans so that they could
-- not be true upon entry into this module.
--
--   05/13/97       9313         D. Turner               M777B_FMF_04763
-- Added logic to allow for negative cost indicies.  The data used for the negative
-- cost index was obtained from running HSS cases with slow-down Rta's
--
--   10/29/97       9629         D. Turner               M777B_FMF_05977
-- Added checks to verify that we have a good rta pointer before attempting to
-- access the LGB with it.  If it is bad, we simply set the Ksa to zero.  This will
-- ruin the rta solution, but should prevent a reset.
--
-- 787 MODIFICATION_HISTORY
-- ==========================================================
-- 6/02/05       480.00        Rob Celesnik
-- Updated Crew Interface signal names from "Fmf_Ioc" to "Fmci".
--
-- 9/12/06      1594.00        Pat Caulfield
-- 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)

```
  --
  -- 02/02/07     26984.03       Keri Kalvelage
  -- To fix SBC compiler warnings, removed unused declarations for
  -- Sav_Step_Dist_Data, Sav_Step_Alt_Data & rec_ctr.
  --
  -- 04/22/08     6276.06       Pat Caulfield
  -- Switched to using get_lat_leg_Status to avoid resets and have a leg validity.
  --
  -- 07/07/2008   5295.00       Rajesh Chaubey
  -- Updated the code to display the 'UNABLE RTA' message for the active route
  -- when the MOD is erased.
  --
  -- 04/04/12     14535.02       Keri Kalvelage
  -- In procedure Compute_Ksa changed the check for abs(Ate_Result > Spd_Adj_Tol)
  -- to now be >=.  The comparison of the Ate_Result and Spd_Adj_Tol = 0 was not
  -- being handled and was causing the Ksa_Adj to be uninitialized value of 0.
  -- This was causing the spd target to go to the unadj econ value and causing a spike.
  --
  -- 07/19/12     11939.02       B. O'Laughlin/Kevin Corbett
  -- Several updates:
  -- 1) Use the Pegasus version of the code related to negative cost index values
  -- (in procedure Limit_Ksa) and add min and max limits to the Ci_Mach_Slope.
  -- 2) Do not update the indexed Rta_Iter_Counter (that is being used by other
  -- predictions files) until the end of a pass of preds.
  -- 3) Modify the way that Flat_Bias_Factor works so that it does not rely on
  -- At_Max_Bias and Apply_Flat_Bias, and Rta_Iter_Counter.  Get rid of variables
  -- At_Max_Bias and Apply_Flat_Bias, since they are not indexed by flight plan.
  -- Replaced uses of Perf_WTS_Lfdata.Apply_Flat_Bias with the following logic
  -- (Perf_WTS_Lfdata.Flat_Bias_Factor >= Perf_WTS_Lfdata.Initial_Flat_Bias_Factor).
  -- Created indexed copy of Flat_Bias_Factor to hold the data for each flight
  -- plan independently and new variable New_Flat_Bias_Factor to hold the
  -- value to be used by the next pass, so that the value is not changed in the
  -- middle of a pass at the RTA waypoint.
  -- 4) Created indexed copy of Rta_CI (it is ok to update this in the middle
  -- of a pass at the RTA waypoint since it shouldn't be used for the remainder
  -- of that pass).
  --
  --!

------------------------------ R E N A M E S -----------------------------------
package r_Msg_Flags renames Perf_Idx_Msg_Flags_Lfdata;  -- shortens name
package R_WTS_Pkg renames Perf_WTS_Lfdata;
```

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

```
package R_Portable renames Portable_Types_Pkg;
----------------------- L O C A L   V A R I A B L E S -------------------------

Eta              : R_Portable.Integer_32; -- est time of arrival (GMT in sec)
Flight_Time      : R_Portable.Integer_32; -- in flight time to rta waypoint
Fpln_Index       : Fmcs_Fp_Guid_Btypes.Act_Or_Prov_Type; -- pred's flight plan index
Init_Valid       : Boolean := True;     -- initialization valid flag
Good_Pointer     : Boolean := True;     -- RTA pointer in LGB valid flag (SCR 9629)
Ksa_New          : R_Portable.Float_32;  -- ksa for the next trip preds
Ksa_Old          : R_Portable.Float_32;  -- ksa used for current trip preds
RtaPtr           : R_Portable.Integer_32 :=
                     Perf_Preds_Lfdata.PrevNavPtr; -- local copy for efficiency
Rta_Time         : R_Portable.Integer_32;  -- required time of arrival
Rta_Idx_Data     : Perf_Rta_Lfdata.Rta_Data_Rec_Type;  -- local copy for efficiency
Rta_Type         : Ac_Position_Types.Time_Constraint_Type;  -- After,Before,AT_Time
Time_Target      : R_Portable.Integer_32;  -- targeted arrival time
Tko_Time         : Ops_Cdk_Perf_Pdb_Mgr_Pkg.
                     Int_32_Tko_Entry_Stat.State;  -- takeoff time and a status
Loop_Ctr         : R_Portable.Integer_32; -- stores ksa, eta data: compare new Ksa guess to prev. ones
Iter_Ctr         : R_WTS_Pkg.Num_Values_Type := 1; -- local var to hold RTA Ctr
KSA_ATE_Slope    : R_Portable.Float_32; -- slope of Ate vs. Ksa curve based on current iteration
                                        -- and next closedt point
Min_ATE          : R_Portable.Integer_32; -- used to keep track of the minimum ATE, considering 9
                                          -- most previous iterations
Max_ATE          : R_Portable.Integer_32; -- used to keep track of the maximum ATE, considering 9
                                          -- most previous iterations
Num_Places       : R_Portable.Integer_32; -- used to limit how many Perf_RTA_Lfdata.Ksa_ETA_Rec
                                          -- array locations will be examined to find the Min_ATE
                                          -- and Max_ATE
Min_Location     : R_Portable.Integer_32; -- Perf_RTA_Lfdata.Ksa_ETA_Rec array location for the
                                          -- Min_ATE
Max_Location     : R_Portable.Integer_32; -- Perf_RTA_Lfdata.Ksa_ETA_Rec array location for the
                                          -- Max_ATE
Save_ATE         : R_Portable.Integer_32; -- local storage for an ATE record from
                                          -- Perf_RTA_Lfdata.Ksa_ETA_Rec array
Save_Ksa         : R_Portable.Float_32;   -- local storage for an Ksa  record from
                                          -- Perf_RTA_Lfdata.Ksa_ETA_Rec array
Save_BadData     : Boolean;               -- local storage for a Bad_Data record from
                                          -- Perf_RTA_Lfdata.Ksa_ETA_Rec array
Min_ATE_Opp_Sign : R_Portable.Integer_32; -- Value of the Minimum ATE, considering 9 most
                                          -- previous iterations, which is of the opposite sign
                                          -- as the current iteration's ATE
```

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

```
Max_ATE_Opp_Sign : R_Portable.Integer_32; -- Value of the Maximum ATE, considering 9 most
                                          -- previous iterations, which is of the opposite sign
                                          -- as the current iteration's ATE
Opposite_Sign   : Boolean; -- If this boolean is true, the ATE for the previous iteration which
                           -- is under consideration is of the opposite sign as the current
                           -- iteration's ATE
Opposite_Sign_Exist : Boolean; -- If this boolean is true, there exists at least one ATE from a
                           -- previous pass which is of the opposite sign as the current
                           -- iteration's ATE
CI_Guess        : R_Portable.Integer_32; -- Initial guess for "RTA" Cost Index
WOD             : R_Portable.Float_32; -- Gross weight / delta
Press_Ratio     : R_Portable.Float_32; -- Atmospheric pressure ratio
Theta           : R_Portable.Float_32; -- Atmospheric temperature ratio
Theta_Std       : R_Portable.Float_32; -- standard day temperature ratio
Vel_Sound       : R_Portable.Float_32; -- speed of sound
Corr_Theta      : R_Portable.Float_32; -- corrected temperature ratio
Base_CI         : R_Portable.Float_32; -- Cost Index at Top of Climb altitude
Corr_Base_CI    : R_Portable.Float_32; -- Corrected Cost Index at Top of Climb altitude
Base_Mach       : R_Portable.Float_32; -- Mach number at Top of Climb altitude based on Base_CI
Tgt_Mach        : R_Portable.Float_32; -- Mach Number at Top of Climb altitude with computed Ksa
                                       -- applied
Previous_Leg    : FMCS_Fp_Guid_Btypes.GuidanceIndx;
Next_Leg        : FMCS_Fp_Guid_Btypes.GuidanceIndx;
Prddataseq      : Portable_Types_Pkg.Integer_32;
Corefp_Status   : Fpp_Status_Type_Tpkg.Fpp_Status_Type_T;

Type Sav_Step_Dist_Type is Array(Alt_Profile_Iftypes.NumScPts) of
                      FMCS_Base_Types.Float_32_Valid.Normal; -- make of the same type as
                                                             -- Perf_Rta_Lfdata.Step_Dist_Rec
CrzAlt : R_Portable.Float_32; -- Local copy of Top of Climb Cruise altitude
Initialize_To_Iter_Ctr : Boolean; -- If true then the Step point records will be rolled back
                                  -- from Iter_Ctr and not (Iter_Ctr + 1)
Copy_From_Opposite_Fpln : Boolean; -- If true, when roll back from Iter_Ctr or Iter_Ctr+1 in
                                   -- Common_Init, get the data from the opposite flightplan
Rollback_Fpln : Fmcs_Fp_Guid_Btypes.Act_Or_Prov_Type; -- The correct Fplnindex to use in rollback



----------------------- L O C A L   C O N S T A N T S -------------------------
-- most of the local constants were moved to Perf_Rta_Lfdata so that they would
-- be global and could be changed from the memory readout page for debug.
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)

```
Half_Day    : R_Portable.Integer_32 := 43200;  -- 43200 sec = 12 hr

------------------- L O C A L   F U N C T I O N  (Time_Map) -------------------
-- Local function Time_Map exists to make the code more readable.  It takes
-- an input time (in seconds) and maps it to be (0 <= time < One_Day).
-- NOTE: if the module dosen't compile, the pragma inline will generate:
-- "Inline expansion of TIME_MAP is not achieved here: body is not available"
-- don't worry about this, it will go away when the module compiles.

function Time_Map(Time_1 : R_Portable.Integer_32)
         return R_Portable.Integer_32 is
begin
  return (Time_1 mod One_Day);
end;
pragma inline (Time_Map);

function Time_Map(Time_1 : R_Portable.Integer_32)
         return R_Portable.Float_32 is
begin
  return R_Portable.Float_32(Time_1 mod One_Day);
end;
pragma inline (Time_Map);

--------------- L O C A L   P R O C E D U R E  (Common_Init)  ----------
-- This procedure is used because Initial_Estimate and Predicted_Sequence
-- both use the following initilizations.
--
procedure Common_Init is
begin

  if Copy_From_Opposite_Fpln then
    if (Fpln_Index = Act_Prov_Index_Manager.Prov_Index) then
      Rollback_Fpln := Act_Prov_Index_Manager.Act_Index;
    else
      Rollback_Fpln := Act_Prov_Index_Manager.Prov_Index;
    end if;
  else
    Rollback_Fpln := Fpln_Index;
  end if;

  if Initialize_To_Iter_Ctr then
    -- 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)

```
    -- into the first array locations so that we have them
    for Loop_Ctr in Alt_Profile_Iftypes.NumScPts loop
       R_WTS_Pkg.Step_Dist_Rec(Fpln_Index,Loop_Ctr,1).Data :=
            R_WTS_Pkg.Step_Dist_Rec(Rollback_Fpln,Loop_Ctr, Iter_Ctr).Data;
       R_WTS_Pkg.Step_Dist_Rec(Fpln_Index,Loop_Ctr,1).Valid :=
            R_WTS_Pkg.Step_Dist_Rec(Rollback_Fpln,Loop_Ctr, Iter_Ctr).Valid;
    end loop;
  else
    for Loop_Ctr in Alt_Profile_Iftypes.NumScPts loop
       R_WTS_Pkg.Step_Dist_Rec(Fpln_Index,Loop_Ctr,1).Data :=
            R_WTS_Pkg.Step_Dist_Rec(Rollback_Fpln,Loop_Ctr, Iter_Ctr+1).Data;
       R_WTS_Pkg.Step_Dist_Rec(Fpln_Index,Loop_Ctr,1).Valid :=
            R_WTS_Pkg.Step_Dist_Rec(Rollback_Fpln,Loop_Ctr, Iter_Ctr+1).Valid;
    end loop;
  end if;

  -- copy step climb data from current 'pass' to index 1
  R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, 1).Ksa_Guess :=
       R_WTS_Pkg.KSA_ETA_REC(Rollback_Fpln, Iter_Ctr).Ksa_Guess;
  R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, 1).Ate_Result :=
       R_WTS_Pkg.KSA_ETA_REC(Rollback_Fpln, Iter_Ctr).Ate_Result;
  R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, 1).Bad_Data :=
       R_WTS_Pkg.KSA_ETA_REC(Rollback_Fpln, Iter_Ctr).Bad_Data;

  for Loop_Ctr in Alt_Profile_Iftypes.NumScPts loop
    for Rec_Ctr in 2..R_WTS_Pkg.Num_Values_Type'last loop
      R_WTS_Pkg.Step_Dist_Rec(Fpln_Index,Loop_Ctr, Rec_Ctr).Data := 0.0;
      R_WTS_Pkg.Step_Dist_Rec(Fpln_Index,Loop_Ctr, Rec_Ctr).Valid := False;
    end loop;
  end loop;

  R_WTS_Pkg.Rta_Iter_Counter(Fpln_Index) := 1;

  Iter_Ctr := R_WTS_Pkg.Rta_Iter_Counter(Fpln_Index);
  Perf_WTS_Lfdata.Flat_Bias_Factor := 0.0;
  Perf_WTS_Lfdata.New_Flat_Bias_Factor := 0.0;
  -- The value of New_Flat_Bias_Factor is now the main way of managing the logic

  -- 1/23/97 refresh the ksa_eta_rec storage array
  Loop_Ctr := 2;

  while Loop_Ctr < R_WTS_Pkg.Num_Ksa_ETA_Records loop
```

```
    R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, Loop_Ctr).Ksa_Guess := 0.0;
    R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, Loop_Ctr).Ate_Result := 0;
    Loop_Ctr := Loop_Ctr + 1;
  end loop;

end Common_Init;

-------------- L O C A L   P R O C E D U R E  (Initial_Est)  ------------------
-- Local procedure Initial_Estimate is run when the Initial_Est input boolean
-- is True.  This means that RTA has just been activated and an initial estimate
-- of the Ksa should be attempted.  Since RTA has just been activated, there
-- are no ETA predictions for the RTA waypoint in the provisional route.
-- However, if the RTA waypoint exists in the active route, then we can use
-- it's ETA to make an initial guess at the Ksa.
--
-- Initial_Estimate is called when Lgb_Seq_Rta_Leg is called from Restart_Check
-- so this is running from CDK's task (but it shouldn't interupt the Perf_BG
-- task in the middle of Leg_Seq_Rta_Leg, because RTA isn't active).
-- Since this is running from Restart_Check, there is no chance that it will
-- be restarted while it has access to the LGB.
--
-- This code is in this module because the initial estimate uses the same Ksa
-- and recommended takeoff time computation logic that is run when trip
-- predictions sequence the RTA waypoint.

procedure Initial_Estimate is
  -- local variables
  Fpln_ID        : FMCS_Fp_Guid_Btypes.Actfpln_Type;
  Route_ID       : Fmcs_Fp_Guid_Btypes.Gbthreadtype;
  Act_Gleg       : Flight_Pln_Leg_Types.Leg_Rec;
  Prov_Gleg      : Flight_Pln_Leg_Types.Leg_Rec;
  Act_Access_Id  : OPS_Lateral_Guidance_Buffer_Manager.Access_ID_Type;
  Prov_Access_Id : OPS_Lateral_Guidance_Buffer_Manager.Access_ID_Type;
  Act_Index      : Fmcs_Fp_Guid_Btypes.Act_Or_Prov_Type;

begin

  -- use provisional flight plan index for rest of module
  Fpln_Index := Act_Prov_Index_Manager.Prov_Index;
  Good_Pointer := True;

  -- 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)

```
-- data is stored.  In this case, its Iter_Ctr+1 since the non-rta step point
-- data is always stored in array location rta iteration counter + 1.
Initialize_To_Iter_Ctr := False;
Copy_From_Opposite_Fpln := True;

-- Initialize steps data, ksa ate storage array, rta iter counter and
-- biasing flags
Common_Init;

-- initialize the VG rta speed filter (off)
Perf_Rp_Guidprms_Ifdata.RTA_Spd_Tgt_Filter_On := False;

-- initialize the message flags
r_Msg_Flags.Unable_Rta_Msg_Issued(Fpln_Index) := False;
r_Msg_Flags.Unable_Rta_Msg_Cleared(Fpln_Index) := False;
r_Msg_Flags.Unable_Flxxx_At_Rta_Fix_Msg_Issued(Fpln_Index) := False;
r_Msg_Flags.Unable_Flxxx_At_Rta_Fix_Msg_Cleared(Fpln_Index) := False;

-- Set up local copy of rta data and initialize only those values
-- that will be read from before they are written to.
Rta_Idx_Data := Perf_Rta_Lfdata.Idx_Data(Fpln_Index);  -- OLD data
Rta_Idx_Data.Preds_Status.Rta_Happy := False;
Rta_Idx_Data.Msg_Counter := 0;

Rta_Idx_Data.Ksa := 0.0;  -- rta was not active when the act preds were done
-- WTS 3/16/97: Refresh the Rta_CostIndex
R_WTS_Pkg.Rta_CI := Ops_CDK_Common_Mgr_Pkg.Cost_Index.Data;
Rta_Idx_Data.Max_Ksa := 200.0;   -- don't limit first guess
Rta_Idx_Data.Min_Ksa := -200.0;  -- don't limit first guess
Rta_Idx_Data.Preds_Status.At_Max_Speed_Up := False;
Rta_Idx_Data.Preds_Status.At_Max_Slow_Down := False;
-- Note: It would be nice (but difficult) to compute the actual thrust
-- limited speeds by calling Max_Spd_At_Alt, we may want to do this someday
-- so the initial VGB speeds will be thrust limited.
Rta_Idx_Data.Min_Spd_At_Crzalt := (Speed => 0.0, Speed_Is_Thrust_Lim => False,
                                   Valid => False, Alt => 0.0, Gwt => 0.0);
Rta_Idx_Data.Max_Spd_At_Crzalt := (Speed => 1.0, Speed_Is_Thrust_Lim => False,
                                   Valid => False, Alt => 0.0, Gwt => 0.0);
-- Rta_Idx_Data.Spd_Gen_Status will be set when the speed generators are run
-- Rta_Idx_Data.Preds_Status, Old_Ksa, Ate, Tup, Tdn, Crz_Dist, and Crz_Time
-- will be set in this module
-- 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)

```

  -- get access to prov lgb
  Ops_Lateral_Guidance_Buffer_Manager.Requestlgb
    (Lgb_Process_Id => Ops_Lateral_Guidance_Buffer_Manager.Preds_Restart,
     Access_Id      => Prov_Access_Id,
     Lgb_Operation  => Flight_Pln_Hdr_Types.Rte_Read,
     Route_Id       => Fmcs_Fp_Guid_Btypes.Gbprov);

  -- get provisional header rta pointer
  RtaPtr := Ops_Lateral_Guidance_Buffer_Manager.RTA_Fix_Ptr(Prov_Access_Id);

  -- SCR 9629 Only attempt to access the LGB if there is a good rta pointer.
  if RtaPtr /= 0 then
    -- get rta leg from the provisional lgb
    Fpp_Common_Lgb_Wrap_Pkg.Get_Lat_Leg_Status
              (Process_ID   => Fmcs_Fp_Guid_Btypes.Preds_Restart_C,
               Leg_Index   => RtaPtr,
               Lateral_Leg => Prov_Gleg,
               Previous_Leg => Previous_Leg,
               Next_Leg => Next_Leg,
               Check_Status => true,
               Corefp_Status => Corefp_Status,
               Hdr_Prddataseq => Prddataseq);

    -- make sure we actually got a leg
    Init_Valid := Corefp_Status = Fppsuccess;

    -- make local copy of rta time constraint
    Rta_Time := Prov_Gleg.Fpln_Data.RTA_Time;
    Rta_Type := Prov_Gleg.Fpln_Data.RTA_Time_Type;
    -- WTS store this rta target as old rta
    R_WTS_Pkg.Old_RTA_Tgt(Fpln_Index) := Rta_Time;

  else -- SCR 9629 bad RTA pointer
    Init_Valid := False;  -- get out of module
  end if;

  -- release access to prov lgb
  Ops_Lateral_Guidance_Buffer_Manager.Releaselgb
    (Access_Id => Prov_Access_Id);

  if (Init_Valid) then
```

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

```

    -- determine which lgb is act lgb
    Fpln_ID := Ops_Lateral_Guidance_Buffer_Manager.Actfpln;
    if (Fpln_ID = FMCS_Fp_Guid_Btypes.Fpln1) then
       Route_ID := FMCS_Fp_Guid_Btypes.Gbrte1;
    elsif (Fpln_ID = FMCS_Fp_Guid_Btypes.Fpln2) then
       Route_ID := FMCS_Fp_Guid_Btypes.Gbrte2;
    else
      Init_Valid := False;  -- get out of module
    end if;
  end if;

  if (Init_Valid) then

    -- get access to act lgb
    Ops_Lateral_Guidance_Buffer_Manager.Requestlgb
      (Lgb_Process_Id => Ops_Lateral_Guidance_Buffer_Manager.Preds_Restart,
       Access_Id      => Act_Access_Id,
       Lgb_Operation  => Flight_Pln_Hdr_Types.Rte_Read,
       Route_Id       => Route_ID);

    -- get active fpln rta pointer
    RtaPtr := Ops_Lateral_Guidance_Buffer_Manager.RTA_Fix_Ptr(Act_Access_Id);

    if RtaPtr /= 0 then
       Fpp_Common_Lgb_Wrap_Pkg.Get_Lat_Leg_Status
             (Process_ID   => Fmcs_Fp_Guid_Btypes.Preds_Restart_C,
              Leg_Index   => RtaPtr,
              Lateral_Leg => Act_Gleg,
              Previous_Leg => Previous_Leg,
              Next_Leg => Next_Leg,
              Check_Status => true,
              Corefp_Status => Corefp_Status,
              Hdr_Prddataseq => Prddataseq);
    else
       Init_Valid := false;
    end if;

    -- release access to act lgb
    Ops_Lateral_Guidance_Buffer_Manager.Releaselgb
      (Access_Id => Act_Access_Id);

```

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

```
    -- check if the act and prov routes are similar (we could add more
    -- checks here: does the act wypt disttodest match, etc.)
    if not (Corefp_Status = Fppsuccess) or else ((not Act_Gleg.Perf_Data.PrdETAFixVal) or else
        (Act_Gleg.Fpln_Data.FixIdent /= Prov_Gleg.Fpln_Data.FixIdent) or else
        (not Prov_Gleg.Fpln_Data.Matchactfpln)) then
      Init_Valid := False;  -- get out of module
    end if;
  end if;

  if (Init_Valid) then   -- setup the rta data
    -- get the rta waypoint eta in the active route
    Eta := Portable_Types_Pkg.Integer_32 ( Act_Gleg.Perf_Data.PrdETAToFix );

    -- get the index of the active route for:  toc, tod, rcmd tko time
    Act_Index := Act_Prov_Index_Manager.Act_Index;

    -- compute distance and time spent in cruise before the rta waypoint
    -- note that t/c data is current a/c data when a/c is in cruise and beyond
    if (Act_Gleg.Perf_Data.Fltmode <= Fmcs_Fp_Guid_Btypes.GBClimb) then
      Rta_Idx_Data.Crz_Time := 0.0;
      Rta_Idx_Data.Crz_Dist := 0.0;
    elsif (Act_Gleg.Perf_Data.Fltmode <= Fmcs_Fp_Guid_Btypes.GBCruise) then
      Rta_Idx_Data.Crz_Time := Time_Map(
        Eta - Idx_Profile_Ifdata.Itocperfdata(Act_Index).Eta.Data);
      Rta_Idx_Data.Crz_Dist :=
        Idx_Profile_Ifdata.Itocperfdata(Act_Index).Disttodest.Data -
        Act_Gleg.Common_Data.Fixdistodest;
    else
      Rta_Idx_Data.Crz_Time := Time_Map(
        Idx_Profile_Ifdata.Itodperfdata (Act_Index).Eta.Data -
        Idx_Profile_Ifdata.Itocperfdata (Act_Index).Eta.Data);
      Rta_Idx_Data.Crz_Dist :=
        Idx_Profile_Ifdata.Itocperfdata(Act_Index).Disttodest.Data -
        Idx_Profile_Ifdata.Itodperfdata(Act_Index).Disttodest.Data;
    end if;

    -- assume that entire time in cruise is available for speed up/slow down
    Rta_Idx_Data.Tup := Rta_Idx_Data.Crz_Time;
    Rta_Idx_Data.Tdn := Rta_Idx_Data.Crz_Time;
    Rta_Idx_Data.Tup_At_Ksa_Minus := Rta_Idx_Data.Crz_Time;
    Rta_Idx_Data.Tdn_At_Ksa_Plus := Rta_Idx_Data.Crz_Time;

```

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

```
    -- note:  Perf_Preds_Lfdata.Aircraft_State.Airborne and
    -- Perf_Rta_Lfdata.Pred_Gmt_Unmodified.Data should have been set
    -- fairly recently if the preds for the active route are valid, so
    -- go ahead and use them (instead of getting a current copy)

    -- compute the time spent flying to the rta waypoint
    if ((not Perf_Preds_Lfdata.Aircraft_State.Airborne) and then
        (Ops_Cdk_Perf_Pdb_Mgr_Pkg.Takeoff_Time_Is_Pilot_Entered)) then
      Ops_Cdk_Perf_Pdb_Mgr_Pkg.Get_Takeoff_Time(Tko_Time,Act_Index);
      -- there is only one pilot entered takeoff time, fpln_index dosen't matter
      Flight_Time := Time_Map(Eta - Tko_Time.Data);
    else
      Flight_Time := Time_Map(Eta - Perf_Rta_Lfdata.Pred_Gmt_Unmodified.Data);
    end if;
  end if;

end Initial_Estimate;

---------------- L O C A L   P R O C E D U R E  (Predicted_Sequence)  -----------
procedure Predicted_Sequence is
  -- local variables
  Time_Delta      : R_Portable.Integer_32;  -- time difference
begin
  -- use predictions flight plan index for rest of module
  Fpln_Index := Perf_Preds_Lfdata.Vtpfplnindex;
  Rta_Idx_Data := Perf_Rta_Lfdata.Idx_Data(Fpln_Index);

  -- save predicted Tup, Tdn, Max_Ksa, Min_Ksa to indexed copies
  Rta_Idx_Data.Tup := Perf_Rta_Lfdata.Pred_Tup;
  Rta_Idx_Data.Tdn := Perf_Rta_Lfdata.Pred_Tdn;
  Rta_Idx_Data.Tup_At_Ksa_Minus := Perf_Rta_Lfdata.Pred_Tup_At_Ksa_Minus;
  Rta_Idx_Data.Tdn_At_Ksa_Plus := Perf_Rta_Lfdata.Pred_Tdn_At_Ksa_Plus;
  Rta_Idx_Data.Max_Ksa := Perf_Rta_Lfdata.Max_Ksa;
  Rta_Idx_Data.Min_Ksa := Perf_Rta_Lfdata.Min_Ksa;

  Flight_Time := R_Portable.Integer_32(
                  Perf_Integrators_Lfdata.IntProgBuf.TProg);

  -- SCR 9629 This SCR was for a bad RTA pointer during the initial call
  --          but we always want to protect against a bad rta pointer
  if Rtaptr /= 0 then
    Good_Pointer := True;
```

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

```

    -- make local copy of rta time constraint
    Rta_Time := Perf_LGB_Lfdata.LGB(RtaPtr).Fpln_Data.RTA_Time;
    Rta_Type := Perf_LGB_Lfdata.LGB(RtaPtr).Fpln_Data.RTA_Time_Type;

    -- reset or increment the rta loopcounter (and reset rta_happy on a firstpass)
    if (Perf_Preds_Lfdata.Vtplogic.Firstpass) then
      Rta_Idx_Data.Msg_Counter := 1;
      Rta_Idx_Data.Preds_Status.Rta_Happy := False;

      -- Before calling Common_Init, must determine where the latest step point
      -- data is stored.  In this case, its Iter_Ctr + 1
      If R_WTS_Pkg.Rta_Iter_Counter(Fpln_Index) >= R_WTS_Pkg.Num_Ksa_ETA_Records then
        -- Set iter_ctr to maximum - 1 because will be resetting to (iter_Ctr + 1)
        -- in common_init
        Iter_Ctr := R_WTS_Pkg.Num_Ksa_ETA_Records - 1;
      else
        Iter_Ctr := R_WTS_Pkg.Rta_Iter_Counter(Fpln_Index);
      end if;
      Initialize_To_Iter_Ctr := False;
      Copy_From_Opposite_Fpln := False;

      if (Perf_WTS_Lfdata.Old_Rta_Tgt(Fpln_Index) /= RTA_Time) then
        -- Re-Initialize steps data, ksa ate storage array, rta iter counter etc.
        -- Steps will be unlatched and free to move
        Common_Init;
      else
        -- for other mods, the arrays will not be cleared, because the step location
        -- checking logic will filter out any bad data before curve fitting to find
        -- the next ksa.  However, steps will be unlatched for any Mods (which
        -- as of 7/18/12 is now done in Vtp_Init by setting Flat_Bias_Factor to
        -- 0.0 when the Firstpass flag is true).
        Perf_WTS_Lfdata.Flat_Bias_Factor := 0.0;      -- not necessary
        Perf_WTS_Lfdata.New_Flat_Bias_Factor := 0.0;  -- not necessary
        Perf_WTS_Lfdata.Rta_Iter_Counter(Fpln_Index) :=
                        Perf_WTS_Lfdata.Rta_Iter_Counter(Fpln_Index) + 1;
      end if;

    elsif (Rta_Idx_Data.Msg_Counter >= 30000) then  -- 30000 is less than maxint
      Rta_Idx_Data.Msg_Counter := 6;  -- prevent overflow

    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)

```
        -- 30000 is less than maxint
        R_WTS_Pkg.Rta_Iter_Counter(Fpln_Index) := 11;
        -- prevent overflow
        -- go to 11 so that array locations not overwritten
      else
        Rta_Idx_Data.Msg_Counter := Rta_Idx_Data.Msg_Counter + 1;
        -- WTS : increment interation counter
        R_WTS_Pkg.Rta_Iter_Counter(Fpln_Index) :=
               R_WTS_Pkg.Rta_Iter_Counter(Fpln_Index) + 1;
      end if;

      -- The previous setup of Iter_Ctr was for the purposes of Common_Init only.
      -- This setup is also necessary.  Don't move this code.
      If R_WTS_Pkg.Rta_Iter_Counter(Fpln_Index) >= R_WTS_Pkg.Num_Ksa_ETA_Records then
        Iter_Ctr := R_WTS_Pkg.Num_Ksa_ETA_Records;
      else
        Iter_Ctr := R_WTS_Pkg.Rta_Iter_Counter(Fpln_Index);
      end if;

      -- there is no guaranty that lgb_store_data has run to store the eta, so
      -- compute the predicted eta for the leg; besides, if there is a recommended
      -- takeoff time, we don't want the eta to be based on it
      if ( (not Perf_Preds_Lfdata.Aircraft_State.Airborne) and then
          Ops_Cdk_Perf_Pdb_Mgr_Pkg.Takeoff_Time_Is_Pilot_Entered ) then

        Ops_Cdk_Perf_Pdb_Mgr_Pkg.Get_Takeoff_Time(Tko_Time,Fpln_Index);
        Time_Delta := Tko_Time.Data - Perf_Rta_Lfdata.Pred_Gmt_Unmodified.Data;

        --  make sure pilot entered takeoff time is in the future
        if ( ((Time_Delta > 0) and then (Time_Delta < Half_Day)) or else
            ((Time_Delta < -Half_Day) and then (Time_Delta > -One_Day)) )
        then  -- pilot entered takeoff time is in future, use it
          Eta := Time_Map(Tko_Time.Data + Flight_Time);
        else  -- use the GMT variable that was not adjusted for pilot entered time
          Eta := Time_Map(Perf_Rta_Lfdata.Pred_Gmt_Unmodified.Data + Flight_Time);
        end if;

      else
        -- use the GMT variable that was not adjusted for recommended takeoff time
        Eta := Time_Map(
             Perf_Rta_Lfdata.Pred_Gmt_Unmodified.Data + Flight_Time);
      end if;
```

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

```

    -- compute distance and time spent in cruise before the rta waypoint
    -- note that t/c data is current a/c data when a/c is in cruise and beyond
    if (Perf_Preds_Lfdata.Desiredphase <= Fmcs_Base_Types.Climb) then
      Rta_Idx_Data.Crz_Time := 0.0;
      Rta_Idx_Data.Crz_Dist := 0.0;
    elsif (Perf_Preds_Lfdata.Desiredphase <= Fmcs_Base_Types.Cruise) then
      Rta_Idx_Data.Crz_Time := Time_Map(R_Portable.Integer_32(
        Eta) - Perf_Profile_Lfdata.Tocperfdata.Eta.Data);
      Rta_Idx_Data.Crz_Dist :=
        Perf_Profile_Lfdata.Tocperfdata.Disttodest.Data -
        Perf_Integrators_Lfdata.IntProgBuf.XProg;
    else
      Rta_Idx_Data.Crz_Time := Time_Map(
        Perf_Profile_Lfdata.Todperfdata.Eta.Data -
        Perf_Profile_Lfdata.Tocperfdata.Eta.Data);
      Rta_Idx_Data.Crz_Dist :=
        Perf_Profile_Lfdata.Tocperfdata.Disttodest.Data -
        Perf_Profile_Lfdata.Todperfdata.Disttodest.Data;
    end if;

  else -- SCR 9629 Bad RTA Pointer
    Good_Pointer := False;
  end if;

end Predicted_Sequence;

--------------- L O C A L   P R O C E D U R E  (Compute_Ate)  ------------------
procedure Compute_Ate is
  -- local variables
  Data_Use_Tolerance : R_Portable.Integer_32 := 1;
begin
  -- Do NOT mess with the Fpln_Index variable.  It has been set already

  -- make local copy of ksa that was used to generate the current trip preds
  Ksa_Old := Rta_Idx_Data.Ksa;

  -- determine time target based on rta type
  if (Rta_Type = Ac_Position_Types.At_Time) then
    Time_Target := Rta_Time;
  elsif (Rta_Type = Ac_Position_Types.After) then
    Time_Target := Rta_Time + Perf_Rta_Lfdata.Message_Tol;
```

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

```
elsif (Rta_Type = Ac_Position_Types.Before) then
  Time_Target := Rta_Time - Perf_Rta_Lfdata.Message_Tol;
end if;

-- compute arrival time error (ate) at the rta waypoint
Rta_Idx_Data.Ate := Eta - Time_Target;
if (Rta_Idx_Data.Ate > Half_Day) then
  Rta_Idx_Data.Ate := Rta_Idx_Data.Ate - One_Day;
elsif (Rta_Idx_Data.Ate < -Half_Day) then
  Rta_Idx_Data.Ate := Rta_Idx_Data.Ate + One_Day;
end if;

-- WTS : Store Ksa_ETA record data.  If neccessary, slide all
-- elements back one place to store last R_WTS_Pkg.Num_Ksa_ETA_Records
-- number of records.
if (R_WTS_Pkg.Rta_Iter_Counter(Fpln_Index) > R_WTS_Pkg.Num_Ksa_ETA_Records) then
  Loop_Ctr := 1;
  While Loop_Ctr < R_WTS_Pkg.Num_Ksa_ETA_Records loop
    R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, Loop_Ctr).Ksa_Guess :=
          R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, Loop_Ctr + 1).Ksa_Guess;
    R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, Loop_Ctr).ATE_Result :=
          R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, Loop_Ctr + 1).ATE_Result;
    R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, Loop_Ctr).Bad_Data :=
          R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, Loop_Ctr + 1).Bad_Data;
    Loop_Ctr := Loop_Ctr + 1;
  end loop;
end if;

R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, Iter_Ctr).Ksa_Guess := Rta_Idx_Data.Ksa;
R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, Iter_Ctr).ATE_Result := Rta_Idx_Data.Ate;

-- determine if aircraft is on time (for speed adjustment logic)
if ( ((Rta_Type = Ac_Position_Types.AT_Time) and then
      (abs(Rta_Idx_Data.Ate) < Perf_Rta_Lfdata.Spd_Adj_Tol))
   or else
     ((Rta_Type = Ac_Position_Types.After) and then
      ( ((abs(Rta_Idx_Data.Ate) < Perf_Rta_Lfdata.Spd_Adj_Tol) and then
         (Ksa_Old <= 0.0))
        or else
        ((Rta_Idx_Data.Ate > Perf_Rta_Lfdata.Spd_Adj_Tol) and then
         (Ksa_Old = 0.0)) ))
   or else
```

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

```
        ((Rta_Type = Ac_Position_Types.Before) and then
         ( ((abs(Rta_Idx_Data.Ate) < Perf_Rta_Lfdata.Spd_Adj_Tol) and then
            (Ksa_Old >= 0.0))
           or else
           ((Rta_Idx_Data.Ate < -Perf_Rta_Lfdata.Spd_Adj_Tol) and then
            (Ksa_Old = 0.0)) ))
      )
  then  -- aircraft is on time, do not adjust ksa
    Rta_Idx_Data.Ate := 0;
    Rta_Idx_Data.Preds_Status.Within_Spd_Adj_Tol := True;

    -- WTS: Before calling Common_Init, have to determine where the latest
    -- step point data is stored
    Initialize_To_Iter_Ctr := True;
    Copy_From_Opposite_Fpln := False;

    -- WTS : Initialize data storage array, ksa ate records, RTA iter counter and
    -- reset biasing flags, copy the step point locations into the first
    -- array locations so that we have them roll counter back to 1
    Common_Init;

    -- WTS :  Max the biasing factor out to freeze steps when on time.
    if (R_WTS_Pkg.Want_To_Use_Flat_Bias) then
      R_WTS_Pkg.New_Flat_Bias_Factor := R_WTS_Pkg.Cost_Bias_Factor;
    end if;

  else
    Rta_Idx_Data.Preds_Status.Within_Spd_Adj_Tol := False;

    -- If we started biasing the cost data on this pass thru perf, sort through
    -- all of the ksa-ate data and "throw out" the data which has a step point
    -- significantly different than where we're trying to hold it at.
    if R_WTS_Pkg.Want_To_Use_Flat_Bias and
       (R_WTS_Pkg.Flat_Bias_Factor >= R_WTS_Pkg.Initial_Flat_Bias_Factor) and
       (Iter_Ctr >= 2) and
       R_WTS_Pkg.Want_To_Sort_Data then

      for Loop_Ctr in 1..(Iter_Ctr - 1) loop
        -- Should check all of the steps, not just the first one
        for Rec_Ctr in Alt_Profile_Iftypes.NumScPts loop
          if R_WTS_Pkg.Step_Dist_Rec
                (Fpln_Index,Rec_Ctr,Loop_Ctr).Valid
```

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

```
                and then
             (abs(R_WTS_Pkg.Step_Dist_Rec
              (Fpln_Index,Rec_Ctr, Loop_Ctr).Data -
             R_WTS_Pkg.Step_Dist_Rec
              (Fpln_Index,Rec_Ctr,Iter_Ctr).Data) >
             R_WTS_Pkg.Max_Step_Movement)
          then
            R_WTS_Pkg.KSA_ETA_REC
              (Fpln_Index, Loop_Ctr).Bad_Data := true;
            exit;
            -- when step is out of wack, get out of inner (all step points) loop
          else -- if this data was previously thrown out and is now good, use it
            R_WTS_Pkg.KSA_ETA_REC
              (Fpln_Index, Loop_Ctr).Bad_Data := false;
          end if;
        end loop; -- loop through all step points
      end loop;
    end if;

  end if;

end Compute_Ate;

--------------- L O C A L   P R O C E D U R E  (Compute_Ksa)  ------------------
-- compute speed adjustment factor (Ksa) for next pass of trip predictions
-- Note:  Ksa is a constant TAS adjustment, in knots
-- Note:  When reducing the (abs) speed adjustment, the appropriate Tavail
-- to use is somewhere between Tup and Tdn (e.g. for reducing a speed up,
-- small Ate's should use a Tavail near Tup, large Ate's should use a
-- Tavail near Tdn).  If we assume a ratio of delta(Tavail/Ksa) = Slope
-- then we can compute a quadratic equation for Tavail.  This equation
-- can be approximated with it's most significant term, giving:
-- Tavail = Tavail(ksa_old) + (x1 * Sqrt(Slope*Ate*Avg_Crz_Gndspd)).
-- where x1 is between 0.707 (Sqrt(2)/2) and 1.0;  0.707 works best for
-- small Ate's and 1.0 is better for large Ate's.
-- Of course this works best when the Ate is small, since the computation
-- of the Slope is only valid for a small range of Ksa.

procedure Compute_Ksa is
  -- local variables
  Avg_Crz_Gndspd : R_Portable.Float_32;  -- average cruise groundspeed
  Ate            : R_Portable.Float_32;  -- arrival time error (float)
```

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

```
  Minimum        : R_Portable.Float_32;  -- min tavail
  Slope          : R_Portable.Float_32;  -- slope of tavail/ksa curve
  Invalid_Ate_Tolerance : R_Portable.Integer_32 := 1;
  Good_Curve_Fit_Data : Boolean := False; -- if all of the ate_result data is
  -- bad, then this variable will remain false and we will use the existing ksa_new
begin

  if (Rta_Idx_Data.Crz_Time <= 0.0) then  -- RTA in clb or a/c in des
    -- do not adjust speed but may adjust rcmd tko time
    -- solution is finished if ksa_old is zero too
    Rta_Idx_Data.Preds_Status.Rta_Happy := (Ksa_Old = 0.0);
    Ksa_New := 0.0;

  elsif ((Rta_Idx_Data.Ate = 0) or else
         (Rta_Idx_Data.Crz_Time < Perf_Rta_Lfdata.Min_Crz_Time)) then
    -- aircraft is on time or RTA fix is too near a/c or t/c, do not adjust ksa
    -- (this also protects Avg_Crz_Gndspd computation below)
    Rta_Idx_Data.Preds_Status.Rta_Happy := True;  -- solution is finished
    Ksa_New := Ksa_Old;  -- do not change ksa

  else  -- adjust ksa
    -- solution is not finished, unless
    -- 1) a rcmd tko time is being computed (Rta_Happy is set later)
    -- 2) there is no time available to adjust speed (Rta_Happy is set later)
    Rta_Idx_Data.Preds_Status.Rta_Happy := False;

    -- If we are at max speed up or max slow down and still can't make RTA, we
    -- shouldn't look for new ksa tries.  It may cause the code to blow up.

    If Rta_Idx_Data.Preds_Status.At_Max_Speed_Up or
       Rta_Idx_Data.Preds_Status.At_Max_Slow_Down then

      if abs(R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, Iter_Ctr).Ate_Result)>=
                   Perf_Rta_Lfdata.Spd_Adj_Tol then
        if R_WTS_Pkg.KSA_ETA_REC(Fpln_Index,Iter_Ctr).Ate_Result > 0
                          then
          -- at max speed up and still late, so can't make RTA.  Bump Ksa old
          -- up by a little so that it will trip the at max speed up logic
          -- again. This should set it to the same value on the next pass
          Ksa_New := Ksa_Old + 1.0;
        elsif
           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)

```
            -- at max slow down and still early, so can't make RTA.  Lower ksa
            -- old some so that we trip the max slow down logic.  This should
            -- reset ksa to the same value on the next pass
            Ksa_New := Ksa_Old - 1.0;
          end if;
       end if; -- at max speed up or slow down
     elsif (R_WTS_Pkg.Rta_Iter_Counter(Fpln_Index) > R_WTS_Pkg.Num_Ksa_ETA_Records and
       (R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, 1).Ate_Result = 0 or
        R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, R_WTS_Pkg.Num_Ksa_ETA_Records).Ate_Result = 0)) then
       -- this is a case where you're doing a slowdown while on the ground,
       -- so Iter_Ctr will be allowed to go past 1 even though the ate
       -- result is 0.  Skip the ksa guess logic to prevent a divide by zero
       Ksa_New := 0.0;
     else
       -- perform normal ksa guess logic
       -- if there are less that two data points available, we can't do a
       -- guess with this method.  If there are two points available, we
       -- will do a linear fit.  If there are more than two, we will do the
       -- linear fit using the two points closest to the RTA target only.
       if (Iter_Ctr = 2) then
         KSA_ATE_Slope := R_Portable.Float_32
             (R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, 2).ATE_Result -
              R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, 1).ATE_Result);
         Good_Curve_Fit_Data := True;
         if (KSA_ATE_Slope = 0.0) then
           -- limit the slope and prevent a possible divide by zero
           KSA_ATE_Slope := -1.0 * R_WTS_Pkg.Default_KSA_Ate_Slope;
         else
           KSA_ATE_Slope := (R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, 2).Ksa_Guess -
                             R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, 1).Ksa_Guess) /
                             KSA_ATE_Slope;
         end if;

         -- we cannot allow a positive slope, and we want to bound the slope
         if (Ksa_ATE_Slope >= 0.0) then
           KSA_ATE_Slope := -1.0 * R_WTS_Pkg.Default_KSA_Ate_Slope;
         elsif (Ksa_ATE_Slope > (-1.0 *  R_WTS_Pkg.KSA_Ate_Slope_Upper_Bound)) then
           KSA_ATE_Slope :=  -1.0 * R_WTS_Pkg.KSA_Ate_Slope_Upper_Bound;
         elsif (Ksa_ATE_Slope < (-1.0 * R_WTS_Pkg.KSA_Ate_Slope_Lower_Bound)) then
           KSA_ATE_Slope :=  -1.0 * R_WTS_Pkg.KSA_Ate_Slope_Lower_Bound;
         end if;

```

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

```
        Ksa_New := R_WTS_Pkg.KSA_ETA_REC(Fpln_Index,Iter_Ctr).Ksa_Guess
                     - KSA_ATE_Slope * R_Portable.Float_32
                         (R_WTS_Pkg.KSA_ETA_REC(Fpln_Index,Iter_Ctr)
                          .ATE_Result);

      elsif (Iter_Ctr /= 1) then

        -- loop through the array, keeping the last data point, plus the
        -- other data point closest to the RTA.  If there is a data point
        -- on "the other side" of the RTA, we want to use this point
        -- because it will give the best next guess, even if it isn't
        -- the closest point to the rta target.  The points furthest
        -- from the RTA will be placed in the lowest array locations so
        -- that they will be dropped off as array locations are overwritten.
        if R_WTS_Pkg.Want_To_Sort_Data then
          Num_Places := Iter_Ctr - 1;
          Min_ATE := 100000000;
          Max_ATE := 0;
          Min_ATE_Opp_Sign := 10000000;
          Max_ATE_Opp_Sign := 0;
          Opposite_Sign_Exist := False;
          Loop_Ctr := 1;
          Max_Location := 1;
          Min_Location := Num_Places;

          Find_Min : For Loop_Ctr in 1..Num_Places loop

            if not(R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, Loop_Ctr).Bad_Data)
            then
              Good_Curve_Fit_Data := True; -- Have at least one good data point
              -- only do the sorting logic if this is good data
              if R_Portable.Float_32(R_WTS_Pkg.KSA_ETA_REC
                     (Fpln_Index, Loop_Ctr).ATE_Result) /
                 R_Portable.Float_32(R_WTS_Pkg.KSA_ETA_REC
                        (Fpln_Index,Iter_Ctr).ATE_Result)
                  < 0.0 then

                Opposite_Sign := True;
                Opposite_Sign_Exist := True;
              else
                Opposite_Sign := False;
              end if;
```

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

```

              if (abs(R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, Loop_Ctr)
                     .ATE_Result) < abs(Min_Ate) and
                      (not Opposite_Sign_Exist)) or
                 (abs(R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, Loop_Ctr)
                     .ATE_Result) < abs(Min_Ate_Opp_Sign) and
                      (Opposite_Sign))
              then
                Min_ATE := R_WTS_Pkg.KSA_ETA_REC(Fpln_Index,
                           Loop_Ctr).ATE_Result;
                Min_Location := Loop_Ctr;
              end if;
              if abs(R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, Loop_Ctr).ATE_Result) >
                     abs(Max_Ate) and (not Opposite_Sign)
              then
                Max_ATE := R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, Loop_Ctr).ATE_Result;
                Max_Location := Loop_Ctr;
              end if;
            else -- not good data, so move this location to the lowest array location
                 -- so that it can be dropped from consideration eventually
              Max_Location := Loop_Ctr;
            end if; -- good_data check
          end loop Find_Min;

          -- only perfrom the Min_Location movement if there was at least one good data
          -- point
          if Good_Curve_Fit_Data then
            if (Min_Location /= Num_Places) then
              if (Iter_Ctr = 3 and
                 Max_Location /= 1) then
                null;
              else
                Save_ATE := R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, Num_Places).ATE_Result;
                R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, Num_Places).ATE_Result :=
                    R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, Min_Location).ATE_Result;
                R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, Min_Location).ATE_Result :=
                    Save_ATE;
                Save_Ksa := R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, Num_Places).Ksa_Guess;
                R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, Num_Places).Ksa_Guess :=
                    R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, Min_Location).Ksa_Guess;
                R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, Min_Location).Ksa_Guess :=
                    Save_Ksa;
```

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

```
                Save_BadData := R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, Num_Places).Bad_Data;
                R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, Num_Places).Bad_Data :=
                    R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, Min_Location).Bad_Data;
                R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, Min_Location).Bad_Data :=
                    Save_BadData;
              end if;
            end if;
          end if; -- the Min data movement

          -- The max location movement is don whether there is good data or not
          if Max_Location /= 1 then
            Save_ATE := R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, 1).ATE_Result;
            R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, 1).ATE_Result :=
                  R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, Max_Location).ATE_Result;
            R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, Max_Location).ATE_Result :=
                  Save_ATE;
            Save_Ksa := R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, 1).Ksa_Guess;
            R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, 1).Ksa_Guess :=
                 R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, Max_Location).Ksa_Guess;
            R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, Max_Location).Ksa_Guess :=
                  Save_Ksa;
            Save_BadData := R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, 1).Bad_Data;
            R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, 1).Bad_Data :=
                R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, Max_Location).Bad_Data;
            R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, Max_Location).Bad_Data :=
                Save_BadData;
          end if;
        end if; -- sorting array

        -- 4/23/97 Want to be able to use this logic if not sorting data
        if (R_WTS_Pkg.Want_To_Sort_Data and Good_Curve_Fit_Data) or
           (not R_WTS_Pkg.Want_To_Sort_Data) then
          -- use standard method to compute ksa_new
          KSA_ATE_Slope := R_Portable.Float_32
            (R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, Iter_Ctr).ATE_Result -
             R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, Iter_Ctr - 1).ATE_Result);
          if (KSA_ATE_Slope = 0.0) then
            -- limit the slope and prevent a possible divide by zero
            KSA_ATE_Slope := -1.0 * R_WTS_Pkg.Default_KSA_Ate_Slope;
          else
            KSA_ATE_Slope :=
              (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)

```
               R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, Iter_Ctr - 1).Ksa_Guess) /
               KSA_ATE_Slope;
          end if;
          -- We cannot allow a positive slope to be used.  If positive, use a default
          -- negative value (chosen so that it will probably undershoot solution).
          if KSA_ATE_Slope > 0.0 then
            KSA_ATE_Slope := -1.0 * R_WTS_Pkg.Default_KSA_Ate_Slope;
          elsif (Ksa_ATE_Slope > (-1.0 *  R_WTS_Pkg.KSA_Ate_Slope_Upper_Bound)) then
            KSA_ATE_Slope :=  -1.0 * R_WTS_Pkg.KSA_Ate_Slope_Upper_Bound;
          elsif (Ksa_ATE_Slope < (-1.0 * R_WTS_Pkg.KSA_Ate_Slope_Lower_Bound)) then
            KSA_ATE_Slope :=  -1.0 * R_WTS_Pkg.KSA_Ate_Slope_Lower_Bound;
          end if;
          Ksa_New := R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, Iter_Ctr).Ksa_Guess -
                       KSA_ATE_Slope * R_Portable.Float_32
                       (R_WTS_Pkg.KSA_ETA_REC(Fpln_Index, Iter_Ctr).ATE_Result);
        else -- use the alternate ksa prediction method
          -- compute average cruise ground speed
          Avg_Crz_Gndspd := 3600.0 * Rta_Idx_Data.Crz_Dist / Rta_Idx_Data.Crz_Time;
          if ((Avg_Crz_Gndspd < 100.0) or else (Avg_Crz_Gndspd > 1000.0)) then
            Avg_Crz_Gndspd := 100.0;
          end if;

          -- make a floating point copy of Ate, since it's used a lot
          Ate := R_Portable.Float_32(Rta_Idx_Data.Ate);

          -- find time available to adjust speed
          if (Ksa_Old >= 0.0) then  -- previous predictions had a speed up
            if (Rta_Idx_Data.Ate >= 0) then  -- eta is late, speed up more
              Rta_Idx_Data.Tavail := Rta_Idx_Data.Tup;
            else  -- (Ate < 0)  -- eta is early, reduce speed up
              -- compute slope of tavail/ksa (Tup_At_Ksa_Minus is Tup at Ksa-k3)
              Slope := (Rta_Idx_Data.Tup_At_Ksa_Minus - Rta_Idx_Data.Tup) /
                       Perf_Rta_Lfdata.k3;
              if (Slope < Perf_Rta_Lfdata.k1) then
                Slope := Perf_Rta_Lfdata.k1;
              end if;
              Rta_Idx_Data.Tavail := Rta_Idx_Data.Tup +
                  (Perf_Rta_Lfdata.k8 * Math_Pkg.Sqrt(Slope * Avg_Crz_Gndspd * (-Ate)));
              -- compute minimum tavail to prevent overshoot when slope is small
              -- and ate is large (since the slope was only computed at one point)
              if (Ksa_Old > 1.0) then
                Minimum := Rta_Idx_Data.Tup +
```

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

```
                           (2.0 * (-Ate) * Avg_Crz_Gndspd / Ksa_Old);
                if (Rta_Idx_Data.Tavail < Minimum) then
                  Rta_Idx_Data.Tavail := Minimum;
                end if;
              end if;
              if (Rta_Idx_Data.Tavail > Rta_Idx_Data.Tdn) then
                Rta_Idx_Data.Tavail := Rta_Idx_Data.Tdn;
              end if;
            end if;
          else  -- ((Ksa_Old < 0.0) -- previous predictions had a slow down
            if (Rta_Idx_Data.Ate <= 0) then  -- eta is early, slow down more
              Rta_Idx_Data.Tavail := Rta_Idx_Data.Tdn;
            else  -- (Ate > 0) -- eta is late, reduce slow down
              -- compute slope of tavail/ksa (Tdn_At_Ksa_Plus is Tdn at Ksa+k3)
              Slope := (Rta_Idx_Data.Tdn_At_Ksa_Plus - Rta_Idx_Data.Tdn) /
                       Perf_Rta_Lfdata.k3;
              if (Slope < Perf_Rta_Lfdata.k2) then
                Slope := Perf_Rta_Lfdata.k2;
              end if;
              Rta_Idx_Data.Tavail := Rta_Idx_Data.Tdn +
                (Perf_Rta_Lfdata.k9 * Math_Pkg.Sqrt(Slope * Avg_Crz_Gndspd * Ate));
              -- compute minimum tavail to prevent overshoot when slope is small
              -- and ate is large (since the slope was only computed at one point)
              if (Ksa_Old < -1.0) then
                Minimum := Rta_Idx_Data.Tdn +
                           (2.0 * Ate * Avg_Crz_Gndspd / (-Ksa_Old));
                if (Rta_Idx_Data.Tavail < Minimum) then
                  Rta_Idx_Data.Tavail := Minimum;
                end if;
              end if;
              if (Rta_Idx_Data.Tavail > Rta_Idx_Data.Tup) then
                Rta_Idx_Data.Tavail := Rta_Idx_Data.Tup;
              end if;
            end if;
          end if;

          -- compute new speed adjustment factor (ksa)
          if (Rta_Idx_Data.Tavail > (2.0 * Ate)) then
            if abs(Rta_Idx_Data.Tavail) > 10.0 then
              Ksa_New := Ksa_Old + Avg_Crz_Gndspd * (Ate / (Rta_Idx_Data.Tavail));
            else
              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)

```
            end if;
          else
            Ksa_New := Ksa_Old + Avg_Crz_Gndspd;
          end if;
          -- end of alternate ksa prediction method
        end if; -- end of standard ksa computation
      else -- if get to here, were not able to do a curve fit, so use alternate method
        -- compute average cruise ground speed
        Avg_Crz_Gndspd := 3600.0 * Rta_Idx_Data.Crz_Dist / Rta_Idx_Data.Crz_Time;
        if ((Avg_Crz_Gndspd < 100.0) or else (Avg_Crz_Gndspd > 1000.0)) then
          Avg_Crz_Gndspd := 100.0;
        end if;

        -- make a floating point copy of Ate, since it's used a lot
        Ate := R_Portable.Float_32(Rta_Idx_Data.Ate);

        -- find time available to adjust speed
        if (Ksa_Old >= 0.0) then  -- previous predictions had a speed up
          if (Rta_Idx_Data.Ate >= 0) then  -- eta is late, speed up more
            Rta_Idx_Data.Tavail := Rta_Idx_Data.Tup;
          else  -- (Ate < 0)  -- eta is early, reduce speed up
            -- compute slope of tavail/ksa (Tup_At_Ksa_Minus is Tup at Ksa-k3)
            Slope := (Rta_Idx_Data.Tup_At_Ksa_Minus - Rta_Idx_Data.Tup) /
                     Perf_Rta_Lfdata.k3;
            if (Slope < Perf_Rta_Lfdata.k1) then
              Slope := Perf_Rta_Lfdata.k1;
            end if;
            Rta_Idx_Data.Tavail := Rta_Idx_Data.Tup +
                 (Perf_Rta_Lfdata.k8 * Math_Pkg.Sqrt(Slope * Avg_Crz_Gndspd * (-Ate)));
            -- compute minimum tavail to prevent overshoot when slope is small
            -- and ate is large (since the slope was only computed at one point)
            if (Ksa_Old > 1.0) then
              Minimum := Rta_Idx_Data.Tup +
                              (2.0 * (-Ate) * Avg_Crz_Gndspd / Ksa_Old);
              if (Rta_Idx_Data.Tavail < Minimum) then
                Rta_Idx_Data.Tavail := Minimum;
              end if;
            end if;
            if (Rta_Idx_Data.Tavail > Rta_Idx_Data.Tdn) then
              Rta_Idx_Data.Tavail := Rta_Idx_Data.Tdn;
            end if;
          end if;
```

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

```
        else  -- ((Ksa_Old < 0.0) -- previous predictions had a slow down
          if (Rta_Idx_Data.Ate <= 0) then  -- eta is early, slow down more
            Rta_Idx_Data.Tavail := Rta_Idx_Data.Tdn;
          else  -- (Ate > 0) -- eta is late, reduce slow down
            -- compute slope of tavail/ksa (Tdn_At_Ksa_Plus is Tdn at Ksa+k3)
            Slope := (Rta_Idx_Data.Tdn_At_Ksa_Plus - Rta_Idx_Data.Tdn) /
                         Perf_Rta_Lfdata.k3;
            if (Slope < Perf_Rta_Lfdata.k2) then
              Slope := Perf_Rta_Lfdata.k2;
            end if;
            Rta_Idx_Data.Tavail := Rta_Idx_Data.Tdn +
                  (Perf_Rta_Lfdata.k9 * Math_Pkg.Sqrt(Slope * Avg_Crz_Gndspd * Ate));
            -- compute minimum tavail to prevent overshoot when slope is small
            -- and ate is large (since the slope was only computed at one point)
            if (Ksa_Old < -1.0) then
              Minimum := Rta_Idx_Data.Tdn +
                             (2.0 * Ate * Avg_Crz_Gndspd / (-Ksa_Old));
              if (Rta_Idx_Data.Tavail < Minimum) then
                Rta_Idx_Data.Tavail := Minimum;
              end if;
            end if;
            if (Rta_Idx_Data.Tavail > Rta_Idx_Data.Tup) then
              Rta_Idx_Data.Tavail := Rta_Idx_Data.Tup;
            end if;
          end if;
        end if;

        -- compute new speed adjustment factor (ksa)
        if (Rta_Idx_Data.Tavail > (2.0 * Ate)) then
          if abs(Rta_Idx_Data.Tavail) > 10.0 then
            Ksa_New := Ksa_Old + Avg_Crz_Gndspd * (Ate / (Rta_Idx_Data.Tavail));
          else
            Ksa_New := Ksa_Old + Avg_Crz_Gndspd * (Ate / (Rta_Idx_Data.Tavail - Ate));
          end if;
        else
          Ksa_New := Ksa_Old + Avg_Crz_Gndspd;
        end if;
        -- end of alternate ksa prediction method
      end if;
    end if;

    -- 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)

```
    -- but don't issue the messages in marginal cases where wind gusts may
    -- be making the solution oscillate between achievable and unachievable.
    -- This margin is currently controlled by k10.
    if ( (Rta_Idx_Data.Tavail <= 0.0) and then
         ((abs(Rta_Idx_Data.Ate) > Perf_Rta_Lfdata.k10) or else
          (Rta_Idx_Data.Msg_Counter >= Perf_Rta_Lfdata.Max_Rta_Loops)) ) then
      Rta_Idx_Data.Preds_Status.Rta_Happy := True;  -- issue 'unable' message
    end if;

  end if;  -- end of KSA adjustment

  -- Do not speed up for At/After RTA's or slow down for At/Before RTA's
  if ( ((Rta_Type = Ac_Position_Types.After) and then (Ksa_New > 0.0)) or else
       ((Rta_Type = Ac_Position_Types.Before) and then (Ksa_New < 0.0)) ) then
    Ksa_New := 0.0;
  end if;

end Compute_Ksa;

--------------- L O C A L   P R O C E D U R E  (Compute_Rcmd_Tko_Time)  --------
-- Note: be very, very careful when modifying this code

procedure Compute_Rcmd_Tko_Time is
  -- local variables
  Time_Delta  : R_Portable.Integer_32; -- time difference
begin
  -- Never predict a slow down to meet a rta time if a delayed takeoff can be
  -- predicted instead.  Trip predictions must be performed at the unadjusted
  -- econ speed (ksa = 0) in order to compute a recommended takeoff time, so
  -- set ksa = 0 to be able to compute a recommended takeoff time next pass.
  if (Ksa_New < 0.0) then
    Ksa_New := 0.0;
  end if;

  -- When (current or next) predictions reflect a rta speed up (ksa > 0) or
  -- when the rta time is an 'at or before', the recommended takeoff time
  -- must be 'NOW'.
  if ((Ksa_Old > 0.0) or else
      (Ksa_New > 0.0) or else
      (Rta_Type = Ac_Position_Types.Before))
  then
    -- recommended takeoff time is 'NOW'
```

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

```
    Rta_Idx_Data.Rcmd_Takeoff.Data.Is_Now := True;
    Rta_Idx_Data.Rcmd_Takeoff.Data.Time :=
      Perf_Rta_Lfdata.Pred_Gmt_Unmodified.Data;
    -- if pred's are stable, set recommended takeoff time valid
    -- do not display "NOW" after a MOD until we are sure of it
    if ( (Rta_Type = Ac_Position_Types.Before) or else
         (Rta_Idx_Data.Preds_Status.Rta_Happy) or else
         (Rta_Idx_Data.Msg_Counter >= Perf_Rta_Lfdata.Max_Rta_Loops) ) then
      Rta_Idx_Data.Rcmd_Takeoff.Valid := True;
    end if;

  -- To compute a recommended takeoff time other than 'NOW', predictions
  -- using the unadjusted econ speed (Ksa_Old = 0) must arrive at the RTA
  -- waypoint before the RTA time (Ksa_New <= 0, Ate <= 0.0).
  elsif ((Ksa_Old = 0.0) and then (Ksa_New <= 0.0) and then
         (Rta_Idx_Data.Ate <= 0)) then
    -- compute recommended takeoff time
    Rta_Idx_Data.Rcmd_Takeoff.Valid := True;
    Rta_Idx_Data.Rcmd_Takeoff.Data.Time := Time_Map(Time_Target - Flight_Time);
    -- If rta time is 'at or after' and the recommended takeoff time is
    -- more than 12 hours in the future or less than 12 hours in the past,
    -- then the recommended takeoff time is 'NOW'.
    Time_Delta := Rta_Idx_Data.Rcmd_Takeoff.Data.Time -
                  Perf_Rta_Lfdata.Pred_Gmt_Unmodified.Data;
    if ((Rta_Type = Ac_Position_Types.After) and then
        ( (Time_Delta > Half_Day) or else
          ((Time_Delta < 0) and then
           (Time_Delta > -Half_Day)) )) then
      Rta_Idx_Data.Rcmd_Takeoff.Data.Is_Now := True;
    else
      Rta_Idx_Data.Rcmd_Takeoff.Data.Is_Now := False;
      Rta_Idx_Data.Preds_Status.Rta_Happy := True;
    end if;
  end if;
  -- note that if ((Ksa_Old < 0) and (Ksa_New = 0)) then the recommended
  -- takeoff time is unchanged (except for 'Before' rta)

  -- note that the rcmd tko time is not output to manager until the end of
  -- predictions so that it will not conflict with the predicted ETA's

end Compute_Rcmd_Tko_Time;

```

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

```
--------------- L O C A L   P R O C E D U R E  (Limit_Ksa)  --------------------
procedure Limit_Ksa is
  -- local variables
  Ksa_Change  : R_Portable.Float_32;  -- Ksa_Change = Ksa_New - Ksa_Old
  Upper_Limit : R_Portable.Float_32;  -- Upper_Limit = Max_Ksa + k6
  Lower_Limit : R_Portable.Float_32;  -- Lower_Limit = Min_Ksa - k6
  Max_Ksa_Step_Local : R_Portable.Float_32;
  Zero_CI_Mach    : R_Portable.Float_32; -- Mach number with CI=0
  CI_Mach_Slope : R_Portable.Float_32; -- Slope of Cost Index vs Mach
  Corr_Hundred_Ci : Portable_Types_Pkg.Float_32; -- corrected cost index
  Hundred_Ci_Mach : Portable_Types_Pkg.Float_32; -- Mach number at step altitude with CI=100

begin
  -- Fpln_index & Iter_Ctr variables set in Initialization: Do Not Modify

  -- reduce ksa change to help prevent overshoot, but do not alter ksa when it
  -- has been set to 0.0 (it is very unlikely to be computed as exactly 0.0)
  if (Ksa_New /= 0.0) then  -- limit ksa rate of change
    Ksa_Change := Ksa_New - Ksa_Old;
    if ((Ksa_Old * Ksa_Change) > 0.0) then  -- check for same sign
      -- approaching the ksa limit, the natural tendancy of Tavail to become
      -- smaller already provides good rate limiting so use a small factor
      Ksa_Change := Perf_Rta_Lfdata.k4 * Ksa_Change;
    else  -- comming down from the ksa limit, the unreliability of Tavail
      -- may require a larger rate limit to help prevent overshoot
      Ksa_Change := Perf_Rta_Lfdata.k5 * Ksa_Change;
    end if;
    Ksa_New := Ksa_Old + Ksa_Change;
  end if;

  -- WTS : Flat Cost Biasing (new version as of 7/18/2012)
  if R_WTS_Pkg.Want_To_Use_Flat_Bias then
    if R_WTS_Pkg.Flat_Bias_Factor < R_WTS_Pkg.Initial_Flat_Bias_Factor then
      -- increment the bias by a very tiny (negligible) amount each pass before
      -- the Iter_To_Start_Biasing, then on the Iter_To_Start_Biasing set the
      -- bias to Initial_Flat_Bias_Factor
      -- (note: the (0.1 * R_WTS_Pkg.Tiny_Bias) is there to deal with any numerical issues)
      if R_WTS_Pkg.Flat_Bias_Factor <
         (R_WTS_Pkg.Tiny_Bias * R_Portable.Float_32(R_WTS_Pkg.Iter_To_Start_Biasing) -
          (0.1 * R_WTS_Pkg.Tiny_Bias)) then
         -- increment the value by a tiny amount, the number of multiples of the
         -- 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)

```
         -- (i.e. how many passes have been done).
         R_WTS_Pkg.New_Flat_Bias_Factor := R_WTS_Pkg.Flat_Bias_Factor +
                                           R_WTS_Pkg.Tiny_Bias;
      else
         R_WTS_Pkg.New_Flat_Bias_Factor := R_WTS_Pkg.Initial_Flat_Bias_Factor;
      end if;
    elsif R_WTS_Pkg.Flat_Bias_Factor < R_WTS_Pkg.Cost_Bias_Factor then
      -- increment the bias by one increment
      R_WTS_Pkg.New_Flat_Bias_Factor := R_WTS_Pkg.Flat_Bias_Factor +
        (R_WTS_Pkg.Cost_Bias_Factor - R_WTS_Pkg.Initial_Flat_Bias_Factor) /
         R_Portable.Float_32(R_WTS_Pkg.Iter_To_Max_Biasing - R_WTS_Pkg.Iter_To_Start_Biasing);
    else
      R_WTS_Pkg.New_Flat_Bias_Factor := R_WTS_Pkg.Cost_Bias_Factor;
    end if;
  end if; -- end WTS

  -- WTS : Been having a lot of trouble with overshoots, so the following
  -- logic is designed to limit the ksa step
  -- don't mess with a ksa that has been set to zero
  if (Ksa_New /= 0.0) then  -- limit ksa rate of change
    Ksa_Change := Ksa_New - Ksa_Old;
    -- first passes after a new rta time are especially bad overshooters
    if Iter_Ctr = 1 and ((R_WTS_Pkg.Old_Rta_Tgt(Fpln_Index) /= Rta_Time) or
       (not Rta_Idx_Data.Preds_Status.Within_Spd_Adj_Tol))
    then
      -- this is a first pass with new rta time, or just a first pass.
      -- limit the ksa change
      -- to a percentage of the computed step
      Ksa_Change := R_WTS_Pkg.First_Pass_Ksa_Step_Percent * Ksa_Change;
    elsif (abs(Rta_Idx_Data.Ate) >= R_WTS_Pkg.Large_Ate_Determiner) then
      -- If we are dealing with a large ate, the code often overshoots, so
      -- limit the ksa change.  This is an elseif because we don't want to
      -- so this limiting we we are doing the first pass one.
      Ksa_Change := R_WTS_Pkg.Ksa_Step_Factor_For_Large_Ate *
                              Ksa_Change;
    elsif (R_WTS_Pkg.Rta_Iter_Counter(Fpln_Index) =
          (R_WTS_Pkg.Iter_To_Start_Biasing)) then
      -- we always seem to overshoot on the first iteration after the
      -- biasing is applied
      Ksa_Change := R_WTS_Pkg.First_Pass_Ksa_Step_Percent * Ksa_Change;
    end if;
    -- 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)

```
    if (abs(Rta_Idx_Data.Ate) <= R_WTS_Pkg.Low_Ksa_Step_Determiner) and
       (abs(Ksa_Change) > R_WTS_Pkg.Max_Ksa_Step_Allowed_Low) then
      if Ksa_Change > 0.0 then
        Ksa_Change := R_WTS_Pkg.Max_Ksa_Step_Allowed_Low;
      else
        Ksa_Change := -1.0 * R_WTS_Pkg.Max_Ksa_Step_Allowed_Low;
      end if;
    elsif (abs(Rta_Idx_Data.Ate) > R_WTS_Pkg.Low_Ksa_Step_Determiner) and
          (abs(Rta_Idx_Data.Ate) < R_WTS_Pkg.High_Ksa_Step_Determiner) then
      Max_Ksa_Step_Local := (R_Portable.Float_32(abs(Rta_Idx_Data.Ate) -
           R_WTS_Pkg.Low_Ksa_Step_Determiner))/
           (R_Portable.Float_32(R_WTS_Pkg.High_Ksa_Step_Determiner -
           R_WTS_Pkg.Low_Ksa_Step_Determiner));
      Max_Ksa_Step_Local := (R_WTS_Pkg.Max_Ksa_Step_Allowed_High -
           R_WTS_Pkg.Max_Ksa_Step_Allowed_Low) * Max_Ksa_Step_Local;
      Max_Ksa_Step_Local := R_WTS_Pkg.Max_Ksa_Step_Allowed_Low +
                   Max_Ksa_Step_Local;
      if (abs(Ksa_Change) > Max_Ksa_Step_Local) then
        if Ksa_Change > 0.0 then
          Ksa_Change := Max_Ksa_Step_Local;
        else
         Ksa_Change := -1.0 * Max_Ksa_Step_Local;
        end if;
      end if;
    elsif (abs(Rta_Idx_Data.Ate) >= R_WTS_Pkg.High_Ksa_Step_Determiner) and
       (abs(Ksa_Change) > R_WTS_Pkg.Max_Ksa_Step_Allowed_High) then
      if Ksa_Change > 0.0 then
        Ksa_Change := R_WTS_Pkg.Max_Ksa_Step_Allowed_High;
      else
        Ksa_Change := -1.0 * R_WTS_Pkg.Max_Ksa_Step_Allowed_High;
      end if;
    end if;
    Ksa_New := Ksa_Old + Ksa_Change;
  end if;
  -- end WTS

  -- compute upper and lower limits for ksa
  Upper_Limit := Rta_Idx_Data.Max_Ksa + Perf_Rta_Lfdata.k6;
  Lower_Limit := Rta_Idx_Data.Min_Ksa - Perf_Rta_Lfdata.k6;

  -- upper limit ksa
  if (Ksa_New >= Upper_Limit) then
```

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

```
    Ksa_New := Upper_Limit;
    Rta_Idx_Data.Preds_Status.At_Max_Speed_Up := True;
    Rta_Idx_Data.Preds_Status.At_Max_Slow_Down := False;

  -- lower limit ksa
  elsif (Ksa_New <= Lower_Limit) then
    Ksa_New := Lower_Limit;
    Rta_Idx_Data.Preds_Status.At_Max_Speed_Up := False;
    Rta_Idx_Data.Preds_Status.At_Max_Slow_Down := True;

  else
    Rta_Idx_Data.Preds_Status.At_Max_Speed_Up := False;
    Rta_Idx_Data.Preds_Status.At_Max_Slow_Down := False;
  end if;

  -- WTS : Split Ksa
  -- To do split ksa correctly, we should compute one cost index and use this for the
  -- entire next pass through Perf.  We will use that data to compute the "RTA"
  -- cost index for the next pass based on the Ksa_New for the next pass.
  -- If we're converged and the counter is rolled back
  -- to 1 then we don't have to compute a new Rta Cost Index.
  if (not Rta_Idx_Data.Preds_Status.Within_Spd_Adj_Tol) then
    Crzalt := Perf_Preds_Lfdata.VGB(VGB_Iftypes.CRZ2L).CMDSpdAlt;
    Perf_Air_Data_Pkg.Air_Static_Data(Crzalt,
                                      Perf_Profile_Lfdata.TOCPerfData.IsaDelta,
                                      Press_Ratio, Theta, Theta_Std, Vel_Sound);
    WOD := Perf_Profile_Lfdata.TOCPerfData.GrossWeight.Data / Press_Ratio;
    Corr_Theta := Theta **
                    Portable_Types_Pkg.Float_32(FMCS_AEDB_CONSTANTS_IFDATA.XTheta);
    Base_CI := Portable_Types_Pkg.Float_32(Ops_CDK_Common_Mgr_Pkg.Cost_Index.Data);
    Corr_Base_CI := Base_CI / (Press_Ratio * Corr_Theta);
    Perf_Aero_Speed_Pkg.EconCrzSpd (WOD, Corr_Base_CI, Base_Mach);
    Tgt_Mach := Base_Mach + Ksa_New/Vel_Sound;
    -- If always using T/C for RTA CI, we need this if block to speed convergence
    -- by using the best guess available for CI
    if (R_WTS_Pkg.Rta_Iter_Counter(Fpln_Index) = 1) then
      CI_Guess := Ops_CDK_Common_Mgr_Pkg.Cost_Index.Data;
    else
      CI_Guess := R_WTS_Pkg.Rta_CI;
    end if;
    R_WTS_Pkg.Rta_CI := Perf_Crz_Pkg.Crz_CIFroMach(CI_Guess => CI_Guess,
                                      Alt_Tgt => Crzalt,
```

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

```
                                  PredWind => Perf_Profile_Lfdata.TOCPerfData.PredWind,
                                  WOD => WOD,
                                  Press_Ratio => Press_Ratio,
                                  Isa_Delta => Perf_Profile_Lfdata.TOCPerfData.IsaDelta,
                                  Corr_Theta => Corr_Theta,
                                  Vel_Sound => Vel_Sound,
                                  FplnTrack => Perf_Profile_Lfdata.TOCPerfData.FplnTrack,
                                  Mach_Tgt => Tgt_Mach);
end if;



  -- Pegasus version:
  -- If a zero cost index is computed, the odds are that it is in fact negative
  -- The econ cruise tables in the PDB only contain positive cost index values.
  if (Perf_Wts_Lfdata.Rta_Ci <= 0) and (not Rta_Idx_Data.Preds_Status.Within_Spd_Adj_Tol) then
    -- Compute the negative cost index corresponding to the Tgt_Mach
    -- This slope is assumed to hold for CI=0 to CI=-Infinity, even though
    -- we know that this is not true.

    -- compute the Mach Number at zero CI
    Perf_Aero_Speed_Pkg.Econcrzspd (Wod, 0.0, Zero_Ci_Mach);

    -- compute the Mach Number at 100 CI
    Corr_Hundred_Ci := 100.0 / (Press_Ratio * Corr_Theta);
    Perf_Aero_Speed_Pkg.Econcrzspd (Wod, Corr_Hundred_Ci, Hundred_Ci_Mach);

    -- compute the cost index vs mach slope
    Ci_Mach_Slope := (Hundred_Ci_Mach - Zero_Ci_Mach) / 100.0;

    -- Protect against unreasonable values and ensure a positive slope
    if (Ci_Mach_Slope > R_WTS_Pkg.Max_Ci_Mach_Slope_Limit) then
       Ci_Mach_Slope := R_WTS_Pkg.Max_Ci_Mach_Slope_Limit;
    elsif (Ci_Mach_Slope < R_WTS_Pkg.Min_Ci_Mach_Slope_Limit) then
       Ci_Mach_Slope := R_WTS_Pkg.Min_Ci_Mach_Slope_Limit;
    end if;

    -- compute the cost index based on linear extrapolation from CI=100 to CI=0
    Perf_Wts_Lfdata.Rta_Ci := Portable_Types_Pkg.Integer_32 ((Tgt_Mach - Zero_Ci_Mach) / Ci_Mach_Slope);
    end if;

-- WTS : set the old_rta_Tgt = current
```

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

```
  R_WTS_Pkg.Old_RTA_Tgt(Fpln_Index) := Rta_Time;
  -- Reset the Found Opt Step Flag to False
  R_WTS_Pkg.Found_Opt_Step := False;
  -- it will be set to true is a step is found on next pass through Perf

end Limit_Ksa;

--------------- L O C A L   P R O C E D U R E  (Do_Message_Logic)  -------------
procedure Do_Message_Logic is
  -- local variables
  Ate         : R_Portable.Integer_32;  -- arrival time error (ETA - RTA)
  Msg_Display : Boolean;              -- current predictions are for the displayed route
  Pred_Alt    : R_Portable.Float_32;    -- predicted altitude
  Step_Alt    : R_Portable.Float_32;    -- step altitude
  Step_Ptr    : R_Portable.Integer_32;  -- planned step climb leg index

begin
  -- find out if these predictions are for the displayed route
  if ((not Perf_Preds_Lfdata.Vtplogic.Haveactfpln) or else
      (Ops_Lateral_Guidance_Buffer_Manager.Provfpln =
       Fmcs_Fp_Guid_Btypes.Noprov)) then
    -- these predictions are for the provisional route (which must be displayed)
    -- or they are for the active route and there is no provisional route
    Msg_Display := True;
  else
    Msg_Display := False;
  end if;

--------------------------- U N A B L E   R T A  -------------------------------
  -- set/clear 'UNABLE RTA' scratchpad message
  -- note the local variable Eta is reused with a different definition
  -- if there is no rcmd tko time, compute Eta from this predictions pass,
  -- if there is a rcmd tko time, compute what the Eta would be if the rcmd tko
  -- time had been used.
  if (Rta_Idx_Data.Rcmd_Takeoff.Valid) then
    if (Rta_Idx_Data.Rcmd_Takeoff.Data.Is_Now) then
      Eta := Perf_Rta_Lfdata.Pred_Gmt_Unmodified.Data + Flight_Time;
    else
      Eta := Rta_Idx_Data.Rcmd_Takeoff.Data.Time + Flight_Time;
    end if;
  else  -- not Rta_Idx_Data.Rcmd_Takeoff.Valid
    Eta := Perf_Preds_Lfdata.Aircraft_State.GMT.Data + Flight_Time;
```

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

```
end if;

-- compute arrival time error (ate) at the rta waypoint
Ate := Eta - Rta_Time;
if (Ate > Half_Day) then
  Ate := Ate - One_Day;
elsif (Ate < -Half_Day) then
  Ate := Ate + One_Day;
end if;

-- determine if aircraft is on time (for message logic)
Rta_Idx_Data.Preds_Status.Within_Msg_Tol :=
  ( ((Rta_Type = Ac_Position_Types.After) and then (Ate >= 0))
    or else
    ((Rta_Type = Ac_Position_Types.Before) and then (Ate <= 0))
    or else
    ((Rta_Type = Ac_Position_Types.AT_Time) and then
     (abs(Ate) <= Perf_Rta_Lfdata.Message_Tol)) );

-- reset the message counter if the rta is on time
if (Rta_Idx_Data.Preds_Status.Within_Msg_Tol) then
  Rta_Idx_Data.Msg_Counter := 0;
end if;

-- reset the 'UNABLE RTA' message Issued flag
if (r_Msg_Flags.Unable_Rta_Msg_Cleared(Fpln_Index)) then
  r_Msg_Flags.Unable_Rta_Msg_Issued(Fpln_Index) := False;
end if;

-- 'UNABLE RTA' message set logic
if ( ((Rta_Idx_Data.Preds_Status.Rta_Happy) or else
      (Rta_Idx_Data.Msg_Counter >= Perf_Rta_Lfdata.Max_Rta_Loops)) and then
     (not Rta_Idx_Data.Preds_Status.Within_Msg_Tol) and then
     (not r_Msg_Flags.Unable_Rta_Msg_Issued(Fpln_Index)) )
then
  -- set flag indicating that the RTA is unable for this route
  r_Msg_Flags.Unable_Rta_Msg_Issued(Fpln_Index) := True;
  r_Msg_Flags.Unable_Rta_Msg_Cleared(Fpln_Index) := False;
  -- if predictions are for the currently displayed route
  if (Msg_Display) then
    -- display 'UNABLE RTA' message in scratchpad
    Fmci_Spad_Manager_Pkg.Display_Message
```

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

```
            (Message_Id => Scratch_Pad_Iftypes.Unable_RTA);
    end if;
  end if;

  -- 'UNABLE RTA' message clear logic
  if (Rta_Idx_Data.Preds_Status.Within_Msg_Tol) then
    -- set flag indicating that the RTA is able for this route
    r_Msg_Flags.Unable_Rta_Msg_Issued(Fpln_Index) := False;
    r_Msg_Flags.Unable_Rta_Msg_Cleared(Fpln_Index) := False;
    -- if predictions are for the currently displayed route
    if (Msg_Display) then
      -- clear 'UNABLE RTA' message from scratchpad
      -- (it doesn't hurt to try to clear it when it's not there)
      Fmci_Spad_Manager_Pkg.Clear_Message
            (Message_Id => Scratch_Pad_Iftypes.Unable_RTA);
    end if;
  end if;

---------------- U N A B L E   F L X X X   A T   R T A   F I X  ----------------
  -- set/clear 'UNABLE FLxxx AT RTA FIX' scratchpad message
  if (Rta_Idx_Data.Preds_Status.Rta_Happy) then
    -- search for a planned (specified) step on a waypoint before the rta wypt
    Step_Ptr := Perf_LGB_Pkg.LGB_Search
                (Starting_Leg_Index => RtaPtr,
                 Search_Thing => Next_Step_Alt_Term,
                 Search_Direction => FMCS_Base_Types.Backward);

    if (Step_Ptr > 0) then
      Step_Alt := Perf_LGB_Lfdata.LGB(Step_Ptr).Fpln_Data.SpAlt1;
      Pred_Alt := Perf_LGB_Lfdata.LGB(RtaPtr).Perf_Data.PrdAlt;

      -- 'UNABLE FLxxx AT RTA FIX' message set logic
      if ((Perf_Preds_Lfdata.DesiredPhase <= FMCS_Base_Types.Cruise) and then
          (not r_Msg_Flags.Unable_Rta_Msg_Issued(Fpln_Index)) and then
          (not r_Msg_Flags.Unable_Flxxx_At_Rta_Fix_Msg_Issued(Fpln_Index))
          and then (Pred_Alt < (Step_Alt - 50.0)))
      then
        -- did not make it to planned step altitude before the rta waypoint
        -- set flag indicating that the FLXXX is unable for this route
        r_Msg_Flags.Unable_Flxxx_At_Rta_Fix_Msg_Issued(Fpln_Index) := True;
        r_Msg_Flags.Unable_Flxxx_At_Rta_Fix_Msg_Cleared(Fpln_Index) := False;
        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)

```
        -- if predictions are for the currently displayed route
        if (Msg_Display) then
          -- display 'UNABLE FLXXX AT RTA FIX' message in scratchpad
          Fmci_Spad_Manager_Pkg.Display_Message
                  (Message_Id => Scratch_Pad_Iftypes.Unable_FLXXX_At_RTA_Fix,
                   Number => Step_Alt);
        end if;
      end if;

      -- 'UNABLE FLxxx AT RTA FIX' message clear logic
      if ( ( (Perf_Preds_Lfdata.DesiredPhase <= FMCS_Base_Types.Cruise) and then
             (Pred_Alt >= (Step_Alt - 50.0)) )
           or else
           ( (Perf_Preds_Lfdata.DesiredPhase > FMCS_Base_Types.Cruise) and then
             (Perf_Crzalt_Lfdata.LastCrzAlt.Data >= (Step_Alt - 50.0)) ) )
      then
        -- set flag indicating that the FLXXX is able for this route
        r_Msg_Flags.Unable_Flxxx_At_Rta_Fix_Msg_Issued(Fpln_Index) := False;
        r_Msg_Flags.Unable_Flxxx_At_Rta_Fix_Msg_Cleared(Fpln_Index) := False;
        -- if predictions are for the currently displayed route
        if (Msg_Display) then
          -- clear 'UNABLE FLXXX AT RTA FIX' message from scratchpad
          -- (it doesn't hurt to try to clear it when it's not there)
          Fmci_Spad_Manager_Pkg.Clear_Message
                  (Message_Id => Scratch_Pad_Iftypes.Unable_FLXXX_At_RTA_Fix);
        end if;
      end if;

    end if;  -- if (Step_Ptr > 0)
  end if;  -- if (Rta_Idx_Data.Preds_Status.Rta_Happy)
end Do_Message_Logic;

--------------- L O C A L   P R O C E D U R E  (Do_Filter_Logic)  --------------
procedure Do_Filter_Logic is
begin
  -- turn on VG rta speed filter once solution is stable and a/c is in crz
  -- turn off VG rta speed filter if speed adjustment changes significantly
  if ((not Perf_Rta_Lfdata.Vg_Rta_Spd_Filter_Inhibit) and then
      (Perf_Rp_Guidprms_Ifdata.Fltphase = Fmcs_Base_Types.Cruise)) then
    if (Rta_Idx_Data.Preds_Status.Within_Spd_Adj_Tol or else
        Rta_Idx_Data.Msg_Counter >= Perf_Rta_Lfdata.Max_Rta_Loops) then
      Perf_Rp_Guidprms_Ifdata.RTA_Spd_Tgt_Filter_On := True;
```

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

```
    elsif (abs(Perf_Rta_Lfdata.Act_Data.Ksa - Ksa_New) > Perf_Rta_Lfdata.k7) then
      Perf_Rp_Guidprms_Ifdata.RTA_Spd_Tgt_Filter_On := False;
    end if;
  else
    Perf_Rp_Guidprms_Ifdata.RTA_Spd_Tgt_Filter_On := False;
  end if;
end Do_Filter_Logic;

--------------------------------------------------------------------------------
-------------------------- M A I N   P R O C E D U R E -------------------------
--------------------------------------------------------------------------------
begin  -- LGB_Seq_Rta_Leg

  -- 7/19/12: Quick fix to not update the indexed Rta_Iter_Counter until the end of preds
  -- since Common_Init has not yet been run to set the Fpln_Index, just save both indexes
  R_WTS_Pkg.Old_Rta_Iter_Counter := R_WTS_Pkg.Rta_Iter_Counter;

  if (Initial_Est) then
    Initial_Estimate;    -- setup data when called from Restart_Check
  else
    Predicted_Sequence;  -- setup data when called from Lgb_Seq_Leg
  end if;

  if (Init_Valid) and (Good_Pointer) then
    Compute_Ate;  -- compute arrival time error and time avail to adj speed
    Compute_Ksa;  -- compute the new speed adjustment factor

    if ((not Perf_Preds_Lfdata.Aircraft_State.Airborne) and then
        (not Ops_Cdk_Perf_Pdb_Mgr_Pkg.Takeoff_Time_Is_Pilot_Entered)) then
      Compute_Rcmd_Tko_Time;  -- convert ksa to recommended takeoff time
    else
      Rta_Idx_Data.Rcmd_Takeoff.Valid := False;
    end if;

    Limit_Ksa;  -- limit the ksa rate of change

    if (not Initial_Est) then
      Do_Message_Logic;  -- set/clear scratchpad messages

      if (Perf_Preds_Lfdata.VTPlogic.Haveactfpln) then
        Do_Filter_Logic;  -- turn the VG speed filter on/off
      end if;
```

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

```
    end if;  -- if (not Initial_Est)

  else  -- not Init_Valid or SCR 9629 bad RTA Pointer
    Ksa_New := 0.0;
  end if;  -- if (Init_Valid)

  -- output data
  Rta_Idx_Data.Old_Ksa := Ksa_Old;
  Rta_Idx_Data.Ksa := Ksa_New;
  Rta_Idx_Data.Preds_Status.Heartbeat := not(Rta_Idx_Data.Preds_Status.Heartbeat);
  Perf_Rta_Lfdata.Idx_Data(Fpln_Index) := Rta_Idx_Data;
  R_WTS_Pkg.Idx_Rta_CI(Fpln_Index) := R_WTS_Pkg.Rta_CI;
  if (Fpln_Index = Act_Prov_Index_Manager.Act_Index) then
    -- save active rta data in a separate location for easy stripcharting
    Perf_Rta_Lfdata.Act_Index := Fpln_Index;
    Perf_Rta_Lfdata.Act_Data := Rta_Idx_Data;
  end if;

  -- 7/19/12, quick fix to not update the indexed Rta_Iter_Counter until the end of preds
  -- Save the new value of R_WTS_Pkg.Rta_Iter_Counter then set the value back
  -- to what it was when this procedure was called.  The saved value will be
  -- used to update the value at the end of predictions.
  R_WTS_Pkg.New_Rta_Iter_Counter(Fpln_Index) := R_WTS_Pkg.Rta_Iter_Counter(Fpln_Index);
  R_WTS_Pkg.Rta_Iter_Counter(Fpln_Index) := R_WTS_Pkg.Old_Rta_Iter_Counter(Fpln_Index);

end LGB_Seq_Rta_Leg;
```

File: CTP_B787_PERF_CRZINITE.ZIP\PERF_LGB_STORE_DATA_SEP.ADA

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

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

```
with Perf_Preds_Lftypes;

with Perf_LGB_Lfdata;
with Perf_Preds_Lfdata;
with Perf_Origin_Dest_Lfdata;
with Perf_Integrators_Lfdata;
with Perf_VTP_Lfdata;
with Perf_Msg_Flags_Lfdata;

with Perf_Pred_Spd_Env_Pkg;
with Perf_SG_Spd_Gen_Pkg;
with Perf_Air_Data_Pkg;
with Perf_ADS_Intent_Pkg;
with Perf_Rta_Lfdata;

use Portable_Types_Pkg;
use Flight_Pln_Leg_Types;
use FMCS_Base_Types;
use FMCS_FP_Guid_BTypes;
use Perf_Preds_Lftypes;

separate (Perf_LGB_Pkg)

procedure LGB_Store_Data is
  --!
  -- ANCHOR:      FMCS_19_21023512
  -- SOURCE:      FMFSDD; FMCS_19_21023000, FMCS_19_21023001, FMCS_19_21023005,
  --                      FMCS_19_21023006
  --            FMFSRD; FMCS_19_20012454, FMCS_19_20006076, FMCS_19_20010030 |
  --| @DESCRIPTION:
  --| This procedure is responsible for storing predicted data from the
  --| integration progress buffer into PERF's copy of the LGB
  --| (PERF_LGB_LFDATA.LGB).
  --|
  --| When flight plan predictions reach (by distance) a waypoint, the required
  --| aircraft state variables (integration progress buffer) at that point are
  --| stored into the corresponding guidance leg in PERF_LGB_LFDATA.LGB.
  --|
  --
  -- SPECIAL_CONSIDERATIONS:
  --
  --   SHARED_DATA_FOR:
```

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

```
--   IOBLK
--
-- REVISION_HISTORY:
--   DATE           SCR #              Programmer              DRCM#
--   01-11-94       1777               D. Groethe              10014
--   Initial version - complete.
--
--   05-18-94       3269               D. Groethe              12763
--   Validate PrdAirSpd for holds and procedure turns.
--
--   08-25-94       4693               D. Jiles                16198
--   Replaced transition_alt with Fl_MSL_Trans_Alt since baro correction
--   should be removed in relation to Fl/MSL alt. instead of speed tranistion
--   altitude.
--
--   04/17/96       8030.08            Karen Hegeman       M777B_FMF_01635
--   Added call to Perf_ADS_Intent_Pkg.Calc_Intermediate_Point to store
--   intermediate intent data when sequencing a hold leg.  When the flight
--   phase is Climb or Cruise, only store the data if sequencing the hold
--   leg will cause a change in the target speed.
--
--   06/25/97       9326.00            Mark Webb           M777B_FMF_05097
--   Added the Max Alt message latch and message index to the invalidate
--   predicted gross weight if block.  This is to prevent the gross weights
--   from being invalidated on legs that are before a Max Alt condition
--   occurs.
--
-- =========================== 787 HISTORY ============================
--
--   12/07/05        519.00            Pat Caulfield
--   Added output of Prddataseq, Predisadev, Predfuelwgt, Pred_Wind,
--   and Preds_Stable.
--
--   05/02/06        788.00            Pat Caulfield
--   Added output of predicted flight phase.
--
--   03/29/07       2676.00            Pat Caulfield
--   Added output of predicted flight path angle (FPA).
--!


  -- 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)

```

    -- DESCRIPTION Tolerance to be used for comparing CAS speeds
    CAS_Tolerance : constant Portable_Types_Pkg.Float_32 := 1.0;

    -- DESCRIPTION Local copy of Perf_Preds_Lfdata.NavPtr used for
    -- efficiency to reduce global memory access.
    NavPtr     : Portable_Types_Pkg.Integer_32 := Perf_Preds_Lfdata.NavPtr;

    -- DESCRIPTION Predicted altitude from the integration progress buffer;
    -- used for efficiency to reduce global memory access.
    Predicted_Alt : Portable_Types_Pkg.Float_32 :=
      Perf_Integrators_Lfdata.IntProgBuf.Hprog;

    -- DESCRIPTION Predicted gross weight from the integration progress buffer;
    -- used for efficiency to reduce global memory access.
    Predicted_GWT : Portable_Types_Pkg.Float_32 :=
      Perf_Integrators_Lfdata.IntProgBuf.GWprog;

    -- DESCRIPTION Predicted flight phase; used for efficiency
    -- to reduce global memory access.
    Predicted_Flight_Phase : FMCS_Base_Types.Flight_Phase_Type :=
      Perf_Preds_Lfdata.DesiredPhase;

    -- DESCRIPTION Transition Altitude
    Trans_Alt : Portable_Types_Pkg.Float_32;

    -- DESCRIPTION PathTerm for the leg - used for efficiency purposes
    LegTerm : FMCS_Base_Types.PathType;

    -- DESCRIPTION Indicates use of flaps extended for best hold speed.
    Use_Flaps : Boolean;

    -- DESCRIPTION Minimum Mach - used in hold speed computation
    MinMach : Portable_Types_Pkg.Float_32;

    -- DESCRIPTION Maximum Mach - used in hold speed computation
    MaxMach : Portable_Types_Pkg.Float_32;

    -- DESCRIPTION Not needed - output from Pred Spd Env call
    Mbfnglo : Portable_Types_Pkg.Float_32;

    -- DESCRIPTION: Best Hold Speed
```

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

```
    Hold_Spd : Portable_Types_Pkg.Float_32;

    -- DESCRIPTION: T=Hold_Spd has been calculated and is being used in place of CAS2.
    Using_Hold_Spd : boolean := false;

    Satnorm : Portable_Types_Pkg.Float_32;
    -- DESCRIPTION: NORMALIZED STAT AIR TEMPERATURE

    Adat_Delta : Portable_Types_Pkg.Float_32;
    -- DESCRIPTION: ATMOSPHERIC PRESSURE RATIO

    Satnormstd : Portable_Types_Pkg.Float_32;
    -- DESCRIPTION: NORMALIZED STAT AIR TEMPERATURE

    Velsound : Portable_Types_Pkg.Float_32;
    -- DESCRIPTION: VELOCITY OF SOUND

    HX_PI_Leg : boolean;
    -- DESCRIPTION: LEG TYPE IS A HOLD OR PROCEDURE TURN

    TempCAS : Portable_Types_Pkg.Float_32;
    -- DESCRIPTION: Temporary variable to hold the conversion of a Mach speed to a CAS
    -- for the calculation of the hold speed

begin  -- LGB_Store_Data

  -- DETERMINE WHICH TRANSITION ALTITUDE TO USE FOR WPT PRED ALT OUTPUT
  if (Predicted_Flight_Phase <= Cruise) then
      Trans_Alt := Perf_Origin_Dest_Lfdata.Origin.FL_MSL_Trans_Alt;
  else
      Trans_Alt := Perf_Origin_Dest_Lfdata.Destination.FL_MSL_Trans_Alt;
  end if;

  -- SET UP THE NODODESPRED LEG FIELD FOR CDK TO NOT DISPLAY PREDICTIONS
  -- FOR A DEFAULT DESCENT PATH.

  if (Predicted_Flight_Phase = Descent) and (Perf_LGB_Lfdata.Last_Constraint_Index = 0) then
      Perf_LGB_Lfdata.LGB(NavPtr).Perf_Data.NoDoDesPred := True;
  else
      Perf_LGB_Lfdata.LGB(NavPtr).Perf_Data.NoDoDesPred := False;
  end if;

```

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

```
-- OVERWRITE THE FLIGHT MODE FIELD FOR THE LEG IF NOT AN ALT CONSTRAINT

if not Perf_LGB_Lfdata.LGB(NavPtr).Fpln_Data.HavePerf then
    ~~if (Predicted_Flight_Phase < Cruise) then~~
        ~~Perf_LGB_Lfdata.LGB(NavPtr).Perf_Data.Fltmode := GBClimb;~~
    ~~elsif (Predicted_Flight_Phase = Cruise) then~~
        ~~Perf_LGB_Lfdata.LGB(NavPtr).Perf_Data.Fltmode := GBCruise;~~
    ~~else -- DESCENT~~
        ~~Perf_LGB_Lfdata.LGB(NavPtr).Perf_Data.Fltmode := GBDescent;~~
    -- If there is an untagged FltMode on a speed constraint, this
    -- means that FPLN was unable to tag it and PERF will have to
    if Perf_LGB_Lfdata.LGB(NavPtr).Perf_Data.SpcSpdVal and then
       Perf_LGB_Lfdata.LGB(NavPtr).Perf_Data.Fltmode = NoGBMode then
        if (Predicted_Flight_Phase < Cruise) then
            Perf_LGB_Lfdata.LGB(NavPtr).Perf_Data.Fltmode := GBClimb;
        else -- DESCENT
            Perf_LGB_Lfdata.LGB(NavPtr).Perf_Data.Fltmode := GBDescent;
        end if;
    -- Once a speed constraint has been tagged for FltMode, it cannot
    -- change. Mini-legs that do not contain speed or altitude constraints
    -- however can be tagged accordingly
    elsif not Perf_LGB_Lfdata.LGB(NavPtr).Perf_Data.SpcSpdVal then
        if (Predicted_Flight_Phase < Cruise) then
            Perf_LGB_Lfdata.LGB(NavPtr).Perf_Data.Fltmode := GBClimb;
        elsif (Predicted_Flight_Phase = Cruise) then
            Perf_LGB_Lfdata.LGB(NavPtr).Perf_Data.Fltmode := GBCruise;
        else -- DESCENT
            Perf_LGB_Lfdata.LGB(NavPtr).Perf_Data.Fltmode := GBDescent;
        end if;
    end if;
end if;

-- WRITE OUT PREDICTED ETA FOR THE LEG

Perf_LGB_Lfdata.LGB(NavPtr).Perf_Data.PrdETAToFix :=
  Portable_Types_Pkg.Integer_32(Perf_Integrators_Lfdata.IntProgBuf.TProg) +
  Perf_Preds_Lfdata.Aircraft_State.GMT.Data;
Perf_LGB_Lfdata.LGB(NavPtr).Perf_Data.PrdETAFixVal :=
  Perf_Preds_Lfdata.Aircraft_State.GMT.Valid;


-- WRITE OUT PREDICTED GROSS WEIGHT FOR THE LEG
```

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

```

  -- INVALIDATE THE PREDICTED GROSS WEIGHT FOR THE LEG WHEN CRUISE ALTITUDE
  -- EXCEEDS THE MAXIMUM ALTITUDE AND AIRCRAFT FLIGHT PHASE IS BEFORE DESCENT.

  if ((Perf_Preds_Lfdata.Fltphase < Descent) and then
      (Perf_Msg_Flags_Lfdata.AboveMaxAlt or else
       ((Perf_Msg_Flags_Lfdata.Max_Alt_Msg_Leg_Indx > 0) and then
        (Perf_Integrators_Lfdata.Intprogbuf.Xprog <
         Perf_Lgb_Lfdata.Lgb(Perf_Msg_Flags_Lfdata.Max_Alt_Msg_Leg_Indx).Common_Data.Fixdistodest)))) then
      Perf_LGB_Lfdata.LGB(NavPtr).Perf_Data.PrdGwtFixVal := False;
  else
      Perf_LGB_Lfdata.LGB(NavPtr).Perf_Data.PrdGwtFixVal := True;
  end if;

  Perf_LGB_Lfdata.LGB(NavPtr).Perf_Data.PrdGwtToFix := Predicted_GWT;

  -- WRITE OUT PREDICTED ALTITUDE FOR THE LEG

  Perf_LGB_Lfdata.LGB(NavPtr).Perf_Data.PrdAltVal := True;

  if (Predicted_Alt < Trans_Alt) then
      Perf_LGB_Lfdata.LGB(NavPtr).Perf_Data.PrdAlt :=
        Predicted_Alt - Perf_Preds_Lfdata.Aircraft_State.Barocorr;
  else
      Perf_LGB_Lfdata.LGB(NavPtr).Perf_Data.PrdAlt := Predicted_Alt;
  end if;

  -- WRITE OUT THE PREDICTED AIR SPEED AND GROUND SPEED FOR THE LEG

  LegTerm := Perf_LGB_Lfdata.LGB(NavPtr).Fpln_Data.PathTerm;
```
~~HX_PI_Leg := (LegTerm = HA) or (LegTerm = HF) or (LegTerm = HM) or (LegTerm = PI);~~
```
  HX_PI_Leg := Perf_Lgb_Pkg.Is_Hold_Leg(NavPtr) or (LegTerm = PI);

  if HX_PI_Leg then -- IF HOLD OR PROCEDURE TURN

      -- COMPUTE BEST HOLD SPEED TO OUTPUT IN THE PREDICTED AIR SPEED FIELD

      -- Use flaps is true if the flaps are out for real and we're either in descent
      -- or predicting the active leg.
      Use_Flaps := Perf_VTP_Lfdata.Act_VTP_Flaps and then ((Perf_Preds_Lfdata.Next_Waypoint_NavPtr =
                   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)

```
      -- COMPUTE THE STATIC AIR DATA FOR THE CURRENT PREDICTIONS STATE
      Perf_Air_Data_Pkg.Air_Static_Data (Pressalt => Perf_Integrators_Lfdata.IntProgBuf.Hprog,
                                         Isadelta => Perf_Preds_Lfdata.Isadelta,
                                         Pressnorm => Adat_Delta,
                                         Satnorm => Satnorm,
                                         Satnormstd => Satnormstd,
                                         Velsound => Velsound);

      -- If there is a specified speed for the current leg:
      if (Perf_Lgb_Lfdata.Lgb(Navptr).Perf_Data.Spcspdval) and then
         (Perf_Lgb_Lfdata.Lgb(Navptr).Fpln_Data.Spcspdpe) and then
         (Perf_Lgb_Lfdata.Lgb(Navptr).Fpln_Data.Hold_Src = Flight_Pln_Leg_Types.PILOT_ENTERED) then

          -- The hold CAS is the specified speed
          Hold_Spd := Perf_Lgb_Lfdata.Lgb(Navptr).Perf_Data.Spcspd;

          -- Limit the hold speed to the profile speed for the active flight mode
          -- If the profile speed is in CAS, the speed can be compared directly to the hold speed
          if (Perf_Preds_Lfdata.VGB(Perf_Preds_Lfdata.VGBPtr).CmdCASVal) and then
             (Hold_Spd > Perf_Preds_Lfdata.VGB(Perf_Preds_Lfdata.VGBPtr).CmdCAS) then

             Hold_Spd := Perf_Preds_Lfdata.VGB(Perf_Preds_Lfdata.VGBPtr).CmdCAS;

          elsif (Perf_Preds_Lfdata.VGB(Perf_Preds_Lfdata.VGBPtr).CmdMachVal) then

             -- If the profile speed is in Mach, convert the Mach speed to a CAS before
             -- making the comparison
             TempCas := Perf_Air_Data_Pkg.Air_Mcas
                          (Mach      => Perf_Preds_Lfdata.VGB(Perf_Preds_Lfdata.VGBPtr).CmdMach,
                           Pressnorm => Adat_Delta);

             -- Limit the hold speed to the profile speed
             if (Hold_Spd > TempCas) then
                Hold_Spd := TempCas;
             end if;
          end if;
      else

         -- There is no specified speed on the hold, so calculate the best hold
         -- speed
      Perf_Pred_Spd_Env_Pkg.Pred_Spd_Env
        (Fltphase    => Predicted_Flight_Phase,
```

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

```
         Despathgen  => False,
         Flapsexist  => Use_Flaps,
         Grossweight => Predicted_GWT,
         Pressalt    => Predicted_Alt,
         Minmach     => MinMach,
         Maxmach     => MaxMach,
         Mbfnglo     => Mbfnglo);

      Perf_SG_Spd_Gen_Pkg.SG_Hold_Spd
        (Flaps_Exist => Use_Flaps,
         Press_Alt   => Predicted_Alt,
         Gw          => Predicted_GWT,
         Min_Mach    => MinMach,
         Max_Mach    => MaxMach,
         Hold_CAS    => Hold_Spd);

         -- limit the BHS to a NavDB defined speed on the hold
         if (Perf_Lgb_Lfdata.Lgb(Navptr).Perf_Data.Spcspdval) and then
            (Perf_Lgb_Lfdata.Lgb(Navptr).Fpln_Data.Spcspdpe) and then
            (Hold_Spd > Perf_Lgb_Lfdata.Lgb(Navptr).Perf_Data.Spcspd) then
            Hold_Spd := Perf_Lgb_Lfdata.Lgb(Navptr).Perf_Data.Spcspd;
         end if;
      end if;
      if Predicted_Flight_Phase < Descent or else
         Perf_Integrators_Lfdata.IntProgBuf.CAS2 > Hold_Spd then

         Perf_LGB_Lfdata.LGB(NavPtr).Perf_Data.PrdAirSpd.Value := Hold_Spd;
         Perf_LGB_Lfdata.LGB(NavPtr).Perf_Data.PrdAirSpd.Speed_Type := CAS;
         Using_Hold_Spd := true;

~~         -- COMPUTE THE STATIC AIR DATA FOR THE CURRENT PREDICTIONS STATE~~
~~         Perf_Air_Data_Pkg.Air_Static_Data (Pressalt => Perf_Integrators_Lfdata.IntProgBuf.Hprog,~~
~~                                            Isadelta => Perf_Preds_Lfdata.Isadelta,~~
~~                                            Pressnorm => Adat_Delta,~~
~~                                            Satnorm => Satnorm,~~
~~                                            Satnormstd => Satnormstd,~~
~~                                            Velsound => Velsound);~~

         -- COMPUTE THE PREDICTED TAS FOR THE LEG BASED ON THE BEST HOLD SPEED
         -- USING THE MACH EQUIVALENT OF THE CAS SPEED TIMES THE SPEED OF SOUND
         Perf_LGB_Lfdata.LGB(NavPtr).Perf_Data.Predicted_TAS := Velsound *
             Perf_Air_Data_Pkg.Air_Casm (Hold_Spd, Adat_Delta);
```

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

```
          Perf_LGB_Lfdata.LGB(NavPtr).Perf_Data.PrdSpdVal := True;
       end if;

       -- INVALIDATE PREDICTED GROUNDSPEED FOR HOLD OR PROCEDURE TURN
       Perf_LGB_Lfdata.LGB(NavPtr).Perf_Data.PrdGndSpdVal := False;

       if Using_Hold_Spd and then ((Predicted_FLight_Phase = Descent) or else
           (not Perf_LGB_Lfdata.LGB(NavPtr).Perf_Data.SpcSpdVal and then
            (abs(Perf_LGB_Lfdata.LGB(NavPtr).Perf_Data.PrdAirSpd.Value -
                 Perf_Integrators_Lfdata.IntProgBuf.CAS2) > CAS_Tolerance))) then
         Perf_ADS_Intent_Pkg.Calc_Intermediate_Point;
       end if;
   else
       -- NOT A HOLD OR PROCEDURE TURN - WRITE OUT THE PREDICTED GROUND SPEED FOR THE LEG.
       Perf_LGB_Lfdata.LGB(NavPtr).Perf_Data.PrdGndSpdVal := True;
       Perf_LGB_Lfdata.LGB(NavPtr).Perf_Data.PrdGndSpd := Perf_Integrators_Lfdata.IntProgBuf.GndSpd2;
   end if;

   -- SELECT CAS OR MACH TO WRITE OUT AS PREDICTED AIRSPEED FOR THE LEG
   if not Using_Hold_Spd then
      if ((Perf_Preds_Lfdata.TgtSpdRec.TgtSpdTag = MachOnly) or
         ((Perf_Preds_Lfdata.TgtSpdRec.TgtSpdTag = CasMach) and
          (Predicted_Alt >= Perf_Preds_Lfdata.TgtSpdRec.CMXalt))) and not
         HX_PI_Leg -- don't output mach for these legs.
      then -- MACH
         Perf_LGB_Lfdata.LGB(NavPtr).Perf_Data.PrdAirSpd.Speed_Type := Mach;
         Perf_LGB_Lfdata.LGB(NavPtr).Perf_Data.PrdAirSpd.Value := Perf_Integrators_Lfdata.IntProgBuf.Mach2;
      else -- CAS
         Perf_LGB_Lfdata.LGB(NavPtr).Perf_Data.PrdAirSpd.Speed_Type := CAS;
         Perf_LGB_Lfdata.LGB(NavPtr).Perf_Data.PrdAirSpd.Value := Perf_Integrators_Lfdata.IntProgBuf.CAS2;
      end if;

      -- WRITE OUT THE PREDICTED TAS FOR THE LEG.
      Perf_LGB_Lfdata.LGB(NavPtr).Perf_Data.Predicted_TAS := Perf_Integrators_Lfdata.IntProgBuf.Tas2;
      Perf_LGB_Lfdata.LGB(NavPtr).Perf_Data.PrdSpdVal := True;
   end if; -- IF NOT USING HOLD SPEED

   -- Copy the guidance header's prediction data sequence counter into the leg to validate
   -- the data.  Core FP tells that predictions are valid when the header and the leg/segment
   -- values are the same, and can invalidate them by incrementing the header version.

   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)

```

  -- 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

```
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)

```
      for Spec ("vgb_iftypes") use "VGB_IFTYPES_.ADA";
      for Spec ("updatewind_iftypes") use "UPDATEWIND_IFTYPES_.ADA";
      for Spec ("transition_tpkg") use "TRANSITION_TPKG_.ADA";
      for Spec ("step_climb_ifdata") use "STEP_CLIMB_IFDATA_.ADA";
      for Spec ("simsoft_interface_dpkg") use "SIMSOFT_INTERFACE_DPKG_.ADA";
      for Spec ("set_package") use "SET_PACKAGE_.ADA";
      for Body ("set_package") use "SET_PACKAGE.ADA";
      for Spec ("scratch_pad_iftypes") use "SCRATCH_PAD_IFTYPES_.ADA";
      for Spec ("radio_tuning_iftypes") use "RADIO_TUNING_IFTYPES_.ADA";
      for Spec ("perf_tko_ref_spd_ifdata") use "PERF_TKO_REF_SPD_IFDATA_.ADA";
      for Spec ("perf_st_spdtape_ifdata") use "PERF_ST_SPDTAPE_IFDATA_.ADA";
      for Spec ("perf_sg_spd_gen_ifdata") use "PERF_SG_SPD_GEN_IFDATA_.ADA";
      for Spec ("perf_rt_iftypes") use "PERF_RT_IFTYPES_.ADA";
      for Spec ("perf_rp_guidprms_lfdata") use "PERF_RP_GUIDPRMS_LFDATA_.ADA";
      for Spec ("perf_rp_guidprms_ifdata") use "PERF_RP_GUIDPRMS_IFDATA_.ADA";
      for Spec ("perf_preds_btypes") use "PERF_PREDS_BTYPES_.ADA";
      for Spec ("perf_offpath_descent_iftypes") use "PERF_OFFPATH_DESCENT_IFTYPES_.ADA";
      for Spec ("perf_offpath_descent_ifdata") use "PERF_OFFPATH_DESCENT_IFDATA_.ADA";
      for Spec ("perf_max_opt_iftypes") use "PERF_MAX_OPT_IFTYPES_.ADA";
      for Spec ("perf_max_opt_ifdata") use "PERF_MAX_OPT_IFDATA_.ADA";
      for Spec ("perf_hold_time_iftypes") use "PERF_HOLD_TIME_IFTYPES_.ADA";
      for Spec ("perf_hold_time_ifdata") use "PERF_HOLD_TIME_IFDATA_.ADA";
      for Spec ("perf_change_flags_iftypes") use "PERF_CHANGE_FLAGS_IFTYPES_.ADA";
      for Spec ("perf_act_spd_env_lfdata") use "PERF_ACT_SPD_ENV_LFDATA_.ADA";
      for Spec ("perf_act_spd_env_ifdata") use "PERF_ACT_SPD_ENV_IFDATA_.ADA";
      for Spec ("pdb_types_pkg") use "PDB_TYPES_PKG_.ADA";
      for Spec ("pdb_table_interp_3_pkg") use "PDB_TABLE_INTERP_3_PKG_.ADA";
      for Body ("pdb_table_interp_3_pkg") use "PDB_TABLE_INTERP_3_PKG.ADA";
      for Spec ("pdb_table_interp_2_pkg") use "PDB_TABLE_INTERP_2_PKG_.ADA";
      for Body ("pdb_table_interp_2_pkg") use "PDB_TABLE_INTERP_2_PKG.ADA";
      for Spec ("pdb_table_interp_1_pkg") use "PDB_TABLE_INTERP_1_PKG_.ADA";
      for Body ("pdb_table_interp_1_pkg") use "PDB_TABLE_INTERP_1_PKG.ADA";
      for Spec ("pdb_selection_pkg") use "PDB_SELECTION_PKG_.ADA";
      for Body ("pdb_selection_pkg") use "PDB_SELECTION_PKG.ADA";
      for Spec ("pdb_parameter_block_data_pkg") use "PDB_PARAMETER_BLOCK_DATA_PKG_.ADA";
      for Spec ("pdb_initialization_pkg") use "PDB_INITIALIZATION_PKG_.ADA";
      for Body ("pdb_initialization_pkg") use "PDB_INITIALIZATION_PKG.ADA";
      for Spec ("pdb_data_pkg") use "PDB_DATA_PKG_.ADA";
      for Spec ("pdb_data_access_pkg") use "PDB_DATA_ACCESS_PKG_.ADA";
      for Body ("pdb_data_access_pkg") use "PDB_DATA_ACCESS_PKG.ADA";
      for Spec ("pdb_application_types_pkg") use "PDB_APPLICATION_TYPES_PKG_.ADA";
      for Spec ("pdb_address_translation_pkg") use "PDB_ADDRESS_TRANSLATION_PKG_.ADA";
```

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

```
      for Body ("pdb_address_translation_pkg") use "PDB_ADDRESS_TRANSLATION_PKG.ADA";
      for Spec ("path_term_types") use "PATH_TERM_TYPES_.ADA";
      for Body ("ops_timer_pkg") use "OPS_TIMER_PKG.ADA";
      for Spec ("fmcs_partnumber_pkg") use "OPS_PARTNUMBER_PKG_.ADA";
      for Spec ("ops_opc_ami_ifdata") use "OPS_OPC_AMI_IFDATA_.ADA";
      for Spec ("ops_lateral_guidbuff_mgr_iftypes") use "OPS_LATERAL_GUIDBUFF_MGR_IFTYPES_.ADA";
      for Spec ("ops_hs_buffer_gnrc") use "OPS_HS_BUFFER_GNRC_.ADA";
      for Body ("ops_hs_buffer_gnrc") use "OPS_HS_BUFFER_GNRC.ADA";
      for Spec ("ops_db_const_pkg") use "OPS_DB_CONST_PKG_.ADA";
      for Body ("ops_db_const_pkg") use "OPS_DB_CONST_PKG.ADA";
      for Spec ("ops_date_compare") use "OPS_DATE_COMP_PKG_.ADA";
      for Body ("ops_date_compare") use "OPS_DATE_COMP_PKG.ADA";
      for Spec ("ops_data_retained_pkg") use "OPS_DATA_RETAINED_PKG_.ADA";
      for Body ("ops_data_retained_pkg") use "OPS_DATA_RETAINED_PKG.ADA";
      for Spec ("ops_cmn_utilities_pkg") use "OPS_CMN_UTILITIES_PKG_.ADA";
      for Body ("ops_cmn_utilities_pkg") use "OPS_CMN_UTILITIES_PKG.ADA";
      for Spec ("ops_cdl_buffer_mgr_iftypes") use "OPS_CDL_BUFFER_MGR_IFTYPES_.ADA";
      for Spec ("ops_cdk_altn_object_iftypes") use "OPS_CDK_ALTN_OBJECT_IFTYPES_.ADA";
      for Spec ("ops_cdk_altn_init_constants_pkg") use "OPS_CDK_ALTN_INIT_CONSTANTS_PKG_.ADA";
      for Spec ("ops_aedb_ifdata") use "OPS_AEDB_IFDATA_.ADA";
      for Body ("nav_wind_pkg.relative_wind") use "NAV_WIND_PKG_RELATIVE_WIND.ADA";
      for Body ("nav_wind_pkg.measure_wind") use "NAV_WIND_PKG_MEASURE_WIND.ADA";
      for Body ("nav_wind_pkg.filter_non_gusty_wind") use "NAV_WIND_PKG_FILTER_NON_GUSTY_WIND.ADA";
      for Body ("nav_wind_pkg.filter_gusty_wind") use "NAV_WIND_PKG_FILTER_GUSTY_WIND.ADA";
      for Spec ("nav_wind_pkg") use "NAV_WIND_PKG_.ADA";
      for Body ("nav_wind_pkg") use "NAV_WIND_PKG.ADA";
      for Body ("wgs84_geoid.elevation") use "NAV_WGS84_GEOID_ELEVATION.ADA";
      for Spec ("wgs84_geoid") use "NAV_WGS84_GEOID_.ADA";
      for Body ("wgs84_geoid") use "NAV_WGS84_GEOID.ADA";
      for Spec ("navigation_types") use "NAV_TYPES_.ADA";
      for Body ("navigation_types") use "NAV_TYPES.ADA";
      for Spec ("third_order_vector") use "NAV_THIRD_ORDER_VECTOR_.ADA";
      for Body ("third_order_vector") use "NAV_THIRD_ORDER_VECTOR.ADA";
      for Body ("nav_sensor_computations.velocity_vector") use "NAV_SENSOR_COMPUTATIONS_VELOCITY_VECTO.ADA";
      for Body ("nav_sensor_computations.position_vector") use "NAV_SENSOR_COMPUTATIONS_POSITION_VECTO.ADA";
      for Body ("nav_sensor_computations.ecef_xyz_to_geodetic_velocity") use "NAV_SENSOR_COMPUTATIONS_GEODETIC_VELOC.ADA";
      for Body ("nav_sensor_computations.ecef_xyz_to_geodetic_position") use "NAV_SENSOR_COMPUTATIONS_GEODETIC_POSIT.ADA";
      for Spec ("nav_sensor_computations") use "NAV_SENSOR_COMPUTATIONS_.ADA";
      for Body ("nav_sensor_computations") use "NAV_SENSOR_COMPUTATIONS.ADA";
      for Spec ("sem_types") use "NAV_SEM_TYPES_.ADA";
      for Spec ("nav_hot_spare_iftypes") use "NAV_HOT_SPARE_IFTYPES_.ADA";
      for Spec ("nav_filter_blocks_pkg") use "NAV_FILTER_BLOCKS_.ADA";
```

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

```
      for Body ("nav_filter_blocks_pkg") use "NAV_FILTER_BLOCKS.ADA";
      for Spec ("navigation_constants") use "NAV_CONSTANTS_.ADA";
      for Body ("navigation_utilities.rotate_bias") use "NAVIGATION_UTILITIES_ROTATE_BIAS.ADA";
      for Spec ("navigation_utilities") use "NAVIGATION_UTILITIES_.ADA";
      for Body ("navigation_utilities") use "NAVIGATION_UTILITIES.ADA";
      for Spec ("nam_waypoint_ifdata") use "NAM_WAYPOINT_IFDATA_.ADA";
      for Spec ("nam_userrec_iftypes") use "NAM_USERREC_IFTYPES_.ADA";
      for Spec ("nam_runway_ifdata") use "NAM_RUNWAY_IFDATA_.ADA";
      for Spec ("nam_navaid_iftypes") use "NAM_NAVAID_IFTYPES_.ADA";
      for Spec ("nam_navaid_ifdata") use "NAM_NAVAID_IFDATA_.ADA";
      for Spec ("nam_image_types") use "NAM_IMAGE_TYPES_.ADA";
      for Spec ("nam_iftypes") use "NAM_IFTYPES_.ADA";
      for Spec ("nam_holding_pattern_tpkg") use "NAM_HOLDING_PATTERN_TPKG_.ADA";
      for Spec ("nam_efis_iftypes") use "NAM_EFIS_IFTYPES_.ADA";
      for Spec ("nam_efis_ifdata") use "NAM_EFIS_IFDATA_.ADA";
      for Body ("nam_efis_ifdata") use "NAM_EFIS_IFDATA.ADA";
      for Spec ("nam_corte_ifdata") use "NAM_CORTE_IFDATA_.ADA";
      for Spec ("nam_base_types") use "NAM_BASE_TYPES_.ADA";
      for Spec ("measurement_valid_types") use "MEASUREMENT_VALID_TYPES_.ADA";
      for Spec ("math_rad_pkg") use "MATH_RAD_PKG.ADA";
      for Spec ("math_rad_64_pkg") use "MATH_RAD_64_PKG.ADA";
      for Spec ("math_primitives_pkg") use "MATH_PRIMITIVES_PKG.ADA";
      for Spec ("math_primitives_64_pkg") use "MATH_PRIMITIVES_64_PKG.ADA";
      for Spec ("math_pkg") use "MATH_PKG_.ADA";
      for Body ("math_pkg") use "MATH_PKG.ADA";
      for Spec ("lg_slow_out_ifdata_pkg") use "LG_SLOW_OUT_IFDATA_PKG_.ADA";
      for Spec ("lg_segment_manager_pkg") use "LG_SEGMENT_MANAGER_PKG_.ADA";
      for Body ("lg_segment_manager_pkg") use "LG_SEGMENT_MANAGER_PKG.ADA";
      for Spec ("lg_iftypes") use "LG_IFTYPES_.ADA";
      for Spec ("lg_group_id_pkg") use "LG_GROUP_ID_PKG_.ADA";
      for Spec ("lg_fast_out_ifdata_pkg") use "LG_FAST_OUT_IFDATA_PKG_.ADA";
      for Spec ("legseg_types") use "LEGSEG_TYPES_.ADA";
      for Spec ("lateral_segment_tpkg") use "LATERAL_SEGMENT_TPKG_.ADA";
      for Spec ("idx_profile_ifdata") use "IDX_PROFILE_IFDATA_.ADA";
      for Spec ("holdupdate_iftypes") use "HOLDUPDATE_IFTYPES_.ADA";
      for Spec ("fprequest_iftypes") use "FPREQUEST_IFTYPES_.ADA";
      for Spec ("fm_sync_imm_types_pkg") use "FM_SYNC_IMM_TYPES_PKG_.ADA";
      for Spec ("fm_navigation_types") use "FM_NAVIGATION_TYPES_.ADA";
      for Spec ("fm_hotspare_iftypes") use "FM_HOTSPARE_IFTYPES_.ADA";
      for Spec ("fm_hotspare_ifdata") use "FM_HOTSPARE_IFDATA_.ADA";
      for Spec ("vg_vert_dev_ifdata") use "FMF_VG_VERT_DEV_IFDATA_.ADA";
      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_btypes") 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)

```
      for Spec ("flx_waypoint_pkg") use "FLX_WAYPOINT_PKG_.ADA";
      for Body ("flx_waypoint_pkg") use "FLX_WAYPOINT_PKG.ADA";
      for Spec ("flx_vhf_navaid_tpkg") use "FLX_VHF_NAVAID_TPKG_.ADA";
      for Spec ("flx_vhf_navaid_pkg") use "FLX_VHF_NAVAID_PKG_.ADA";
      for Body ("flx_vhf_navaid_pkg") use "FLX_VHF_NAVAID_PKG.ADA";
      for Spec ("flx_utils_pkg") use "FLX_UTILS_PKG_.ADA";
      for Body ("flx_utils_pkg") use "FLX_UTILS_PKG.ADA";
      for Spec ("flx_tap_legs_tpkg") use "FLX_TAP_LEGS_TPKG_.ADA";
      for Spec ("flx_tap_legs_pkg") use "FLX_TAP_LEGS_PKG_.ADA";
      for Body ("flx_tap_legs_pkg") use "FLX_TAP_LEGS_PKG.ADA";
      for Spec ("flx_star_tpkg") use "FLX_STAR_TPKG_.ADA";
      for Spec ("flx_star_pkg") use "FLX_STAR_PKG_.ADA";
      for Body ("flx_star_pkg") use "FLX_STAR_PKG.ADA";
      for Spec ("flx_sid_tpkg") use "FLX_SID_TPKG_.ADA";
      for Spec ("flx_sid_pkg") use "FLX_SID_PKG_.ADA";
      for Body ("flx_sid_pkg") use "FLX_SID_PKG.ADA";
      for Spec ("flx_runway_tpkg") use "FLX_RUNWAY_TPKG_.ADA";
      for Spec ("flx_runway_pkg") use "FLX_RUNWAY_PKG_.ADA";
      for Body ("flx_runway_pkg") use "FLX_RUNWAY_PKG.ADA";
      for Spec ("flx_ndrb_pkg") use "FLX_NDRB_PKG_.ADA";
      for Body ("flx_ndrb_pkg") use "FLX_NDRB_PKG.ADA";
      for Spec ("flx_nav_airport_pkg") use "FLX_NAV_AIRPORT_PKG_.ADA";
      for Body ("flx_nav_airport_pkg") use "FLX_NAV_AIRPORT_PKG.ADA";
      for Body ("flx_navigation_pkg.get_closest_two_navaids_by_freq") use "FLX_NAVIGATION_PKG_CLO_2_NAV_BY_FRQ.ADA";
      for Body ("flx_navigation_pkg.build_nav_list") use "FLX_NAVIGATION_PKG_BUILD_NAV_LIST.ADA";
      for Body ("flx_navigation_pkg.build_nav_tuning_list") use "FLX_NAVIGATION_PKG_BLD_NAV_TUNE_LST.ADA";
      for Spec ("flx_navigation_pkg") use "FLX_NAVIGATION_PKG_.ADA";
      for Body ("flx_navigation_pkg") use "FLX_NAVIGATION_PKG.ADA";
      for Spec ("flx_navigation_ifdata") use "FLX_NAVIGATION_IFDATA_.ADA";
      for Body ("flx_navigation_ifdata") use "FLX_NAVIGATION_IFDATA.ADA";
      for Spec ("flx_localizer_tpkg") use "FLX_LOCALIZER_TPKG_.ADA";
      for Spec ("flx_localizer_pkg") use "FLX_LOCALIZER_PKG_.ADA";
      for Body ("flx_localizer_pkg") use "FLX_LOCALIZER_PKG.ADA";
      for Spec ("flx_iftypes") use "FLX_IFTYPES_.ADA";
      for Spec ("flx_holding_pattern_tpkg") use "FLX_HOLDING_PATTERN_TPKG_.ADA";
      for Spec ("flx_holding_pattern_pkg") use "FLX_HOLDING_PATTERN_PKG_.ADA";
      for Body ("flx_holding_pattern_pkg") use "FLX_HOLDING_PATTERN_PKG.ADA";
      for Spec ("flx_gate_tpkg") use "FLX_GATE_TPKG_.ADA";
      for Spec ("flx_gate_pkg") use "FLX_GATE_PKG_.ADA";
      for Body ("flx_gate_pkg") use "FLX_GATE_PKG.ADA";
      for Body ("flx_efis_pkg.build_wpt_candidate_list") use "FLX_EFIS_PKG_BUILD_WPT_CANDIDATE_LIST.ADA";
      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)

```
      for Body ("flx_efis_pkg.build_nav_candidate_list") use "FLX_EFIS_PKG_BUILD_NAV_CANDIDATE_LIST.ADA";
      for Body ("flx_efis_pkg.build_apt_candidate_list") use "FLX_EFIS_PKG_BUILD_APT_CANDIDATE_LIST.ADA";
      for Spec ("flx_efis_pkg") use "FLX_EFIS_PKG_.ADA";
      for Body ("flx_efis_pkg") use "FLX_EFIS_PKG.ADA";
      for Spec ("flx_db_utils_pkg") use "FLX_DB_UTILS_PKG_.ADA";
      for Body ("flx_db_utils_pkg") use "FLX_DB_UTILS_PKG.ADA";
      for Spec ("flx_corte_tpkg") use "FLX_CORTE_TPKG_.ADA";
      for Spec ("flx_corte_pkg") use "FLX_CORTE_PKG_.ADA";
      for Body ("flx_corte_pkg") use "FLX_CORTE_PKG.ADA";
      for Spec ("flx_bite_pkg") use "FLX_BITE_PKG_.ADA";
      for Body ("flx_bite_pkg") use "FLX_BITE_PKG.ADA";
      for Spec ("flx_approach_tpkg") use "FLX_APPROACH_TPKG_.ADA";
      for Spec ("flx_approach_pkg") use "FLX_APPROACH_PKG_.ADA";
      for Body ("flx_approach_pkg") use "FLX_APPROACH_PKG.ADA";
      for Spec ("flx_any_navaid_tpkg") use "FLX_ANY_NAVAID_TPKG_.ADA";
      for Spec ("flx_any_navaid_pkg") use "FLX_ANY_NAVAID_PKG_.ADA";
      for Body ("flx_any_navaid_pkg") use "FLX_ANY_NAVAID_PKG.ADA";
      for Spec ("flx_airway_tpkg") use "FLX_AIRWAY_TPKG_.ADA";
      for Spec ("flx_airway_pkg") use "FLX_AIRWAY_PKG_.ADA";
      for Body ("flx_airway_pkg") use "FLX_AIRWAY_PKG.ADA";
      for Spec ("flx_airport_tpkg") use "FLX_AIRPORT_TPKG_.ADA";
      for Spec ("flx_airport_pkg") use "FLX_AIRPORT_PKG_.ADA";
      for Body ("flx_airport_pkg") use "FLX_AIRPORT_PKG.ADA";
      for Spec ("flight_pln_leg_types") use "FLIGHT_PLN_LEG_TYPES_.ADA";
      for Spec ("flight_pln_hold_ent_types") use "FLIGHT_PLN_HOLD_ENT_TYPES_.ADA";
      for Spec ("flight_pln_hdr_types") use "FLIGHT_PLN_HDR_TYPES_.ADA";
      for Spec ("fix_info_iftypes") use "FIX_INFO_IFTYPES_.ADA";
      for Spec ("fix_info_ifdata") use "FIX_INFO_IFDATA_.ADA";
      for Spec ("ops_bite_ifdata_pkg") use "FCS_OPS_BITE_IFDATA_.ADA";
      for Spec ("fcs_iin_generic_data_def_pkg") use "FCS_IIN_GENERIC_DATA_DEF_PKG_.ADA";
      for Spec ("assert_pkg") use "FCS_ASSERT_PKG_.ADA";
      for Body ("assert_pkg") use "FCS_ASSERT_PKG.ADA";
      for Spec ("efis_ops_ifdata") use "EFIS_OPS_IFDATA_.ADA";
      for Spec ("dtrrec_types") use "DTRREC_TYPES_.ADA";
      for Spec ("dst_brg_utilities_pkg") use "DST_BRG_UTILITIES_PKG_.ADA";
      for Body ("dst_brg_utilities_pkg") use "DST_BRG_UTILITIES_PKG.ADA";
      for Spec ("dirto_wpt_iftypes") use "DIRTO_WPT_IFTYPES_.ADA";
      for Spec ("descent_path_iftypes") use "DESCENT_PATH_IFTYPES_.ADA";
      for Spec ("deldirect_iftypes") use "DELDIRECT_IFTYPES_.ADA";
      for Spec ("definefixrec_iftypes") use "DEFINEFIXREC_IFTYPES_.ADA";
      for Spec ("cs_oss_access_gnrc") use "CS_OSS_ACCESS_GNRC_.ADA";
      for Body ("cs_oss_access_gnrc") use "CS_OSS_ACCESS_GNRC.ADA";
```

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

```
   for Spec ("cs_database_access_iftypes") use "CS_DATABASE_ACCESS_IFTYPES_.ADA";
   for Spec ("cs_database_access_gnrc") use "CS_DATABASE_ACCESS_GNRC_.ADA";
   for Body ("cs_database_access_gnrc") use "CS_DATABASE_ACCESS_GNRC.ADA";
   for Spec ("compute_phinom_pkg") use "COMPUTE_PHINOM_PKG_.ADA";
   for Body ("compute_phinom_pkg") use "COMPUTE_PHINOM_PKG.ADA";
   for Spec ("compat_check_result_pkg") use "COMPAT_CHECK_RESULT_PKG_.ADA";
   for Body ("compat_check_result_pkg") use "COMPAT_CHECK_RESULT_PKG.ADA";
   for Spec ("common_io_pkg") use "COMMON_IO_PKG_.ADA";
   for Body ("common_io_pkg") use "COMMON_IO_PKG.ADA";
   for Spec ("close_intc_crs_from_iftypes") use "CLOSE_INTC_CRS_FROM_IFTYPES_.ADA";
   for Spec ("cfp_vg_iftypes") use "CFP_VG_IFTYPES_.ADA";
   for Spec ("cfp_perf_step_iftypes") use "CFP_PERF_STEP_IFTYPES_.ADA";
   for Spec ("cfp_io_download_iftypes") use "CFP_IO_DOWNLOAD_IFTYPES_.ADA";
   for Spec ("cfp_io_download_ifdata") use "CFP_IO_DOWNLOAD_IFDATA_.ADA";
   for Spec ("cfp_efis_iftypes") use "CFP_EFIS_IFTYPES_.ADA";
   for Spec ("cfp_efis_ifdata") use "CFP_EFIS_IFDATA_.ADA";
   for Spec ("cfp_directories_iftypes") use "CFP_DIRECTORIES_IFTYPES_.ADA";
   for Spec ("cfp_cdk_ifdata") use "CFP_CDK_IFDATA_.ADA";
   for Spec ("cfp_calcmagvar_pkg") use "CFP_CALCMAGVAR_PKG_.ADA";
   for Body ("cfp_calcmagvar_pkg") use "CFP_CALCMAGVAR_PKG.ADA";
   for Spec ("cex_bite_iftypes") use "CEX_BITE_IFTYPES_.ADA";
   for Spec ("cex_bite_ifdata") use "CEX_BITE_IFDATA_.ADA";
   for Spec ("cdu_output_iftypes") use "CDU_OUTPUT_IFTYPES_.ADA";
   for Spec ("cdl_dltype_iftypes") use "CDL_DLTYPE_IFTYPES_.ADA";
   for Spec ("cdl_dltokentype_data") use "CDL_DLTOKENTYPE_DATA_.ADA";
   for Spec ("cdl_cdk_takeoff_iftypes") use "CDL_CDK_TAKEOFF_UPLINK_IFTYPES_.ADA";
   for Spec ("cdl_cdk_perf_init_iftypes") use "CDL_CDK_PERF_INIT_IFTYPES_.ADA";
   for Spec ("cdl_cdk_perf_init_ifdata") use "CDL_CDK_PERF_INIT_IFDATA_.ADA";
   for Spec ("cdl_cdk_fpln_iftypes") use "CDL_CDK_FPLN_IFTYPES_.ADA";
   for Spec ("cdl_cdk_flight_plan_uplink_ifdata") use "CDL_CDK_FLIGHT_PLAN_UPLINK_IFDATA_.ADA";
   for Spec ("cdl_cdk_alternates_iftypes") use "CDL_CDK_ALTERNATES_IFTYPES_.ADA";
   for Spec ("cdl_cdk_alternates_ifdata") use "CDL_CDK_ALTERNATES_IFDATA_.ADA";
   for Spec ("cdk_reference_iftypes") use "CDK_REFERENCE_IFTYPES_.ADA";
   for Spec ("cdk_page_iftypes") use "CDK_PAGE_IFTYPES_.ADA";
   for Spec ("cdk_offpath_iftypes") use "CDK_OFFPATH_IFTYPES_.ADA";
   for Spec ("cdk_nav_radio_preselect_iftypes") use "CDK_NAV_RADIO_PRESELECT_IFTYPES_.ADA";
   for Spec ("cdk_nav_radio_iftypes") use "CDK_NAV_RADIO_IFTYPES_.ADA";
   for Spec ("cdk_key_data_iftypes") use "CDK_KEY_DATA_IFTYPES_.ADA";
   for Spec ("cdk_internal_ltypes") use "CDK_INTERNAL_LTYPES_.ADA";
   for Spec ("cdk_fix_entry_iftypes") use "CDK_FIX_ENTRY_IFTYPES_.ADA";
   for Body ("cdk_fix_entry_iftypes") use "CDK_FIX_ENTRY_IFTYPES.ADA";
   for Spec ("cabin_pressure_ifdata") use "CABIN_PRESSURE_IFDATA_.ADA";
```

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

```
      for Spec ("bld_intc_crs_from_iftypes") use "BLD_INTC_CRS_FROM_IFTYPES_.ADA";
      for Spec ("bldvia_iftypes") use "BLDVIA_IFTYPES_.ADA";
      for Spec ("bldvert_iftypes") use "BLDVERT_IFTYPES_.ADA";
      for Spec ("bldrta_iftypes") use "BLDRTA_IFTYPES_.ADA";
      for Spec ("blddeparr_iftypes") use "BLDDEPARR_IFTYPES_.ADA";
      for Spec ("bldcorte_iftypes") use "BLDCORTE_IFTYPES_.ADA";
      for Spec ("bldapt_iftypes") use "BLDAPT_IFTYPES_.ADA";
      for Spec ("bldalternate_iftypes") use "BLDALTERNATE_IFTYPES_.ADA";
      for Spec ("bldaltcnstr_iftypes") use "BLDALTCNSTR_IFTYPES_.ADA";
      for Spec ("ioc_wst_wordstring_types") use "BITE_IOC_WST_WORDSTRING_TYPES_.ADA";
      for Spec ("bite_advanced_fault_record_func_pkg") use "BITE_ADVANCED_FAULT_RECORD_FUNC_PKG_.ADA";
      for Body ("bite_advanced_fault_record_func_pkg") use "BITE_ADVANCED_FAULT_RECORD_FUNC_PKG.ADA";
      for Spec ("base_domain_services_tpkg") use "BASE_DOMAIN_SERVICES_TPKG_.ADA";
      for Spec ("options_and_data_pkg") use "B787_OPTIONS_AND_DATA_PKG_.ADA";
      for Body ("options_and_data_pkg") use "B787_OPTIONS_AND_DATA_PKG.ADA";
      for Spec ("flx_user_id_dpkg") use "B787_FLX_USER_ID_DPKG_.ADA";
      for Spec ("arr_dep_iftypes") use "ARR_DEP_IFTYPES_.ADA";
      for Spec ("arinc_base_types_pkg") use "ARINC_BASE_TYPES_PKG_.ADA";
      for Spec ("arinc_629_rep_constants_pkg") use "ARINC_629_REP_CONSTANTS_PKG_.ADA";
      for Spec ("ami_iftypes") use "AMI_IFTYPES_.ADA";
      for Spec ("alt_profile_iftypes") use "ALT_PROFILE_IFTYPES_.ADA";
      for Spec ("alternate_airport_iftypes") use "ALTERNATE_AIRPORT_IFTYPES_.ADA";
      for Spec ("ac_position_types") use "AC_POSITION_TYPES_.ADA";
      for Spec ("ac_config_types") use "AC_CONFIG_TYPES_.ADA";
      for Spec ("acars_buffer_iftypes") use "ACARS_BUFFER_IFTYPES_.ADA";

   --fm naming
      for Body ("ops_fm_partition_init_pkg") use "OPS_FM_PARTITION_INIT_PKG.ADA";
      for Body ("ops_fm_init_status_pkg") use "OPS_FM_INIT_STATUS_PKG.ADA";
      for Body ("ops_fm_database_verification_pkg") use "OPS_FM_DATABASE_VERIFICATION_PKG.ADA";
      for Body ("ops_efis_bg_pkg") use "OPS_EFIS_BG_PKG.ADA";
      for Body ("bite_exec_pkg") use "FMF_BITE_EXEC_PKG.ADA";
      for Body ("efis_store_buff_pkg") use "EFIS_STORE_BUF.ADA";
      for Spec ("bite_large_bp_service_pkg") use "BITE_LARGE_BP_SERVICE_PKG_.ADA";
      for Body ("bite_large_bp_service_pkg") use "BITE_LARGE_BP_SERVICE_PKG.ADA";
      for Spec ("perf_wts_lfdata") use "PERF_WTS_LFDATA_.ADA";
      for Spec ("perf_wind_lftypes") use "PERF_WIND_LFTYPES_.ADA";
      for Spec ("perf_wind_lfdata") use "PERF_WIND_LFDATA_.ADA";
      for Body ("perf_vtp_pkg.vtp_post_processing") use "PERF_VTP_POST_PROCESSING_SEP.ADA";
      for Spec ("perf_vtp_pkg") use "PERF_VTP_PKG_.ADA";
      for Body ("perf_vtp_pkg") use "PERF_VTP_PKG.ADA";
      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)

```
      for Body ("perf_rp_guidprms_pkg.rp_command_tsp") use "PERF_RP_COMMAND_TSP_SEP.ADA";
      for Spec ("perf_restart_preds_pkg") use "PERF_RESTART_PREDS_PKG_.ADA";
      for Body ("perf_restart_preds_pkg") use "PERF_RESTART_PREDS_PKG.ADA";
      for Body ("perf_restart_preds_pkg.restart_clear_perf_data") use "PERF_RESTART_CLEAR_PERF_DATA_SEP.ADA";
      for Body ("perf_restart_preds_pkg.restart_check") use "PERF_RESTART_CHECK_SEP.ADA";
      for Spec ("perf_recmd_crz_fl_pkg") use "PERF_RECMD_CRZ_FL_PKG_.ADA";
      for Body ("perf_recmd_crz_fl_pkg") use "PERF_RECMD_CRZ_FL_PKG.ADA";
      for Body ("perf_recmd_crz_fl_pkg.recmd_crz_fl_comp") use "PERF_RECMD_CRZ_FL_COMP_SEP.ADA";
      for Spec ("perf_punt_pkg") use "PERF_PUNT_PKG_.ADA";
      for Body ("perf_punt_pkg") use "PERF_PUNT_PKG.ADA";
      for Spec ("perf_punt_lftypes") use "PERF_PUNT_LFTYPES_.ADA";
      for Spec ("perf_punt_lfdata") use "PERF_PUNT_LFDATA_.ADA";
      for Body ("perf_pte_pkg.pte_sel_task") use "PERF_PTE_SEL_TASK_SEP.ADA";
      for Body ("perf_pte_pkg.pte_sel_priority_task") use "PERF_PTE_SEL_PRIORITY_TASK_SEP.ADA";
      for Spec ("perf_pte_pkg") use "PERF_PTE_PKG_.ADA";
      for Body ("perf_pte_pkg") use "PERF_PTE_PKG.ADA";
      for Body ("perf_pte_pkg.pte_init_timers") use "PERF_PTE_INIT_TIMERS_SEP.ADA";
      for Body ("perf_pte_pkg.pte_init_powerup_data") use "PERF_PTE_INIT_POWERUP_DATA_SEP.ADA";
      for Body ("perf_pte_pkg.pte_get_subtask_data") use "PERF_PTE_GET_SUBTASK_DATA_SEP.ADA";
      for Body ("perf_pte_pkg.pte_get_lgb") use "PERF_PTE_GET_LGB_SEP.ADA";
      for Body ("perf_pte_pkg.pte_get_exec_data") use "PERF_PTE_GET_EXEC_DATA_SEP.ADA";
      for Body ("perf_pte_pkg.pte_exec") use "PERF_PTE_EXEC_SEP.ADA";
      for Body ("perf_pte_pkg.pte_calc_timers") use "PERF_PTE_CALC_TIMERS_SEP.ADA";
      for Body ("perf_pte_pkg.calc_preds_available") use "PERF_PTE_CALC_PREDS_AVAILABLE_SEP.ADA";
      for Spec ("perf_profile_lfdata") use "PERF_PROFILE_LFDATA_.ADA";
      for Spec ("perf_pred_spd_env_pkg") use "PERF_PRED_SPD_ENV_PKG_.ADA";
      for Body ("perf_pred_spd_env_pkg") use "PERF_PRED_SPD_ENV_PKG.ADA";
      for Spec ("perf_preds_lftypes") use "PERF_PREDS_LFTYPES_.ADA";
      for Spec ("perf_preds_lfdata") use "PERF_PREDS_LFDATA_.ADA";
      for Spec ("perf_point_termination_types") use "PERF_POINT_TERMINATION_TYPES_.ADA";
      for Spec ("perf_persistent_lfdata") use "PERF_PERSISTENT_LFDATA_.ADA";
      for Spec ("perf_origin_dest_lfdata") use "PERF_ORIGIN_DEST_LFDATA_.ADA";
      for Body ("perf_opd_pkg.opd_procterms") use "PERF_OPD_PROCTERMS_SEP.ADA";
      for Body ("perf_opd_pkg.opd_predexec") use "PERF_OPD_PREDEXEC_SEP.ADA";
      for Spec ("perf_opd_pkg") use "PERF_OPD_PKG_.ADA";
      for Body ("perf_opd_pkg") use "PERF_OPD_PKG.ADA";
      for Body ("perf_opd_pkg.opd_init") use "PERF_OPD_INIT_SEP.ADA";
      for Body ("perf_opd_pkg.opd_calcradius") use "PERF_OPD_CALCRADIUS_SEP.ADA";
      for Spec ("perf_opa_pkg") use "PERF_OPA_PKG_.ADA";
      for Body ("perf_opa_pkg") use "PERF_OPA_PKG.ADA";
      for Spec ("perf_onpath_lfdata") use "PERF_ONPATH_LFDATA_.ADA";
      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            ~~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_lvlacldcl") 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)

```
      for Spec ("perf_integrators_lfdata") use "PERF_INTEGRATORS_LFDATA_.ADA";
      for Spec ("perf_idx_wind_lfdata") use "PERF_IDX_WIND_LFDATA_.ADA";
      for Spec ("perf_idx_top_of_des_lfdata") use "PERF_IDX_TOP_OF_DES_LFDATA_.ADA";
      for Spec ("perf_idx_origin_dest_lfdata") use "PERF_IDX_ORIGIN_DEST_LFDATA_.ADA";
      for Spec ("perf_idx_msg_flags_lfdata") use "PERF_IDX_MSG_FLAGS_LFDATA_.ADA";
      for Spec ("perf_idx_crzalt_lfdata") use "PERF_IDX_CRZALT_LFDATA_.ADA";
      for Spec ("perf_hld_pkg") use "PERF_HLD_PKG_.ADA";
      for Body ("perf_hld_pkg") use "PERF_HLD_PKG.ADA";
      for Spec ("perf_hdu_pkg") use "PERF_HDU_PKG_.ADA";
      for Body ("perf_hdu_pkg") use "PERF_HDU_PKG.ADA";
      for Spec ("perf_guid_header_lftypes") use "PERF_GUID_HEADER_LFTYPES_.ADA";
      for Spec ("perf_fwt_pkg") use "PERF_FWT_PKG_.ADA";
      for Body ("perf_fwt_pkg") use "PERF_FWT_PKG.ADA";
      for Spec ("perf_fwt_lfdata") use "PERF_FWT_LFDATA_.ADA";
      for Body ("perf_fwt_pkg.fwt_init") use "PERF_FWT_INIT_SEP.ADA";
      for Body ("perf_fwt_pkg.fwt_get_data") use "PERF_FWT_GET_DATA_SEP.ADA";
      for Body ("perf_fwt_pkg.fwt_exec") use "PERF_FWT_EXEC_SEP.ADA";
      for Spec ("perf_fuelintime_pkg") use "PERF_FUELINTIME_PKG_.ADA";
      for Body ("perf_fuelintime_pkg") use "PERF_FUELINTIME_PKG.ADA";
      for Spec ("perf_fpi_pkg") use "PERF_FPI_PKG_.ADA";
      for Body ("perf_fpi_pkg") use "PERF_FPI_PKG.ADA";
      for Body ("perf_fpi_pkg.fpi_exec") use "PERF_FPI_EXEC_SEP.ADA";
      for Body ("perf_fpi_pkg.fpi_estwindtemp") use "PERF_FPI_ESTWINDTEMP_SEP.ADA";
      for Body ("perf_fpi_pkg.fpi_estdistgw") use "PERF_FPI_ESTDISTGW_SEP.ADA";
      for Body ("perf_fix_info_pkg.fix_setterms") use "PERF_FIX_SETTERMS_SEP.ADA";
      for Body ("perf_fix_info_pkg.fix_procterms") use "PERF_FIX_PROCTERMS_SEP.ADA";
      for Spec ("perf_fix_info_pkg") use "PERF_FIX_INFO_PKG_.ADA";
      for Body ("perf_fix_info_pkg") use "PERF_FIX_INFO_PKG.ADA";
      for Spec ("perf_fix_info_lfdata") use "PERF_FIX_INFO_LFDATA_.ADA";
      for Body ("perf_fg_exec_pkg.fg_sim_gwt_step") use "PERF_FG_SIM_GWT_STEP_SEP.ADA";
      for Body ("perf_fg_exec_pkg.fg_powerup_init") use "PERF_FG_POWERUP_INIT_SEP.ADA";
      for Spec ("perf_fg_lfdata") use "PERF_FG_LFDATA_.ADA";
      for Body ("perf_fg_exec_pkg.fg_init") use "PERF_FG_INIT_SEP.ADA";
      for Body ("perf_fg_exec_pkg.fg_get_data") use "PERF_FG_GET_DATA_SEP.ADA";
      for Body ("perf_fg_exec_pkg.fg_flight_complete") use "PERF_FG_FLIGHT_COMPLETE_SEP.ADA";
      for Body ("perf_fg_exec_pkg.fg_exec") use "PERF_FG_EXEC_SEP.ADA";
      for Spec ("perf_fg_exec_pkg") use "PERF_FG_EXEC_PKG_.ADA";
      for Body ("perf_fg_exec_pkg") use "PERF_FG_EXEC_PKG.ADA";
      for Body ("perf_act_spd_env_pkg.fast_act_spd_env") use "PERF_FAST_ACT_SPD_ENV_SEP.ADA";
      for Body ("perf_act_spd_env_pkg.fast_act_spd_env_put_data") use "PERF_FAST_ACT_SPD_ENV_PUT_DATA_SEP.ADA";
      for Body ("perf_act_spd_env_pkg.fast_act_spd_env_get_data") use "PERF_FAST_ACT_SPD_ENV_GET_DATA_SEP.ADA";
      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_procterms") 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_procconstraint") use "PERF_DPC_PROCCONSTRAINT_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_procterms") 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_procterms") use "PERF_CRZ_PROCTERMS_SEP.ADA";
 549  548        for Body ("perf_crz_pkg.crz_procstep") use "PERF_CRZ_PROCSTEP_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_DOMAXALTMSG_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)

```
      for Body ("perf_cmd_spd_pkg.cmd_descent_cas") use "PERF_CMD_DESCENT_CAS_SEP.ADA";
      for Body ("perf_cmd_spd_pkg.cmd_crz_spd") use "PERF_CMD_CRZ_SPD_SEP.ADA";
      for Body ("perf_cmd_spd_pkg.cmd_crz_delta_ci") use "PERF_CMD_CRZ_DELTA_CI_SEP.ADA";
      for Body ("perf_cmd_spd_pkg.cmd_climb_cas") use "PERF_CMD_CLIMB_CAS_SEP.ADA";
      for Body ("perf_cmd_spd_pkg.cmd_clb_spd") use "PERF_CMD_CLB_SPD_SEP.ADA";
      for Body ("perf_clb_pkg.clb_setwptterms") use "PERF_CLB_SETWPTTERMS_SEP.ADA";
      for Body ("perf_clb_pkg.clb_setterms") use "PERF_CLB_SETTERMS_SEP.ADA";
      for Body ("perf_clb_pkg.clb_procterms") use "PERF_CLB_PROCTERMS_SEP.ADA";
      for Body ("perf_clb_pkg.clb_predexec") use "PERF_CLB_PREDEXEC_SEP.ADA";
      for Spec ("perf_clb_pkg") use "PERF_CLB_PKG_.ADA";
      for Body ("perf_clb_pkg") use "PERF_CLB_PKG.ADA";
      for Spec ("perf_bld_pkg") use "PERF_BLD_PKG_.ADA";
      for Body ("perf_bld_pkg") use "PERF_BLD_PKG.ADA";
      for Body ("perf_atm_pkg.atm_preddeswind") use "PERF_ATM_PREDDESWIND_SEP.ADA";
      for Body ("perf_atm_pkg.atm_predcrzwind") use "PERF_ATM_PREDCRZWIND_SEP.ADA";
      for Body ("perf_atm_pkg.atm_predclbwind") use "PERF_ATM_PREDCLBWIND_SEP.ADA";
      for Spec ("perf_atm_pkg") use "PERF_ATM_PKG_.ADA";
      for Body ("perf_atm_pkg") use "PERF_ATM_PKG.ADA";
      for Body ("perf_atm_pkg.atm_model") use "PERF_ATM_MODEL_SEP.ADA";
      for Body ("perf_atm_pkg.atm_interpwind") use "PERF_ATM_INTERPWIND_SEP.ADA";
      for Body ("perf_atm_pkg.atm_forcwind") use "PERF_ATM_FORCWIND_SEP.ADA";
      for Body ("perf_atm_pkg.atm_forctempdev") use "PERF_ATM_FORCTEMPDEV_SEP.ADA";
      for Body ("perf_atm_pkg.atm_forcdeswind") use "PERF_ATM_FORCDESWIND_SEP.ADA";
      for Body ("perf_atm_pkg.atm_calctempdev") use "PERF_ATM_CALCTEMPDEV_SEP.ADA";
      for Body ("perf_atm_pkg.atm_calcpredwind") use "PERF_ATM_CALCPREDWIND_SEP.ADA";
      for Body ("perf_atm_pkg.atm_alt_forcwind") use "PERF_ATM_ALT_FORCWIND_SEP.ADA";
      for Body ("perf_atm_pkg.atm_altnwind") use "PERF_ATM_ALTNWIND_SEP.ADA";
      for Body ("perf_atm_pkg.atm_altntemp") use "PERF_ATM_ALTNTEMP_SEP.ADA";
      for Body ("perf_altn_pkg.altn_pred_to_overhead") use "PERF_ALTN_PRED_TO_OVERHEAD_SEP.ADA";
      for Body ("perf_altn_pkg.altn_predict") use "PERF_ALTN_PREDICT_SEP.ADA";
      for Spec ("perf_altn_pkg") use "PERF_ALTN_PKG_.ADA";
      for Body ("perf_altn_pkg") use "PERF_ALTN_PKG.ADA";
      for Spec ("perf_altn_lfdata") use "PERF_ALTN_LFDATA_.ADA";
      for Body ("perf_altn_pkg.altn_init") use "PERF_ALTN_INIT_SEP.ADA";
      for Body ("perf_altn_pkg.altn_est_despath") use "PERF_ALTN_EST_DESPATH_SEP.ADA";
      for Body ("perf_altn_pkg.altn_calc_alt") use "PERF_ALTN_CALC_ALT_SEP.ADA";
      for Spec ("perf_altd_pkg") use "PERF_ALTD_PKG_.ADA";
      for Body ("perf_altd_pkg") use "PERF_ALTD_PKG.ADA";
      for Spec ("perf_air_data_pkg") use "PERF_AIR_DATA_PKG_.ADA";
      for Body ("perf_air_data_pkg") use "PERF_AIR_DATA_PKG.ADA";
      for Body ("perf_aero_takeoff_pkg.wind_comp") use "PERF_AERO_TKO_WIND_COMP_SEP.ADA";
      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)

```
      for Body ("perf_aero_takeoff_pkg.takeoff_speeds") use "PERF_AERO_TKO_TAKEOFF_SPEEDS_SEP.ADA";
      for Body ("perf_aero_takeoff_pkg.stabtrimset") use "PERF_AERO_TKO_STABTRIMSET_SEP.ADA";
      for Body ("perf_aero_takeoff_pkg.speed_monitor_initialize") use "PERF_AERO_TKO_SPD_MONITOR_INIT_SEP.ADA";
      for Body ("perf_aero_takeoff_pkg.select_database") use "PERF_AERO_TKO_SELECT_DATABASE_SEP.ADA";
      for Body ("perf_aero_takeoff_pkg.scheduled_speeds") use "PERF_AERO_TKO_SCHEDULED_SPEEDS_SEP.ADA";
      for Body ("perf_aero_takeoff_pkg.put_vspeeds") use "PERF_AERO_TKO_PUT_VSPEEDS_SEP.ADA";
      for Body ("perf_aero_takeoff_pkg.input_proc") use "PERF_AERO_TKO_INPUT_PROC_SEP.ADA";
      for Body ("perf_aero_takeoff_pkg.gen_engine_data") use "PERF_AERO_TKO_GEN_ENGINE_DATA_SEP.ADA";
      for Body ("perf_aero_takeoff_pkg.eng_data_interp") use "PERF_AERO_TKO_ENG_DATA_INTERP_SEP.ADA";
      for Body ("perf_aero_takeoff_pkg.balanced_field.bal_fld_st_eqn") use "PERF_AERO_TKO_BAL_FLD_ST_EQN_SEP.ADA";
      for Body ("perf_aero_takeoff_pkg.balanced_field.bal_fld_integ") use "PERF_AERO_TKO_BAL_FLD_INTEG_SEP.ADA";
      for Body ("perf_aero_takeoff_pkg.balanced_field.bal_fld_comp_v1mcg") use "PERF_AERO_TKO_BAL_FLD_COMP_V1MCG_SEP.ADA";
      for Body ("perf_aero_takeoff_pkg.balanced_field") use "PERF_AERO_TKO_BALANCED_FIELD_SEP.ADA";
      for Body ("perf_aero_takeoff_pkg.atmos_comp") use "PERF_AERO_TKO_ATMOS_COMP_SEP.ADA";
      for Spec ("perf_aero_takeoff_pkg") use "PERF_AERO_TAKEOFF_PKG_.ADA";
      for Body ("perf_aero_takeoff_pkg") use "PERF_AERO_TAKEOFF_PKG.ADA";
      for Spec ("perf_aero_speed_pkg") use "PERF_AERO_SPEED_PKG_.ADA";
      for Body ("perf_aero_speed_pkg") use "PERF_AERO_SPEED_PKG.ADA";
      for Spec ("perf_aero_reference_pkg") use "PERF_AERO_REFERENCE_PKG_.ADA";
      for Body ("perf_aero_reference_pkg") use "PERF_AERO_REFERENCE_PKG.ADA";
      for Spec ("perf_aero_limit_pkg") use "PERF_AERO_LIMIT_PKG_.ADA";
      for Body ("perf_aero_limit_pkg") use "PERF_AERO_LIMIT_PKG.ADA";
      for Body ("perf_ads_intent_pkg.ads_set_terms") use "PERF_ADS_SET_TERMS_SEP.ADA";
      for Body ("perf_ads_intent_pkg.ads_process_terms") use "PERF_ADS_PROCESS_TERMS_SEP.ADA";
      for Spec ("perf_ads_intent_pkg") use "PERF_ADS_INTENT_PKG_.ADA";
      for Body ("perf_ads_intent_pkg") use "PERF_ADS_INTENT_PKG.ADA";
      for Body ("perf_ads_intent_pkg.ads_get_init_data") use "PERF_ADS_GET_INIT_DATA_SEP.ADA";
      for Body ("perf_ads_intent_pkg.calc_off_route_fixed_intent_points") use "PERF_ADS_CALC_OFF_RTE_FIXED_INTENT_SEP.ADA";
      for Body ("perf_ads_intent_pkg.calc_intermediate_point") use "PERF_ADS_CALC_INTERMEDIATE_POINT_SEP.ADA";
      for Body ("perf_ads_intent_pkg.calc_fixed_point") use "PERF_ADS_CALC_FIXED_POINT_SEP.ADA";
      for Spec ("perf_act_spd_env_pkg") use "PERF_ACT_SPD_ENV_PKG_.ADA";
      for Body ("perf_act_spd_env_pkg") use "PERF_ACT_SPD_ENV_PKG.ADA";
      for Spec ("perf_acstate_mgr") use "PERF_ACSTATE_MGR_.ADA";
      for Body ("perf_acstate_mgr") use "PERF_ACSTATE_MGR.ADA";
      for Spec ("ops_vgb_mgr_pkg") use "OPS_VGB_MGR_PKG_.ADA";
      for Body ("ops_vgb_mgr_pkg") use "OPS_VGB_MGR_PKG.ADA";
      for Spec ("ops_tob_pkg") use "OPS_TOB_PKG_.ADA";
      for Body ("ops_tob_pkg") use "OPS_TOB_PKG.ADA";
      for Spec ("ops_tob_ifdata") use "OPS_TOB_IFDATA_.ADA";
      for Spec ("ops_tob_hs_mgr_pkg") use "OPS_TOB_HS_MGR_PKG_.ADA";
      for Body ("ops_tob_hs_mgr_pkg") use "OPS_TOB_HS_MGR_PKG.ADA";
      for Spec ("ops_standby_one_ltypes") use "OPS_STANDBY_ONE_LTYPES_.ADA";
```

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

```
      for Spec ("ops_printer_pkg") use "OPS_PRINTER_PKG_.ADA";
      for Body ("ops_printer_pkg") use "OPS_PRINTER_PKG.ADA";
      for Spec ("ops_perf_fg_pkg") use "OPS_PERF_FG_PKG_.ADA";
      for Body ("ops_perf_fg_pkg") use "OPS_PERF_FG_PKG.ADA";
      for Spec ("ops_perf_change_flags_mgr_pkg") use "OPS_PERF_CHANGE_FLAGS_MGR_PKG_.ADA";
      for Body ("ops_perf_change_flags_mgr_pkg") use "OPS_PERF_CHANGE_FLAGS_MGR_PKG.ADA";
      for Spec ("ops_perf_bg_pkg") use "OPS_PERF_BG_PKG_.ADA";
      for Body ("ops_perf_bg_pkg") use "OPS_PERF_BG_PKG.ADA";
      for Spec ("ops_onehertz_pkg") use "OPS_ONEHERTZ_PKG_.ADA";
      for Body ("ops_onehertz_pkg") use "OPS_ONEHERTZ_PKG.ADA";
      for Spec ("ops_nvm_manager") use "OPS_NVM_MANAGER_.ADA";
      for Body ("ops_nvm_manager") use "OPS_NVM_MANAGER.ADA";
      for Spec ("ops_lateral_guidance_buffer_manager") use "OPS_LATERAL_GUIDANCE_BUFFER_MANAGER_.ADA";
      for Body ("ops_lateral_guidance_buffer_manager") use "OPS_LATERAL_GUIDANCE_BUFFER_MANAGER.ADA";
      for Spec ("ops_icbout_pkg") use "OPS_ICBOUT_PKG_.ADA";
      for Body ("ops_icbout_pkg") use "OPS_ICBOUT_PKG.ADA";
      for Spec ("ops_hs_sync_mon") use "OPS_HS_SYNC_MON_PKG_.ADA";
      for Body ("ops_hs_sync_mon") use "OPS_HS_SYNC_MON_PKG.ADA";
      for Spec ("ops_hs_chkpnt_control_pkg") use "OPS_HS_CHKPNT_CONTROL_PKG_.ADA";
      for Body ("ops_hs_chkpnt_control_pkg") use "OPS_HS_CHKPNT_CONTROL_PKG.ADA";
      for Spec ("ops_hs_buffer_pkg") use "OPS_HS_BUFFER_PKG_.ADA";
      for Body ("ops_hs_buffer_pkg") use "OPS_HS_BUFFER_PKG.ADA";
      for Spec ("ops_hotspare_exec_pkg") use "OPS_HOTSPARE_EXEC_PKG_.ADA";
      for Body ("ops_hotspare_exec_pkg") use "OPS_HOTSPARE_EXEC_PKG.ADA";
      for Spec ("ops_foreground_pkg") use "OPS_FOREGROUND_PKG_.ADA";
      for Body ("ops_foreground_pkg") use "OPS_FOREGROUND_PKG.ADA";
      for Spec ("ops_fm_zombie_pkg") use "OPS_FM_ZOMBIE_PKG_.ADA";
      for Body ("ops_fm_zombie_pkg") use "OPS_FM_ZOMBIE_PKG.ADA";
      for Spec ("ops_fm_utilities_pkg") use "OPS_FM_UTILITIES_PKG_.ADA";
      for Body ("ops_fm_utilities_pkg") use "OPS_FM_UTILITIES_PKG.ADA";
      for Spec ("ops_fm_semaphore_id_pkg") use "OPS_FM_SEMAPHORE_ID_PKG_.ADA";
      for Spec ("ops_process_id_pkg") use "OPS_FM_PROCESS_ID_PKG_.ADA";
      for Spec ("ops_fm_partition_init_pkg") use "OPS_FM_PARTITION_INIT_PKG_.ADA";
      for Body ("ops_fm_partition_init_pkg") use "OPS_FM_PARTITION_INIT_PKG.ADA";
      for Body ("ops_fm_master_determination.determine_transition_allowance") use "OPS_FM_MASTER_DET_TRANS_ALLOW.ADA";
      for Body ("ops_fm_master_determination.transitions_exec") use "OPS_FM_MASTER_DET_TRANSITIONS_EXEC.ADA";
      for Body ("ops_fm_master_determination.power_up_partition") use "OPS_FM_MASTER_DET_POWERUP.ADA";
      for Body ("ops_fm_master_determination.determine_offside_master_status") use "OPS_FM_MASTER_DET_OFFS_MS.ADA";
      for Body ("ops_fm_master_determination.determine_offside_kill_signal") use "OPS_FM_MASTER_DET_OFFS_KILL.ADA";
      for Body ("ops_fm_master_determination.calculate_tm_status") use "OPS_FM_MASTER_DET_CALC_TM_MS.ADA";
      for Body ("ops_fm_master_determination.calculate_offside_health") use "OPS_FM_MASTER_DET_CALC_OFFS_HEALTH.ADA";
      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)

```
      for Spec ("ops_fm_master_determination") use "OPS_FM_MASTER_DETERMINATION_PKG_.ADA";
      for Body ("ops_fm_master_determination") use "OPS_FM_MASTER_DETERMINATION_PKG.ADA";
      for Spec ("ops_fm_latch_states_pkg") use "OPS_FM_LATCH_STATES_PKG_.ADA";
      for Body ("ops_fm_latch_states_pkg") use "OPS_FM_LATCH_STATES_PKG.ADA";
      for Spec ("ops_fm_init_status_pkg") use "OPS_FM_INIT_STATUS_PKG_.ADA";
      for Spec ("ops_fm_fault_response_pkg") use "OPS_FM_FAULT_RESPONSE_PKG_.ADA";
      for Body ("ops_fm_fault_response_pkg") use "OPS_FM_FAULT_RESPONSE_PKG.ADA";
      for Spec ("ops_fm_event_id_pkg") use "OPS_FM_EVENT_ID_PKG_.ADA";
      for Spec ("ops_fm_erp_pkg") use "OPS_FM_ERROR_RECOVERY_PROCESS_PKG_.ADA";
      for Body ("ops_fm_erp_pkg") use "OPS_FM_ERROR_RECOVERY_PROCESS_PKG.ADA";
      for Spec ("ops_fm_database_verification_pkg") use "OPS_FM_DATABASE_VERIFICATION_PKG_.ADA";
      for Body ("ops_fm_database_verification_pkg") use "OPS_FM_DATABASE_VERIFICATION_PKG.ADA";
      for Spec ("ops_fm_database_utils_pkg") use "OPS_FM_DATABASE_UTILS_PKG_.ADA";
      for Spec ("ops_fm_bite_pkg") use "OPS_FM_BITE_PKG_.ADA";
      for Body ("ops_fm_bite_pkg") use "OPS_FM_BITE_PKG.ADA";
      for Spec ("ops_efis_path_pkg") use "OPS_EFIS_PATH_PKG_.ADA";
      for Body ("ops_efis_path_pkg") use "OPS_EFIS_PATH_PKG.ADA";
      for Spec ("ops_efis_fg_pkg") use "OPS_EFIS_FG_PKG_.ADA";
      for Body ("ops_efis_fg_pkg") use "OPS_EFIS_FG_PKG.ADA";
      for Spec ("ops_efis_bg_pkg") use "OPS_EFIS_BG_PKG_.ADA";
      for Body ("ops_efis_bg_pkg") use "OPS_EFIS_BG_PKG.ADA";
      for Spec ("ops_dual_simsoft_pkg") use "OPS_DUAL_SIMSOFT_PKG_.ADA";
      for Body ("ops_dual_simsoft_pkg") use "OPS_DUAL_SIMSOFT_PKG.ADA";
      for Spec ("ops_cfp_nav_mgr") use "OPS_CFP_NAV_MGR_.ADA";
      for Body ("ops_cfp_nav_mgr") use "OPS_CFP_NAV_MGR.ADA";
      for Spec ("ops_cfp_internal_mgr_pkg") use "OPS_CFP_INTERNAL_MGR_PKG_.ADA";
      for Body ("ops_cfp_internal_mgr_pkg") use "OPS_CFP_INTERNAL_MGR_PKG.ADA";
      for Spec ("ops_cex_cdukey_pkg") use "OPS_CEX_CDUKEY_PKG_.ADA";
      for Body ("ops_cex_cdukey_pkg") use "OPS_CEX_CDUKEY_PKG.ADA";
      for Spec ("ops_cdu_guid_mgr") use "OPS_CDU_GUID_MGR_.ADA";
      for Body ("ops_cdu_guid_mgr") use "OPS_CDU_GUID_MGR.ADA";
      for Spec ("ops_cdl_interface_mgr_pkg") use "OPS_CDL_INTERFACE_MGR_PKG_.ADA";
      for Body ("ops_cdl_interface_mgr_pkg") use "OPS_CDL_INTERFACE_MGR_PKG.ADA";
      for Spec ("ops_cdl_buffer_mgr_pkg") use "OPS_CDL_BUFFER_MGR_PKG_.ADA";
      for Body ("ops_cdl_buffer_mgr_pkg") use "OPS_CDL_BUFFER_MGR_PKG.ADA";
      for Spec ("ops_cdk_perf_pdb_mgr_pkg") use "OPS_CDK_PERF_PDB_MGR_PKG_.ADA";
      for Body ("ops_cdk_perf_pdb_mgr_pkg") use "OPS_CDK_PERF_PDB_MGR_PKG.ADA";
      for Spec ("ops_cdk_page_data_mgr_pkg") use "OPS_CDK_PAGE_DATA_MGR_PKG_.ADA";
      for Body ("ops_cdk_page_data_mgr_pkg") use "OPS_CDK_PAGE_DATA_MGR_PKG.ADA";
      for Spec ("ops_cdk_io_mgr_pkg") use "OPS_CDK_IO_MGR_PKG_.ADA";
      for Body ("ops_cdk_io_mgr_pkg") use "OPS_CDK_IO_MGR_PKG.ADA";
      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)

```
      for Body ("ops_cdk_internal_cdk_mgr_pkg") use "OPS_CDK_INTERNAL_CDK_MGR_PKG.ADA";
      for Spec ("ops_cdk_fuel_weight_mgr_pkg") use "OPS_CDK_FUEL_WEIGHT_MGR_PKG_.ADA";
      for Body ("ops_cdk_fuel_weight_mgr_pkg") use "OPS_CDK_FUEL_WEIGHT_MGR_PKG.ADA";
      for Spec ("ops_cdk_fm_nav_mgr_pkg") use "OPS_CDK_FM_NAV_MGR_PKG_.ADA";
      for Body ("ops_cdk_fm_nav_mgr_pkg") use "OPS_CDK_FM_NAV_MGR_PKG.ADA";
      for Spec ("ops_cdk_flt_plan_mgr_pkg") use "OPS_CDK_FLT_PLAN_MGR_PKG_.ADA";
      for Body ("ops_cdk_flt_plan_mgr_pkg") use "OPS_CDK_FLT_PLAN_MGR_PKG.ADA";
      for Spec ("ops_cdk_duplicate_fix_mgr_pkg") use "OPS_CDK_DUPLICATE_FIX_MGR_PKG_.ADA";
      for Body ("ops_cdk_duplicate_fix_mgr_pkg") use "OPS_CDK_DUPLICATE_FIX_MGR_PKG.ADA";
      for Spec ("ops_cdk_common_mgr_pkg") use "OPS_CDK_COMMON_MGR_PKG_.ADA";
      for Body ("ops_cdk_common_mgr_pkg") use "OPS_CDK_COMMON_MGR_PKG.ADA";
      for Body ("ops_cdk_altn_data_mgr_pkg.put_preds") use "OPS_CDK_ALTN_DATA_MGR_PKG__PUT_PREDS.ADA";
      for Body ("ops_cdk_altn_data_mgr_pkg.gen_list_ul") use "OPS_CDK_ALTN_DATA_MGR_PKG__GEN_LIST_UL.ADA";
      for Body ("ops_cdk_altn_data_mgr_pkg.enter_arpt") use "OPS_CDK_ALTN_DATA_MGR_PKG__ENTER_ARPT.ADA";
      for Body ("ops_cdk_altn_data_mgr_pkg.sort_by_time") use "OPS_CDK_ALTN_DATA_MGR_PKG_SORT_BY_TIME.ADA";
      for Body ("ops_cdk_altn_data_mgr_pkg.sort_by_dist") use "OPS_CDK_ALTN_DATA_MGR_PKG_SORT_BY_DIST.ADA";
      for Body ("ops_cdk_altn_data_mgr_pkg.gen_list_ndb") use "OPS_CDK_ALTN_DATA_MGR_PKG_GEN_LIST_NDB.ADA";
      for Spec ("ops_cdk_altn_data_mgr_pkg") use "OPS_CDK_ALTN_DATA_MGR_PKG_.ADA";
      for Body ("ops_cdk_altn_data_mgr_pkg") use "OPS_CDK_ALTN_DATA_MGR_PKG.ADA";
      for Spec ("offset_segment_set_pkg") use "OFFSET_SEGMENT_SET_PKG_.ADA";
      for Spec ("lt_initialize_wrapper_pkg") use "LT_INITIALIZE_WRAPPER_PKG_.ADA";
      for Body ("lt_initialize_wrapper_pkg") use "LT_INITIALIZE_WRAPPER_PKG.ADA";
      for Body ("lg_data_managers_pkg.lg_write_slow_to_fast_data") use "LG_WRITE_SLOW_TO_FAST_DATA_SEP.ADA";
      for Body ("lg_data_managers_pkg.lg_write_slow_interface_data") use "LG_WRITE_SLOW_INTERFACE_DATA_SEP.ADA";
      for Body ("lg_data_managers_pkg.lg_write_fast_to_slow_data") use "LG_WRITE_FAST_TO_SLOW_DATA_SEP.ADA";
      for Body ("lg_data_managers_pkg.lg_write_fast_interface_data") use "LG_WRITE_FAST_INTERFACE_DATA_SEP.ADA";
      for Body ("lg_path_def_pkg.lg_updatelegs") use "LG_UPDATELEGS_SEP.ADA";
      for Spec ("lg_unable_hold_airspace_message") use "LG_UNABLE_HOLD_AIRSPACE_MESSAGE_.ADA";
      for Body ("lg_unable_hold_airspace_message") use "LG_UNABLE_HOLD_AIRSPACE_MESSAGE.ADA";
      for Body ("lg_mode_control_pkg.lg_testlnaveng") use "LG_TESTLNAVENG_SEP.ADA";
      for Body ("lg_path_def_pkg.lg_tactical_segments") use "LG_TACTICAL_SEGMENTS_SEP.ADA";
      for Body ("lg_slow_progress_data_pkg.lg_sumlegdists") use "LG_SUMLEGDISTS_SEP.ADA";
      for Body ("lg_seq_legs_pkg.lg_storewptdata") use "LG_STOREWPTDATA_SEP.ADA";
      for Spec ("lg_slow_to_fast_lfdata") use "LG_SLOW_TO_FAST_LFDATA_.ADA";
      for Spec ("lg_slow_task_lfdata") use "LG_SLOW_TASK_LFDATA_.ADA";
      for Spec ("lg_slow_progress_data_pkg") use "LG_SLOW_PROGRESS_DATA_PKG_.ADA";
      for Body ("lg_slow_progress_data_pkg") use "LG_SLOW_PROGRESS_DATA_PKG.ADA";
      for Spec ("lg_slow_in_lfdata") use "LG_SLOW_IN_LFDATA_.ADA";
      for Body ("lg_mode_control_pkg.lg_set_ref_point") use "LG_SET_REF_POINT_SEP.ADA";
      for Body ("lg_slow_progress_data_pkg.lg_setnrplrp") use "LG_SETNRPLRP_SEP.ADA";
      for Spec ("lg_seq_legs_pkg") use "LG_SEQ_LEGS_PKG_.ADA";
      for Body ("lg_seq_legs_pkg") use "LG_SEQ_LEGS_PKG.ADA";
```

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

```
      for Body ("lg_seq_legs_pkg.lg_seq_lat_leg") use "LG_SEQ_LAT_LEG_SEP.ADA";
      for Body ("lg_seq_legs_pkg.lg_sequencing_paramaters") use "LG_SEQUENCING_PARAMATERS_SEP.ADA";
      for Body ("lg_seq_legs_pkg.lg_sequencelegs") use "LG_SEQUENCELEGS_SEP.ADA";
      for Body ("lg_slow_progress_data_pkg.lg_sel_vgdist_comp") use "LG_SEL_VGDIST_COMP_SEP.ADA";
      for Body ("lg_slow_progress_data_pkg.lg_sel_control_mode") use "LG_SEL_CONTROL_MODE_SEP.ADA";
      for Body ("lg_path_def_pkg.lg_select_hx_speed") use "LG_SELECT_HX_SPEED_SEP.ADA";
      for Body ("lg_data_managers_pkg.lg_read_slow_to_fast_data") use "LG_READ_SLOW_TO_FAST_DATA_SEP.ADA";
      for Body ("lg_data_managers_pkg.lg_read_slow_interface_data") use "LG_READ_SLOW_INTERFACE_DATA_SEP.ADA";
      for Body ("lg_data_managers_pkg.lg_read_fast_to_slow_data") use "LG_READ_FAST_TO_SLOW_DATA_SEP.ADA";
      for Body ("lg_data_managers_pkg.lg_read_fast_interface_data") use "LG_READ_FAST_INTERFACE_DATA_SEP.ADA";
      for Body ("lg_path_def_pkg.lg_populate_active_leg_record") use "LG_POPULATE_ACTIVE_LEG_RECORD_SEP.ADA";
      for Body ("lg_fast_progress_data_pkg.lg_path_leg_prog") use "LG_PATH_LEG_PROG_SEP.ADA";
      for Spec ("lg_path_def_pkg") use "LG_PATH_DEF_PKG_.ADA";
      for Body ("lg_path_def_pkg") use "LG_PATH_DEF_PKG.ADA";
      for Spec ("lg_mode_control_pkg") use "LG_MODE_CONTROL_PKG_.ADA";
      for Body ("lg_mode_control_pkg") use "LG_MODE_CONTROL_PKG.ADA";
      for Spec ("lg_lt_wrapper_pkg") use "LG_LT_WRAPPER_PKG_.ADA";
      for Body ("lg_lt_wrapper_pkg") use "LG_LT_WRAPPER_PKG.ADA";
      for Spec ("lg_lt_segment_list_wrapper_pkg") use "LG_LT_SEGMENT_LIST_WRAPPER_PKG_.ADA";
      for Body ("lg_lt_segment_list_wrapper_pkg") use "LG_LT_SEGMENT_LIST_WRAPPER_PKG.ADA";
      for Body ("lg_path_def_pkg.populate_lt_aircraft_state") use "LG_LT_POPULATE_AC_STATE_SEP.ADA";
      for Spec ("lg_lt_interface_tpkg") use "LG_LT_INTERFACE_TPKG_.ADA";
      for Spec ("lg_lt_ifdata") use "LG_LT_IFDATA_.ADA";
      for Body ("lg_seq_legs_pkg.lg_line_segment_intercept") use "LG_LINE_SEGMENT_INTERCEPT_SEP.ADA";
      for Spec ("lg_lftypes") use "LG_LFTYPES_.ADA";
      for Body ("lg_mode_control_pkg.lg_lat_path_cap") use "LG_LAT_PATH_CAP_SEP.ADA";
      for Body ("lg_fast_progress_data_pkg.lg_latpatherror") use "LG_LATPATHERROR_SEP.ADA";
      for Body ("lg_executives_pkg.lg_init") use "LG_INIT_SEP.ADA";
      for Body ("lg_seq_legs_pkg.lg_initiate_trans") use "LG_INITIATE_TRANS_SEP.ADA";
      for Body ("lg_path_def_pkg.lg_hx_size_limit") use "LG_HX_SIZE_LIMIT_SEP.ADA";
      for Body ("lg_fast_progress_data_pkg.lg_hx_pi_leg_prog") use "LG_HX_PI_LEG_PROG_SEP.ADA";
      for Body ("lg_fast_progress_data_pkg.lg_heading_leg_prog") use "LG_HEADING_LEG_PROG_SEP.ADA";
      for Body ("lg_executives_pkg.lg_guidcontrol") use "LG_GUIDCONTROL_SEP.ADA";
      for Spec ("lg_fast_to_slow_lfdata") use "LG_FAST_TO_SLOW_LFDATA_.ADA";
      for Spec ("lg_fast_task_lfdata") use "LG_FAST_TASK_LFDATA_.ADA";
      for Spec ("lg_fast_progress_data_pkg") use "LG_FAST_PROGRESS_DATA_PKG_.ADA";
      for Body ("lg_fast_progress_data_pkg") use "LG_FAST_PROGRESS_DATA_PKG.ADA";
      for Spec ("lg_fast_in_lfdata") use "LG_FAST_IN_LFDATA_.ADA";
      for Body ("lg_executives_pkg.lg_fastguid") use "LG_FASTGUID_SEP.ADA";
      for Spec ("lg_executives_pkg") use "LG_EXECUTIVES_PKG_.ADA";
      for Body ("lg_executives_pkg") use "LG_EXECUTIVES_PKG.ADA";
      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_cmptransit") use "LG_CMPTRANSIT_SEP.ADA";
 844   843        for Body ("lg_fast_progress_data_pkg.lg_cmptdistocrp") use "LG_CMPTDISTOCRP_SEP.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_caplimdist") 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)

```
      for Body ("fpp_wrap_segment_pkg") use "FPP_WRAP_SEGMENT_PKG.ADA";
      for Spec ("fpp_wrap_request_process_pkg") use "FPP_WRAP_REQUEST_PROCESS_PKG_.ADA";
      for Body ("fpp_wrap_request_process_pkg") use "FPP_WRAP_REQUEST_PROCESS_PKG.ADA";
      for Spec ("fpp_wrap_point_pkg") use "FPP_WRAP_POINT_PKG_.ADA";
      for Body ("fpp_wrap_point_pkg") use "FPP_WRAP_POINT_PKG.ADA";
      for Spec ("fpp_wrap_controller_pkg") use "FPP_WRAP_CONTROLLER_PKG_.ADA";
      for Body ("fpp_wrap_controller_pkg") use "FPP_WRAP_CONTROLLER_PKG.ADA";
      for Spec ("fpp_wrap_config_data_pkg") use "FPP_WRAP_CONFIG_DATA_PKG_.ADA";
      for Body ("fpp_wrap_config_data_pkg") use "FPP_WRAP_CONFIG_DATA_PKG.ADA";
      for Spec ("fpp_status_type_tpkg") use "FPP_STATUS_TYPE_TPKG_.ADA";
      for Spec ("fpp_proj_utils_pkg") use "FPP_PROJ_UTILS_PKG_.ADA";
      for Body ("fpp_proj_utils_pkg") use "FPP_PROJ_UTILS_PKG.ADA";
      for Spec ("fpp_interface_type") use "FPP_INTERFACE_TYPE_.ADA";
      for Spec ("fpp_handle_wrap_pkg") use "FPP_HANDLE_WRAP_PKG_.ADA";
      for Body ("fpp_handle_wrap_pkg") use "FPP_HANDLE_WRAP_PKG.ADA";
      for Spec ("fpp_fpa_wrap_pkg") use "FPP_FPA_WRAP_PKG_.ADA";
      for Body ("fpp_fpa_wrap_pkg") use "FPP_FPA_WRAP_PKG.ADA";
      for Spec ("fpp_common_lgb_wrap_pkg") use "FPP_COMMON_LGB_WRAP_PKG_.ADA";
      for Body ("fpp_common_lgb_wrap_pkg") use "FPP_COMMON_LGB_WRAP_PKG.ADA";
      for Spec ("fpp_awy_utils_wrap_pkg") use "FPP_AWY_UTILS_WRAP_PKG_.ADA";
      for Body ("fpp_awy_utils_wrap_pkg") use "FPP_AWY_UTILS_WRAP_PKG.ADA";
      for Spec ("ops_io_hotspare_ifpkg") use "FM_OPS_IO_HOTSPARE_IFPKG_.ADA";
      for Body ("ops_io_hotspare_ifpkg") use "FM_OPS_IO_HOTSPARE_IFPKG.ADA";
      for Spec ("fm_nav_toga_update") use "FM_NAV_TOGA_UPDATE_.ADA";
      for Body ("fm_nav_toga_update") use "FM_NAV_TOGA_UPDATE.ADA";
      for Body ("fm_nav_support.fm_onenav") use "FM_NAV_SUPPORT_FM_ONENAV.ADA";
      for Body ("fm_nav_support.fm_fastnav") use "FM_NAV_SUPPORT_FM_FASTNAV.ADA";
      for Spec ("fm_nav_support") use "FM_NAV_SUPPORT_.ADA";
      for Body ("fm_nav_support") use "FM_NAV_SUPPORT.ADA";
      for Body ("fm_nav_simsoft_interface.onehz_update") use "FM_NAV_SIMSOFT_INTERFACE_ONEHZ_UPDATE.ADA";
      for Body ("fm_nav_simsoft_interface.fast_update") use "FM_NAV_SIMSOFT_INTERFACE_FAST_UPDATE.ADA";
      for Spec ("fm_nav_simsoft_interface") use "FM_NAV_SIMSOFT_INTERFACE_.ADA";
      for Body ("fm_nav_simsoft_interface") use "FM_NAV_SIMSOFT_INTERFACE.ADA";
      for Spec ("fm_nav_rnp_interface") use "FM_NAV_RNP_INTERFACE_.ADA";
      for Body ("fm_nav_rnp_interface") use "FM_NAV_RNP_INTERFACE.ADA";
      for Body ("fm_nav_rnp.compute_rnp") use "FM_NAV_RNP_COMPUTE_RNP.ADA";
      for Spec ("fm_nav_rnp") use "FM_NAV_RNP_.ADA";
      for Body ("fm_nav_rnp") use "FM_NAV_RNP.ADA";
      for Spec ("radio_sample") use "FM_NAV_RADIO_SAMPLE_.ADA";
      for Body ("radio_sample") use "FM_NAV_RADIO_SAMPLE.ADA";
      for Spec ("radio_navaid") use "FM_NAV_RADIO_NAVAID_.ADA";
      for Body ("radio_navaid") use "FM_NAV_RADIO_NAVAID.ADA";
```

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

```
      for Body ("radio_management.navaid_in_line_of_sight") use "FM_NAV_RADIO_MANAGEMENT_NAVAID_IN_LOS.ADA";
      for Body ("radio_management.in_vor_zone_of_confusion") use "FM_NAV_RADIO_MANAGEMENT_IN_VOR_ZOC.ADA";
      for Body ("radio_management.approximate_distance") use "FM_NAV_RADIO_MANAGEMENT_APPROX_DIST.ADA";
      for Spec ("radio_management") use "FM_NAV_RADIO_MANAGEMENT_.ADA";
      for Body ("radio_management") use "FM_NAV_RADIO_MANAGEMENT.ADA";
      for Body ("fm_nav_pos_update.select_update_type") use "FM_NAV_POS_UPDATE_SELECT_UPDATE_TYPE.ADA";
      for Spec ("fm_nav_pos_update") use "FM_NAV_POS_UPDATE_.ADA";
      for Body ("fm_nav_pos_update") use "FM_NAV_POS_UPDATE.ADA";
      for Body ("fm_nav_partition_interface.onehz_update") use "FM_NAV_PARTITION_INTERFACE_ONEHZ_UPD.ADA";
      for Body ("fm_nav_partition_interface.fast_update") use "FM_NAV_PARTITION_INTERFACE_FAST_UPDATE.ADA";
      for Spec ("fm_nav_partition_interface") use "FM_NAV_PARTITION_INTERFACE_.ADA";
      for Body ("fm_nav_partition_interface") use "FM_NAV_PARTITION_INTERFACE.ADA";
      for Body ("fm_nav_ops_interface.onehz_update") use "FM_NAV_OPS_INTERFACE_ONEHZ_UPDATE.ADA";
      for Spec ("fm_nav_ops_interface") use "FM_NAV_OPS_INTERFACE_.ADA";
      for Body ("fm_nav_ops_interface") use "FM_NAV_OPS_INTERFACE.ADA";
      for Body ("fm_onenav_io_interface.put_fm_onenav_data") use "FM_NAV_ONENAV_IO_INTFACE_PUT_1NAV_DATA.ADA";
      for Body ("fm_onenav_io_interface.get_fm_onenav_external_data") use "FM_NAV_ONENAV_IO_INTFACE_EXTERN_DATA.ADA";
      for Spec ("fm_onenav_io_interface") use "FM_NAV_ONENAV_IO_INTFACE_.ADA";
      for Body ("fm_onenav_io_interface") use "FM_NAV_ONENAV_IO_INTFACE.ADA";
      for Body ("fm_nav_messages.vrnp_message") use "FM_NAV_MESSAGES_VRNP_MESSAGE.ADA";
      for Body ("fm_nav_messages.vp_message") use "FM_NAV_MESSAGES_VP_MESSAGE.ADA";
      for Body ("fm_nav_messages.vert_rnp_message") use "FM_NAV_MESSAGES_VERT_RNP_MESSAGES.ADA";
      for Body ("fm_nav_messages.rw_ils_freq_message") use "FM_NAV_MESSAGES_RW_ILS_FREQ_MESSAGE.ADA";
      for Body ("fm_nav_messages.rw_ils_crs_message") use "FM_NAV_MESSAGES_RW_ILS_CRS_MESSAGE.ADA";
      for Body ("fm_nav_messages.runway_monitor") use "FM_NAV_MESSAGES_RUNWAY_MONITOR.ADA";
      for Body ("fm_nav_messages.pre_nav_messages") use "FM_NAV_MESSAGES_PRE_NAV_MESSAGES.ADA";
      for Body ("fm_nav_messages.post_nav_messages") use "FM_NAV_MESSAGES_POST_NAV_MESSAGES.ADA";
      for Body ("fm_nav_messages.nit_message") use "FM_NAV_MESSAGES_NIT_MESSAGE.ADA";
      for Body ("fm_nav_messages.messages") use "FM_NAV_MESSAGES_MESSAGES.ADA";
      for Body ("fm_nav_messages.iti_message") use "FM_NAV_MESSAGES_ITI_MESSAGE.ADA";
      for Body ("fm_nav_messages.initialize") use "FM_NAV_MESSAGES_INITIALIZE.ADA";
      for Spec ("fm_nav_messages") use "FM_NAV_MESSAGES_.ADA";
      for Body ("fm_nav_messages") use "FM_NAV_MESSAGES.ADA";
      for Body ("fm_nav_inertial_interface.onehz_update") use "FM_NAV_INERTIAL_INTERFACE_ONEHZ_UPDATE.ADA";
      for Body ("fm_nav_inertial_interface.fast_update") use "FM_NAV_INERTIAL_INTERFACE_FAST_UPDATE.ADA";
      for Spec ("fm_nav_inertial_interface") use "FM_NAV_INERTIAL_INTERFACE_.ADA";
      for Body ("fm_nav_inertial_interface") use "FM_NAV_INERTIAL_INTERFACE.ADA";
      for Spec ("ops_fm_nav_hs_mgr_pkg") use "FM_NAV_HS_MGR_PKG_.ADA";
      for Body ("ops_fm_nav_hs_mgr_pkg") use "FM_NAV_HS_MGR_PKG.ADA";
      for Spec ("guidance_interface") use "FM_NAV_GUIDANCE_INTERFACE_.ADA";
      for Body ("guidance_interface") use "FM_NAV_GUIDANCE_INTERFACE.ADA";
      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)

```
      for Spec ("fm_nav_flight_phase") use "FM_NAV_FLIGHT_PHASE_.ADA";
      for Body ("fm_nav_flight_phase") use "FM_NAV_FLIGHT_PHASE.ADA";
      for Body ("fm_fastnav_io_interface.get_external_fm_fastnav_data") use "FM_NAV_FASTNAV_IO_INTFACE_EXTERN_DATA.ADA";
      for Spec ("fm_fastnav_io_interface") use "FM_NAV_FASTNAV_IO_INTFACE_.ADA";
      for Body ("fm_fastnav_io_interface") use "FM_NAV_FASTNAV_IO_INTFACE.ADA";
      for Body ("fm_nav_cockpit_interface.onehz_update") use "FM_NAV_COCKPIT_INTERFACE_ONEHZ_UPDATE.ADA";
      for Body ("fm_nav_cockpit_interface.fast_update") use "FM_NAV_COCKPIT_INTERFACE_FAST_UPDATE.ADA";
      for Spec ("fm_nav_cockpit_interface") use "FM_NAV_COCKPIT_INTERFACE_.ADA";
      for Body ("fm_nav_cockpit_interface") use "FM_NAV_COCKPIT_INTERFACE.ADA";
      for Spec ("fm_nav_checkpoint") use "FM_NAV_CHECKPOINT_.ADA";
      for Body ("fm_nav_checkpoint") use "FM_NAV_CHECKPOINT.ADA";
      for Body ("fm_nav_cfp_interface.reciprocal_runway") use "FM_NAV_CFP_INTERFACE_RECIPROCAL_RWY.ADA";
      for Body ("fm_nav_cfp_interface.onehz_update") use "FM_NAV_CFP_INTERFACE_ONEHZ_UPDATE.ADA";
      for Body ("fm_nav_cfp_interface.get_proc_vor_crs") use "FM_NAV_CFP_INTERFACE_GET_PROC_VOR_CRS.ADA";
      for Body ("fm_nav_cfp_interface.get_leg_data") use "FM_NAV_CFP_INTERFACE_GET_LEG_DATA.ADA";
      for Body ("fm_nav_cfp_interface.get_header_data") use "FM_NAV_CFP_INTERFACE_GET_HEADER_DATA.ADA";
      for Body ("fm_nav_cfp_interface.find_downpath_vor") use "FM_NAV_CFP_INTERFACE_FIND_DOWNPATH_VOR.ADA";
      for Spec ("fm_nav_cfp_interface") use "FM_NAV_CFP_INTERFACE_.ADA";
      for Body ("fm_nav_cfp_interface") use "FM_NAV_CFP_INTERFACE.ADA";
      for Body ("fm_nav_cdk_interface.onehz_update") use "FM_NAV_CDK_INTERFACE_ONEHZ_UPDATE.ADA";
      for Body ("fm_nav_cdk_interface.fast_update") use "FM_NAV_CDK_INTERFACE_FAST_UPDATE.ADA";
      for Spec ("fm_nav_cdk_interface") use "FM_NAV_CDK_INTERFACE_.ADA";
      for Body ("fm_nav_cdk_interface") use "FM_NAV_CDK_INTERFACE.ADA";
      for Body ("fm_nav_anp_monitor.compute_vert_anp") use "FM_NAV_ANP_MONITOR_COMPUTE_VERT_ANP.ADA";
      for Body ("fm_nav_anp_monitor.compute_unable_rnp") use "FM_NAV_ANP_MONITOR_COMPUTE_UNABLE_RNP.ADA";
      for Spec ("fm_nav_anp_monitor") use "FM_NAV_ANP_MONITOR_.ADA";
      for Body ("fm_nav_anp_monitor") use "FM_NAV_ANP_MONITOR.ADA";
      for Spec ("navigation_data") use "FM_NAVIGATION_DATA_.ADA";
      for Body ("navigation_data") use "FM_NAVIGATION_DATA.ADA";
      for Body ("fm_navaid_selector.select_route_vor") use "FM_NAVAID_SELECTOR_SELECT_ROUTE_VOR.ADA";
      for Body ("fm_navaid_selector.select_proc_vor") use "FM_NAVAID_SELECTOR_SELECT_PROC_VOR.ADA";
      for Body ("fm_navaid_selector.select_localizer") use "FM_NAVAID_SELECTOR_SELECT_LOCALIZER.ADA";
      for Body ("fm_navaid_selector.localizer_navaid") use "FM_NAVAID_SELECTOR_LOCALIZER_NAVAID.ADA";
      for Body ("fm_navaid_selector.display_navaids") use "FM_NAVAID_SELECTOR_DISPLAY_NAVAIDS.ADA";
      for Spec ("fm_navaid_selector") use "FM_NAVAID_SELECTOR_.ADA";
      for Body ("fm_navaid_selector") use "FM_NAVAID_SELECTOR.ADA";
      for Spec ("flx_semaphore_pkg") use "FM_FLX_SEMAPHORE_PKG_.ADA";
      for Body ("flx_semaphore_pkg") use "FM_FLX_SEMAPHORE_PKG.ADA";
      for Spec ("vg_slow_to_fast_lfdata") use "FMF_VG_SLOW_TO_FAST_LFDATA_.ADA";
      for Body ("guidance_executive.vg_init") use "FMF_VG_INIT_SEP.ADA";
      for Spec ("vg_hot_spare_lfdata") use "FMF_VG_HOT_SPARE_LFDATA_.ADA";
      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_tospdtgtsel") use "FMF_VGS_TOSPDTGTSEL_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_selecttlclim") use "FMF_VGS_SELECTTLCLIM_SEP.ADA";
1010  1009        for Body ("vgs_speed_tgt_sel.vgs_selcasmach") use "FMF_VGS_SELCASMACH_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_cruisecntrl") 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)

```
      for Body ("vgf_engagement.vgf_vnav_engout") use "FMF_VGF_VNAV_ENGOUT_SEP.ADA";
      for Body ("vgf_logic.vgf_transitions_executive") use "FMF_VGF_TRANSITIONS_EXECUTIVE_SEP.ADA";
      for Body ("vg_control_laws.vgf_thrust_control") use "FMF_VGF_THRUST_CONTROL_SEP.ADA";
      for Body ("vg_control_laws.vgf_tas_filter") use "FMF_VGF_TAS_FILTER_SEP.ADA";
      for Body ("vgf_logic.vgf_submode_logic") use "FMF_VGF_SUBMODE_LOGIC_SEP.ADA";
      for Body ("vg_control_laws.vgf_spd_tgt_rate_lim") use "FMF_VGF_SPD_TGT_RATE_LIM_SEP.ADA";
      for Body ("vg_control_laws.vgf_spd_tgt_asf_bias") use "FMF_VGF_SPD_TGT_ASF_BIAS_SEP.ADA";
      for Body ("vg_control_laws.vgf_spd_rev_cntrl_sel") use "FMF_VGF_SPD_REV_CNTRL_SEL_SEP.ADA";
      for Body ("vg_control_laws.vgf_soe_control") use "FMF_VGF_SOE_CONTROL_SEP.ADA";
      for Spec ("vgf_snapshot_lfdata") use "FMF_VGF_SNAPSHOT_LFDATA_.ADA";
      for Body ("vg_data_managers.vgf_read_data") use "FMF_VGF_READ_DATA_SEP.ADA";
      for Body ("vg_control_laws.vgf_poe_control") use "FMF_VGF_POE_CONTROL_SEP.ADA";
      for Body ("vg_control_laws.vgf_pitch_command") use "FMF_VGF_PITCH_COMMAND_SEP.ADA";
      for Body ("vgf_logic.vgf_path_error") use "FMF_VGF_PATH_ERROR_SEP.ADA";
      for Body ("vgf_logic.vgf_path_error_filter") use "FMF_VGF_PATH_ERROR_FILTER_SEP.ADA";
      for Spec ("vgf_logic") use "FMF_VGF_LOGIC_PKG_.ADA";
      for Body ("vgf_logic") use "FMF_VGF_LOGIC_PKG.ADA";
      for Body ("vgf_engagement.vgf_init_engagement") use "FMF_VGF_INIT_ENGAGEMENT_SEP.ADA";
      for Body ("vg_executives.vg_fastguid") use "FMF_VGF_FASTGUID_SEP.ADA";
      for Body ("vgf_logic.vgf_epr_adjust_mode") use "FMF_VGF_EPR_ADJUST_MODE_SEP.ADA";
      for Body ("vg_control_laws.vgf_enhanced_speed_reversion") use "FMF_VGF_ENHANCED_SPEED_REVERSION_SEP.ADA";
      for Body ("vgf_engagement.vgf_engagement") use "FMF_VGF_ENGAGEMENT_SEP.ADA";
      for Spec ("vgf_engagement") use "FMF_VGF_ENGAGEMENT_PKG_.ADA";
      for Body ("vgf_engagement") use "FMF_VGF_ENGAGEMENT_PKG.ADA";
      for Body ("vgf_logic.vgf_descent_transitions") use "FMF_VGF_DESCENT_TRANSITIONS_SEP.ADA";
      for Body ("vgf_logic.vgf_descent_path_smooth") use "FMF_VGF_DESCENT_PATH_SMOOTH_SEP.ADA";
      for Body ("vgf_logic.vgf_cruise_transitions") use "FMF_VGF_CRUISE_TRANSITIONS_SEP.ADA";
      for Body ("vg_control_laws.vgf_cmpt_wind_acc") use "FMF_VGF_CMPT_WIND_ACC_SEP.ADA";
      for Body ("vg_control_laws.vgf_cmpt_potential_fpa") use "FMF_VGF_CMPT_POTENTIAL_FPA_SEP.ADA";
      for Body ("vg_control_laws.vgf_cl_spd_select") use "FMF_VGF_CL_SPD_SELECT_SEP.ADA";
      for Body ("vgf_logic.vgf_climb_transitions") use "FMF_VGF_CLIMB_TRANSITIONS_SEP.ADA";
      for Body ("vgf_logic.vgf_autothrottle_mode") use "FMF_VGF_AUTOTHROTTLE_MODE_SEP.ADA";
      for Body ("vg_control_laws.vgf_asf_target") use "FMF_VGF_ASFTARGET_SEP.ADA";
      for Body ("vg_control_laws.vgf_asf_control") use "FMF_VGF_ASFCONTROL_SEP.ADA";
      for Body ("vg_control_laws.vgf_asf_activate") use "FMF_VGF_ASFACTIVATE_SEP.ADA";
      for Body ("vg_control_laws.vgf_asc_target") use "FMF_VGF_ASC_TARGET_SEP.ADA";
      for Body ("vg_control_laws.vgf_asc_control") use "FMF_VGF_ASC_CONTROL_SEP.ADA";
      for Body ("vg_control_laws.vgf_asc_activate") use "FMF_VGF_ASC_ACTIVATE_SEP.ADA";
      for Body ("vgf_logic.vgf_altitude_target") use "FMF_VGF_ALTITUDE_TARGET_SEP.ADA";
      for Body ("vgf_logic.vgf_above_path") use "FMF_VGF_ABOVE_PATH_SEP.ADA";
      for Body ("fmf_simsoft_io_pkg.put_outputs") use "FMF_SIMSOFT_IO_PKG_PUT_OUTPUTS.ADA";
      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)

```
      for Spec ("fmf_simsoft_io_pkg") use "FMF_SIMSOFT_IO_PKG_.ADA";
      for Body ("fmf_simsoft_io_pkg") use "FMF_SIMSOFT_IO_PKG.ADA";
      for Body ("fmf_simsoft_exec_pkg.rte_rsng_edg_sgnls") use "FMF_SIMSOFT_EXEC_PKG_RTE_RSNG_EDG.ADA";
      for Body ("fmf_simsoft_exec_pkg.rte_flng_edg_sgnls") use "FMF_SIMSOFT_EXEC_PKG_RTE_FLNG_EDG.ADA";
      for Body ("fmf_simsoft_exec_pkg.exec") use "FMF_SIMSOFT_EXEC_PKG_EXEC.ADA";
      for Body ("fmf_simsoft_exec_pkg.cr8_fts_response") use "FMF_SIMSOFT_EXEC_PKG_CR8_FTS_RESPONSE.ADA";
      for Spec ("fmf_simsoft_exec_pkg") use "FMF_SIMSOFT_EXEC_PKG_.ADA";
      for Body ("fmf_simsoft_exec_pkg") use "FMF_SIMSOFT_EXEC_PKG.ADA";
      for Body ("fmf_simsoft_cnfig_pkg.rsng_edg_init") use "FMF_SIMSOFT_CNFIG_PKG_RSNG_EDG_INIT.ADA";
      for Body ("fmf_simsoft_cnfig_pkg.init") use "FMF_SIMSOFT_CNFIG_PKG_INIT.ADA";
      for Body ("fmf_simsoft_cnfig_pkg.flng_edg_init") use "FMF_SIMSOFT_CNFIG_PKG_FLNG_EDG_INIT.ADA";
      for Spec ("fmf_simsoft_cnfig_pkg") use "FMF_SIMSOFT_CNFIG_PKG_.ADA";
      for Body ("fmf_simsoft_cnfig_pkg") use "FMF_SIMSOFT_CNFIG_PKG.ADA";
      for Spec ("fmf_secondary_throttle_hold_pkg") use "FMF_SECONDARY_THROTTLE_HOLD_PKG_.ADA";
      for Body ("fmf_secondary_throttle_hold_pkg") use "FMF_SECONDARY_THROTTLE_HOLD_PKG.ADA";
      for Body ("fmf_printer_snapshot_data_mgr_pkg.snapshot_data") use "FMF_PRINTER_SNAPSHOT_DATA_SEP.ADA";
      for Spec ("fmf_printer_snapshot_data_mgr_pkg") use "FMF_PRINTER_SNAPSHOT_DATA_MGR_PKG_.ADA";
      for Body ("fmf_printer_snapshot_data_mgr_pkg") use "FMF_PRINTER_SNAPSHOT_DATA_MGR_PKG.ADA";
      for Spec ("fmf_printer_lftypes_pkg") use "FMF_PRINTER_LFTYPES_PKG_.ADA";
      for Spec ("fmf_perf_takeoff_bump_pkg") use "FMF_PERF_TAKEOFF_BUMP_PKG_.ADA";
      for Body ("fmf_perf_takeoff_bump_pkg") use "FMF_PERF_TAKEOFF_BUMP_PKG.ADA";
      for Spec ("perf_su_spd_utils_pkg") use "FMF_PERF_SU_SPD_UTILS_PKG_.ADA";
      for Body ("perf_su_spd_utils_pkg") use "FMF_PERF_SU_SPD_UTILS_PKG.ADA";
      for Spec ("perf_st_spdtape_pkg") use "FMF_PERF_ST_SPDTAPE_PKG_.ADA";
      for Body ("perf_st_spdtape_pkg") use "FMF_PERF_ST_SPDTAPE_PKG.ADA";
      for Body ("fmf_perf_epm_pkg.epm_trm_psp") use "FMF_PERF_EPM_TRM_PSP_SEP.ADA";
      for Body ("fmf_perf_epm_pkg.epm_trip_pred") use "FMF_PERF_EPM_TRIP_PRED_SEP.ADA";
      for Body ("fmf_perf_epm_pkg.epm_tko_spd") use "FMF_PERF_EPM_TKO_SPD_SEP.ADA";
      for Body ("fmf_perf_epm_pkg.epm_reverse_thrust") use "FMF_PERF_EPM_REVERSE_THRUST_SEP.ADA";
      for Body ("fmf_perf_epm_pkg.epm_rating_thrust_model_2") use "FMF_PERF_EPM_RATING_THRUST_2_SEP.ADA";
      for Body ("fmf_perf_epm_pkg.epm_rating") use "FMF_PERF_EPM_RATING_SEP.ADA";
      for Spec ("fmf_perf_epm_pkg") use "FMF_PERF_EPM_PKG_.ADA";
      for Body ("fmf_perf_epm_pkg") use "FMF_PERF_EPM_PKG.ADA";
      for Body ("fmf_perf_epm_pkg.epm_n1_to_epr") use "FMF_PERF_EPM_N1_TO_EPR_SEP.ADA";
      for Body ("fmf_perf_epm_pkg.epm_min_tko_spd") use "FMF_PERF_EPM_MIN_TKO_SPD_SEP.ADA";
      for Body ("fmf_perf_epm_pkg.epm_idle") use "FMF_PERF_EPM_IDLE_SEP.ADA";
      for Body ("fmf_perf_epm_pkg.epm_fuel_flow_model_2") use "FMF_PERF_EPM_FUEL_FLOW_2_SEP.ADA";
      for Body ("fmf_perf_epm_pkg.epm_epr_to_n1") use "FMF_PERF_EPM_EPR_TO_N1_SEP.ADA";
      for Body ("fmf_perf_epm_pkg.epm_bleed_flag") use "FMF_PERF_EPM_BLEED_FLAG_SEP.ADA";
      for Spec ("fmf_perf_aer_acdrag_pkg") use "FMF_PERF_AER_ACDRAG_PKG_.ADA";
      for Body ("fmf_perf_aer_acdrag_pkg") use "FMF_PERF_AER_ACDRAG_PKG.ADA";
      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)

```
      for Spec ("ops_hs_cross_cab_pkg") use "FMF_OPS_HS_CROSS_CAB_PKG_.ADA";
      for Body ("ops_hs_cross_cab_pkg") use "FMF_OPS_HS_CROSS_CAB_PKG.ADA";
      for Spec ("fmf_hsp_types_pkg") use "FMF_HSP_TYPES_PKG_.ADA";
      for Spec ("fmf_hsp_sync_state_pkg") use "FMF_HSP_SYNC_STATE_PKG_.ADA";
      for Body ("fmf_hsp_sync_state_pkg") use "FMF_HSP_SYNC_STATE_PKG.ADA";
      for Body ("fmf_hsp_sync_msg_pkg.send_seq_no_block") use "FMF_HSP_SYNC_MSG_PKG_SEND_SEQ_NO_BLOCK.ADA";
      for Body ("fmf_hsp_sync_msg_pkg.send_init_reqst") use "FMF_HSP_SYNC_MSG_PKG_SEND_INIT_REQST.ADA";
      for Body ("fmf_hsp_sync_msg_pkg.send_init_ack") use "FMF_HSP_SYNC_MSG_PKG_SEND_INIT_ACK.ADA";
      for Body ("fmf_hsp_sync_msg_pkg.send_data_block") use "FMF_HSP_SYNC_MSG_PKG_SEND_DATA_BLOCK.ADA";
      for Body ("fmf_hsp_sync_msg_pkg.send_data_ack") use "FMF_HSP_SYNC_MSG_PKG_SEND_DATA_ACK.ADA";
      for Spec ("fmf_hsp_sync_msg_pkg") use "FMF_HSP_SYNC_MSG_PKG_.ADA";
      for Body ("fmf_hsp_sync_msg_pkg") use "FMF_HSP_SYNC_MSG_PKG.ADA";
      for Body ("fmf_hsp_control_pkg.tx_data") use "FMF_HSP_CONTROL_PKG_TX_DATA.ADA";
      for Body ("fmf_hsp_control_pkg.state_reset") use "FMF_HSP_CONTROL_PKG_STATE_RESET.ADA";
      for Body ("fmf_hsp_control_pkg.send_init_reqst") use "FMF_HSP_CONTROL_PKG_SEND_INIT_REQST.ADA";
      for Body ("fmf_hsp_control_pkg.send_init_ack") use "FMF_HSP_CONTROL_PKG_SEND_INIT_ACK.ADA";
      for Body ("fmf_hsp_control_pkg.send_ack_nak") use "FMF_HSP_CONTROL_PKG_SEND_ACK_NAK.ADA";
      for Body ("fmf_hsp_control_pkg.rx_data") use "FMF_HSP_CONTROL_PKG_RX_DATA.ADA";
      for Body ("fmf_hsp_control_pkg.proc_input_frm") use "FMF_HSP_CONTROL_PKG_PROC_INPUT_FRM.ADA";
      for Body ("fmf_hsp_control_pkg.proc_ack_nak") use "FMF_HSP_CONTROL_PKG_PROC_ACK_NAK.ADA";
      for Body ("fmf_hsp_control_pkg.master_idle_proc") use "FMF_HSP_CONTROL_PKG_MASTER_IDLE_PROC.ADA";
      for Body ("fmf_hsp_control_pkg.log_protocol_error") use "FMF_HSP_CONTROL_PKG_LOG_PROTOCOL_ERROR.ADA";
      for Body ("fmf_hsp_control_pkg.init_reqst_pending") use "FMF_HSP_CONTROL_PKG_INIT_REQST_PENDING.ADA";
      for Body ("fmf_hsp_control_pkg.init_ack_pending") use "FMF_HSP_CONTROL_PKG_INIT_ACK_PENDING.ADA";
      for Body ("fmf_hsp_control_pkg.initialize") use "FMF_HSP_CONTROL_PKG_INITIALIZE.ADA";
      for Body ("fmf_hsp_control_pkg.hs_exec") use "FMF_HSP_CONTROL_PKG_HS_EXEC.ADA";
      for Body ("fmf_hsp_control_pkg.frame_control") use "FMF_HSP_CONTROL_PKG_FRAME_CONTROL.ADA";
      for Spec ("fmf_hsp_control_pkg") use "FMF_HSP_CONTROL_PKG_.ADA";
      for Body ("fmf_hsp_control_pkg") use "FMF_HSP_CONTROL_PKG.ADA";
      for Body ("fmf_hsp_blocking_pkg.rollback") use "FMF_HSP_BLOCKING_PKG_ROLLBACK.ADA";
      for Body ("fmf_hsp_blocking_pkg.put_block") use "FMF_HSP_BLOCKING_PKG_PUT_BLOCK.ADA";
      for Body ("fmf_hsp_blocking_pkg.init") use "FMF_HSP_BLOCKING_PKG_INIT.ADA";
      for Body ("fmf_hsp_blocking_pkg.get_block") use "FMF_HSP_BLOCKING_PKG_GET_BLOCK.ADA";
      for Spec ("fmf_hsp_blocking_pkg") use "FMF_HSP_BLOCKING_PKG_.ADA";
      for Body ("fmf_hsp_blocking_pkg") use "FMF_HSP_BLOCKING_PKG.ADA";
      for Spec ("fmf_hsp_access_pkg") use "FMF_HSP_ACCESS_PKG_.ADA";
      for Body ("fmf_hsp_access_pkg") use "FMF_HSP_ACCESS_PKG.ADA";
      for Body ("guidance_executive.guid_setup") use "FMF_GUID_SETUP_SEP.ADA";
      for Body ("guidance_executive.guid_onehertz_exec") use "FMF_GUID_ONEHERTZ_EXEC_SEP.ADA";
      for Body ("guidance_executive.guid_foregnd_exec") use "FMF_GUID_FOREGND_EXEC_SEP.ADA";
      for Spec ("guidance_spare_buffered_ifdata") use "FMF_GUIDANCE_SPARE_BUFFERED_IFDATA_.ADA";
      for Spec ("guidance_ifdata") use "FMF_GUIDANCE_IFDATA_.ADA";
```

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

```
      for Body ("guidance_ifdata") use "FMF_GUIDANCE_IFDATA.ADA";
      for Spec ("guidance_executive") use "FMF_GUIDANCE_EXECUTIVE_PKG_.ADA";
      for Body ("guidance_executive") use "FMF_GUIDANCE_EXECUTIVE_PKG.ADA";
      for Spec ("guidance_data_managers_pkg") use "FMF_GUIDANCE_DATA_MANAGERS_PKG_.ADA";
      for Body ("guidance_data_managers_pkg") use "FMF_GUIDANCE_DATA_MANAGERS_PKG.ADA";
      for Spec ("fmf_dual_partition_ifdata") use "FMF_DUAL_PARTITION_IFDATA_.ADA";
      for Body ("fmf_dual_partition_ifdata") use "FMF_DUAL_PARTITION_IFDATA.ADA";
      for Spec ("fmf_chronometer") use "FMF_CHRONOMETER_.ADA";
      for Body ("fmf_chronometer") use "FMF_CHRONOMETER.ADA";
      for Spec ("fmf_cam_timer_pkg") use "FMF_CAM_TIMER_PKG_.ADA";
      for Body ("fmf_cam_timer_pkg") use "FMF_CAM_TIMER_PKG.ADA";
      for Spec ("fmf_cam_process_data") use "FMF_CAM_PROCESS_DATA_.ADA";
      for Body ("fmf_cam_process_data") use "FMF_CAM_PROCESS_DATA.ADA";
      for Spec ("fmf_cam_msg_pkg") use "FMF_CAM_MSG_PKG_.ADA";
      for Body ("fmf_cam_msg_pkg") use "FMF_CAM_MSG_PKG.ADA";
      for Spec ("fmf_cam_init_for_startup") use "FMF_CAM_INIT_FOR_STARTUP_.ADA";
      for Body ("fmf_cam_init_for_startup") use "FMF_CAM_INIT_FOR_STARTUP.ADA";
      for Spec ("fmf_cam_activity_pkg") use "FMF_CAM_ACTIVITY_PKG_.ADA";
      for Body ("fmf_cam_activity_pkg") use "FMF_CAM_ACTIVITY_PKG.ADA";
      for Spec ("bite_periodic_data_update_pkg") use "FMF_BITE_PERIODIC_DATA_UPDATE_PKG_.ADA";
      for Body ("bite_periodic_data_update_pkg") use "FMF_BITE_PERIODIC_DATA_UPDATE_PKG.ADA";
      for Spec ("bite_exec_pkg") use "FMF_BITE_EXEC_PKG_.ADA";
      for Body ("bite_exec_pkg") use "FMF_BITE_EXEC_PKG.ADA";
      for Spec ("bite_cmcf_pkg") use "FMF_BITE_CMCF_PKG_.ADA";
      for Body ("bite_cmcf_pkg") use "FMF_BITE_CMCF_PKG.ADA";
      for Spec ("fmf_atc_uplink_preloading_pkg") use "FMF_ATC_UPLINK_PRELOADING_PKG_.ADA";
      for Body ("fmf_atc_uplink_preloading_pkg") use "FMF_ATC_UPLINK_PRELOADING_PKG.ADA";
      for Spec ("fmf_atc_uplink_obj_mgr") use "FMF_ATC_UPLINK_OBJ_MGR_.ADA";
      for Body ("fmf_atc_uplink_obj_mgr") use "FMF_ATC_UPLINK_OBJ_MGR.ADA";
      for Spec ("fmf_atc_uplink_loading_pkg") use "FMF_ATC_UPLINK_LOADING_PKG_.ADA";
      for Body ("fmf_atc_uplink_loading_pkg") use "FMF_ATC_UPLINK_LOADING_PKG.ADA";
      for Spec ("fmf_atc_uplink_decoding_pkg") use "FMF_ATC_UPLINK_DECODING_PKG_.ADA";
      for Body ("fmf_atc_uplink_decoding_pkg") use "FMF_ATC_UPLINK_DECODING_PKG.ADA";
      for Spec ("fmf_atc_types") use "FMF_ATC_TYPES_.ADA";
      for Body ("fmf_atc_route_request_pkg.bld_rte_clr") use "FMF_ATC_ROUTE_REQUEST_PKG__BLD_RTE_CLR.ADA";
      for Spec ("fmf_atc_route_request_pkg") use "FMF_ATC_ROUTE_REQUEST_PKG_.ADA";
      for Body ("fmf_atc_route_request_pkg") use "FMF_ATC_ROUTE_REQUEST_PKG.ADA";
      for Spec ("fmf_atc_route_clearance_pkg") use "FMF_ATC_ROUTE_CLEARANCE_PKG_.ADA";
      for Body ("fmf_atc_route_clearance_pkg") use "FMF_ATC_ROUTE_CLEARANCE_PKG.ADA";
      for Spec ("fmf_atc_report_crossing_req_pkg") use "FMF_ATC_REPORT_CROSSING_REQ_PKG_.ADA";
      for Body ("fmf_atc_report_crossing_req_pkg") use "FMF_ATC_REPORT_CROSSING_REQ_PKG.ADA";
      for Spec ("fmf_atc_rc_additional_pkg") use "FMF_ATC_RC_ADDITIONAL_PKG_.ADA";
```

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

```
      for Body ("fmf_atc_rc_additional_pkg") use "FMF_ATC_RC_ADDITIONAL_PKG.ADA";
      for Spec ("fmf_atc_procedure_request_pkg") use "FMF_ATC_PROCEDURE_REQUEST_PKG_.ADA";
      for Body ("fmf_atc_procedure_request_pkg") use "FMF_ATC_PROCEDURE_REQUEST_PKG.ADA";
      for Spec ("fmf_atc_output_parameters_pkg") use "FMF_ATC_OUTPUT_PARAMETERS_PKG_.ADA";
      for Body ("fmf_atc_output_parameters_pkg") use "FMF_ATC_OUTPUT_PARAMETERS_PKG.ADA";
      for Spec ("fmf_atc_manager_pkg") use "FMF_ATC_MANAGER_PKG_.ADA";
      for Body ("fmf_atc_manager_pkg") use "FMF_ATC_MANAGER_PKG.ADA";
      for Spec ("fmf_atc_internal_data_obj_mgr") use "FMF_ATC_INTERNAL_DATA_OBJ_MGR_.ADA";
      for Body ("fmf_atc_internal_data_obj_mgr") use "FMF_ATC_INTERNAL_DATA_OBJ_MGR.ADA";
      for Spec ("fmf_atc_flight_number_pkg") use "FMF_ATC_FLIGHT_NUMBER_PKG_.ADA";
      for Body ("fmf_atc_flight_number_pkg") use "FMF_ATC_FLIGHT_NUMBER_PKG.ADA";
      for Spec ("fmf_atc_downlink_obj_mgr") use "FMF_ATC_DOWNLINK_OBJ_MGR_.ADA";
      for Body ("fmf_atc_downlink_obj_mgr") use "FMF_ATC_DOWNLINK_OBJ_MGR.ADA";
      for Spec ("fmf_atc_divert_pos_request_pkg") use "FMF_ATC_DIVERT_POS_REQUEST_PKG_.ADA";
      for Body ("fmf_atc_divert_pos_request_pkg") use "FMF_ATC_DIVERT_POS_REQUEST_PKG.ADA";
      for Spec ("fmf_atc_dist_to_wpt_req_pkg") use "FMF_ATC_DIST_TO_WPT_REQ_PKG_.ADA";
      for Body ("fmf_atc_dist_to_wpt_req_pkg") use "FMF_ATC_DIST_TO_WPT_REQ_PKG.ADA";
      for Body ("fmf_secondary_throttle_hold_pkg.fmf_at2ndhold_sep") use "FMF_AT2NDHOLD_SEP.ADA";
      for Spec ("fmf_ami_access_pkg") use "FMF_AMI_ACCESS_PKG_.ADA";
      for Body ("fmf_ami_access_pkg") use "FMF_AMI_ACCESS_PKG.ADA";
      for Spec ("fmcs_partition_itypes_pkg") use "FMCS_FM_PARTITION_ITYPES_PKG_.ADA";
      for Spec ("fmci_widget_event_tpkg") use "FMCI_WIDGET_EVENT_TPKG_.ADA";
      for Spec ("fmci_validate_address_pkg") use "FMCI_VALIDATE_ADDRESS_PKG_.ADA";
      for Body ("fmci_validate_address_pkg") use "FMCI_VALIDATE_ADDRESS_PKG.ADA";
      for Spec ("fmci_utility_tpkg") use "FMCI_UTILITY_TPKG_.ADA";
      for Body ("fmci_utility_pkg.process_nd_spad_entry") use "FMCI_UTILITY_PKG_PROC_ND_SPAD_ENT.ADA";
      for Spec ("fmci_utility_pkg") use "FMCI_UTILITY_PKG_.ADA";
      for Body ("fmci_utility_pkg") use "FMCI_UTILITY_PKG.ADA";
      for Spec ("fmci_spad_util_pkg") use "FMCI_SPAD_UTIL_PKG_.ADA";
      for Body ("fmci_spad_util_pkg") use "FMCI_SPAD_UTIL_PKG.ADA";
      for Spec ("fmci_spad_manager_pkg") use "FMCI_SPAD_MANAGER_PKG_.ADA";
      for Body ("fmci_spad_manager_pkg") use "FMCI_SPAD_MANAGER_PKG.ADA";
      for Spec ("fmci_pos_ref4_dpkg") use "FMCI_POS_REF4_DPKG_.ADA";
      for Body ("fmci_pos_ref4_dpkg") use "FMCI_POS_REF4_DPKG.ADA";
      for Spec ("fmci_message_util_pkg") use "FMCI_MESSAGE_UTIL_PKG_.ADA";
      for Body ("fmci_message_util_pkg") use "FMCI_MESSAGE_UTIL_PKG.ADA";
      for Spec ("fmci_message_iface_pkg") use "FMCI_MESSAGE_IFACE_PKG_.ADA";
      for Body ("fmci_message_iface_pkg") use "FMCI_MESSAGE_IFACE_PKG.ADA";
      for Spec ("fmci_memory_page_pkg") use "FMCI_MEMORY_PAGE_PKG_.ADA";
      for Body ("fmci_memory_page_pkg") use "FMCI_MEMORY_PAGE_PKG.ADA";
      for Body ("fmci_memory_page_pkg.dump_perf_data") use "FMCI_MEMORY_PAGE_DUMP_PERF_DATA.ADA";
      for Body ("fmci_memory_page_pkg.dietm") use "FMCI_MEMORY_PAGE_DIE_TM.ADA";
```

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

```
      for Body ("fmci_memory_page_pkg.dienav") use "FMCI_MEMORY_PAGE_DIE_NAV.ADA";
      for Body ("fmci_memory_page_pkg.diefm") use "FMCI_MEMORY_PAGE_DIE_FM.ADA";
      for Body ("fmci_memory_page_pkg.build_save") use "FMCI_MEMORY_PAGE_BUILD_SAVE.ADA";
      for Body ("fmci_memory_page_pkg.build_recall") use "FMCI_MEMORY_PAGE_BUILD_RECALL.ADA";
      for Body ("fmci_memory_page_pkg.build_part_number") use "FMCI_MEMORY_PAGE_BUILD_PART_NUMBER.ADA";
      for Body ("fmci_memory_page_pkg.build_mro_partition") use "FMCI_MEMORY_PAGE_BUILD_MRO_PARTITION.ADA";
      for Body ("fmci_memory_page_pkg.build_line") use "FMCI_MEMORY_PAGE_BUILD_LINE.ADA";
      for Body ("fmci_memory_page_pkg.build_clear_page") use "FMCI_MEMORY_PAGE_BUILD_CLEAR_PAGE.ADA";
      for Body ("fmci_memory_page_pkg.build_clear") use "FMCI_MEMORY_PAGE_BUILD_CLEAR_ALL.ADA";
      for Spec ("fmci_fm_memory_mem_dpkg") use "FMCI_FM_MEMORY_MEM_DPKG_.ADA";
      for Body ("fmci_fm_memory_mem_dpkg") use "FMCI_FM_MEMORY_MEM_DPKG.ADA";
      for Spec ("fmci_event_tpkg") use "FMCI_EVENT_TPKG_.ADA";
      for Body ("fmci_event_pkg.process_fm_event") use "FMCI_EVENT_PROCESS_FM_EVENT.ADA";
      for Body ("fmci_event_pkg.process_efis_event") use "FMCI_EVENT_PROCESS_EFIS_EVENT.ADA";
      for Body ("fmci_event_pkg.process_ci_event") use "FMCI_EVENT_PROCESS_CI_EVENT.ADA";
      for Spec ("fmci_event_pkg") use "FMCI_EVENT_PKG_.ADA";
      for Body ("fmci_event_pkg") use "FMCI_EVENT_PKG.ADA";
      for Spec ("fmci_event_in_pkg") use "FMCI_EVENT_IN_PKG_.ADA";
      for Body ("fmci_event_in_pkg") use "FMCI_EVENT_IN_PKG.ADA";
      for Spec ("fmci_event_in_dpkg") use "FMCI_EVENT_IN_DPKG_.ADA";
      for Body ("fmci_event_in_pkg.decoder_exception_processing") use "FMCI_EVENT_IN_DECODE_EXCEPTION.ADA";
      for Body ("fmci_event_in_pkg.decoder_event_processing") use "FMCI_EVENT_IN_DECODE_EVENT.ADA";
      for Spec ("fmci_error_code_dpkg") use "FMCI_ERROR_CODE_DPKG_.ADA";
      for Spec ("fmci_efis_center_spad_pkg") use "FMCI_EFIS_CENTER_SPAD_PKG_.ADA";
      for Body ("fmci_efis_center_spad_pkg") use "FMCI_EFIS_CENTER_SPAD_PKG.ADA";
      for Spec ("fmci_display_tpkg") use "FMCI_DISPLAY_TPKG_.ADA";
      for Spec ("fmci_display_pkg") use "FMCI_DISPLAY_PKG_.ADA";
      for Body ("fmci_display_pkg") use "FMCI_DISPLAY_PKG.ADA";
      for Spec ("fmci_display_dpkg") use "FMCI_DISPLAY_DPKG_.ADA";
      for Body ("fmci_display_dpkg") use "FMCI_DISPLAY_DPKG.ADA";
      for Body ("fmci_display_pkg.convert_page_data") use "FMCI_DISPLAY_CONVERT_PAGE_DATA.ADA";
      for Body ("fmci_display_pkg.build_page_data") use "FMCI_DISPLAY_BUILD_PAGE_DATA.ADA";
      for Spec ("fmci_cit_key_pkg") use "FMCI_CIT_KEY_PKG_.ADA";
      for Body ("fmci_cit_key_pkg") use "FMCI_CIT_KEY_PKG.ADA";
      for Spec ("fmci_bp_req_que_manager_pkg") use "FMCI_BP_REQ_QUE_MANAGER_PKG_.ADA";
      for Body ("fmci_bp_req_que_manager_pkg") use "FMCI_BP_REQ_QUE_MANAGER_PKG.ADA";
      for Body ("fmci_bp_req_que_manager_pkg.convert_widget_2_event") use "FMCI_BP_REQ_MANAGER_WIDGET_2_EVENT.ADA";
      for Body ("fmci_bp_req_que_manager_pkg.build_bp_event") use "FMCI_BP_REQ_MANAGER_BUILD_BP_EVENT.ADA";
      for Spec ("fmci_bite_dpkg") use "FMCI_BITE_TPKG_.ADA";
      for Spec ("flx_fm_register_navdb_pkg") use "FLX_FM_REGISTER_NAVDB_PKG_.ADA";
      for Body ("flx_fm_register_navdb_pkg") use "FLX_FM_REGISTER_NAVDB_PKG.ADA";
      for Spec ("eosid_types") use "EOSID_TYPES_.ADA";
```

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

```
      for Spec ("efis_waypoint_pkg") use "EFIS_WAYPOINT_PKG_.ADA";
      for Body ("efis_waypoint_pkg") use "EFIS_WAYPOINT_PKG.ADA";
      for Body ("efis_vsd_utilities_pkg.object_within_footprint") use "EFIS_VSD_UTILTIES_OBJ_FOOTPRINT_SEP.ADA";
      for Spec ("efis_vsd_utilities_pkg") use "EFIS_VSD_UTILITIES_PKG_.ADA";
      for Body ("efis_vsd_utilities_pkg") use "EFIS_VSD_UTILITIES_PKG.ADA";
      for Body ("efis_vsd_utilities_pkg.determine_path_or_track_mode") use "EFIS_VSD_UTILITIES_PATH_OR_TRACK_SEP.ADA";
      for Body ("efis_vsd_utilities_pkg.object_within_display_range") use "EFIS_VSD_UTILITIES_DISPLAY_RANGE_SEP.ADA";
      for Body ("efis_vsd_utilities_pkg.calculate_swath") use "EFIS_VSD_UTILITIES_CALC_SWATH_SEP.ADA";
      for Body ("efis_vsd_utilities_pkg.calculate_footprint") use "EFIS_VSD_UTILITIES_CALC_FOOTPRINT_SEP.ADA";
      for Body ("efis_vsd_pkg.process_waypoints") use "EFIS_VSD_PROCESS_WAYPOINTS_SEP.ADA";
      for Body ("efis_vsd_pkg.process_vnav_path") use "EFIS_VSD_PROCESS_VNAV_PATH_SEP.ADA";
      for Body ("efis_vsd_pkg.process_runway") use "EFIS_VSD_PROCESS_RUNWAY_SEP.ADA";
      for Body ("efis_vsd_pkg.process_missed_approach_point") use "EFIS_VSD_PROCESS_MISS_APP_POINT_SEP.ADA";
      for Body ("efis_vsd_pkg.process_reference_approach_vector_and_decision_gates") use
» "EFIS_VSD_PROCESS_DECISION_GATES_SEP.ADA";
      for Spec ("efis_vsd_pkg") use "EFIS_VSD_PKG_.ADA";
      for Body ("efis_vsd_pkg") use "EFIS_VSD_PKG.ADA";
      for Body ("efis_vsd_io_pkg.send_1hz_data") use "EFIS_VSD_IO_SEND_1HZ_DATA_SEP.ADA";
      for Body ("efis_vsd_io_pkg.send_10hz_data") use "EFIS_VSD_IO_SEND_10HZ_DATA_SEP.ADA";
      for Spec ("efis_vsd_io_pkg") use "EFIS_VSD_IO_PKG_.ADA";
      for Body ("efis_vsd_io_pkg") use "EFIS_VSD_IO_PKG.ADA";
      for Body ("efis_vsd_io_pkg.get_external_data") use "EFIS_VSD_IO_GET_DATA_SEP.ADA";
      for Spec ("efis_vsd_io_dpkg") use "EFIS_VSD_IO_DPKG_.ADA";
      for Body ("efis_vsd_pkg.get_waypoint_data_from_buffer") use "EFIS_VSD_GET_WPOINT_DATA_SEP.ADA";
      for Body ("efis_vsd_pkg.get_runway_data_from_header") use "EFIS_VSD_GET_RUNWAY_DATA_SEP.ADA";
      for Body ("efis_vsd_pkg.get_reference_approach_data_from_buffer") use "EFIS_VSD_GET_REF_APPROACH_DATA_SEP.ADA";
      for Body ("efis_vsd_pkg.get_missed_approach_point_data_from_buffer") use "EFIS_VSD_GET_MP_DATA_SEP.ADA";
      for Body ("efis_vsd_pkg.get_lgb_data") use "EFIS_VSD_GET_LGB_DATA_SEP.ADA";
      for Body ("efis_vsd_pkg.get_lgb_data_for_iss") use "EFIS_VSD_GET_LGB_DATA_FOR_ISS_SEP.ADA";
      for Spec ("efis_viewable_window_pkg") use "EFIS_VIEWABLE_WINDOW_PKG_.ADA";
      for Body ("efis_viewable_window_pkg") use "EFIS_VIEWABLE_WINDOW_PKG.ADA";
      for Spec ("efis_text_utilities_pkg") use "EFIS_TEXT_UTILITIES_PKG_.ADA";
      for Body ("efis_text_utilities_pkg") use "EFIS_TEXT_UTILITIES_PKG.ADA";
      for Spec ("efis_symbols_lines_tpkg") use "EFIS_SYMBOLS_LINES_TPKG_.ADA";
      for Spec ("efis_store_utilities_pkg") use "EFIS_STORE_UTILITIES_PKG_.ADA";
      for Body ("efis_store_utilities_pkg") use "EFIS_STORE_UTILITIES_PKG.ADA";
      for Spec ("efis_store_buff_pkg") use "EFIS_STORE_BUF_.ADA";
      for Body ("efis_store_buff_pkg") use "EFIS_STORE_BUF.ADA";
      for Body ("efis_special_data_pkg.process_tuned_navaids") use "EFIS_SPECIAL_TUNED_NAVAIDS_SEP.ADA";
      for Body ("efis_special_data_pkg.process_srp_points") use "EFIS_SPECIAL_SRP_POINTS_SEP.ADA";
      for Body ("efis_special_data_pkg.convert_saf32_to_char_degrees") use "EFIS_SPECIAL_SAF32_TO_CHAR_SEP.ADA";
      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)

```
      for Body ("efis_special_data_pkg.process_special_data") use "EFIS_SPECIAL_PROCESS_DATA_SEP.ADA";
      for Body ("efis_special_data_pkg.process_orig_dest_airports") use "EFIS_SPECIAL_ORIG_DEST_AIRPORTS_SEP.ADA";
      for Body ("efis_special_data_pkg.process_offpath_descent") use "EFIS_SPECIAL_OFFPATH_DESCENT_SEP.ADA";
      for Body ("efis_special_data_pkg.get_global_data") use "EFIS_SPECIAL_GLOBAL_DATA_SEP.ADA";
      for Spec ("efis_special_data_pkg") use "EFIS_SPECIAL_DATA_PKG_.ADA";
      for Body ("efis_special_data_pkg") use "EFIS_SPECIAL_DATA_PKG.ADA";
      for Body ("efis_special_data_pkg.update_alt_airports_org_des") use "EFIS_SPECIAL_ALT_ORG_DES_SEP.ADA";
      for Body ("efis_special_data_pkg.update_alt_aprt_waypoints") use "EFIS_SPECIAL_ALT_APRT_WAYPOINTS_SEP.ADA";
      for Body ("efis_special_data_pkg.update_alt_aprt_list") use "EFIS_SPECIAL_ALT_APRT_LIST_SEP.ADA";
      for Body ("efis_special_data_pkg.process_alternate_airports") use "EFIS_SPECIAL_ALTERNATE_AIRPORTS_SEP.ADA";
      for Spec ("efis_search_pkg") use "EFIS_SEARCH_PKG_.ADA";
      for Body ("efis_search_pkg") use "EFIS_SEARCH_PKG.ADA";
      for Spec ("efis_search_lfdata") use "EFIS_SEARCH_LFDATA_.ADA";
      for Spec ("efis_rwy_def_pkg") use "EFIS_RWY_DEF_PKG_.ADA";
      for Body ("efis_rwy_def_pkg") use "EFIS_RWY_DEF_PKG.ADA";
      for Spec ("efis_profile_point_pkg") use "EFIS_PROFILE_POINT_PKG_.ADA";
      for Body ("efis_profile_point_pkg") use "EFIS_PROFILE_POINT_PKG.ADA";
      for Body ("efis_profile_point_pkg.format_altitude_profile_points") use "EFIS_PROFILE_POINT_FORMAT_POINT_SEP.ADA";
      for Body ("efis_profile_point_pkg.efis_alt_prof_point_location") use "EFIS_PROFILE_POINT_EFIAPPLOC_SEP.ADA";
      for Body ("efis_profile_point_pkg.calculate_altitude_locations") use "EFIS_PROFILE_POINT_CAL_ALT_LOC_SEP.ADA";
      for Spec ("efis_proc_offscale_alternates_pkg") use "EFIS_PROC_OFFSCALE_ALTERNATES_PKG_.ADA";
      for Body ("efis_proc_offscale_alternates_pkg") use "EFIS_PROC_OFFSCALE_ALTERNATES_PKG.ADA";
      for Spec ("efis_path_protected_ifdata_pkg") use "EFIS_PATH_PROTECTED_IFDATA_.ADA";
      for Spec ("efis_path_perf_data_pkg") use "EFIS_PATH_PERF_DATA_PKG_.ADA";
      for Body ("efis_path_perf_data_pkg") use "EFIS_PATH_PERF_DATA_PKG.ADA";
      for Spec ("efis_path_lg_manager") use "EFIS_PATH_LG_MANAGER_.ADA";
      for Body ("efis_path_lg_manager") use "EFIS_PATH_LG_MANAGER.ADA";
      for Spec ("efis_path_lftypes") use "EFIS_PATH_LFTYPES_.ADA";
      for Body ("efis_path_exec_lt_pkg.efis_store_segments_into_lgb") use "EFIS_PATH_EXEC_STORE_SEP.ADA";
      for Body ("efis_path_exec_lt_pkg.efis_path_smooth_plan") use "EFIS_PATH_EXEC_SMOOTH_SEP.ADA";
      for Body ("efis_path_exec_lt_pkg.efis_path_setup") use "EFIS_PATH_EXEC_SETUP_SEP.ADA";
      for Body ("efis_path_exec_lt_pkg.efis_path_prov_setup") use "EFIS_PATH_EXEC_PROV_SEP.ADA";
      for Body ("efis_path_exec_lt_pkg.efis_path_ppos_hold_push_ahead") use "EFIS_PATH_EXEC_PPOS_SEP.ADA";
      for Body ("efis_path_exec_lt_pkg.efis_path_plan_select") use "EFIS_PATH_EXEC_PLN_SEL_SEP.ADA";
      for Spec ("efis_path_exec_lt_pkg") use "EFIS_PATH_EXEC_LT_PKG_.ADA";
      for Body ("efis_path_exec_lt_pkg") use "EFIS_PATH_EXEC_LT_PKG.ADA";
      for Body ("efis_path_exec_lt_pkg.efis_path_distance") use "EFIS_PATH_EXEC_DIST_SEP.ADA";
      for Body ("efis_path_exec_lt_pkg.efis_path_14k_update_holds") use "EFIS_PATH_EXEC_14K_UPDATE_SEP.ADA";
      for Spec ("efis_nd_10hz_sublayer_manager_pkg") use "EFIS_ND_10HZ_SUBLAYER_MANAGER_PKG_.ADA";
      for Body ("efis_nd_10hz_sublayer_manager_pkg") use "EFIS_ND_10HZ_SUBLAYER_MANAGER_PKG.ADA";
      for Spec ("efis_ndbdata_pkg") use "EFIS_NDBDATA_PKG_.ADA";
      for Body ("efis_ndbdata_pkg") use "EFIS_NDBDATA_PKG.ADA";
```

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

```
      for Body ("efis_ndbdata_pkg.ndb_wpt_or_cwpt_process") use "EFIS_NDBDATA_NDB_WPT_OR_CWP_SEP.ADA";
      for Body ("efis_ndbdata_pkg.ndb_ndrb_process") use "EFIS_NDBDATA_NDB_NDRB_SEP.ADA";
      for Body ("efis_ndbdata_pkg.ndb_nav_process") use "EFIS_NDBDATA_NDB_NAV_SEP.ADA";
      for Body ("efis_ndbdata_pkg.ndb_buffer_access_manager") use "EFIS_NDBDATA_NDB_BUFF_ACCESS_MGR_SEP.ADA";
      for Body ("efis_ndbdata_pkg.ndb_apt_process") use "EFIS_NDBDATA_NDB_APT_SEP.ADA";
      for Spec ("efis_map_req_pkg") use "EFIS_MAP_REQ_PKG_.ADA";
      for Body ("efis_map_req_pkg") use "EFIS_MAP_REQ_PKG.ADA";
      for Spec ("efis_map_parameter_lfdata") use "EFIS_MAP_PARAMETER_LFDATA_.ADA";
      for Spec ("efis_map_background_lfdata") use "EFIS_MAP_BACKGROUND_LFDATA_.ADA";
      for Spec ("efis_lt_wrapper_pkg") use "EFIS_LT_WRAPPER_PKG_.ADA";
      for Body ("efis_lt_wrapper_pkg") use "EFIS_LT_WRAPPER_PKG.ADA";
      for Spec ("efis_lt_segment_list_wrapper_pkg") use "EFIS_LT_SEGMENT_LIST_WRAPPER_PKG_.ADA";
      for Body ("efis_lt_segment_list_wrapper_pkg") use "EFIS_LT_SEGMENT_LIST_WRAPPER_PKG.ADA";
      for Spec ("efis_lt_leg_list_wrapper_pkg") use "EFIS_LT_LEG_LIST_WRAPPER_PKG_.ADA";
      for Body ("efis_lt_leg_list_wrapper_pkg") use "EFIS_LT_LEG_LIST_WRAPPER_PKG.ADA";
      for Spec ("efis_leg_seg_proc_pkg") use "EFIS_LEG_SEG_PROC_PKG_.ADA";
      for Body ("efis_leg_seg_proc_pkg") use "EFIS_LEG_SEG_PROC_PKG.ADA";
      for Body ("efis_leg_seg_proc_pkg.insert_hold_proc_symbols") use "EFIS_LEG_SEG_HOLD_PROC_SYMBOLS_SEP.ADA";
      for Body ("efis_leg_seg_proc_pkg.hold_proc_display_as_symbol") use "EFIS_LEG_SEG_DISPLAY_AS_SYMBOL_SEP.ADA";
      for Spec ("efis_iss_utilty_pkg") use "EFIS_ISS_UTILITY_PKG_.ADA";
      for Body ("efis_iss_utilty_pkg") use "EFIS_ISS_UTILITY_PKG.ADA";
      for Body ("efis_iss_traj_manager_pkg.refresh_events") use "EFIS_ISS_TRAJ_MG_PKG_REFRESH_EVENTS.ADA";
      for Body ("efis_iss_traj_manager_pkg.determine_traj_intent") use "EFIS_ISS_TRAJ_MG_PKG_DETERMINE_TRAJ.ADA";
      for Body ("efis_iss_traj_manager_pkg.process_iss_trajectory") use "EFIS_ISS_TRAJ_MANAGER_PKG_PRO_ISS_TRAJ.ADA";
      for Spec ("efis_iss_traj_manager_pkg") use "EFIS_ISS_TRAJ_MANAGER_PKG_.ADA";
      for Body ("efis_iss_traj_manager_pkg") use "EFIS_ISS_TRAJ_MANAGER_PKG.ADA";
      for Spec ("efis_iss_size_pkg") use "EFIS_ISS_SIZE_PKG_.ADA";
      for Spec ("efis_iss_output_tpkg") use "EFIS_ISS_OUTPUT_TPKG_.ADA";
      for Body ("efis_iss_output_pkg.determine_endpoint_type") use "EFIS_ISS_OUTPUT_PKG_DETER_ENDPT.ADA";
      for Spec ("efis_iss_output_pkg") use "EFIS_ISS_OUTPUT_PKG_.ADA";
      for Body ("efis_iss_output_pkg") use "EFIS_ISS_OUTPUT_PKG.ADA";
      for Spec ("efis_iss_output_dpkg") use "EFIS_ISS_OUTPUT_DPKG_.ADA";
      for Body ("efis_iss_output_dpkg") use "EFIS_ISS_OUTPUT_DPKG.ADA";
      for Spec ("efis_iss_iftypes") use "EFIS_ISS_IFTYPES_.ADA";
      for Spec ("efis_iss_dpkg") use "EFIS_ISS_DPKG_.ADA";
      for Spec ("efis_io_iftypes") use "EFIS_IO_IFTYPES_.ADA";
      for Spec ("efis_init_pkg") use "EFIS_INIT_PKG_.ADA";
      for Body ("efis_init_pkg") use "EFIS_INIT_PKG.ADA";
      for Spec ("efis_get_acstate_data_pkg") use "EFIS_GET_ACSTATE_DATA_PKG_.ADA";
      for Body ("efis_get_acstate_data_pkg") use "EFIS_GET_ACSTATE_DATA_PKG.ADA";
      for Spec ("efis_generic_pkg") use "EFIS_GENERIC_PKG_.ADA";
      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)

```
      for Spec ("efis_flight_data_pkg") use "EFIS_FLIGHT_DATA_PKG_.ADA";
      for Body ("efis_flight_data_pkg") use "EFIS_FLIGHT_DATA_PKG.ADA";
      for Body ("efis_flight_data_pkg.find_inactive_legs") use "EFIS_FLIGHT_DATA_INACTIVE_SEP.ADA";
      for Body ("efis_flight_data_pkg.find_active_legs") use "EFIS_FLIGHT_DATA_ACTIVE_SEP.ADA";
      for Body ("efis_flight_data_pkg.find_active_legs.draw_swath") use "EFIS_FLIGHT_DATA_ACTIVE_DRAW_SWATH_SEP.ADA";
      for Spec ("efis_ext_interfaces_pkg") use "EFIS_EXT_INTERFACES_PKG_.ADA";
      for Body ("efis_ext_interfaces_pkg") use "EFIS_EXT_INTERFACES_PKG.ADA";
      for Body ("efis_edit_pkg.determine_edit_area") use "EFIS_EDIT_PKG_DET_EDIT_AREA.ADA";
      for Body ("efis_edit_pkg.determine_corner_points") use "EFIS_EDIT_PKG_DET_CORNER_POINTS.ADA";
      for Body ("efis_edit_pkg.define_window") use "EFIS_EDIT_PKG_DEFINE_WINDOW.ADA";
      for Spec ("efis_edit_pkg") use "EFIS_EDIT_PKG_.ADA";
      for Body ("efis_edit_pkg") use "EFIS_EDIT_PKG.ADA";
      for Spec ("efis_dynamic_data_pkg") use "EFIS_DYNAMIC_DATA_PKG_.ADA";
      for Body ("efis_dynamic_data_pkg") use "EFIS_DYNAMIC_DATA_PKG.ADA";
      for Spec ("efis_duplicate_utility_pkg") use "EFIS_DUPLICATE_UTILITY_PKG_.ADA";
      for Body ("efis_duplicate_utility_pkg") use "EFIS_DUPLICATE_UTILITY_PKG.ADA";
      for Spec ("efis_debug_control_pkg") use "EFIS_DEBUG_CONTROL_PKG_.ADA";
      for Spec ("efis_convert_pkg") use "EFIS_CONVERT_PKG_.ADA";
      for Body ("efis_convert_pkg") use "EFIS_CONVERT_PKG.ADA";
      for Spec ("efis_ci_interface_mgr_pkg") use "EFIS_CI_INTERFACE_MGR_PKG_.ADA";
      for Body ("efis_ci_interface_mgr_pkg") use "EFIS_CI_INTERFACE_MGR_PKG.ADA";
      for Body ("efis_bkgn_buff_cntrl_pkg.process_nd_optional_layer") use "EFIS_BKGN_BUFF_CNTRL_PROC_OPT_SEP.ADA";
      for Body ("efis_bkgn_buff_cntrl_pkg.process_minimap") use "EFIS_BKGN_BUFF_CNTRL_PROC_MINI_SEP.ADA";
      for Body ("efis_bkgn_buff_cntrl_pkg.process_build_map") use "EFIS_BKGN_BUFF_CNTRL_PROC_MAP_SEP.ADA";
      for Spec ("efis_bkgn_buff_cntrl_pkg") use "EFIS_BKGN_BUFF_CNTRL_PKG_.ADA";
      for Body ("efis_bkgn_buff_cntrl_pkg") use "EFIS_BKGN_BUFF_CNTRL_PKG.ADA";
      for Body ("efis_bkgn_buff_cntrl_pkg.map_edit_area") use "EFIS_BKGN_BUFF_CNTRL_MAP_AREA_SEP.ADA";
      for Body ("efis_bkgn_buff_cntrl_pkg.build_background_buffer") use "EFIS_BKGN_BUFF_CNTRL_BUILD_MAP_SEP.ADA";
      for Spec ("efis_base_lftypes") use "EFIS_BASE_LFTYPES_.ADA";
      for Spec ("efis_661_vsd_layer_pkg") use "EFIS_661_VSD_LAYER_PKG_.ADA";
      for Body ("efis_661_vsd_layer_pkg") use "EFIS_661_VSD_LAYER_PKG.ADA";
      for Spec ("efis_661_nd_opt_layer_pkg") use "EFIS_661_ND_OPT_LAYER_PKG_.ADA";
      for Body ("efis_661_nd_opt_layer_pkg") use "EFIS_661_ND_OPT_LAYER_PKG.ADA";
      for Spec ("efis_661_nd_layer_pkg") use "EFIS_661_ND_LAYER_PKG_.ADA";
      for Body ("efis_661_nd_layer_pkg") use "EFIS_661_ND_LAYER_PKG.ADA";
      for Spec ("efis_661_nd_10hz_sublayer_pkg") use "EFIS_661_ND_10HZ_SUBLAYER_PKG_.ADA";
      for Body ("efis_661_nd_10hz_sublayer_pkg") use "EFIS_661_ND_10HZ_SUBLAYER_PKG.ADA";
      for Spec ("efis_661_minimap_layer_pkg") use "EFIS_661_MINIMAP_LAYER_PKG_.ADA";
      for Body ("efis_661_minimap_layer_pkg") use "EFIS_661_MINIMAP_LAYER_PKG.ADA";
      for Spec ("efis_661_iftypes") use "EFIS_661_IFTYPES_.ADA";
      for Spec ("efis_661_ifdata") use "EFIS_661_IFDATA_.ADA";
      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)

```
      for Body ("earth_center_cood_math_pkg.convert_to_earth_center") use "EARTH_CENTER_COOD_MATH_PKG_TO_EAR_CTR.ADA";
      for Body ("earth_center_cood_math_pkg.size_to_earth_radius") use "EARTH_CENTER_COOD_MATH_PKG_SIZE_TO.ADA";
      for Body ("earth_center_cood_math_pkg.rotate_and_shift") use "EARTH_CENTER_COOD_MATH_PKG_ROT_SHIFT.ADA";
      for Body ("earth_center_cood_math_pkg.nm_ft") use "EARTH_CENTER_COOD_MATH_PKG_NM_FT.ADA";
      for Body ("earth_center_cood_math_pkg.magnitude") use "EARTH_CENTER_COOD_MATH_PKG_MAGNITUDE.ADA";
      for Body ("earth_center_cood_math_pkg.intersection_point_of_two_lines") use "EARTH_CENTER_COOD_MATH_PKG_INTRSCT.ADA";
      for Body ("earth_center_cood_math_pkg.convert_ft_nm") use "EARTH_CENTER_COOD_MATH_PKG_FT_TO_NM.ADA";
      for Body ("earth_center_cood_math_pkg.local_earth_radius") use "EARTH_CENTER_COOD_MATH_PKG_EARTH_RAD.ADA";
      for Body ("earth_center_cood_math_pkg.dot_product") use "EARTH_CENTER_COOD_MATH_PKG_DOT_PRODUCT.ADA";
      for Body ("earth_center_cood_math_pkg.center_point_of_curve") use "EARTH_CENTER_COOD_MATH_PKG_CURVE_CTR.ADA";
      for Body ("earth_center_cood_math_pkg.cross_product") use "EARTH_CENTER_COOD_MATH_PKG_CROSS_PRD.ADA";
      for Spec ("earth_center_cood_math_pkg") use "EARTH_CENTER_COOD_MATH_PKG_.ADA";
      for Body ("earth_center_cood_math_pkg") use "EARTH_CENTER_COOD_MATH_PKG.ADA";
      for Spec ("descent_path_mgr") use "DESCENT_PATH_MGR_.ADA";
      for Body ("descent_path_mgr") use "DESCENT_PATH_MGR.ADA";
      for Spec ("cut_takeoff_vspeeds_check_pkg") use "CUT_TAKEOFF_VSPEEDS_CHECK_PKG_.ADA";
      for Body ("cut_takeoff_vspeeds_check_pkg") use "CUT_TAKEOFF_VSPEEDS_CHECK_PKG.ADA";
      for Spec ("cut_takeoff_ref_states_pkg") use "CUT_TAKEOFF_REF_STATES_PKG_.ADA";
      for Body ("cut_takeoff_ref_states_pkg") use "CUT_TAKEOFF_REF_STATES_PKG.ADA";
      for Spec ("cut_string_pkg") use "CUT_STRING_PKG_.ADA";
      for Body ("cut_string_pkg") use "CUT_STRING_PKG.ADA";
      for Spec ("cut_string_conversion_pkg") use "CUT_STRING_CONVERSION_PKG_.ADA";
      for Body ("cut_string_conversion_pkg") use "CUT_STRING_CONVERSION_PKG.ADA";
      for Spec ("cut_std_inclusion_pkg") use "CUT_STD_INCLUSION_PKG_.ADA";
      for Spec ("cut_rte_util_pkg") use "CUT_RTE_UTIL_PKG_.ADA";
      for Body ("cut_rte_util_pkg") use "CUT_RTE_UTIL_PKG.ADA";
      for Spec ("cut_key_conversion_pkg") use "CUT_KEY_CONVERSION_PKG_.ADA";
      for Body ("cut_key_conversion_pkg") use "CUT_KEY_CONVERSION_PKG.ADA";
      for Spec ("cut_gen_inclusion_pkg") use "CUT_GEN_INCLUSION_PKG_.ADA";
      for Body ("cut_gen_inclusion_pkg") use "CUT_GEN_INCLUSION_PKG.ADA";
      for Spec ("cut_datalink_pkg") use "CUT_DATALINK_PKG_.ADA";
      for Body ("cut_datalink_pkg") use "CUT_DATALINK_PKG.ADA";
      for Spec ("cut_cpi_operation_pkg") use "CUT_CPI_OPERATION_PKG_.ADA";
      for Body ("cut_cpi_operation_pkg") use "CUT_CPI_OPERATION_PKG.ADA";
      for Body ("common_lgb.putlgbleg") use "COMMON_LGB_PUTLGBLEG.ADA";
      for Body ("common_lgb.putlgbhdr") use "COMMON_LGB_PUTLGBHDR.ADA";
      for Spec ("common_lgb_int_nonresync_dpkg") use "COMMON_LGB_INT_NONRESYNC_DPKG_.ADA";
      for Body ("common_lgb.initlgb") use "COMMON_LGB_INITLGB.ADA";
      for Body ("common_lgb.getlgbleg") use "COMMON_LGB_GETLGBLEG.ADA";
      for Body ("common_lgb.getlgbhdr") use "COMMON_LGB_GETLGBHDR.ADA";
      for Body ("common_lgb.convert_leg_data") use "COMMON_LGB_CONVERT_LEG_DATA.ADA";
      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_dbgettype_pkg") use "CND_DBGETTYPE_PKG_.ADA";
1492  1491        for Body ("cnd_dbgettype_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)

```
      for Body ("cky_rte_legs_common_pkg.process_speed_altitude") use "CKY_RTE_LEGS_COMMON_PKG_PROC_SPALT.ADA";
      for Spec ("cky_rte_legs_common_pkg") use "CKY_RTE_LEGS_COMMON_PKG_.ADA";
      for Body ("cky_rte_legs_common_pkg") use "CKY_RTE_LEGS_COMMON_PKG.ADA";
      for Spec ("cky_rte_data_page_pkg") use "CKY_RTE_DATA_PAGE_PKG_.ADA";
      for Body ("cky_rte_data_page_pkg") use "CKY_RTE_DATA_PAGE_PKG.ADA";
      for Spec ("cky_rte_common_pkg") use "CKY_RTE_COMMON_PKG_.ADA";
      for Body ("cky_rte_common_pkg") use "CKY_RTE_COMMON_PKG.ADA";
      for Spec ("cky_ref_nav_data_page_pkg") use "CKY_REF_NAV_DATA_PAGE_PKG_.ADA";
      for Body ("cky_ref_nav_data_page_pkg") use "CKY_REF_NAV_DATA_PAGE_PKG.ADA";
      for Spec ("cky_prov_request_pkg") use "CKY_PROV_REQUEST_PKG_.ADA";
      for Body ("cky_prov_request_pkg") use "CKY_PROV_REQUEST_PKG.ADA";
      for Spec ("cky_progress_page_4_pkg") use "CKY_PROGRESS_PAGE_4_PKG_.ADA";
      for Body ("cky_progress_page_4_pkg") use "CKY_PROGRESS_PAGE_4_PKG.ADA";
      for Spec ("cky_progress_page_3_pkg") use "CKY_PROGRESS_PAGE_3_PKG_.ADA";
      for Body ("cky_progress_page_3_pkg") use "CKY_PROGRESS_PAGE_3_PKG.ADA";
      for Spec ("cky_progress_page_2_pkg") use "CKY_PROGRESS_PAGE_2_PKG_.ADA";
      for Body ("cky_progress_page_2_pkg") use "CKY_PROGRESS_PAGE_2_PKG.ADA";
      for Spec ("cky_progress_page_1_pkg") use "CKY_PROGRESS_PAGE_1_PKG_.ADA";
      for Body ("cky_progress_page_1_pkg") use "CKY_PROGRESS_PAGE_1_PKG.ADA";
      for Spec ("cky_pos_report_page_pkg") use "CKY_POS_REPORT_PAGE_PKG_.ADA";
      for Body ("cky_pos_report_page_pkg") use "CKY_POS_REPORT_PAGE_PKG.ADA";
      for Spec ("cky_pos_ref_page_4_4_pkg") use "CKY_POS_REF_PAGE_4_4_PKG_.ADA";
      for Body ("cky_pos_ref_page_4_4_pkg") use "CKY_POS_REF_PAGE_4_4_PKG.ADA";
      for Spec ("cky_pos_ref_page_3_4_pkg") use "CKY_POS_REF_PAGE_3_4_PKG_.ADA";
      for Body ("cky_pos_ref_page_3_4_pkg") use "CKY_POS_REF_PAGE_3_4_PKG.ADA";
      for Spec ("cky_pos_ref_page_2_4_pkg") use "CKY_POS_REF_PAGE_2_4_PKG_.ADA";
      for Body ("cky_pos_ref_page_2_4_pkg") use "CKY_POS_REF_PAGE_2_4_PKG.ADA";
      for Spec ("cky_pos_init_page_pkg") use "CKY_POS_INIT_PAGE_PKG_.ADA";
      for Body ("cky_pos_init_page_pkg") use "CKY_POS_INIT_PAGE_PKG.ADA";
      for Spec ("cky_post_flight_plan_pkg") use "CKY_POST_FLIGHT_PLAN_PKG_.ADA";
      for Body ("cky_post_flight_plan_pkg") use "CKY_POST_FLIGHT_PLAN_PKG.ADA";
      for Spec ("cky_perf_init_page_pkg") use "CKY_PERF_INIT_PAGE_PKG_.ADA";
      for Body ("cky_perf_init_page_pkg") use "CKY_PERF_INIT_PAGE_PKG.ADA";
      for Spec ("cky_offpath_des_page_pkg") use "CKY_OFFPATH_DES_PAGE_PKG_.ADA";
      for Body ("cky_offpath_des_page_pkg") use "CKY_OFFPATH_DES_PAGE_PKG.ADA";
      for Spec ("cky_nav_radio_page_pkg") use "CKY_NAV_RADIO_PAGE_PKG_.ADA";
      for Body ("cky_nav_radio_page_pkg") use "CKY_NAV_RADIO_PAGE_PKG.ADA";
      for Spec ("cky_mode_keys_pkg") use "CKY_MODE_KEYS_PKG_.ADA";
      for Body ("cky_mode_keys_pkg") use "CKY_MODE_KEYS_PKG.ADA";
      for Spec ("cky_maint_index_page_pkg") use "CKY_MAINT_INDEX_PAGE_PKG_.ADA";
      for Body ("cky_maint_index_page_pkg") use "CKY_MAINT_INDEX_PAGE_PKG.ADA";
      for Spec ("cky_key_pkg") use "CKY_KEY_PKG_.ADA";
```

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

```
        for Body ("cky_key_pkg") use "CKY_KEY_PKG.ADA";
        for Spec ("cky_irs_mon_page_pkg") use "CKY_IRS_MON_PAGE_PKG_.ADA";
        for Body ("cky_irs_mon_page_pkg") use "CKY_IRS_MON_PAGE_PKG.ADA";
        for Spec ("cky_init_ref_index_page_pkg") use "CKY_INIT_REF_INDEX_PAGE_PKG_.ADA";
        for Body ("cky_init_ref_index_page_pkg") use "CKY_INIT_REF_INDEX_PAGE_PKG.ADA";
        for Spec ("cky_ident_page_pkg") use "CKY_IDENT_PAGE_PKG_.ADA";
        for Body ("cky_ident_page_pkg") use "CKY_IDENT_PAGE_PKG.ADA";
        for Spec ("cky_hold_page_pkg") use "CKY_HOLD_PAGE_PKG_.ADA";
        for Body ("cky_hold_page_pkg") use "CKY_HOLD_PAGE_PKG.ADA";
        for Spec ("cky_hold_at_page_pkg") use "CKY_HOLD_AT_PAGE_PKG_.ADA";
        for Body ("cky_hold_at_page_pkg") use "CKY_HOLD_AT_PAGE_PKG.ADA";
        for Spec ("cky_fuel_weight_entry_pkg") use "CKY_FUEL_WEIGHT_ENTRY_PKG_.ADA";
        for Body ("cky_fuel_weight_entry_pkg") use "CKY_FUEL_WEIGHT_ENTRY_PKG.ADA";
        for Spec ("cky_fmc_comm_page_pkg") use "CKY_FMC_COMM_PAGE_PKG_.ADA";
        for Body ("cky_fmc_comm_page_pkg") use "CKY_FMC_COMM_PAGE_PKG.ADA";
        for Body ("cky_fix_info_page_pkg.process_1l_entry") use "CKY_FIX_INFO_PAGE_PKG_PROC_1L_ENT.ADA";
        for Spec ("cky_fix_info_page_pkg") use "CKY_FIX_INFO_PAGE_PKG_.ADA";
        for Body ("cky_fix_info_page_pkg") use "CKY_FIX_INFO_PAGE_PKG.ADA";
        for Body ("cky_fix_entry_pkg.verify_runway") use "CKY_FIX_ENTRY_PKG__VERIFY_RUNWAY.ADA";
        for Body ("cky_fix_entry_pkg.verify_pb") use "CKY_FIX_ENTRY_PKG__VERIFY_PB.ADA";
        for Body ("cky_fix_entry_pkg.search_ndb") use "CKY_FIX_ENTRY_PKG__SEARCH_NDB.ADA";
        for Body ("cky_fix_entry_pkg.request_pb_pb") use "CKY_FIX_ENTRY_PKG__REQUEST_PB_PB.ADA";
        for Body ("cky_fix_entry_pkg.request_pbd") use "CKY_FIX_ENTRY_PKG__REQUEST_PBD.ADA";
        for Body ("cky_fix_entry_pkg.request_ndb") use "CKY_FIX_ENTRY_PKG__REQUEST_NDB.ADA";
        for Body ("cky_fix_entry_pkg.request_leg") use "CKY_FIX_ENTRY_PKG__REQUEST_LEG.ADA";
        for Body ("cky_fix_entry_pkg.get_ndb") use "CKY_FIX_ENTRY_PKG__GET_NDB.ADA";
        for Body ("cky_fix_entry_pkg.get_bearings") use "CKY_FIX_ENTRY_PKG__GET_BEARINGS.ADA";
        for Body ("cky_fix_entry_pkg.check_xing_pt") use "CKY_FIX_ENTRY_PKG__CHECK_XING_PT.ADA";
        for Body ("cky_fix_entry_pkg.check_runway") use "CKY_FIX_ENTRY_PKG__CHECK_RUNWAY.ADA";
        for Body ("cky_fix_entry_pkg.check_pb_term") use "CKY_FIX_ENTRY_PKG__CHECK_PB_TERM.ADA";
        for Body ("cky_fix_entry_pkg.check_pb_pb") use "CKY_FIX_ENTRY_PKG__CHECK_PB_PB.ADA";
        for Body ("cky_fix_entry_pkg.check_pbd") use "CKY_FIX_ENTRY_PKG__CHECK_PBD.ADA";
        for Body ("cky_fix_entry_pkg.check_pb") use "CKY_FIX_ENTRY_PKG__CHECK_PB.ADA";
        for Body ("cky_fix_entry_pkg.check_ndb") use "CKY_FIX_ENTRY_PKG__CHECK_NDB.ADA";
        for Body ("cky_fix_entry_pkg.check_fp") use "CKY_FIX_ENTRY_PKG__CHECK_FP.ADA";
        for Body ("cky_fix_entry_pkg.check_fix") use "CKY_FIX_ENTRY_PKG__CHECK_FIX.ADA";
        for Body ("cky_fix_entry_pkg.check_atw") use "CKY_FIX_ENTRY_PKG__CHECK_ATW.ADA";
        for Body ("cky_fix_entry_pkg.check_airway_term") use "CKY_FIX_ENTRY_PKG__CHECK_AIRWAY_TERM.ADA";
        for Spec ("cky_fix_entry_pkg") use "CKY_FIX_ENTRY_PKG_.ADA";
        for Body ("cky_fix_entry_pkg") use "CKY_FIX_ENTRY_PKG.ADA";
        for Spec ("cky_field_util_pkg") use "CKY_FIELD_UTIL_PKG_.ADA";
        for Body ("cky_field_util_pkg") use "CKY_FIELD_UTIL_PKG.ADA";
```

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

```
      for Spec ("cky_determine_rte_type_pkg") use "CKY_DETERMINE_RTE_TYPE_PKG_.ADA";
      for Body ("cky_determine_rte_type_pkg") use "CKY_DETERMINE_RTE_TYPE_PKG.ADA";
      for Body ("cky_des_page_pkg.process_3r") use "CKY_DES_PAGE_PKG__PROCESS_3R.ADA";
      for Spec ("cky_des_page_pkg") use "CKY_DES_PAGE_PKG_.ADA";
      for Body ("cky_des_page_pkg") use "CKY_DES_PAGE_PKG.ADA";
      for Spec ("cky_des_forecast_page_pkg") use "CKY_DES_FORECAST_PAGE_PKG_.ADA";
      for Body ("cky_des_forecast_page_pkg") use "CKY_DES_FORECAST_PAGE_PKG.ADA";
      for Spec ("cky_dep_arr_index_page_pkg") use "CKY_DEP_ARR_INDEX_PAGE_PKG_.ADA";
      for Body ("cky_dep_arr_index_page_pkg") use "CKY_DEP_ARR_INDEX_PAGE_PKG.ADA";
      for Spec ("cky_dep_arr_common_pkg") use "CKY_DEP_ARR_COMMON_PKG_.ADA";
      for Body ("cky_dep_arr_common_pkg") use "CKY_DEP_ARR_COMMON_PKG.ADA";
      for Spec ("cky_departures_page_pkg") use "CKY_DEPARTURES_PAGE_PKG_.ADA";
      for Body ("cky_departures_page_pkg") use "CKY_DEPARTURES_PAGE_PKG.ADA";
      for Spec ("cky_crz_page_pkg") use "CKY_CRZ_PAGE_PKG_.ADA";
      for Body ("cky_crz_page_pkg") use "CKY_CRZ_PAGE_PKG.ADA";
      for Spec ("cky_constants_pkg") use "CKY_CONSTANTS_PKG_.ADA";
      for Spec ("cky_clb_page_pkg") use "CKY_CLB_PAGE_PKG_.ADA";
      for Body ("cky_clb_page_pkg") use "CKY_CLB_PAGE_PKG.ADA";
      for Spec ("cky_arrivals_page_pkg") use "CKY_ARRIVALS_PAGE_PKG_.ADA";
      for Body ("cky_arrivals_page_pkg") use "CKY_ARRIVALS_PAGE_PKG.ADA";
      for Spec ("cky_approach_ref_page_pkg") use "CKY_APPROACH_REF_PAGE_PKG_.ADA";
      for Body ("cky_approach_ref_page_pkg") use "CKY_APPROACH_REF_PAGE_PKG.ADA";
      for Spec ("cky_apf_page_2_pkg") use "CKY_APF_PAGE_2_PKG_.ADA";
      for Body ("cky_apf_page_2_pkg") use "CKY_APF_PAGE_2_PKG.ADA";
      for Spec ("cky_apf_page_1_pkg") use "CKY_APF_PAGE_1_PKG_.ADA";
      for Body ("cky_apf_page_1_pkg") use "CKY_APF_PAGE_1_PKG.ADA";
      for Spec ("cky_altn_plan_page_pkg") use "CKY_ALTN_PLAN_PAGE_PKG_.ADA";
      for Body ("cky_altn_plan_page_pkg") use "CKY_ALTN_PLAN_PAGE_PKG.ADA";
      for Spec ("cky_altn_page_pkg") use "CKY_ALTN_PAGE_PKG_.ADA";
      for Body ("cky_altn_page_pkg") use "CKY_ALTN_PAGE_PKG.ADA";
      for Spec ("cky_altn_page_access_pkg") use "CKY_ALTN_PAGE_ACCESS_PKG_.ADA";
      for Body ("cky_altn_page_access_pkg") use "CKY_ALTN_PAGE_ACCESS_PKG.ADA";
      for Spec ("cky_altn_list_page_pkg") use "CKY_ALTN_LIST_PAGE_PKG_.ADA";
      for Body ("cky_altn_list_page_pkg") use "CKY_ALTN_LIST_PAGE_PKG.ADA";
      for Spec ("cfp_vert_ltypes") use "CFP_VERT_LTYPES_.ADA";
      for Spec ("cfp_vert_ldata") use "CFP_VERT_LDATA_.ADA";
      for Spec ("cfp_utils") use "CFP_UTILS_.ADA";
      for Body ("cfp_utils") use "CFP_UTILS.ADA";
      for Spec ("deactivate_fpln") use "CFP_STDBY_DEACTIVATE_.ADA";
      for Body ("deactivate_fpln") use "CFP_STDBY_DEACTIVATE.ADA";
      for Spec ("clear_provisional") use "CFP_STDBY_CLEARPROV_.ADA";
      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_iftypes") 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)

```
      for Spec ("cfp_fpoktodelete") use "CFP_FPOKTODELETE_.ADA";
      for Body ("cfp_fpoktodelete") use "CFP_FPOKTODELETE.ADA";
      for Spec ("cfp_fpoffset_pkg") use "CFP_FPOFFSET_PKG_.ADA";
      for Body ("cfp_fpoffset_pkg") use "CFP_FPOFFSET_PKG.ADA";
      for Spec ("cfp_fpnewalt_pkg") use "CFP_FPNEWALT_PKG_.ADA";
      for Body ("cfp_fpnewalt_pkg") use "CFP_FPNEWALT_PKG.ADA";
      for Spec ("cfp_fpholdtime") use "CFP_FPHOLDTIME_.ADA";
      for Body ("cfp_fpholdtime") use "CFP_FPHOLDTIME.ADA";
      for Spec ("cfp_fpexec_vert") use "CFP_FPEXEC_VERT_.ADA";
      for Body ("cfp_fpexec_vert") use "CFP_FPEXEC_VERT.ADA";
      for Spec ("cfp_fpexecute") use "CFP_FPEXECUTE_.ADA";
      for Body ("cfp_fpexecute") use "CFP_FPEXECUTE.ADA";
      for Spec ("cfp_fpefis") use "CFP_FPEFIS_.ADA";
      for Body ("cfp_fpefis") use "CFP_FPEFIS.ADA";
      for Spec ("cfp_fpdiscon") use "CFP_FPDISCON_.ADA";
      for Body ("cfp_fpdiscon") use "CFP_FPDISCON.ADA";
      for Spec ("cfp_fpdeparr_pkg") use "CFP_FPDEPARR_PKG_.ADA";
      for Body ("cfp_fpdeparr_pkg") use "CFP_FPDEPARR_PKG.ADA";
      for Spec ("cfp_fpdelvialeg") use "CFP_FPDELVIALEG_.ADA";
      for Body ("cfp_fpdelvialeg") use "CFP_FPDELVIALEG.ADA";
      for Spec ("cfp_fpdelete") use "CFP_FPDELETE_.ADA";
      for Body ("cfp_fpdelete") use "CFP_FPDELETE.ADA";
      for Spec ("cfp_fpckvprofile") use "CFP_FPCKVPROFILE_.ADA";
      for Body ("cfp_fpckvprofile") use "CFP_FPCKVPROFILE.ADA";
      for Spec ("cfp_fpbldvia") use "CFP_FPBLDVIA_.ADA";
      for Body ("cfp_fpbldvia") use "CFP_FPBLDVIA.ADA";
      for Spec ("cfp_fpbldvfrleg") use "CFP_FPBLDVFRLEG_.ADA";
      for Body ("cfp_fpbldvfrleg") use "CFP_FPBLDVFRLEG.ADA";
      for Spec ("cfp_fpbldrxleg") use "CFP_FPBLDRXLEG_.ADA";
      for Body ("cfp_fpbldrxleg") use "CFP_FPBLDRXLEG.ADA";
      for Spec ("cfp_fpblddefvert_pkg") use "CFP_FPBLDDEFVERT_PKG_.ADA";
      for Body ("cfp_fpblddefvert_pkg") use "CFP_FPBLDDEFVERT_PKG.ADA";
      for Spec ("cfp_fpbldclb") use "CFP_FPBLDCLB_.ADA";
      for Body ("cfp_fpbldclb") use "CFP_FPBLDCLB.ADA";
      for Spec ("cfp_fpbldapt") use "CFP_FPBLDAPT_.ADA";
      for Body ("cfp_fpbldapt") use "CFP_FPBLDAPT.ADA";
      for Spec ("cfp_fpawysrch_pkg") use "CFP_FPAWYSRCH_PKG_.ADA";
      for Body ("cfp_fpawysrch_pkg") use "CFP_FPAWYSRCH_PKG.ADA";
      for Spec ("cfp_fpawyintc_pkg") use "CFP_FPAWYINTC_PKG_.ADA";
      for Body ("cfp_fpawyintc_pkg") use "CFP_FPAWYINTC_PKG.ADA";
      for Spec ("cfp_fpaltdelete") use "CFP_FPALTDELETE_.ADA";
      for Body ("cfp_fpaltdelete") use "CFP_FPALTDELETE.ADA";
```

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

```
      for Spec ("cfp_fpaltcnstr") use "CFP_FPALTCNSTR_.ADA";
      for Body ("cfp_fpaltcnstr") use "CFP_FPALTCNSTR.ADA";
      for Spec ("cfp_fpactvvert") use "CFP_FPACTVVERT_.ADA";
      for Body ("cfp_fpactvvert") use "CFP_FPACTVVERT.ADA";
      for Spec ("cfp_fpactivate") use "CFP_FPACTIVATE_.ADA";
      for Body ("cfp_fpactivate") use "CFP_FPACTIVATE.ADA";
      for Spec ("cfp_fpabeampts") use "CFP_FPABEAMPTS_.ADA";
      for Body ("cfp_fpabeampts") use "CFP_FPABEAMPTS.ADA";
      for Spec ("cfp_flight_complete_pkg") use "CFP_FLIGHT_COMPLETE_PKG_.ADA";
      for Body ("cfp_flight_complete_pkg") use "CFP_FLIGHT_COMPLETE_PKG.ADA";
      for Spec ("cfp_findllxingpt_pkg") use "CFP_FINDLLXINGPT_PKG_.ADA";
      for Body ("cfp_findllxingpt_pkg") use "CFP_FINDLLXINGPT_PKG.ADA";
      for Spec ("cfp_delete_act_fpln") use "CFP_DELETE_ACT_FPLN_.ADA";
      for Body ("cfp_delete_act_fpln") use "CFP_DELETE_ACT_FPLN.ADA";
      for Spec ("cfp_createprov") use "CFP_CREATEPROV_.ADA";
      for Body ("cfp_createprov") use "CFP_CREATEPROV.ADA";
      for Spec ("cfp_co_route_pkg") use "CFP_CO_ROUTE_PKG_.ADA";
      for Body ("cfp_co_route_pkg") use "CFP_CO_ROUTE_PKG.ADA";
      for Spec ("cfp_computeabm") use "CFP_COMPUTEABM_.ADA";
      for Body ("cfp_computeabm") use "CFP_COMPUTEABM.ADA";
      for Spec ("cfp_clearrtes") use "CFP_CLEARRTES_.ADA";
      for Body ("cfp_clearrtes") use "CFP_CLEARRTES.ADA";
      for Spec ("cfp_clearprov") use "CFP_CLEARPROV_.ADA";
      for Body ("cfp_clearprov") use "CFP_CLEARPROV.ADA";
      for Spec ("cfp_chkoffdes") use "CFP_CHKOFFDES_.ADA";
      for Body ("cfp_chkoffdes") use "CFP_CHKOFFDES.ADA";
      for Body ("cfp_cdurtedist") use "CFP_CDURTEDIST.ADA";
      for Spec ("cfp_cdufp_pkg") use "CFP_CDUFP_PKG_.ADA";
      for Body ("cfp_cdufp_pkg") use "CFP_CDUFP_PKG.ADA";
      for Spec ("cfp_cdualtintv_pkg") use "CFP_CDUALTINTV_PKG_.ADA";
      for Body ("cfp_cdualtintv_pkg") use "CFP_CDUALTINTV_PKG.ADA";
      for Spec ("cfp_calcintrsect_pkg") use "CFP_CALCINTRSECT_PKG_.ADA";
      for Body ("cfp_calcintrsect_pkg") use "CFP_CALCINTRSECT_PKG.ADA";
      for Spec ("cfp_cabin_pressure") use "CFP_CABIN_PRESSURE_.ADA";
      for Body ("cfp_cabin_pressure") use "CFP_CABIN_PRESSURE.ADA";
      for Spec ("cfp_awytoawyintc_pkg") use "CFP_AWYTOAWYINTC_PKG_.ADA";
      for Body ("cfp_awytoawyintc_pkg") use "CFP_AWYTOAWYINTC_PKG.ADA";
      for Spec ("cfp_alternates_pkg") use "CFP_ALTERNATES_PKG_.ADA";
      for Body ("cfp_alternates_pkg") use "CFP_ALTERNATES_PKG.ADA";
      for Spec ("cex_transition_pkg") use "CEX_TRANSITION_PKG_.ADA";
      for Body ("cex_transition_pkg") use "CEX_TRANSITION_PKG.ADA";
      for Spec ("cex_lgb_utils_pkg") use "CEX_LGB_UTILS_PKG_.ADA";
```

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

```
      for Body ("cex_lgb_utils_pkg") use "CEX_LGB_UTILS_PKG.ADA";
      for Spec ("cex_exec_ldata") use "CEX_EXEC_LDATA_.ADA";
      for Spec ("cex_cinit_pkg") use "CEX_CINIT_PKG_.ADA";
      for Body ("cex_cinit_pkg") use "CEX_CINIT_PKG.ADA";
      for Spec ("cex_cdu_util_pkg") use "CEX_CDU_UTIL_PKG_.ADA";
      for Body ("cex_cdu_util_pkg") use "CEX_CDU_UTIL_PKG.ADA";
      for Spec ("cex_cdu_pkg") use "CEX_CDU_PKG_.ADA";
      for Body ("cex_cdu_pkg") use "CEX_CDU_PKG.ADA";
      for Spec ("cex_cdu_enq_event_pkg") use "CEX_CDU_ENQ_EVENT_PKG_.ADA";
      for Body ("cex_cdu_enq_event_pkg") use "CEX_CDU_ENQ_EVENT_PKG.ADA";
      for Spec ("cex_cdu_cyclic_pkg") use "CEX_CDU_CYCLIC_PKG_.ADA";
      for Body ("cex_cdu_cyclic_pkg") use "CEX_CDU_CYCLIC_PKG.ADA";
      for Spec ("cex_cdusrvc") use "CEX_CDUSRVC_.ADA";
      for Body ("cex_cdusrvc") use "CEX_CDUSRVC.ADA";
      for Spec ("cex_cdurtednld") use "CEX_CDURTEDNLD_.ADA";
      for Body ("cex_cdurtednld") use "CEX_CDURTEDNLD.ADA";
      for Spec ("cex_cduefimap_proc") use "CEX_CDUEFIMAP_PROC_.ADA";
      for Body ("cex_cduefimap_proc") use "CEX_CDUEFIMAP_PROC.ADA";
      for Spec ("cex_bite_interface_pkg") use "CEX_BITE_INTERFACE_PKG_.ADA";
      for Body ("cex_bite_interface_pkg") use "CEX_BITE_INTERFACE_PKG.ADA";
      for Spec ("cdm_disp_buffer_mon_pkg") use "CDM_DISP_BUFFER_MON_PKG_.ADA";
      for Body ("cdm_disp_buffer_mon_pkg") use "CDM_DISP_BUFFER_MON_PKG.ADA";
      for Spec ("cdl_wind_uplink_pkg") use "CDL_WIND_UPLINK_PKG_.ADA";
      for Body ("cdl_wind_uplink_pkg") use "CDL_WIND_UPLINK_PKG.ADA";
      for Spec ("cdl_wind_request_downlink_pkg") use "CDL_WIND_REQUEST_DOWNLINK_PKG_.ADA";
      for Body ("cdl_wind_request_downlink_pkg") use "CDL_WIND_REQUEST_DOWNLINK_PKG.ADA";
      for Body ("cdl_wind_uplink_pkg.uplink_wd_wm") use "CDL_UPLINK_WD_WM_SEP.ADA";
      for Body ("cdl_takeoff_uplink_pkg.uplink_rw") use "CDL_UPLINK_RW_SEP.ADA";
      for Body ("cdl_request_uplink_pkg.uplink_req") use "CDL_UPLINK_REQ_SEP.ADA";
      for Body ("cdl_wind_uplink_pkg.uplink_pwx") use "CDL_UPLINK_PWX_SEP.ADA";
      for Body ("cdl_position_trigger_uplink_pkg.uplink_pos_trigger") use "CDL_UPLINK_POS_TRIGGER_SEP.ADA";
      for Body ("cdl_takeoff_uplink_pkg.uplink_ldi") use "CDL_UPLINK_LDI_SEP.ADA";
      for Body ("cdl_wind_uplink_pkg.uplink_dd") use "CDL_UPLINK_DD_SEP.ADA";
      for Body ("cdl_takeoff_uplink_pkg.uplink_cg") use "CDL_UPLINK_CG_SEP.ADA";
      for Body ("cdl_alternate_uplink_pkg.uplink_alt") use "CDL_UPLINK_ALT_SEP.ADA";
      for Body ("cdl_alternate_uplink_pkg.uplink_alt_inhibit") use "CDL_UPLINK_ALT_INHIBIT_SEP.ADA";
      for Body ("cdl_alternate_uplink_pkg.uplink_alt_flight_list") use "CDL_UPLINK_ALT_FLIGHT_LIST_SEP.ADA";
      for Body ("cdl_alternate_uplink_pkg.uplink_alt_company_preferred") use "CDL_UPLINK_ALT_COMPANY_PREFERRED_SEP.ADA";
      for Spec ("cdl_takeoff_uplink_pkg") use "CDL_TAKEOFF_UPLINK_PKG_.ADA";
      for Body ("cdl_takeoff_uplink_pkg") use "CDL_TAKEOFF_UPLINK_PKG.ADA";
      for Spec ("cdl_takeoff_downlink_pkg") use "CDL_TAKEOFF_DOWNLINK_PKG_.ADA";
      for Body ("cdl_takeoff_downlink_pkg") use "CDL_TAKEOFF_DOWNLINK_PKG.ADA";
```

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

```
      for Body ("cdl_flight_plan_uplink_pkg.step") use "CDL_STEP_SEP.ADA";
      for Body ("cdl_flight_plan_uplink_pkg.speed_altitude") use "CDL_SPEED_ALTITUDE_SEP.ADA";
      for Body ("cdl_takeoff_uplink_pkg.search_ldi_rwy") use "CDL_SEARCH_LDI_RWY_SEP.ADA";
      for Body ("cdl_fix_utilities_pkg.search_fpln_fix") use "CDL_SEARCH_FPLN_FIX_SEP.ADA";
      for Spec ("cdl_search_fix_list_pkg") use "CDL_SEARCH_FIX_LIST_PKG_.ADA";
      for Body ("cdl_search_fix_list_pkg") use "CDL_SEARCH_FIX_LIST_PKG.ADA";
      for Body ("cdl_flight_plan_uplink_pkg.runway") use "CDL_RUNWAY_SEP.ADA";
      for Spec ("cdl_response_downlink_pkg") use "CDL_RESPONSE_DOWNLINK_PKG_.ADA";
      for Body ("cdl_response_downlink_pkg") use "CDL_RESPONSE_DOWNLINK_PKG.ADA";
      for Spec ("cdl_request_uplink_pkg") use "CDL_REQUEST_UPLINK_PKG_.ADA";
      for Body ("cdl_request_uplink_pkg") use "CDL_REQUEST_UPLINK_PKG.ADA";
      for Body ("cdl_position_trigger_uplink_pkg.reporting_points") use "CDL_REPORTING_POINTS_SEP.ADA";
      for Spec ("cdl_rejection_downlink_pkg") use "CDL_REJECTION_DOWNLINK_PKG_.ADA";
      for Body ("cdl_rejection_downlink_pkg") use "CDL_REJECTION_DOWNLINK_PKG.ADA";
      for Spec ("cdl_position_trigger_uplink_pkg") use "CDL_POSITION_TRIGGER_UPLINK_PKG_.ADA";
      for Body ("cdl_position_trigger_uplink_pkg") use "CDL_POSITION_TRIGGER_UPLINK_PKG.ADA";
      for Spec ("cdl_position_downlink_pkg") use "CDL_POSITION_DOWNLINK_PKG_.ADA";
      for Body ("cdl_position_downlink_pkg") use "CDL_POSITION_DOWNLINK_PKG.ADA";
      for Spec ("cdl_periodic_pkg") use "CDL_PERIODIC_PKG_.ADA";
      for Body ("cdl_periodic_pkg") use "CDL_PERIODIC_PKG.ADA";
      for Spec ("cdl_perf_init_uplink_pkg") use "CDL_PERF_INIT_UPLINK_PKG_.ADA";
      for Body ("cdl_perf_init_uplink_pkg") use "CDL_PERF_INIT_UPLINK_PKG.ADA";
      for Spec ("cdl_perf_init_downlink_pkg") use "CDL_PERF_INIT_DOWNLINK_PKG_.ADA";
      for Body ("cdl_perf_init_downlink_pkg") use "CDL_PERF_INIT_DOWNLINK_PKG.ADA";
      for Body ("cdl_flight_plan_uplink_pkg.origin_destination") use "CDL_ORIGIN_DESTINATION_SEP.ADA";
      for Body ("cdl_flight_plan_uplink_pkg.offset") use "CDL_OFFSET_SEP.ADA";
      for Body ("cdl_flight_plan_uplink_pkg.new_fpn") use "CDL_NEW_FPN_SEP.ADA";
      for Spec ("cdl_loc_buffer_data") use "CDL_LOC_BUFFER_DATA_.ADA";
      for Body ("cdl_flight_plan_uplink_pkg.lat_lon_crossing") use "CDL_LAT_LON_CROSSING_SEP.ADA";
      for Body ("cdl_flight_plan_uplink_pkg.intercept_course_from") use "CDL_INTERCEPT_COURSE_FROM_SEP.ADA";
      for Spec ("cdl_init_token_rules_pkg") use "CDL_INIT_TOKEN_RULES_PKG_.ADA";
      for Body ("cdl_init_token_rules_pkg") use "CDL_INIT_TOKEN_RULES_PKG.ADA";
      for Body ("cdl_takeoff_uplink_pkg.initialize_standard_data") use "CDL_INITIALIZE_STANDARD_DATA_SEP.ADA";
      for Body ("cdl_flight_plan_uplink_pkg.hold") use "CDL_HOLD_SEP.ADA";
      for Spec ("cdl_global_data") use "CDL_GLOBAL_DATA_.ADA";
      for Spec ("cdl_get_element_token_pkg") use "CDL_GET_ELEMENT_TOKEN_PKG_.ADA";
      for Body ("cdl_get_element_token_pkg") use "CDL_GET_ELEMENT_TOKEN_PKG.ADA";
      for Spec ("cdl_fpx_init_pkg") use "CDL_FPX_INIT_PKG_.ADA";
      for Body ("cdl_fpx_init_pkg") use "CDL_FPX_INIT_PKG.ADA";
      for Spec ("cdl_fpxrectype_data") use "CDL_FPXRECTYPE_DATA_.ADA";
      for Spec ("cdl_fpn_uplink_transition_pkg") use "CDL_FPN_UPLINK_TRANSITION_PKG_.ADA";
      for Body ("cdl_fpn_uplink_transition_pkg") use "CDL_FPN_UPLINK_TRANSITION_PKG.ADA";
```

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

```
      for Body ("cdl_flight_plan_uplink_pkg.fpn_uplink") use "CDL_FPN_UPLINK_SEP.ADA";
      for Spec ("cdl_flight_plan_uplink_pkg") use "CDL_FLIGHT_PLAN_UPLINK_PKG_.ADA";
      for Body ("cdl_flight_plan_uplink_pkg") use "CDL_FLIGHT_PLAN_UPLINK_PKG.ADA";
      for Body ("cdl_flight_plan_uplink_pkg.flight_plan_element") use "CDL_FLIGHT_PLAN_ELEMENT_SEP.ADA";
      for Spec ("cdl_flight_plan_downlink_pkg") use "CDL_FLIGHT_PLAN_DOWNLINK_PKG_.ADA";
      for Body ("cdl_flight_plan_downlink_pkg") use "CDL_FLIGHT_PLAN_DOWNLINK_PKG.ADA";
      for Body ("cdl_flight_plan_uplink_pkg.flight_number") use "CDL_FLIGHT_NUMBER_SEP.ADA";
      for Spec ("cdl_fix_utilities_pkg") use "CDL_FIX_UTILITIES_PKG_.ADA";
      for Body ("cdl_fix_utilities_pkg") use "CDL_FIX_UTILITIES_PKG.ADA";
      for Spec ("cdl_executive_pkg") use "CDL_EXECUTIVE_PKG_.ADA";
      for Body ("cdl_executive_pkg") use "CDL_EXECUTIVE_PKG.ADA";
      for Spec ("cdl_error_codes_data") use "CDL_ERROR_CODES_DATA_.ADA";
      for Body ("cdl_wind_request_downlink_pkg.downlink_req_wind") use "CDL_DOWNLINK_REQ_WIND_SEP.ADA";
      for Body ("cdl_request_uplink_pkg.downlink_req") use "CDL_DOWNLINK_REQ_SEP.ADA";
      for Body ("cdl_alternates_request_downlink_pkg.downlink_req_alternates") use "CDL_DOWNLINK_REQ_ALTERNATES_SEP.ADA";
      for Spec ("cdl_downlink_buffer_data") use "CDL_DOWNLINK_BUFFER_DATA_.ADA";
      for Spec ("cdl_dlupendawy_data") use "CDL_DLUPENDAWY_DATA_.ADA";
      for Spec ("cdl_dltermtype_types") use "CDL_DLTERMTYPE_TYPES_.ADA";
      for Spec ("cdl_dlsrchtypes_types") use "CDL_DLSRCHTYPES_TYPES_.ADA";
      for Body ("cdl_flight_plan_uplink_pkg.direct_fix") use "CDL_DIRECT_FIX_SEP.ADA";
      for Body ("cdl_flight_plan_uplink_pkg.departure_arrival") use "CDL_DEPARTURE_ARRIVAL_SEP.ADA";
      for Spec ("cdl_demand_pkg") use "CDL_DEMAND_PKG_.ADA";
      for Body ("cdl_demand_pkg") use "CDL_DEMAND_PKG.ADA";
      for Spec ("cdl_data_link_init_pkg") use "CDL_DATA_LINK_INIT_PKG_.ADA";
      for Body ("cdl_data_link_init_pkg") use "CDL_DATA_LINK_INIT_PKG.ADA";
      for Spec ("cdl_datalinerec_types") use "CDL_DATALINEREC_TYPES_.ADA";
      for Body ("cdl_flight_plan_uplink_pkg.cruise_speed_segment") use "CDL_CRUISE_SPEED_SEGMENT_SEP.ADA";
      for Spec ("cdl_constants_data") use "CDL_CONSTANTS_DATA_.ADA";
      for Body ("cdl_flight_plan_uplink_pkg.company_route") use "CDL_COMPANY_ROUTE_SEP.ADA";
      for Spec ("cdl_check_sequence_number_pkg") use "CDL_CHECK_SEQUENCE_NUMBER_PKG_.ADA";
      for Body ("cdl_check_sequence_number_pkg") use "CDL_CHECK_SEQUENCE_NUMBER_PKG.ADA";
      for Body ("cdl_fix_utilities_pkg.check_pbd") use "CDL_CHECK_PBD_SEP.ADA";
      for Body ("cdl_fix_utilities_pkg.check_lat_lon") use "CDL_CHECK_LAT_LON_SEP.ADA";
      for Spec ("cdl_check_fpn_uplink_status_pkg") use "CDL_CHECK_FPN_UPLINK_STATUS_PKG_.ADA";
      for Body ("cdl_check_fpn_uplink_status_pkg") use "CDL_CHECK_FPN_UPLINK_STATUS_PKG.ADA";
      for Body ("cdl_fix_utilities_pkg.check_fix_format") use "CDL_CHECK_FIX_FORMAT_SEP.ADA";
      for Body ("cdl_fix_utilities_pkg.check_crs_int") use "CDL_CHECK_CRS_INT_SEP.ADA";
      for Body ("cdl_flight_plan_uplink_pkg.check_altitude") use "CDL_CHECK_ALTITUDE_SEP.ADA";
      for Body ("cdl_flight_plan_uplink_pkg.approach") use "CDL_APPROACH_SEP.ADA";
      for Body ("cdl_wind_request_downlink_pkg.append_xq") use "CDL_APPEND_XQ_SEP.ADA";
      for Spec ("cdl_append_ts_ga_ca_pkg") use "CDL_APPEND_TS_GA_CA_PKG_.ADA";
      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)

```
      for Spec ("cdl_append_sp_pkg") use "CDL_APPEND_SP_PKG_.ADA";
      for Body ("cdl_append_sp_pkg") use "CDL_APPEND_SP_PKG.ADA";
      for Spec ("cdl_append_rx_pkg") use "CDL_APPEND_RX_PKG_.ADA";
      for Body ("cdl_append_rx_pkg") use "CDL_APPEND_RX_PKG.ADA";
      for Spec ("cdl_append_rp_pkg") use "CDL_APPEND_RP_PKG_.ADA";
      for Body ("cdl_append_rp_pkg") use "CDL_APPEND_RP_PKG.ADA";
      for Spec ("cdl_append_px_pkg") use "CDL_APPEND_PX_PKG_.ADA";
      for Body ("cdl_append_px_pkg") use "CDL_APPEND_PX_PKG.ADA";
      for Spec ("cdl_append_fn_pkg") use "CDL_APPEND_FN_PKG_.ADA";
      for Body ("cdl_append_fn_pkg") use "CDL_APPEND_FN_PKG.ADA";
      for Spec ("cdl_append_error_pkg") use "CDL_APPEND_ERROR_PKG_.ADA";
      for Body ("cdl_append_error_pkg") use "CDL_APPEND_ERROR_PKG.ADA";
      for Spec ("cdl_append_dt_pkg") use "CDL_APPEND_DT_PKG_.ADA";
      for Body ("cdl_append_dt_pkg") use "CDL_APPEND_DT_PKG.ADA";
      for Body ("cdl_wind_request_downlink_pkg.append_dq") use "CDL_APPEND_DQ_SEP.ADA";
      for Body ("cdl_alternates_request_downlink_pkg.append_ax") use "CDL_APPEND_AX_SEP.ADA";
      for Body ("cdl_alternates_request_downlink_pkg.append_aq") use "CDL_APPEND_AQ_SEP.ADA";
      for Spec ("cdl_aoc_message_recording_pkg") use "CDL_AOC_MESSAGE_RECORDING_PKG_.ADA";
      for Body ("cdl_aoc_message_recording_pkg") use "CDL_AOC_MESSAGE_RECORDING_PKG.ADA";
      for Spec ("cdl_alternate_uplink_pkg") use "CDL_ALTERNATES_UPLINK_PKG_.ADA";
      for Body ("cdl_alternate_uplink_pkg") use "CDL_ALTERNATES_UPLINK_PKG.ADA";
      for Spec ("cdl_alternates_request_downlink_pkg") use "CDL_ALTERNATES_REQUEST_DOWNLINK_PKG_.ADA";
      for Body ("cdl_alternates_request_downlink_pkg") use "CDL_ALTERNATES_REQUEST_DOWNLINK_PKG.ADA";
      for Body ("cdl_flight_plan_uplink_pkg.along_track_waypoint") use "CDL_ALONG_TRACK_WAYPOINT_SEP.ADA";
      for Body ("cdl_flight_plan_uplink_pkg.airway") use "CDL_AIRWAY_SEP.ADA";
      for Spec ("cdk_takeoff_ref_page_lftypes") use "CDK_TAKEOFF_REF_PAGE_LFTYPES_.ADA";
      for Spec ("cdk_takeoff_ref_page_2_lftypes") use "CDK_TAKEOFF_REF_PAGE_2_LFTYPES_.ADA";
      for Spec ("cdk_takeoff_ref_page_1_lftypes") use "CDK_TAKEOFF_REF_PAGE_1_LFTYPES_.ADA";
      for Spec ("cdk_takeoff_lftypes") use "CDK_TAKEOFF_LFTYPES_.ADA";
      for Spec ("cdk_refnav_ltypes") use "CDK_REFNAV_LTYPES_.ADA";
      for Spec ("cdk_progress_ltypes") use "CDK_PROGRESS_LTYPES_.ADA";
      for Spec ("cdk_offset_valid_pkg") use "CDK_OFFSET_VALID_PKG_.ADA";
      for Body ("cdk_offset_valid_pkg") use "CDK_OFFSET_VALID_PKG.ADA";
      for Spec ("cdk_key_reaction_lfdata") use "CDK_KEY_REACTION_LFDATA_.ADA";
      for Spec ("cdk_entry_ltypes") use "CDK_ENTRY_LTYPES_.ADA";
      for Spec ("cdk_display_format_ltypes") use "CDK_DISPLAY_FORMAT_LTYPES_.ADA";
      for Spec ("cdk_des_page_ltypes") use "CDK_DES_PAGE_LTYPES_.ADA";
      for Spec ("cdk_dep_arr_support_pkg") use "CDK_DEP_ARR_SUPPORT_PKG_.ADA";
      for Body ("cdk_dep_arr_support_pkg") use "CDK_DEP_ARR_SUPPORT_PKG.ADA";
      for Spec ("data_conversion_pkg") use "CDK_DATA_CONVERSION_PKG_.ADA";
      for Body ("data_conversion_pkg") use "CDK_DATA_CONVERSION_PKG.ADA";
      for Spec ("cdk_datalink_ltypes") use "CDK_DATALINK_LTYPES_.ADA";
```

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

```
      for Spec ("cdk_constants_pkg") use "CDK_CONSTANTS_PKG_.ADA";
      for Spec ("cdk_altn_airports_candidates_pkg") use "CDK_ALTN_AIRPORTS_CANDIDATES_PKG_.ADA";
      for Body ("cdk_altn_airports_candidates_pkg") use "CDK_ALTN_AIRPORTS_CANDIDATES_PKG.ADA";
      for Spec ("cdkm_thrust_lim_mon_pkg") use "CDKM_THRUST_LIM_MON_PKG_.ADA";
      for Body ("cdkm_thrust_lim_mon_pkg") use "CDKM_THRUST_LIM_MON_PKG.ADA";
      for Spec ("cdkm_progress_page_2_mon_pkg") use "CDKM_PROGRESS_PAGE_2_MON_PKG_.ADA";
      for Body ("cdkm_progress_page_2_mon_pkg") use "CDKM_PROGRESS_PAGE_2_MON_PKG.ADA";
      for Spec ("cdkm_progress_page_1_mon_pkg") use "CDKM_PROGRESS_PAGE_1_MON_PKG_.ADA";
      for Body ("cdkm_progress_page_1_mon_pkg") use "CDKM_PROGRESS_PAGE_1_MON_PKG.ADA";
      for Spec ("cdkm_pos_ref_page_3_4_pkg") use "CDKM_POS_REF_PAGE_3_4_PKG_.ADA";
      for Body ("cdkm_pos_ref_page_3_4_pkg") use "CDKM_POS_REF_PAGE_3_4_PKG.ADA";
      for Spec ("cdkm_pos_ref_page_2_4_pkg") use "CDKM_POS_REF_PAGE_2_4_PKG_.ADA";
      for Body ("cdkm_pos_ref_page_2_4_pkg") use "CDKM_POS_REF_PAGE_2_4_PKG.ADA";
      for Spec ("cdkm_pos_init_page_pkg") use "CDKM_POS_INIT_PAGE_PKG_.ADA";
      for Body ("cdkm_pos_init_page_pkg") use "CDKM_POS_INIT_PAGE_PKG.ADA";
      for Spec ("cdkm_nav_radio_page_pkg") use "CDKM_NAV_RADIO_PAGE_PKG_.ADA";
      for Body ("cdkm_nav_radio_page_pkg") use "CDKM_NAV_RADIO_PAGE_PKG.ADA";
      for Spec ("cdi_wind_page_pkg") use "CDI_WIND_PAGE_PKG_.ADA";
      for Body ("cdi_wind_page_pkg") use "CDI_WIND_PAGE_PKG.ADA";
      for Spec ("cdi_time_date_init_page_pkg") use "CDI_TIME_DATE_INIT_PAGE_PKG_.ADA";
      for Body ("cdi_time_date_init_page_pkg") use "CDI_TIME_DATE_INIT_PAGE_PKG.ADA";
      for Spec ("cdi_thrust_lim_page_pkg") use "CDI_THRUST_LIM_PAGE_PKG_.ADA";
      for Body ("cdi_thrust_lim_page_pkg") use "CDI_THRUST_LIM_PAGE_PKG.ADA";
      for Spec ("cdi_takeoff_ref_states_pkg") use "CDI_TAKEOFF_REF_STATES_PKG_.ADA";
      for Body ("cdi_takeoff_ref_states_pkg") use "CDI_TAKEOFF_REF_STATES_PKG.ADA";
      for Body ("cdi_takeoff_ref_page_pkg.get_data") use "CDI_TAKEOFF_REF_PAGE_PKG__GET_DATA.ADA";
      for Spec ("cdi_takeoff_ref_page_pkg") use "CDI_TAKEOFF_REF_PAGE_PKG_.ADA";
      for Body ("cdi_takeoff_ref_page_pkg") use "CDI_TAKEOFF_REF_PAGE_PKG.ADA";
      for Spec ("cdi_takeoff_ref_page_2_pkg") use "CDI_TAKEOFF_REF_PAGE_2_PKG_.ADA";
      for Body ("cdi_takeoff_ref_page_2_pkg") use "CDI_TAKEOFF_REF_PAGE_2_PKG.ADA";
      for Spec ("cdi_takeoff_ref_page_1_pkg") use "CDI_TAKEOFF_REF_PAGE_1_PKG_.ADA";
      for Body ("cdi_takeoff_ref_page_1_pkg") use "CDI_TAKEOFF_REF_PAGE_1_PKG.ADA";
      for Spec ("cdi_takeoff_ref_2_states_pkg") use "CDI_TAKEOFF_REF_2_STATES_PKG_.ADA";
      for Body ("cdi_takeoff_ref_2_states_pkg") use "CDI_TAKEOFF_REF_2_STATES_PKG.ADA";
      for Spec ("cdi_takeoff_ref_1_states_pkg") use "CDI_TAKEOFF_REF_1_STATES_PKG_.ADA";
      for Body ("cdi_takeoff_ref_1_states_pkg") use "CDI_TAKEOFF_REF_1_STATES_PKG.ADA";
      for Spec ("cdi_std_disp_pkg") use "CDI_STD_DISP_PKG_.ADA";
      for Body ("cdi_std_disp_pkg") use "CDI_STD_DISP_PKG.ADA";
      for Spec ("cdi_select_wpt_page_pkg") use "CDI_SELECT_WPT_PAGE_PKG_.ADA";
      for Body ("cdi_select_wpt_page_pkg") use "CDI_SELECT_WPT_PAGE_PKG.ADA";
      for Spec ("cdi_rte_page_6r_pkg") use "CDI_RTE_PAGE_6R_PKG_.ADA";
      for Body ("cdi_rte_page_6r_pkg") use "CDI_RTE_PAGE_6R_PKG.ADA";
```

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

```
      for Spec ("cdi_rte_page_2_x_pkg") use "CDI_RTE_PAGE_2_X_PKG_.ADA";
      for Body ("cdi_rte_page_2_x_pkg") use "CDI_RTE_PAGE_2_X_PKG.ADA";
      for Spec ("cdi_rte_page_1_x_pkg") use "CDI_RTE_PAGE_1_X_PKG_.ADA";
      for Body ("cdi_rte_page_1_x_pkg") use "CDI_RTE_PAGE_1_X_PKG.ADA";
      for Spec ("cdi_rte_legs_page_pkg") use "CDI_RTE_LEGS_PAGE_PKG_.ADA";
      for Body ("cdi_rte_legs_page_pkg") use "CDI_RTE_LEGS_PAGE_PKG.ADA";
      for Spec ("cdi_rte_legs_dirto_page_pkg") use "CDI_RTE_LEGS_DIRTO_PAGE_PKG_.ADA";
      for Body ("cdi_rte_legs_dirto_page_pkg") use "CDI_RTE_LEGS_DIRTO_PAGE_PKG.ADA";
      for Spec ("cdi_rte_legs_common_pkg") use "CDI_RTE_LEGS_COMMON_PKG_.ADA";
      for Body ("cdi_rte_legs_common_pkg") use "CDI_RTE_LEGS_COMMON_PKG.ADA";
      for Spec ("cdi_rte_dir_dep_arr_pkg") use "CDI_RTE_DIR_DEP_ARR_PKG_.ADA";
      for Body ("cdi_rte_dir_dep_arr_pkg") use "CDI_RTE_DIR_DEP_ARR_PKG.ADA";
      for Body ("cdi_rte_directory_indices_pkg.via_trans") use "CDI_RTE_DIRECTORY_INDICES_PKG_VIATRANS.ADA";
      for Body ("cdi_rte_directory_indices_pkg.rte_page_index") use "CDI_RTE_DIRECTORY_INDICES_PKG_RTE_PAGE.ADA";
      for Spec ("cdi_rte_directory_indices_pkg") use "CDI_RTE_DIRECTORY_INDICES_PKG_.ADA";
      for Body ("cdi_rte_directory_indices_pkg") use "CDI_RTE_DIRECTORY_INDICES_PKG.ADA";
      for Spec ("cdi_rte_data_page_pkg") use "CDI_RTE_DATA_PAGE_PKG_.ADA";
      for Body ("cdi_rte_data_page_pkg") use "CDI_RTE_DATA_PAGE_PKG.ADA";
      for Spec ("cdi_ref_nav_data_page_pkg") use "CDI_REF_NAV_DATA_PAGE_PKG_.ADA";
      for Body ("cdi_ref_nav_data_page_pkg") use "CDI_REF_NAV_DATA_PAGE_PKG.ADA";
      for Spec ("cdi_progress_page_4_pkg") use "CDI_PROGRESS_PAGE_4_PKG_.ADA";
      for Body ("cdi_progress_page_4_pkg") use "CDI_PROGRESS_PAGE_4_PKG.ADA";
      for Spec ("cdi_progress_page_3_pkg") use "CDI_PROGRESS_PAGE_3_PKG_.ADA";
      for Body ("cdi_progress_page_3_pkg") use "CDI_PROGRESS_PAGE_3_PKG.ADA";
      for Spec ("cdi_progress_page_2_pkg") use "CDI_PROGRESS_PAGE_2_PKG_.ADA";
      for Body ("cdi_progress_page_2_pkg") use "CDI_PROGRESS_PAGE_2_PKG.ADA";
      for Spec ("cdi_progress_page_1_pkg") use "CDI_PROGRESS_PAGE_1_PKG_.ADA";
      for Body ("cdi_progress_page_1_pkg") use "CDI_PROGRESS_PAGE_1_PKG.ADA";
      for Spec ("cdi_pos_report_page_pkg") use "CDI_POS_REPORT_PAGE_PKG_.ADA";
      for Body ("cdi_pos_report_page_pkg") use "CDI_POS_REPORT_PAGE_PKG.ADA";
      for Spec ("cdi_pos_ref_page_4_4_pkg") use "CDI_POS_REF_PAGE_4_4_PKG_.ADA";
      for Body ("cdi_pos_ref_page_4_4_pkg") use "CDI_POS_REF_PAGE_4_4_PKG.ADA";
      for Spec ("cdi_pos_ref_page_3_4_pkg") use "CDI_POS_REF_PAGE_3_4_PKG_.ADA";
      for Body ("cdi_pos_ref_page_3_4_pkg") use "CDI_POS_REF_PAGE_3_4_PKG.ADA";
      for Spec ("cdi_pos_ref_page_2_4_pkg") use "CDI_POS_REF_PAGE_2_4_PKG_.ADA";
      for Body ("cdi_pos_ref_page_2_4_pkg") use "CDI_POS_REF_PAGE_2_4_PKG.ADA";
      for Spec ("cdi_pos_init_page_pkg") use "CDI_POS_INIT_PAGE_PKG_.ADA";
      for Body ("cdi_pos_init_page_pkg") use "CDI_POS_INIT_PAGE_PKG.ADA";
      for Spec ("cdi_perf_init_page_pkg") use "CDI_PERF_INIT_PAGE_PKG_.ADA";
      for Body ("cdi_perf_init_page_pkg") use "CDI_PERF_INIT_PAGE_PKG.ADA";
      for Spec ("cdi_offpath_des_page_pkg") use "CDI_OFFPATH_DES_PAGE_PKG_.ADA";
      for Body ("cdi_offpath_des_page_pkg") use "CDI_OFFPATH_DES_PAGE_PKG.ADA";
```

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

```
      for Spec ("cdi_nav_radio_page_pkg") use "CDI_NAV_RADIO_PAGE_PKG_.ADA";
      for Body ("cdi_nav_radio_page_pkg") use "CDI_NAV_RADIO_PAGE_PKG.ADA";
      for Spec ("cdi_maint_index_page_pkg") use "CDI_MAINT_INDEX_PAGE_PKG_.ADA";
      for Body ("cdi_maint_index_page_pkg") use "CDI_MAINT_INDEX_PAGE_PKG.ADA";
      for Spec ("cdi_irs_mon_page_pkg") use "CDI_IRS_MON_PAGE_PKG_.ADA";
      for Body ("cdi_irs_mon_page_pkg") use "CDI_IRS_MON_PAGE_PKG.ADA";
      for Spec ("cdi_init_ref_index_page_pkg") use "CDI_INIT_REF_INDEX_PAGE_PKG_.ADA";
      for Body ("cdi_init_ref_index_page_pkg") use "CDI_INIT_REF_INDEX_PAGE_PKG.ADA";
      for Spec ("cdi_ident_page_pkg") use "CDI_IDENT_PAGE_PKG_.ADA";
      for Body ("cdi_ident_page_pkg") use "CDI_IDENT_PAGE_PKG.ADA";
      for Spec ("cdi_hold_page_pkg") use "CDI_HOLD_PAGE_PKG_.ADA";
      for Body ("cdi_hold_page_pkg") use "CDI_HOLD_PAGE_PKG.ADA";
      for Spec ("cdi_hold_at_page_pkg") use "CDI_HOLD_AT_PAGE_PKG_.ADA";
      for Body ("cdi_hold_at_page_pkg") use "CDI_HOLD_AT_PAGE_PKG.ADA";
      for Spec ("cdi_fmc_comm_page_pkg") use "CDI_FMC_COMM_PAGE_PKG_.ADA";
      for Body ("cdi_fmc_comm_page_pkg") use "CDI_FMC_COMM_PAGE_PKG.ADA";
      for Spec ("cdi_fix_info_page_pkg") use "CDI_FIX_INFO_PAGE_PKG_.ADA";
      for Body ("cdi_fix_info_page_pkg") use "CDI_FIX_INFO_PAGE_PKG.ADA";
      for Spec ("cdi_display_pkg") use "CDI_DISPLAY_PKG_.ADA";
      for Body ("cdi_display_pkg") use "CDI_DISPLAY_PKG.ADA";
      for Spec ("cdi_des_page_pkg") use "CDI_DES_PAGE_PKG_.ADA";
      for Body ("cdi_des_page_pkg") use "CDI_DES_PAGE_PKG.ADA";
      for Spec ("cdi_des_forecast_page_pkg") use "CDI_DES_FORECAST_PAGE_PKG_.ADA";
      for Body ("cdi_des_forecast_page_pkg") use "CDI_DES_FORECAST_PAGE_PKG.ADA";
      for Spec ("cdi_dep_arr_index_page_pkg") use "CDI_DEP_ARR_INDEX_PAGE_PKG_.ADA";
      for Body ("cdi_dep_arr_index_page_pkg") use "CDI_DEP_ARR_INDEX_PAGE_PKG.ADA";
      for Spec ("cdi_dep_arr_common_pkg") use "CDI_DEP_ARR_COMMON_PKG_.ADA";
      for Body ("cdi_dep_arr_common_pkg") use "CDI_DEP_ARR_COMMON_PKG.ADA";
      for Spec ("cdi_departures_page_pkg") use "CDI_DEPARTURES_PAGE_PKG_.ADA";
      for Body ("cdi_departures_page_pkg") use "CDI_DEPARTURES_PAGE_PKG.ADA";
      for Spec ("cdi_crz_page_pkg") use "CDI_CRZ_PAGE_PKG_.ADA";
      for Body ("cdi_crz_page_pkg") use "CDI_CRZ_PAGE_PKG.ADA";
      for Spec ("cdi_constants_pkg") use "CDI_CONSTANTS_PKG_.ADA";
      for Spec ("cdi_clb_page_pkg") use "CDI_CLB_PAGE_PKG_.ADA";
      for Body ("cdi_clb_page_pkg") use "CDI_CLB_PAGE_PKG.ADA";
      for Spec ("cdi_arrivals_page_pkg") use "CDI_ARRIVALS_PAGE_PKG_.ADA";
      for Body ("cdi_arrivals_page_pkg") use "CDI_ARRIVALS_PAGE_PKG.ADA";
      for Spec ("cdi_approach_ref_page_pkg") use "CDI_APPROACH_REF_PAGE_PKG_.ADA";
      for Body ("cdi_approach_ref_page_pkg") use "CDI_APPROACH_REF_PAGE_PKG.ADA";
      for Spec ("cdi_apf_page_2_pkg") use "CDI_APF_PAGE_2_PKG_.ADA";
      for Body ("cdi_apf_page_2_pkg") use "CDI_APF_PAGE_2_PKG.ADA";
      for Spec ("cdi_apf_page_1_pkg") use "CDI_APF_PAGE_1_PKG_.ADA";
```

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

```
        for Body ("cdi_apf_page_1_pkg") use "CDI_APF_PAGE_1_PKG.ADA";
        for Spec ("cdi_altn_plan_page_pkg") use "CDI_ALTN_PLAN_PAGE_PKG_.ADA";
        for Body ("cdi_altn_plan_page_pkg") use "CDI_ALTN_PLAN_PAGE_PKG.ADA";
        for Spec ("cdi_altn_page_pkg") use "CDI_ALTN_PAGE_PKG_.ADA";
        for Body ("cdi_altn_page_pkg") use "CDI_ALTN_PAGE_PKG.ADA";
        for Spec ("cdi_altn_list_page_pkg") use "CDI_ALTN_LIST_PAGE_PKG_.ADA";
        for Body ("cdi_altn_list_page_pkg") use "CDI_ALTN_LIST_PAGE_PKG.ADA";
        for Spec ("cdck_fpln_request_status_tpkg") use "CDCK_FPLN_REQUEST_STATUS_TPKG_.ADA";
        for Spec ("brown_lgbm") use "BROWN_LGBM_.ADA";
        for Body ("brown_lgbm") use "BROWN_LGBM.ADA";
        for Spec ("bite_large_bp_service_pkg") use "BITE_LARGE_BP_SERVICE_PKG_.ADA";
        for Body ("bite_large_bp_service_pkg") use "BITE_LARGE_BP_SERVICE_PKG.ADA";
        for Spec ("bite_fmf_ac_state_data_pkg") use "BITE_FMF_AC_STATE_DATA_PKG_.ADA";
        for Body ("bite_fmf_ac_state_data_pkg") use "BITE_FMF_AC_STATE_DATA_PKG.ADA";
        for Spec ("atc_msg_encoder_pkg") use "ATC_MSG_ENCODER_PKG_.ADA";
        for Body ("atc_msg_encoder_pkg") use "ATC_MSG_ENCODER_PKG.ADA";
        for Spec ("atc_msg_decoder_pkg") use "ATC_MSG_DECODER_PKG_.ADA";
        for Body ("atc_msg_decoder_pkg") use "ATC_MSG_DECODER_PKG.ADA";
        for Spec ("atc_msg_common_types_pkg") use "ATC_MSG_COMMON_TYPES_PKG_.ADA";
        for Spec ("atc_fmf_cmf_interface_types") use "ATC_FMF_CMF_INTERFACE_TYPES_.ADA";
        for Spec ("atc_encode_components_pkg") use "ATC_ENCODE_COMPONENTS_PKG_.ADA";
        for Body ("atc_encode_components_pkg") use "ATC_ENCODE_COMPONENTS_PKG.ADA";
        for Spec ("atc_encoder_utility_pkg") use "ATC_ENCODER_UTILITY_PKG_.ADA";
        for Body ("atc_encoder_utility_pkg") use "ATC_ENCODER_UTILITY_PKG.ADA";
        for Spec ("atc_decode_components_pkg") use "ATC_DECODE_COMPONENTS_PKG_.ADA";
        for Body ("atc_decode_components_pkg") use "ATC_DECODE_COMPONENTS_PKG.ADA";
        for Spec ("atc_decode_clearance_pkg") use "ATC_DECODE_CLEARANCE_PKG_.ADA";
        for Body ("atc_decode_clearance_pkg") use "ATC_DECODE_CLEARANCE_PKG.ADA";
        for Spec ("atc_decoder_utility_pkg") use "ATC_DECODER_UTILITY_PKG_.ADA";
        for Body ("atc_decoder_utility_pkg") use "ATC_DECODER_UTILITY_PKG.ADA";
        for Spec ("atc_decoder_rte_info_pkg") use "ATC_DECODER_RTE_INFO_PKG_.ADA";
        for Body ("atc_decoder_rte_info_pkg") use "ATC_DECODER_RTE_INFO_PKG.ADA";
        for Spec ("atc_datalink_io_pkg") use "ATC_DATALINK_IO_PKG_.ADA";
        for Body ("atc_datalink_io_pkg") use "ATC_DATALINK_IO_PKG.ADA";
        for Spec ("atc_common_interface_types") use "ATC_COMMON_INTERFACE_TYPES_.ADA";
        for Spec ("atc_cmf_fmf_interface_types") use "ATC_CMF_FMF_INTERFACE_TYPES_.ADA";
        for Spec ("assert_fm_codes_pkg") use "ASSERT_FM_CODES_PKG_.ADA";
        for Spec ("arr_dep_ldata_obj_mgr") use "ARR_DEP_LDATA_OBJ_MGR_.ADA";
        for Body ("arr_dep_ldata_obj_mgr") use "ARR_DEP_LDATA_OBJ_MGR.ADA";
        for Spec ("arr_dep_ldata") use "ARR_DEP_LDATA_.ADA";
        for Spec ("aoc_datalink_io_pkg") use "AOC_DATALINK_IO_PKG_.ADA";
        for Body ("aoc_datalink_io_pkg") use "AOC_DATALINK_IO_PKG.ADA";
```

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

```
      for Spec ("angle_const_pkg") use "ANGLE_CONST_PKG_.ADA";
      for Spec ("airway_types") use "AIRWAY_TYPES_.ADA";
      for Spec ("ads_manager_pkg") use "ADS_MANAGER_PKG_.ADA";
      for Body ("ads_manager_pkg") use "ADS_MANAGER_PKG.ADA";
      for Spec ("ads_lftypes") use "ADS_LFTYPES_.ADA";
      for Spec ("ads_iftypes") use "ADS_IFTYPES_.ADA";
      for Spec ("ads_fm_interface_mgr_pkg") use "ADS_FM_INTERFACE_MGR_PKG_.ADA";
      for Body ("ads_fm_interface_mgr_pkg") use "ADS_FM_INTERFACE_MGR_PKG.ADA";
      for Spec ("act_prov_index_manager") use "ACT_PROV_INDEX_MANAGER_.ADA";
      for Body ("act_prov_index_manager") use "ACT_PROV_INDEX_MANAGER.ADA";
      for Spec ("acars_periodic_pkg") use "ACARS_PERIODIC_PKG_.ADA";
      for Body ("acars_periodic_pkg") use "ACARS_PERIODIC_PKG.ADA";
      for Spec ("acars_incoming_buffer_pkg") use "ACARS_INCOMING_BUFFER_PKG_.ADA";
      for Spec ("acars_crc_pkg") use "ACARS_CRC_PKG_.ADA";
      for Body ("acars_crc_pkg") use "ACARS_CRC_PKG.ADA";
      for Spec ("acars_buffer_ltypes") use "ACARS_BUFFER_LTYPES_.ADA";
      for Spec ("acars_buffer_interface_mgr_pkg") use "ACARS_BUFFER_INTERFACE_MGR_PKG_.ADA";
      for Body ("acars_buffer_interface_mgr_pkg") use "ACARS_BUFFER_INTERFACE_MGR_PKG.ADA";

      for Specification_Suffix ("ada") use "_.ada";
      for Implementation_Suffix ("ada") use ".ada";
      for Separate_Suffix use ".ada";
   end Naming;

   Tornado := external ("WIND_BASE");
   Hi_Scoe := external ("SCOE_BASE");
   Hi_Platform := "wrSbc750gx_scoe";
   for Languages use ("Ada");
   for Object_Dir use "..\OBJ";
   for Main use ("CTP_B787_PERF_ALTNESTDES.ADA");
   for Exec_Dir use "..";

   package Pretty_Printer is
   end Pretty_Printer;

   package Builder is
      for Default_Switches ("ada") use ("-x", "--RTS=cert", "-j2", "-m", "-I" & Hi_Scoe & "\platforms\" & Hi_Platform &
» "\include");
      for Executable ("fm_startup.ada") use "fm_ada_startup";
   end Builder;

   package Binder is
```

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

```
     for Default_Switches ("ada") use ("-E");
  end Binder;

  package Linker is
    for Default_Switches ("ada") use ("--LINK=ldppc",
        "-nostdlib",
        "-r",
        "-d",
        Hi_Scoe & "\platforms\" & Hi_Platform & "\lib\adaLCH.PPC604gnu.cert.o",
        Hi_Scoe & "\platforms\" & Hi_Platform & "\lib\tftp.PPC604gnu.cert.o",
        "-L..\..\LIB",
        "--start-group",
        "-l_io_c_fmf",
        "-l_bite_c_fmf",
        "-l_bsvc_c_fmf",
        "-l_flxcore_c_fmf",
        "-l_flxprj_c_fmf",
        "-l_ci_c",
        "-l_hmi_c",
        "-l_fpprj_c",
        "-l_ltcore_c",
        "-l_fpcore_c",
        "-l_psvc_c",
        "-l_dbam_c",
        "--end-group");
  end Linker;

  package Compiler is
    for Default_Switches ("ada") use (
        "-gdwarf-2",
        "-ansi",
        "-gnatf",
        "-gnatn",
        "-gnato",
        "-fno-common",
        "-mstrict-align",
        "-fno-crossjumping",
        "-fno-strict-aliasing",
        "-fstack-check");
  end Compiler;

  package Ide is
```

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

```
      for Compiler_Command ("ada") use "powerpc-wrs-vxworksae-gnatmake";
      for Gnatlist use "powerpc-wrs-vxworksae-gnatls";
      for Debugger_Command use "powerpc-wrs-vxworksae-gdb";
      for Program_Host use "SBC-session";
      for Communication_Protocol use "wtx";
   end Ide;

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_CI

```
  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
```