UNCLASSIFIED

AD NUMBER

ADB120254

NEW LIMITATION CHANGE

TO

Approved for public release, distribution unlimited

FROM

Distribution authorized to U.S. Gov't. agencies and their contractors; Administrative/Operational Use; MAR 1988. Other requests shall be referred to Air Force Armament Lab., Eglin AFB, FL 32542.

AUTHORITY

AFSC/MNOI [WL/Eglin] ltr dtd 13 Feb 1992

THIS PAGE IS UNCLASSIFIED

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

DTIC FILE COPY

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Distribution authorized to U.S. Government Agencies and their contractors; CT (over)
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S) AFATL-TR-88-18, Vol 7
6a. NAME OF PERFORMING ORGANIZATION McDonnell Douglas Astronautics Company 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION Aeromechanics Division
6c. ADDRESS (City, State, and ZIP Code) P.O. Box 516 St. Louis, MO 63166	7b. ADDRESS (City, State, and ZIP Code) Air Force Armament Laboratory Eglin AFB, FL 32542-5434
8a. NAME OF FUNDING/SPONSORING ORGANIZATION STARS Joint Program Office 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F08635-86-C-0025
8c. ADDRESS (City, State, and ZIP Code) Room 3D139 (1211 Fern St) The Pentagon Washington DC 20301-3081	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. 63756D; PROJECT NO. 921C; TASK NO GZ; WORK UNIT ACCESSION NO. 57

11. TITLE (Include Security Classification)
Common Ada Missile Package (CAMP) Project: Missile Software Parts, Vol 7: Detail Design Documents (Vol 7-12)

12. PERSONAL AUTHOR(S)
D. McNicholl, S. Cohen, C. Palmer, et al.

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Technical Note	FROM Sep 85 TO Mar 88	March 1988	526

16. SUPPLEMENTARY NOTATION
SUBJECT TO EXPORT CONTROL LAWS.
Availability of this report is specified on verso of front cover. (over)

17. COSATI CODES		
FIELD	GROUP	SUB-GROUP

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
Reusable Software, Missile Software, Software Generators
Ada, Parts Composition Systems, Software Parts

19. ABSTRACT (Continue on reverse if necessary and identify by block number)
The objective of the CAMP program is to demonstrate the feasibility of reusable Ada software parts in a real-time embedded application area; the domain chosen for the demonstration was that of missile flight software systems. This required that the existence of commonality within that domain be verified (in order to justify the development of parts for that domain), and that software parts be designed which address those areas identified. An associated parts system was developed to support parts usage. Volume 1 of this document is the User's Guide to the CAMP Software parts, Volume 2 is the Version Description Document; Volume 3 is the Software Product Specification; Volumes 4-6 contain the Top-Level Design Document; and, Volumes 7-12 contain the Detail Design Documents.

AD-B120 254

DTIC
ELECTE
APR 07 1988
E

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION
☐ UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT ☐ DTIC USERS	Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL Christine Anderson	22b. TELEPHONE (Include Area Code) (904) 882-2961 22c. OFFICE SYMBOL AFATL/FXG

DD Form 1473, JUN 86 *Previous editions are obsolete.*

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

3. DISTRIBUTION/AVAILABILITY OF REPORT (CONCLUDED)

~~this report documents test and evaluation,~~ distribution limitation applied March 1988. Other requests for this document must be referred to AFATL/FXG, Eglin AFB, Florida 32542-5434.

16. SUPPLEMENTARY NOTATION (CONCLUDED)

These technical notes accompany the CAMP final report AFATL-TR-85-93 (3 Vols)

AFATL-TR-88-18, Vol 7

SOFTWARE DETAILED DESIGN DOCUMENT

FOR THE

MISSILE SOFTWARE PARTS

OF THE

COMMON ADA MISSILE PACKAGE (CAMP) PROJECT

CONTRACT F08635-86-C-0025

CDRL SEQUENCE NO. C007

Accession For	
NTIS GRA&I	X
DTIC TAB	
Unannounced	
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
C-2	57

30 OCTOBER 1987

Distribution authorized to U.S. Government agencies and their contractors only; ~~this report documents test and evaluation;~~ distribution limitation applied July 1987. Other requests for this document must be referred to the Air Force Armament Laboratory (FXG) Eglin Air Force Base, Florida 32542-5434.

CT

DESTRUCTION NOTICE – For classified documents, follow the procedures in DoD 5220.22-M, Industrial Security Manual, Section II-19 or DoD 5200.1-R, Information Security Program Regulation, Chapter IX. For unclassified, limited documents, destroy by any method that will prevent disclosure of contents or reconstruction of the document.

WARNING: This document contains technical data whose export is restricted by the Arms Export Control Act (Title 22, U.S.C., Sec. 2751, et seq.) or the Export Administration Act of 1979, as amended (Title 50, U.S.C., App. 2401, et seq.). Violations of these export laws are subject to severe criminal penalties. Disseminate in accordance with the provisions of AFR 80-34.

AIR FORCE ARMAMENT LABORATORY
Air Force Systems Command ▮ United States Air Force ▮ Eglin Air Force Base, Florida

SOFTWARE DETAILED DESIGN DOCUMENT

FOR THE

MISSILE SOFTWARE PARTS

OF THE

COMMON ADA MISSILE PACKAGE (CAMP)
PROJECT

CONTRACT F08635-86-C-0025

CDRL SEQUENCE NO. C007

30 OCTOBER 1987

Prepared for

Air Force Armament Laboratory
Aeromechanics Division
Guidance & Control Branch
Eglin AFB, Florida 32542

Prepared by

McDonnell Douglas Astronautics Company
P.O. Box 516
St. Louis, MO 63166

88 4 6 126

PREFACE

This document specifies the detailed design of the missile software parts identified during the Common Ada Missile Packages (CAMP) program.

This document contains data generated under the work task described by sections 5.3.1.2 and 5.3.2.3 of DOD-STD-2167, Defense System Software Development. It has been prepared in accordance with DI-MCCR-80031, Software Detailed Design Document.

The structure of DI-MCCR-80031 requires tailoring to meet the needs of reusable software components. The DID is intended for use in describing the design of a single application Computer Software Configuration Item (CSCI). The DID has, therefore, been tailored for describing the design of parts used individually or on a group basis. In addition, elements of DI-MCCR-80031 were already incorporated into the paragraph structure of the Top-Level Design Document (TLDD) because the DID for that document does not provide enough detail to be useful as a top-level design description for reusable parts. Appendix I summarizes the modifications made to this DID to tailor it for the description of the detailed design of reusable missile software parts.

TABLE OF CONTENTS

1 SCOPE 1
1.1 IDENTIFICATION 1
1.2 PURPOSE 1
1.3 INTRODUCTION 3

2 APPLICABLE DOCUMENTS 7
2.1 GOVERNMENT DOCUMENTS 7
2.2 NON-GOVERNMENT DOCUMENTS 7

3 REQUIREMENTS 9
3.1 INTERFACE DESIGN 9
3.2 GLOBAL DATA AND DATA TYPES 9

3.3 DETAILED DESIGN 11
3.3.1 DATA TYPES 11

3.3.1.1 BASIC DATA TYPES (BODY) TLCSC P621 (CATALOG #P1093-0) 13
3.3.1.1.1 REQUIREMENTS ALLOCATION 13
3.3.1.1.2 LOCAL ENTITIES DESIGN 13
3.3.1.1.3 INPUT/OUTPUT 13
3.3.1.1.4 LOCAL DATA 13
3.3.1.1.5 PROCESS CONTROL 13
3.3.1.1.6 PROCESSING 13
3.3.1.1.7 UTILIZATION OF OTHER ELEMENTS 25
3.3.1.1.8 LIMITATIONS 25
3.3.1.1.9 LLCSC DESIGN 25
3.3.1.1.10 UNIT DESIGN 25

3.3.1.2 KALMAN FILTER DATA TYPES (BODY) TLCSC P622 (CATALOG #P158-0) 39
3.3.1.2.1 REQUIREMENTS ALLOCATION 39
3.3.1.2.2 LOCAL ENTITIES DESIGN 39
3.3.1.2.3 INPUT/OUTPUT 39
3.3.1.2.4 LOCAL DATA 39
3.3.1.2.5 PROCESS CONTROL 39
3.3.1.2.6 PROCESSING 39
3.3.1.2.7 UTILIZATION OF OTHER ELEMENTS 40
3.3.1.2.8 LIMITATIONS 41
3.3.1.2.9 LLCSC DESIGN 41
3.3.1.2.10 UNIT DESIGN 41

3.3.1.3 AUTOPILOT DATA TYPES (BODY) TLCSC P623 (CATALOG #P93-0) 45
3.3.1.3.1 REQUIREMENTS ALLOCATION 45
3.3.1.3.2 LOCAL ENTITIES DESIGN 45
3.3.1.3.3 INPUT/OUTPUT 45
3.3.1.3.4 LOCAL DATA 45
3.3.1.3.5 PROCESS CONTROL 45
3.3.1.3.6 PROCESSING 45
3.3.1.3.7 UTILIZATION OF OTHER ELEMENTS 49
3.3.1.3.8 LIMITATIONS 49
3.3.1.3.9 LLCSC DESIGN 49
3.3.1.3.10 UNIT DESIGN 50

3.3.2 NAVIGATION 55

3.3.2.1 COMMON_NAVIGATION_PARTS (PACKAGE BODY) TLCSC P001 (CATALOG #P208-0) 57
3.3.2.1.1 REQUIREMENTS ALLOCATION 57
3.3.2.1.2 LOCAL ENTITIES DESIGN 57
3.3.2.1.3 INPUT/OUTPUT 57
3.3.2.1.4 LOCAL DATA 57
3.3.2.1.5 PROCESS CONTROL 57
3.3.2.1.6 PROCESSING 57
3.3.2.1.7 UTILIZATION OF OTHER ELEMENTS 58
3.3.2.1.8 LIMITATIONS 58
3.3.2.1.9 LLCSC DESIGN 58
3.3.2.1.9.1 ALTITUDE_INTEGRATION PACKAGE DESIGN (CATALOG #P209-0) 58
3.3.2.1.9.1.1 REQUIREMENTS ALLOCATION 59
3.3.2.1.9.1.2 LOCAL ENTITIES DESIGN 59
3.3.2.1.9.1.3 INPUT/OUTPUT 59
3.3.2.1.9.1.4 LOCAL DATA 60
3.3.2.1.9.1.5 PROCESS CONTROL 60
3.3.2.1.9.1.6 PROCESSING 60
3.3.2.1.9.1.7 UTILIZATION OF OTHER ELEMENTS 60
3.3.2.1.9.1.8 LIMITATIONS 60
3.3.2.1.9.1.9 LLCSC DESIGN 61
3.3.2.1.9.1.10 UNIT DESIGN 61
3.3.2.1.9.1.10.1 REINITIALIZE UNIT DESIGN 61
3.3.2.1.9.1.10.1.1 REQUIREMENTS ALLOCATION 61
3.3.2.1.9.1.10.1.2 LOCAL ENTITIES DESIGN 61
3.3.2.1.9.1.10.1.3 INPUT/OUTPUT 61
3.3.2.1.9.1.10.1.4 LOCAL DATA 61
3.3.2.1.9.1.10.1.5 PROCESS CONTROL 61
3.3.2.1.9.1.10.1.6 PROCESSING 61
3.3.2.1.9.1.10.1.7 UTILIZATION OF OTHER ELEMENTS 62
3.3.2.1.9.1.10.1.8 LIMITATIONS 62
3.3.2.1.9.1.10.2 INTEGRATE UNIT DESIGN 62
3.3.2.1.9.1.10.2.1 REQUIREMENTS ALLOCATION 63
3.3.2.1.9.1.10.2.2 LOCAL ENTITIES DESIGN 63
3.3.2.1.9.1.10.2.3 INPUT/OUTPUT 63
3.3.2.1.9.1.10.2.4 LOCAL DATA 63
3.3.2.1.9.1.10.2.5 PROCESS CONTROL 63
3.3.2.1.9.1.10.2.6 PROCESSING 63
3.3.2.1.9.1.10.2.7 UTILIZATION OF OTHER ELEMENTS 64
3.3.2.1.9.1.10.2.8 LIMITATIONS 65
3.3.2.1.9.2 UPDATE_VELOCITY PACKAGE DESIGN (CATALOG #P214-0) 65
3.3.2.1.9.2.1 REQUIREMENTS ALLOCATION 65
3.3.2.1.9.2.2 LOCAL ENTITIES DESIGN 65
3.3.2.1.9.2.3 INPUT/OUTPUT 65
3.3.2.1.9.2.4 LOCAL DATA 67
3.3.2.1.9.2.5 PROCESS CONTROL 67
3.3.2.1.9.2.6 PROCESSING 67
3.3.2.1.9.2.7 UTILIZATION OF OTHER ELEMENTS 67
3.3.2.1.9.2.8 LIMITATIONS 67
3.3.2.1.9.2.9 LLCSC DESIGN 67
3.3.2.1.9.2.10 UNIT DESIGN 67
3.3.2.1.9.2.10.1 REINITIALIZE UNIT DESIGN 68
3.3.2.1.9.2.10.1.1 REQUIREMENTS ALLOCATION 68

3.3.2.1.9.2.10.1.2 LOCAL ENTITIES DESIGN 68
3.3.2.1.9.2.10.1.3 INPUT/OUTPUT 68
3.3.2.1.9.2.10.1.4 LOCAL DATA 68
3.3.2.1.9.2.10.1.5 PROCESS CONTROL 68
3.3.2.1.9.2.10.1.6 PROCESSING 68
3.3.2.1.9.2.10.1.7 UTILIZATION OF OTHER ELEMENTS 69
3.3.2.1.9.2.10.1.8 LIMITATIONS 69
3.3.2.1.9.2.10.2 UPDATE UNIT DESIGN 69
3.3.2.1.9.2.10.2.1 REQUIREMENTS ALLOCATION 69
3.3.2.1.9.2.10.2.2 LOCAL ENTITIES DESIGN 70
3.3.2.1.9.2.10.2.3 INPUT/OUTPUT 70
3.3.2.1.9.2.10.2.4 LOCAL DATA 70
3.3.2.1.9.2.10.2.5 PROCESS CONTROL 70
3.3.2.1.9.2.10.2.6 PROCESSING 70
3.3.2.1.9.2.10.2.7 UTILIZATION OF OTHER ELEMENTS 71
3.3.2.1.9.2.10.2.8 LIMITATIONS 72
3.3.2.1.9.2.10.3 CURRENT_VELOCITY UNIT DESIGN 72
3.3.2.1.9.2.10.3.1 REQUIREMENTS ALLOCATION 72
3.3.2.1.9.2.10.3.2 LOCAL ENTITIES DESIGN 73
3.3.2.1.9.2.10.3.3 INPUT/OUTPUT 73
3.3.2.1.9.2.10.3.4 LOCAL DATA 73
3.3.2.1.9.2.10.3.5 PROCESS CONTROL 73
3.3.2.1.9.2.10.3.6 PROCESSING 73
3.3.2.1.9.2.10.3.7 UTILIZATION OF OTHER ELEMENTS 73
3.3.2.1.9.2.10.3.8 LIMITATIONS 74
3.3.2.1.10 UNIT DESIGN 74
3.3.2.1.10.1 COMPUTE_GROUND_VELOCITY UNIT DESIGN (CATALOG #P210-0) 74
3.3.2.1.10.1.1 REQUIREMENTS ALLOCATION 74
3.3.2.1.10.1.2 LOCAL ENTITIES DESIGN 74
3.3.2.1.10.1.3 INPUT/OUTPUT 74
3.3.2.1.10.1.4 LOCAL DATA 75
3.3.2.1.10.1.5 PROCESS CONTROL 75
3.3.2.1.10.1.6 PROCESSING 75
3.3.2.1.10.1.7 UTILIZATION OF OTHER ELEMENTS 76
3.3.2.1.10.1.8 LIMITATIONS 76
3.3.2.1.10.2 COMPUTE_GRAVITATIONAL_ACCELERATION_LAT_IN UNIT DESIGN (CATALOG #P211-0) 76
3.3.2.1.10.2.1 REQUIREMENTS ALLOCATION 76
3.3.2.1.10.2.2 LOCAL ENTITIES DESIGN 76
3.3.2.1.10.2.3 INPUT/OUTPUT 76
3.3.2.1.10.2.4 LOCAL DATA 78
3.3.2.1.10.2.5 PROCESS CONTROL 78
3.3.2.1.10.2.6 PROCESSING 78
3.3.2.1.10.2.7 UTILIZATION OF OTHER ELEMENTS 79
3.3.2.1.10.2.8 LIMITATIONS 79
3.3.2.1.10.3 COMPUTE_GRAVITATIONAL_ACCELERATION_SIN_LAT_IN UNIT DESIGN (CATALOG #P212-0) 79
3.3.2.1.10.3.1 REQUIREMENTS ALLOCATION 79
3.3.2.1.10.3.2 LOCAL ENTITIES DESIGN 79
3.3.2.1.10.3.3 INPUT/OUTPUT 79
3.3.2.1.10.3.4 LOCAL DATA 81
3.3.2.1.10.3.5 PROCESS CONTROL 81
3.3.2.1.10.3.6 PROCESSING 81
3.3.2.1.10.3.7 UTILIZATION OF OTHER ELEMENTS 81
3.3.2.1.10.3.8 LIMITATIONS 81
3.3.2.1.10.4 COMPUTE_HEADING UNIT DESIGN (CATALOG #P213-0) 81

3.3.2.1.10.4.1 REQUIREMENTS ALLOCATION 82
3.3.2.1.10.4.2 LOCAL ENTITIES DESIGN 82
3.3.2.1.10.4.3 INPUT/OUTPUT 82
3.3.2.1.10.4.4 LOCAL DATA 83
3.3.2.1.10.4.5 PROCESS CONTROL 83
3.3.2.1.10.4.6 PROCESSING 83
3.3.2.1.10.4.7 UTILIZATION OF OTHER ELEMENTS 83
3.3.2.1.10.4.8 LIMITATIONS 83
3.3.2.1.10.5 SCALAR_VELOCITY UNIT DESIGN (CATALOG #P215-0) 83
3.3.2.1.10.5.1 REQUIREMENTS ALLOCATION 83
3.3.2.1.10.5.2 LOCAL ENTITIES DESIGN 83
3.3.2.1.10.5.3 INPUT/OUTPUT 83
3.3.2.1.10.5.4 LOCAL DATA 84
3.3.2.1.10.5.5 PROCESS CONTROL 84
3.3.2.1.10.5.6 PROCESSING 84
3.3.2.1.10.5.7 UTILIZATION OF OTHER ELEMENTS 85
3.3.2.1.10.5.8 LIMITATIONS 85
3.3.2.1.10.6 COMPUTE_ROTATION_INCREMENTS (FUNCTION BODY) UNIT DESIGN (CATALOG #P216-0) 85
3.3.2.1.10.6.1 REQUIREMENTS ALLOCATION 85
3.3.2.1.10.6.2 LOCAL ENTITIES DESIGN 85
3.3.2.1.10.6.3 INPUT/OUTPUT 85
3.3.2.1.10.6.4 LOCAL DATA 86
3.3.2.1.10.6.5 PROCESS CONTROL 86
3.3.2.1.10.6.6 PROCESSING 86
3.3.2.1.10.6.7 UTILIZATION OF OTHER ELEMENTS 87
3.3.2.1.10.6.8 LIMITATIONS 87

3.3.2.2 NORTH POINTING NAVIGATION PARTS (BODY) TLCSC P003 (CATALOG #P257-0) 99
3.3.2.2.1 REQUIREMENTS ALLOCATION 99
3.3.2.2.2 LOCAL ENTITIES DESIGN 99
3.3.2.2.3 INPUT/OUTPUT 99
3.3.2.2.4 LOCAL DATA 99
3.3.2.2.5 PROCESS CONTROL 99
3.3.2.2.6 PROCESSING 99
3.3.2.2.7 UTILIZATION OF OTHER ELEMENTS 100
3.3.2.2.8 LIMITATIONS 100
3.3.2.2.9 LLCSC DESIGN 100
3.3.2.2.9.1 RADIUS_OF_CURVATURE PACKAGE DESIGN (CATALOG #P259-0) 100
3.3.2.2.9.1.1 REQUIREMENTS ALLOCATION 100
3.3.2.2.9.1.2 LOCAL ENTITIES DESIGN 100
3.3.2.2.9.1.3 INPUT/OUTPUT 100
3.3.2.2.9.1.4 LOCAL DATA 102
3.3.2.2.9.1.5 PROCESS CONTROL 102
3.3.2.2.9.1.6 PROCESSING 102
3.3.2.2.9.1.7 UTILIZATION OF OTHER ELEMENTS 103
3.3.2.2.9.1.8 LIMITATIONS 104
3.3.2.2.9.1.9 LLCSC DESIGN 104
3.3.2.2.9.1.10 UNIT DESIGN 104
3.3.2.2.9.2 EARTH_ROTATION_RATE PACKAGE DESIGN (CATALOG #P261-0) 104
3.3.2.2.9.2.1 REQUIREMENTS ALLOCATION 104
3.3.2.2.9.2.2 LOCAL ENTITIES DESIGN 104
3.3.2.2.9.2.3 INPUT/OUTPUT 104
3.3.2.2.9.2.4 LOCAL DATA 105

3.3.2.2.9.2.5 PROCESS CONTROL 106
3.3.2.2.9.2.6 PROCESSING 106
3.3.2.2.9.2.7 UTILIZATION OF OTHER ELEMENTS 107
3.3.2.2.9.2.8 LIMITATIONS 107
3.3.2.2.9.2.9 LLCSC DESIGN 107
3.3.2.2.9.2.10 UNIT DESIGN 107
3.3.2.2.9.3 EARTH_RELATIVE_NAVIGATION_ROTATION_RATE PACKAGE DESIGN (CATALOG #P262-0) 107
3.3.2.2.9.3.1 REQUIREMENTS ALLOCATION 108
3.3.2.2.9.3.2 LOCAL ENTITIES DESIGN 108
3.3.2.2.9.3.3 INPUT/OUTPUT 108
3.3.2.2.9.3.4 LOCAL DATA 109
3.3.2.2.9.3.5 PROCESS CONTROL 109
3.3.2.2.9.3.6 PROCESSING 109
3.3.2.2.9.3.7 UTILIZATION OF OTHER ELEMENTS 110
3.3.2.2.9.3.8 LIMITATIONS 110
3.3.2.2.9.3.9 LLCSC DESIGN 110
3.3.2.2.9.3.10 UNIT DESIGN 110
3.3.2.2.9.4 LATITUDE INTEGRATION PACKAGE DESIGN (CATALOG #P263-0) 111
3.3.2.2.9.4.1 REQUIREMENTS ALLOCATION 111
3.3.2.2.9.4.2 LOCAL ENTITIES DESIGN 111
3.3.2.2.9.4.3 INPUT/OUTPUT 111
3.3.2.2.9.4.4 LOCAL DATA 112
3.3.2.2.9.4.5 PROCESS CONTROL 112
3.3.2.2.9.4.6 PROCESSING 112
3.3.2.2.9.4.7 UTILIZATION OF OTHER ELEMENTS 112
3.3.2.2.9.4.8 LIMITATIONS 113
3.3.2.2.9.4.9 LLCSC DESIGN 113
3.3.2.2.9.4.10 UNIT DESIGN 113
3.3.2.2.9.4.10.1 REINITIALIZE UNIT DESIGN 113
3.3.2.2.9.4.10.1.1 REQUIREMENTS ALLOCATION 113
3.3.2.2.9.4.10.1.2 LOCAL ENTITIES DESIGN 113
3.3.2.2.9.4.10.1.3 INPUT/OUTPUT 113
3.3.2.2.9.4.10.1.4 LOCAL DATA 113
3.3.2.2.9.4.10.1.5 PROCESS CONTROL 113
3.3.2.2.9.4.10.1.6 PROCESSING 114
3.3.2.2.9.4.10.1.7 UTILIZATION OF OTHER ELEMENTS 114
3.3.2.2.9.4.10.1.8 LIMITATIONS 115
3.3.2.2.9.4.10.2 INTEGRATE UNIT DESIGN 115
3.3.2.2.9.4.10.2.1 REQUIREMENTS ALLOCATION 115
3.3.2.2.9.4.10.2.2 LOCAL ENTITIES DESIGN 115
3.3.2.2.9.4.10.2.3 INPUT/OUTPUT 115
3.3.2.2.9.4.10.2.4 LOCAL DATA 115
3.3.2.2.9.4.10.2.5 PROCESS CONTROL 115
3.3.2.2.9.4.10.2.6 PROCESSING 115
3.3.2.2.9.4.10.2.7 UTILIZATION OF OTHER ELEMENTS 116
3.3.2.2.9.4.10.2.8 LIMITATIONS 117
3.3.2.2.9.5 LONGITUDE INTEGRATION PACKAGE DESIGN (CATALOG #P264-0) 117
3.3.2.2.9.5.1 REQUIREMENTS ALLOCATION 117
3.3.2.2.9.5.2 LOCAL ENTITIES DESIGN 117
3.3.2.2.9.5.3 INPUT/OUTPUT 117
3.3.2.2.9.5.4 LOCAL DATA 118
3.3.2.2.9.5.5 PROCESS CONTROL 118
3.3.2.2.9.5.6 PROCESSING 119
3.3.2.2.9.5.7 UTILIZATION OF OTHER ELEMENTS 119

3.3.2.2.9.5.8 LIMITATIONS 119
3.3.2.2.9.5.9 LLCSC DESIGN 119
3.3.2.2.9.5.10 UNIT DESIGN 119
3.3.2.2.9.5.10.1 REINITIALIZE UNIT DESIGN 119
3.3.2.2.9.5.10.1.1 REQUIREMENTS ALLOCATION 119
3.3.2.2.9.5.10.1.2 LOCAL ENTITIES DESIGN 119
3.3.2.2.9.5.10.1.3 INPUT/OUTPUT 119
3.3.2.2.9.5.10.1.4 LOCAL DATA 120
3.3.2.2.9.5.10.1.5 PROCESS CONTROL 120
3.3.2.2.9.5.10.1.6 PROCESSING 120
3.3.2.2.9.5.10.1.7 UTILIZATION OF OTHER ELEMENTS 120
3.3.2.2.9.5.10.1.8 LIMITATIONS 121
3.3.2.2.9.5.10.2 INTEGRATE UNIT DESIGN 121
3.3.2.2.9.5.10.2.1 REQUIREMENTS ALLOCATION 121
3.3.2.2.9.5.10.2.2 LOCAL ENTITIES DESIGN 122
3.3.2.2.9.5.10.2.3 INPUT/OUTPUT 122
3.3.2.2.9.5.10.2.4 LOCAL DATA 122
3.3.2.2.9.5.10.2.5 PROCESS CONTROL 122
3.3.2.2.9.5.10.2.6 PROCESSING 122
3.3.2.2.9.5.10.2.7 UTILIZATION OF OTHER ELEMENTS 123
3.3.2.2.9.5.10.2.8 LIMITATIONS 124
3.3.2.2.10 UNIT DESIGN 124
3.3.2.2.10.1 COMPUTE CORIOLIS ACCELERATION UNIT DESIGN (CATALOG #P258-0) 124
3.3.2.2.10.1.1 REQUIREMENTS ALLOCATION 124
3.3.2.2.10.1.2 LOCAL ENTITIES DESIGN 124
3.3.2.2.10.1.3 INPUT/OUTPUT 125
3.3.2.2.10.1.4 LOCAL DATA 126
3.3.2.2.10.1.5 PROCESS CONTROL 126
3.3.2.2.10.1.6 PROCESSING 126
3.3.2.2.10.1.7 UTILIZATION OF OTHER ELEMENTS 127
3.3.2.2.10.1.8 LIMITATIONS 127
3.3.2.2.10.2 TOTAL_PLATFORM_ROTATION_RATES UNIT DESIGN (CATALOG #P260-0) 127
3.3.2.2.10.2.1 REQUIREMENTS ALLOCATION 127
3.3.2.2.10.2.2 LOCAL ENTITIES DESIGN 127
3.3.2.2.10.2.3 INPUT/OUTPUT 127
3.3.2.2.10.2.4 LOCAL DATA 128
3.3.2.2.10.2.5 PROCESS CONTROL 128
3.3.2.2.10.2.6 PROCESSING 128
3.3.2.2.10.2.7 UTILIZATION OF OTHER ELEMENTS 128
3.3.2.2.10.2.8 LIMITATIONS 129

3.3.2.3 WANDER_AZIMUTH_NAVIGATION_PARTS (PACKAGE BODY) TLCSC P002 (CATALOG #P234-0) 137
3.3.2.3.1 REQUIREMENTS ALLOCATION 137
3.3.2.3.2 LOCAL ENTITIES DESIGN 137
3.3.2.3.3 INPUT/OUTPUT 137
3.3.2.3.4 LOCAL DATA 137
3.3.2.3.5 PROCESS CONTROL 137
3.3.2.3.6 PROCESSING 138
3.3.2.3.7 UTILIZATION OF OTHER ELEMENTS 140
3.3.2.3.8 LIMITATIONS 140
3.3.2.3.9 LLCSC DESIGN 140
3.3.2.3.9.1 CORIOLIS_ACCELERATION_FROM_TOTAL_RATES PACKAGE DESIGN (CATALOG #P240-0) 140
3.3.2.3.9.1.1 REQUIREMENTS ALLOCATION 140

3.3.2.3.9.1.2 LOCAL ENTITIES DESIGN 140
3.3.2.3.9.1.3 INPUT/OUTPUT 140
3.3.2.3.9.1.4 LOCAL DATA 141
3.3.2.3.9.1.5 PROCESS CONTROL 142
3.3.2.3.9.1.6 PROCESSING 142
3.3.2.3.9.1.7 UTILIZATION OF OTHER ELEMENTS 143
3.3.2.3.9.1.8 LIMITATIONS 143
3.3.2.3.9.1.9 LLCSC DESIGN 143
3.3.2.3.9.1.10 UNIT DESIGN 143
3.3.2.3.9.2 EARTH_ROTATION_RATE PACKAGE DESIGN (CATALOG #P243-0) 143
3.3.2.3.9.2.1 REQUIREMENTS ALLOCATION 143
3.3.2.3.9.2.2 LOCAL ENTITIES DESIGN 144
3.3.2.3.9.2.3 INPUT/OUTPUT 144
3.3.2.3.9.2.4 LOCAL DATA 145
3.3.2.3.9.2.5 PROCESS CONTROL 145
3.3.2.3.9.2.6 PROCESSING 145
3.3.2.3.9.2.7 UTILIZATION OF OTHER ELEMENTS 146
3.3.2.3.9.2.8 LIMITATIONS 146
3.3.2.3.9.2.9 LLCSC DESIGN 146
3.3.2.3.9.2.10 UNIT DESIGN 147
3.3.2.3.10 UNIT DESIGN 147
3.3.2.3.10.1 COMPUTE_EAST_VELOCITY UNIT DESIGN (CATALOG #P235-0) 147
3.3.2.3.10.1.1 REQUIREMENTS ALLOCATION 147
3.3.2.3.10.1.2 LOCAL ENTITIES DESIGN 147
3.3.2.3.10.1.3 INPUT/OUTPUT 147
3.3.2.3.10.1.4 LOCAL DATA 148
3.3.2.3.10.1.5 PROCESS CONTROL 148
3.3.2.3.10.1.6 PROCESSING 148
3.3.2.3.10.1.7 UTILIZATION OF OTHER ELEMENTS 149
3.3.2.3.10.1.8 LIMITATIONS 149
3.3.2.3.10.2 COMPUTE_NORTH_VELOCITY UNIT DESIGN (CATALOG #P236-0) 149
3.3.2.3.10.2.1 REQUIREMENTS ALLOCATION 149
3.3.2.3.10.2.2 LOCAL ENTITIES DESIGN 149
3.3.2.3.10.2.3 INPUT/OUTPUT 150
3.3.2.3.10.2.4 LOCAL DATA 151
3.3.2.3.10.2.5 PROCESS CONTROL 151
3.3.2.3.10.2.6 PROCESSING 151
3.3.2.3.10.2.7 UTILIZATION OF OTHER ELEMENTS 151
3.3.2.3.10.2.8 LIMITATIONS 152
3.3.2.3.10.3 COMPUTE_EARTH_RELATIVE_HORIZONTAL_VELOCITIES UNIT DESIGN (CATALOG #P237-0) 152
3.3.2.3.10.3.1 REQUIREMENTS ALLOCATION 152
3.3.2.3.10.3.2 LOCAL ENTITIES DESIGN 152
3.3.2.3.10.3.3 INPUT/OUTPUT 152
3.3.2.3.10.3.4 LOCAL DATA 153
3.3.2.3.10.3.5 PROCESS CONTROL 153
3.3.2.3.10.3.6 PROCESSING 153
3.3.2.3.10.3.7 UTILIZATION OF OTHER ELEMENTS 154
3.3.2.3.10.3.8 LIMITATIONS 154
3.3.2.3.10.4 COMPUTE_TOTAL_ANGULAR_VELOCITY UNIT DESIGN (CATALOG #P238-0) 154
3.3.2.3.10.4.1 REQUIREMENTS ALLOCATION 155
3.3.2.3.10.4.2 LOCAL ENTITIES DESIGN 155
3.3.2.3.10.4.3 INPUT/OUTPUT 155

3.3.2.3.10.4.4 LOCAL DATA 156
3.3.2.3.10.4.5 PROCESS CONTROL 156
3.3.2.3.10.4.6 PROCESSING 156
3.3.2.3.10.4.7 UTILIZATION OF OTHER ELEMENTS 156
3.3.2.3.10.4.8 LIMITATIONS 156
3.3.2.3.10.5 COMPUTE CORIOLIS ACCELERATION UNIT DESIGN (CATALOG #P239-0) 156
3.3.2.3.10.5.1 REQUIREMENTS ALLOCATION 157
3.3.2.3.10.5.2 LOCAL ENTITIES DESIGN 157
3.3.2.3.10.5.3 INPUT/OUTPUT 157
3.3.2.3.10.5.4 LOCAL DATA 158
3.3.2.3.10.5.5 PROCESS CONTROL 158
3.3.2.3.10.5.6 PROCESSING 158
3.3.2.3.10.5.7 UTILIZATION OF OTHER ELEMENTS 159
3.3.2.3.10.5.8 LIMITATIONS 159
3.3.2.3.10.6 COMPUTE CURVATURES UNIT DESIGN (CATALOG #P241-0) 159
3.3.2.3.10.6.1 REQUIREMENTS ALLOCATION 159
3.3.2.3.10.6.2 LOCAL ENTITIES DESIGN 159
3.3.2.3.10.6.3 INPUT/OUTPUT 159
3.3.2.3.10.6.4 LOCAL DATA 161
3.3.2.3.10.6.5 PROCESS CONTROL 161
3.3.2.3.10.6.6 PROCESSING 161
3.3.2.3.10.6.7 UTILIZATION OF OTHER ELEMENTS 162
3.3.2.3.10.6.8 LIMITATIONS 162
3.3.2.3.10.7 TOTAL PLATFORM ROTATION RATE (FUNCTION BODY) UNIT DESIGN (CATALOG #P242-0) 162
3.3.2.3.10.7.1 REQUIREMENTS ALLOCATION 163
3.3.2.3.10.7.2 LOCAL ENTITIES DESIGN 163
3.3.2.3.10.7.3 INPUT/OUTPUT 163
3.3.2.3.10.7.4 LOCAL DATA 164
3.3.2.3.10.7.5 PROCESS CONTROL 164
3.3.2.3.10.7.6 PROCESSING 164
3.3.2.3.10.7.7 UTILIZATION OF OTHER ELEMENTS 164
3.3.2.3.10.7.8 LIMITATIONS 164
3.3.2.3.10.8 COMPUTE EARTH RELATIVE NAVIGATION ROTATION RATE UNIT DESIGN (CATALOG #P244-0) 164
3.3.2.3.10.8.1 REQUIREMENTS ALLOCATION 165
3.3.2.3.10.8.2 LOCAL ENTITIES DESIGN 165
3.3.2.3.10.8.3 INPUT/OUTPUT 165
3.3.2.3.10.8.4 LOCAL DATA 166
3.3.2.3.10.8.5 PROCESS CONTROL 166
3.3.2.3.10.8.6 PROCESSING 166
3.3.2.3.10.8.7 UTILIZATION OF OTHER ELEMENTS 167
3.3.2.3.10.8.8 LIMITATIONS 167
3.3.2.3.10.9 COMPUTE LATITUDE UNIT DESIGN (CATALOG #P245-0) 167
3.3.2.3.10.9.1 REQUIREMENTS ALLOCATION 167
3.3.2.3.10.9.2 LOCAL ENTITIES DESIGN 167
3.3.2.3.10.9.3 INPUT/OUTPUT 167
3.3.2.3.10.9.4 LOCAL DATA 168
3.3.2.3.10.9.5 PROCESS CONTROL 168
3.3.2.3.10.9.6 PROCESSING 168
3.3.2.3.10.9.7 UTILIZATION OF OTHER ELEMENTS 169
3.3.2.3.10.9.8 LIMITATIONS 169
3.3.2.3.10.10 COMPUTE LATITUDE USING ARCTANGENT UNIT DESIGN (CATALOG #P246-0) 169
3.3.2.3.10.10.1 REQUIREMENTS ALLOCATION 169

3.3.2.3.10.10.2 LOCAL ENTITIES DESIGN 169
3.3.2.3.10.10.3 INPUT/OUTPUT 169
3.3.2.3.10.10.4 LOCAL DATA 170
3.3.2.3.10.10.5 PROCESS CONTROL 170
3.3.2.3.10.10.6 PROCESSING 170
3.3.2.3.10.10.7 UTILIZATION OF OTHER ELEMENTS 171
3.3.2.3.10.10.8 LIMITATIONS 171
3.3.2.3.10.11 COMPUTE_LONGITUDE UNIT DESIGN (CATALOG #P247-0) 171
3.3.2.3.10.11.1 REQUIREMENTS ALLOCATION 171
3.3.2.3.10.11.2 LOCAL ENTITIES DESIGN 171
3.3.2.3.10.11.3 INPUT/OUTPUT 171
3.3.2.3.10.11.4 LOCAL DATA 172
3.3.2.3.10.11.5 PROCESS CONTROL 173
3.3.2.3.10.11.6 PROCESSING 173
3.3.2.3.10.11.7 UTILIZATION OF OTHER ELEMENTS 173
3.3.2.3.10.11.8 LIMITATIONS 173
3.3.2.3.10.12 COMPUTE_WANDER_AZIMUTH_ANGLE UNIT DESIGN (CATALOG #P248-0) 173
3.3.2.3.10.12.1 REQUIREMENTS ALLOCATION 173
3.3.2.3.10.12.2 LOCAL ENTITIES DESIGN 173
3.3.2.3.10.12.3 INPUT/OUTPUT 174
3.3.2.3.10.12.4 LOCAL DATA 174
3.3.2.3.10.12.5 PROCESS CONTROL 175
3.3.2.3.10.12.6 PROCESSING 175
3.3.2.3.10.12.7 UTILIZATION OF OTHER ELEMENTS 175
3.3.2.3.10.12.8 LIMITATIONS 175
3.3.2.3.10.13 COMPUTE_EAST_VELOCITY_WITH_SIN_COS_IN UNIT DESIGN (CATALOG #P1099-0) 175
3.3.2.3.10.13.1 REQUIREMENTS ALLOCATION 175
3.3.2.3.10.13.2 LOCAL ENTITIES DESIGN 175
3.3.2.3.10.13.3 INPUT/OUTPUT 176
3.3.2.3.10.13.4 LOCAL DATA 176
3.3.2.3.10.13.5 PROCESS CONTROL 177
3.3.2.3.10.13.6 PROCESSING 177
3.3.2.3.10.13.7 UTILIZATION OF OTHER ELEMENTS 177
3.3.2.3.10.13.8 LIMITATIONS 177
3.3.2.3.10.14 COMPUTE_NORTH_VELOCITY_WITH_SIN_COS_IN UNIT DESIGN (CATALOG #P1101-0) 178
3.3.2.3.10.14.1 REQUIREMENTS ALLOCATION 178
3.3.2.3.10.14.2 LOCAL ENTITIES DESIGN 178
3.3.2.3.10.14.3 INPUT/OUTPUT 178
3.3.2.3.10.14.4 LOCAL DATA 179
3.3.2.3.10.14.5 PROCESS CONTROL 179
3.3.2.3.10.14.6 PROCESSING 179
3.3.2.3.10.14.7 UTILIZATION OF OTHER ELEMENTS 180
3.3.2.3.10.14.8 LIMITATIONS 180
3.3.2.3.10.15 COMPUTE_EARTH_RELATIVE_HORIZONTAL_VELOCITIES_WITH_SIN_COS_IN UNIT DESIGN (CATALOG #P1107-0) 180
3.3.2.3.10.15.1 REQUIREMENTS ALLOCATION 180
3.3.2.3.10.15.2 LOCAL ENTITIES DESIGN 180
3.3.2.3.10.15.3 INPUT/OUTPUT 180
3.3.2.3.10.15.4 LOCAL DATA 181
3.3.2.3.10.15.5 PROCESS CONTROL 182
3.3.2.3.10.15.6 PROCESSING 182
3.3.2.3.10.15.7 UTILIZATION OF OTHER ELEMENTS 182
3.3.2.3.10.15.8 LIMITATIONS 182

3.3.2.3.10.16 COMPUTE_LATITUDE_USING_TWO_VALUE_ARCTANGENT UNIT DESIGN (CATALOG #P1104-0) 182
3.3.2.3.10.16.1 REQUIREMENTS ALLOCATION 182
3.3.2.3.10.16.2 LOCAL ENTITIES DESIGN 183
3.3.2.3.10.16.3 INPUT/OUTPUT 183
3.3.2.3.10.16.4 LOCAL DATA 184
3.3.2.3.10.16.5 PROCESS CONTROL 184
3.3.2.3.10.16.6 PROCESSING 184
3.3.2.3.10.16.7 UTILIZATION OF OTHER ELEMENTS 185
3.3.2.3.10.16.8 LIMITATIONS 185
3.3.2.3.10.17 COMPUTE_LONGITUDE_USING_TWO_VALUE_ARCTANGENT UNIT DESIGN (CATALOG #P1106-0) 185
3.3.2.3.10.17.1 REQUIREMENTS ALLOCATION 185
3.3.2.3.10.17.2 LOCAL ENTITIES DESIGN 185
3.3.2.3.10.17.3 INPUT/OUTPUT 185
3.3.2.3.10.17.4 LOCAL DATA 186
3.3.2.3.10.17.5 PROCESS CONTROL 186
3.3.2.3.10.17.6 PROCESSING 186
3.3.2.3.10.17.7 UTILIZATION OF OTHER ELEMENTS 187
3.3.2.3.10.17.8 LIMITATIONS 187
3.3.2.3.10.18 COMPUTE_WANDER_AZIMUTH_ANGLE_USING_TWO_VALUE_ARCTANGENT UNIT DESIGN (CATALOG #P1109-0) 187
3.3.2.3.10.18.1 REQUIREMENTS ALLOCATION 187
3.3.2.3.10.18.2 LOCAL ENTITIES DESIGN 187
3.3.2.3.10.18.3 INPUT/OUTPUT 187
3.3.2.3.10.18.4 LOCAL DATA 188
3.3.2.3.10.18.5 PROCESS CONTROL 188
3.3.2.3.10.18.6 PROCESSING 188
3.3.2.3.10.18.7 UTILIZATION OF OTHER ELEMENTS 189
3.3.2.3.10.18.8 LIMITATIONS 189

3.3.2.4 DIRECTION_COSINE_MATRIX_OPERATIONS (PACKAGE BODY) TLCSC P644 (CATALOG #P286-0) 213
3.3.2.4.1 REQUIREMENTS ALLOCATION 213
3.3.2.4.2 LOCAL ENTITIES DESIGN 213
3.3.2.4.3 INPUT/OUTPUT 213
3.3.2.4.4 LOCAL DATA 214
3.3.2.4.5 PROCESS CONTROL 214
3.3.2.4.6 PROCESSING 214
3.3.2.4.7 UTILIZATION OF OTHER ELEMENTS 214
3.3.2.4.8 LIMITATIONS 214
3.3.2.4.9 LLCSC DESIGN 214

3.3.2.4.9.1 DCM_GENERAL_OPERATIONS PACKAGE DESIGN (CATALOG #P287-0) 215
3.3.2.4.9.1.1 REQUIREMENTS ALLOCATION 215
3.3.2.4.9.1.2 LOCAL ENTITIES DESIGN 215
3.3.2.4.9.1.3 INPUT/OUTPUT 215
3.3.2.4.9.1.4 LOCAL DATA 215
3.3.2.4.9.1.5 PROCESS CONTROL 215
3.3.2.4.9.1.6 PROCESSING 215
3.3.2.4.9.1.7 UTILIZATION OF OTHER ELEMENTS 216
3.3.2.4.9.1.8 LIMITATIONS 216
3.3.2.4.9.1.9 LLCSC DESIGN 216
3.3.2.4.9.1.9.1 DCM_TRAPEZOIDAL_INTEGRATION PACKAGE DESIGN (CATALOG #P289-0) 217
3.3.2.4.9.1.9.1.1 REQUIREMENTS ALLOCATION 217

3.3.2.4.9.1.9.1.2 LOCAL ENTITIES DESIGN 217
3.3.2.4.9.1.9.1.3 INPUT/OUTPUT 217
3.3.2.4.9.1.9.1.4 LOCAL DATA 220
3.3.2.4.9.1.9.1.5 PROCESS CONTROL 220
3.3.2.4.9.1.9.1.6 PROCESSING 220
3.3.2.4.9.1.9.1.7 UTILIZATION OF OTHER ELEMENTS 221
3.3.2.4.9.1.9.1.8 LIMITATIONS 221
3.3.2.4.9.1.9.1.9 LLCSC DESIGN 221
3.3.2.4.9.1.9.1.10 UNIT DESIGN 221
3.3.2.4.9.1.9.1.10.1 REINITIALIZE_ANGULAR_VELOCITIES UNIT DESIGN 221
3.3.2.4.9.1.9.1.10.1.1 REQUIREMENTS ALLOCATION 221
3.3.2.4.9.1.9.1.10.1.2 LOCAL ENTITIES DESIGN 221
3.3.2.4.9.1.9.1.10.1.3 INPUT/OUTPUT 221
3.3.2.4.9.1.9.1.10.1.4 LOCAL DATA 222
3.3.2.4.9.1.9.1.10.1.5 PROCESS CONTROL 222
3.3.2.4.9.1.9.1.10.1.6 PROCESSING 222
3.3.2.4.9.1.9.1.10.1.7 UTILIZATION OF OTHER ELEMENTS 222
3.3.2.4.9.1.9.1.10.1.8 LIMITATIONS 223
3.3.2.4.9.1.9.1.10.2 PERFORM_TRAPEZOIDAL_INTEGRATION_OF_DCM UNIT DESIGN 223
3.3.2.4.9.1.9.1.10.2.1 REQUIREMENTS ALLOCATION 223
3.3.2.4.9.1.9.1.10.2.2 LOCAL ENTITIES DESIGN 223
3.3.2.4.9.1.9.1.10.2.3 INPUT/OUTPUT 223
3.3.2.4.9.1.9.1.10.2.4 LOCAL DATA 226
3.3.2.4.9.1.9.1.10.2.5 PROCESS CONTROL 227
3.3.2.4.9.1.9.1.10.2.6 PROCESSING 227
3.3.2.4.9.1.9.1.10.2.7 UTILIZATION OF OTHER ELEMENTS 228
3.3.2.4.9.1.9.1.10.2.8 LIMITATIONS 229
3.3.2.4.9.1.10 UNIT DESIGN 229
3.3.2.4.9.1.10.1 DCM_INITIALIZED_FROM_REFERENCE UNIT DESIGN (CATALOG #P288-0) 229
3.3.2.4.9.1.10.1.1 REQUIREMENTS ALLOCATION 229
3.3.2.4.9.1.10.1.2 LOCAL ENTITIES DESIGN 230
3.3.2.4.9.1.10.1.3 INPUT/OUTPUT 230
3.3.2.4.9.1.10.1.4 LOCAL DATA 232
3.3.2.4.9.1.10.1.5 PROCESS CONTROL 233
3.3.2.4.9.1.10.1.6 PROCESSING 233
3.3.2.4.9.1.10.1.7 UTILIZATION OF OTHER ELEMENTS 234
3.3.2.4.9.1.10.1.8 LIMITATIONS 234
3.3.2.4.9.1.10.2 PERFORM_RECTANGULAR_INTEGRATION_OF_DCM UNIT DESIGN (CATALOG #P290-0) 235
3.3.2.4.9.1.10.2.1 REQUIREMENTS ALLOCATION 235
3.3.2.4.9.1.10.2.2 LOCAL ENTITIES DESIGN 235
3.3.2.4.9.1.10.2.3 INPUT/OUTPUT 235
3.3.2.4.9.1.10.2.4 LOCAL DATA 237
3.3.2.4.9.1.10.2.5 PROCESS CONTROL 237
3.3.2.4.9.1.10.2.6 PROCESSING 238
3.3.2.4.9.1.10.2.7 UTILIZATION OF OTHER ELEMENTS 239
3.3.2.4.9.1.10.2.8 LIMITATIONS 239
3.3.2.4.9.1.10.3 REORTHONORMALIZE_DCM UNIT DESIGN (CATALOG #P291-0) 239
3.3.2.4.9.1.10.3.1 REQUIREMENTS ALLOCATION 239
3.3.2.4.9.1.10.3.2 LOCAL ENTITIES DESIGN 239
3.3.2.4.9.1.10.3.3 INPUT/OUTPUT 239
3.3.2.4.9.1.10.3.4 LOCAL DATA 242
3.3.2.4.9.1.10.3.5 PROCESS CONTROL 242

3.3.2.4.9.1.10.3.6 PROCESSING 242
3.3.2.4.9.1.10.3.7 UTILIZATION OF OTHER ELEMENTS 244
3.3.2.4.9.1.10.3.8 LIMITATIONS 244
3.3.2.4.9.1.10.4 FRAME_MISALIGNMENT UNIT DESIGN (CATALOG #P292-0) 244
3.3.2.4.9.1.10.4.1 REQUIREMENTS ALLOCATION 244
3.3.2.4.9.1.10.4.2 LOCAL ENTITIES DESIGN 244
3.3.2.4.9.1.10.4.3 INPUT/OUTPUT 244
3.3.2.4.9.1.10.4.4 LOCAL DATA 247
3.3.2.4.9.1.10.4.5 PROCESS CONTROL 247
3.3.2.4.9.1.10.4.6 PROCESSING 247
3.3.2.4.9.1.10.4.7 UTILIZATION OF OTHER ELEMENTS 248
3.3.2.4.9.1.10.4.8 LIMITATIONS 248
3.3.2.4.9.1.10.5 ALIGNED_DCM_MATRIX UNIT DESIGN (CATALOG #P293-0) 248
3.3.2.4.9.1.10.5.1 REQUIREMENTS ALLOCATION 248
3.3.2.4.9.1.10.5.2 LOCAL ENTITIES DESIGN 249
3.3.2.4.9.1.10.5.3 INPUT/OUTPUT 249
3.3.2.4.9.1.10.5.4 LOCAL DATA 251
3.3.2.4.9.1.10.5.5 PROCESS CONTROL 251
3.3.2.4.9.1.10.5.6 PROCESSING 251
3.3.2.4.9.1.10.5.7 UTILIZATION OF OTHER ELEMENTS 252
3.3.2.4.9.1.10.5.8 LIMITATIONS 252
3.3.2.4.9.1.10.6 COMPUTE_FIRST_ROW_FROM_ORTHONORMAL UNIT DESIGN (CATALOG #P294-0) 253
3.3.2.4.9.1.10.6.1 REQUIREMENTS ALLOCATION 253
3.3.2.4.9.1.10.6.2 LOCAL ENTITIES DESIGN 253
3.3.2.4.9.1.10.6.3 INPUT/OUTPUT 253
3.3.2.4.9.1.10.6.4 LOCAL DATA 255
3.3.2.4.9.1.10.6.5 PROCESS CONTROL 255
3.3.2.4.9.1.10.6.6 PROCESSING 255
3.3.2.4.9.1.10.6.7 UTILIZATION OF OTHER ELEMENTS 255
3.3.2.4.9.1.10.6.8 LIMITATIONS 255
3.3.2.4.9.1.10.7 DCM_FROM_QUATERNION UNIT DESIGN (CATALOG #P295-0) 255
3.3.2.4.9.1.10.7.1 REQUIREMENTS ALLOCATION 255
3.3.2.4.9.1.10.7.2 LOCAL ENTITIES DESIGN 256
3.3.2.4.9.1.10.7.3 INPUT/OUTPUT 256
3.3.2.4.9.1.10.7.4 LOCAL DATA 258
3.3.2.4.9.1.10.7.5 PROCESS CONTROL 258
3.3.2.4.9.1.10.7.6 PROCESSING 258
3.3.2.4.9.1.10.7.7 UTILIZATION OF OTHER ELEMENTS 260
3.3.2.4.9.1.10.7.8 LIMITATIONS 260

3.3.2.4.9.2 CNE_OPERATIONS PACKAGE DESIGN (CATALOG #P296-0) 261
3.3.2.4.9.2.1 REQUIREMENTS ALLOCATION 261
3.3.2.4.9.2.2 LOCAL ENTITIES DESIGN 261
3.3.2.4.9.2.3 INPUT/OUTPUT 262
3.3.2.4.9.2.4 LOCAL DATA 265
3.3.2.4.9.2.5 PROCESS CONTROL 265
3.3.2.4.9.2.6 PROCESSING 265
3.3.2.4.9.2.7 UTILIZATION OF OTHER ELEMENTS 267
3.3.2.4.9.2.8 LIMITATIONS 267
3.3.2.4.9.2.9 LLCSC DESIGN 267
3.3.2.4.9.2.9.1 CNE_INTEGRATION PACKAGE DESIGN (CATALOG #P298-0) 267
3.3.2.4.9.2.9.1.1 REQUIREMENTS ALLOCATION 267

3.3.2.4.9.2.9.1.2 LOCAL ENTITIES DESIGN 267
3.3.2.4.9.2.9.1.3 INPUT/OUTPUT 267
3.3.2.4.9.2.9.1.4 LOCAL DATA 270
3.3.2.4.9.2.9.1.5 PROCESS CONTROL 270
3.3.2.4.9.2.9.1.6 PROCESSING 270
3.3.2.4.9.2.9.1.7 UTILIZATION OF OTHER ELEMENTS 271
3.3.2.4.9.2.9.1.8 LIMITATIONS 272
3.3.2.4.9.2.9.1.9 LLCSC DESIGN 272
3.3.2.4.9.2.9.1.10 UNIT DESIGN 272
3.3.2.4.9.2.9.2 ALIGNMENT_PARTS PACKAGE DESIGN (CATALOG #P299-0) 272
3.3.2.4.9.2.9.2.1 REQUIREMENTS ALLOCATION 272
3.3.2.4.9.2.9.2.2 LOCAL ENTITIES DESIGN 273
3.3.2.4.9.2.9.2.3 INPUT/OUTPUT 273
3.3.2.4.9.2.9.2.4 LOCAL DATA 274
3.3.2.4.9.2.9.2.5 PROCESS CONTROL 274
3.3.2.4.9.2.9.2.6 PROCESSING 275
3.3.2.4.9.2.9.2.7 UTILIZATION OF OTHER ELEMENTS 276
3.3.2.4.9.2.9.2.8 LIMITATIONS 276
3.3.2.4.9.2.9.2.9 LLCSC DESIGN 276
3.3.2.4.9.2.9.2.10 UNIT DESIGN 276
3.3.2.4.9.2.9.3 CNE_FROM_QUATERNION PACKAGE DESIGN (CATALOG #P300-0) 277
3.3.2.4.9.2.9.3.1 REQUIREMENTS ALLOCATION 277
3.3.2.4.9.2.9.3.2 LOCAL ENTITIES DESIGN 277
3.3.2.4.9.2.9.3.3 INPUT/OUTPUT 277
3.3.2.4.9.2.9.3.4 LOCAL DATA 278
3.3.2.4.9.2.9.3.5 PROCESS CONTROL 278
3.3.2.4.9.2.9.3.6 PROCESSING 278
3.3.2.4.9.2.9.3.7 UTILIZATION OF OTHER ELEMENTS 279
3.3.2.4.9.2.9.3.8 LIMITATIONS 280
3.3.2.4.9.2.9.3.9 LLCSC DESIGN 280
3.3.2.4.9.2.9.3.10 UNIT DESIGN 280
3.3.2.4.9.2.10 UNIT DESIGN 280
3.3.2.4.9.2.10.1 CNE_INITIALIZED_FROM_EARTH_POSITION UNIT DESIGN (CATALOG #P297-0) 280
3.3.2.4.9.2.10.1.1 REQUIREMENTS ALLOCATION 280
3.3.2.4.9.2.10.1.2 LOCAL ENTITIES DESIGN 280
3.3.2.4.9.2.10.1.3 INPUT/OUTPUT 280
3.3.2.4.9.2.10.1.4 LOCAL DATA 281
3.3.2.4.9.2.10.1.5 PROCESS CONTROL 282
3.3.2.4.9.2.10.1.6 PROCESSING 282
3.3.2.4.9.2.10.1.7 UTILIZATION OF OTHER ELEMENTS 283
3.3.2.4.9.2.10.1.8 LIMITATIONS 283

3.3.2.4.10 UNIT DESIGN 285

3.3.3 KALMAN FILTER 311

3.3.3.1 KALMAN FILTER COMMON PARTS (BODY) TLCSC 651 (CATALOG #P163-0) 313
3.3.3.1.1 REQUIREMENTS ALLOCATION 313
3.3.3.1.2 LOCAL ENTITIES DESIGN 313
3.3.3.1.3 INPUT/OUTPUT 313
3.3.3.1.4 LOCAL DATA 313
3.3.3.1.5 PROCESS CONTROL 313
3.3.3.1.6 PROCESSING 313

3.3.3.1.7 UTILIZATION OF OTHER ELEMENTS 314
3.3.3.1.8 LIMITATIONS 314
3.3.3.1.9 LLCSC DESIGN 314
3.3.3.1.9.1 STATE_TRANSITION_AND_PROCESS_NOISE_MATRICES_MANAGER PACKAGE DESIGN (CATALOG #P164-0) 314
3.3.3.1.9.1.1 REQUIREMENTS ALLOCATION 314
3.3.3.1.9.1.2 LOCAL ENTITIES DESIGN 314
3.3.3.1.9.1.3 INPUT/OUTPUT 314
3.3.3.1.9.1.4 LOCAL DATA 315
3.3.3.1.9.1.5 PROCESS CONTROL 315
3.3.3.1.9.1.6 PROCESSING 316
3.3.3.1.9.1.7 UTILIZATION OF OTHER ELEMENTS 316
3.3.3.1.9.1.8 LIMITATIONS 316
3.3.3.1.9.1.9 LLCSC DESIGN 316
3.3.3.1.9.1.10 UNIT DESIGN 316
3.3.3.1.9.1.10.1 INITIALIZE UNIT DESIGN 316
3.3.3.1.9.1.10.1.1 REQUIREMENTS ALLOCATION 316
3.3.3.1.9.1.10.1.2 LOCAL ENTITIES DESIGN 316
3.3.3.1.9.1.10.1.3 INPUT/OUTPUT 316
3.3.3.1.9.1.10.1.4 LOCAL DATA 316
3.3.3.1.9.1.10.1.5 PROCESS CONTROL 317
3.3.3.1.9.1.10.1.6 PROCESSING 317
3.3.3.1.9.1.10.1.7 UTILIZATION OF OTHER ELEMENTS 317
3.3.3.1.9.1.10.1.8 LIMITATIONS 318
3.3.3.1.9.1.10.2 PROPAGATE UNIT DESIGN 318
3.3.3.1.9.1.10.2.1 REQUIREMENTS ALLOCATION 318
3.3.3.1.9.1.10.2.2 LOCAL ENTITIES DESIGN 318
3.3.3.1.9.1.10.2.3 INPUT/OUTPUT 318
3.3.3.1.9.1.10.2.4 LOCAL DATA 319
3.3.3.1.9.1.10.2.5 PROCESS CONTROL 319
3.3.3.1.9.1.10.2.6 PROCESSING 319
3.3.3.1.9.1.10.2.7 UTILIZATION OF OTHER ELEMENTS 319
3.3.3.1.9.1.10.2.8 LIMITATIONS 320
3.3.3.1.9.1.10.3 GET_CURRENT_UNIT DESIGN 320
3.3.3.1.9.1.10.3.1 REQUIREMENTS ALLOCATION 320
3.3.3.1.9.1.10.3.2 LOCAL ENTITIES DESIGN 320
3.3.3.1.9.1.10.3.3 INPUT/OUTPUT 321
3.3.3.1.9.1.10.3.4 LOCAL DATA 321
3.3.3.1.9.1.10.3.5 PROCESS CONTROL 321
3.3.3.1.9.1.10.3.6 PROCESSING 321
3.3.3.1.9.1.10.3.7 UTILIZATION OF OTHER ELEMENTS 321
3.3.3.1.9.1.10.3.8 LIMITATIONS 322
3.3.3.1.9.1.10.4 PROPAGATED_PHI UNIT DESIGN 322
3.3.3.1.9.1.10.4.1 REQUIREMENTS ALLOCATION 322
3.3.3.1.9.1.10.4.2 LOCAL ENTITIES DESIGN 322
3.3.3.1.9.1.10.4.3 INPUT/OUTPUT 322
3.3.3.1.9.1.10.4.4 LOCAL DATA 323
3.3.3.1.9.1.10.4.5 PROCESS CONTROL 323
3.3.3.1.9.1.10.4.6 PROCESSING 323
3.3.3.1.9.1.10.4.7 UTILIZATION OF OTHER ELEMENTS 323
3.3.3.1.9.1.10.4.8 LIMITATIONS 324
3.3.3.1.9.2 ERROR_COVARIANCE_MATRIX_MANAGER PACKAGE DESIGN (CATALOG #P165-0) 324
3.3.3.1.9.2.1 REQUIREMENTS ALLOCATION 324
3.3.3.1.9.2.2 LOCAL ENTITIES DESIGN 324
3.3.3.1.9.2.3 INPUT/OUTPUT 324
3.3.3.1.9.2.4 LOCAL DATA 325

3.3.3.1.9.2.5 PROCESS CONTROL 325
3.3.3.1.9.2.6 PROCESSING 325
3.3.3.1.9.2.7 UTILIZATION OF OTHER ELEMENTS 325
3.3.3.1.9.2.8 LIMITATIONS 325
3.3.3.1.9.2.9 LLCSC DESIGN 325
3.3.3.1.9.2.10 UNIT DESIGN 325
3.3.3.1.9.2.10.1 INITIALIZE UNIT DESIGN 325
3.3.3.1.9.2.10.1.1 REQUIREMENTS ALLOCATION 325
3.3.3.1.9.2.10.1.2 LOCAL ENTITIES DESIGN 326
3.3.3.1.9.2.10.1.3 INPUT/OUTPUT 326
3.3.3.1.9.2.10.1.4 LOCAL DATA 326
3.3.3.1.9.2.10.1.5 PROCESS CONTROL 326
3.3.3.1.9.2.10.1.6 PROCESSING 326
3.3.3.1.9.2.10.1.7 UTILIZATION OF OTHER ELEMENTS 326
3.3.3.1.9.2.10.1.8 LIMITATIONS 327
3.3.3.1.9.2.10.2 PROPAGATE UNIT DESIGN 327
3.3.3.1.9.2.10.2.1 REQUIREMENTS ALLOCATION 327
3.3.3.1.9.2.10.2.2 LOCAL ENTITIES DESIGN 327
3.3.3.1.9.2.10.2.3 INPUT/OUTPUT 327
3.3.3.1.9.2.10.2.4 LOCAL DATA 328
3.3.3.1.9.2.10.2.5 PROCESS CONTROL 328
3.3.3.1.9.2.10.2.6 PROCESSING 328
3.3.3.1.9.2.10.2.7 UTILIZATION OF OTHER ELEMENTS 328
3.3.3.1.9.2.10.2.8 LIMITATIONS 329
3.3.3.1.9.2.10.3 P UNIT DESIGN 329
3.3.3.1.9.2.10.3.1 REQUIREMENTS ALLOCATION 329
3.3.3.1.9.2.10.3.2 LOCAL ENTITIES DESIGN 329
3.3.3.1.9.2.10.3.3 INPUT/OUTPUT 329
3.3.3.1.9.2.10.3.4 LOCAL DATA 329
3.3.3.1.9.2.10.3.5 PROCESS CONTROL 329
3.3.3.1.9.2.10.3.6 PROCESSING 329
3.3.3.1.9.2.10.3.7 UTILIZATION OF OTHER ELEMENTS 330
3.3.3.1.9.2.10.3.8 LIMITATIONS 330
3.3.3.1.9.3 STATE_TRANSITION_MATRIX_MANAGER PACKAGE DESIGN (CATALOG #P166-0) 330
3.3.3.1.9.3.1 REQUIREMENTS ALLOCATION 330
3.3.3.1.9.3.2 LOCAL ENTITIES DESIGN 330
3.3.3.1.9.3.3 INPUT/OUTPUT 331
3.3.3.1.9.3.4 LOCAL DATA 331
3.3.3.1.9.3.5 PROCESS CONTROL 331
3.3.3.1.9.3.6 PROCESSING 331
3.3.3.1.9.3.7 UTILIZATION OF OTHER ELEMENTS 332
3.3.3.1.9.3.8 LIMITATIONS 332
3.3.3.1.9.3.9 LLCSC DESIGN 332
3.3.3.1.9.3.10 UNIT DESIGN 332
3.3.3.1.9.3.10.1 INITIALIZE UNIT DESIGN 332
3.3.3.1.9.3.10.1.1 REQUIREMENTS ALLOCATION 332
3.3.3.1.9.3.10.1.2 LOCAL ENTITIES DESIGN 332
3.3.3.1.9.3.10.1.3 INPUT/OUTPUT 332
3.3.3.1.9.3.10.1.4 LOCAL DATA 332
3.3.3.1.9.3.10.1.5 PROCESS CONTROL 332
3.3.3.1.9.3.10.1.6 PROCESSING 333
3.3.3.1.9.3.10.1.7 UTILIZATION OF OTHER ELEMENTS 333
3.3.3.1.9.3.10.1.8 LIMITATIONS 333
3.3.3.1.9.3.10.2 PROPAGATE UNIT DESIGN 333
3.3.3.1.9.3.10.2.1 REQUIREMENTS ALLOCATION 334
3.3.3.1.9.3.10.2.2 LOCAL ENTITIES DESIGN 334

3.3.3.1.9.3.10.2.3 INPUT/OUTPUT 334
3.3.3.1.9.3.10.2.4 LOCAL DATA 334
3.3.3.1.9.3.10.2.5 PROCESS CONTROL 334
3.3.3.1.9.3.10.2.6 PROCESSING 334
3.3.3.1.9.3.10.2.7 UTILIZATION OF OTHER ELEMENTS 334
3.3.3.1.9.3.10.2.8 LIMITATIONS 335
3.3.3.1.9.3.10.3 PROPAGATED PHI UNIT DESIGN 335
3.3.3.1.9.3.10.3.1 REQUIREMENTS ALLOCATION 335
3.3.3.1.9.3.10.3.2 LOCAL ENTITIES DESIGN 335
3.3.3.1.9.3.10.3.3 INPUT/OUTPUT 335
3.3.3.1.9.3.10.3.4 LOCAL DATA 336
3.3.3.1.9.3.10.3.5 PROCESS CONTROL 336
3.3.3.1.9.3.10.3.6 PROCESSING 336
3.3.3.1.9.3.10.3.7 UTILIZATION OF OTHER ELEMENTS 336
3.3.3.1.9.3.10.3.8 LIMITATIONS 337
3.3.3.1.10 UNIT DESIGN 337

3.3.3.2 KALMAN FILTER COMPACT H PARTS (BODY) TLCSC (CATALOG #P137-0) 345
3.3.3.2.1 REQUIREMENTS ALLOCATION 345
3.3.3.2.2 LOCAL ENTITIES DESIGN 345
3.3.3.2.3 INPUT/OUTPUT 345
3.3.3.2.4 LOCAL DATA 345
3.3.3.2.5 PROCESS CONTROL 345
3.3.3.2.6 PROCESSING 345
3.3.3.2.7 UTILIZATION OF OTHER ELEMENTS 346
3.3.3.2.8 LIMITATIONS 346
3.3.3.2.9 LLCSC DESIGN 346
3.3.3.2.9.1 SEQUENTIALLY UPDATE COVARIANCE MATRIX AND STATE VECTOR PACKAGE DESIGN (CATALOG #P141-0) 346
3.3.3.2.9.1.1 REQUIREMENTS ALLOCATION 347
3.3.3.2.9.1.2 LOCAL ENTITIES DESIGN 347
3.3.3.2.9.1.3 INPUT/OUTPUT 347
3.3.3.2.9.1.4 LOCAL DATA 348
3.3.3.2.9.1.5 PROCESS CONTROL 348
3.3.3.2.9.1.6 PROCESSING 348
3.3.3.2.9.1.7 UTILIZATION OF OTHER ELEMENTS 349
3.3.3.2.9.1.8 LIMITATIONS 349
3.3.3.2.9.1.9 LLCSC DESIGN 350
3.3.3.2.9.1.10 UNIT DESIGN 350
3.3.3.2.9.1.10.1 UPDATE UNIT DESIGN 350
3.3.3.2.9.1.10.1.1 REQUIREMENTS ALLOCATION 350
3.3.3.2.9.1.10.1.2 LOCAL ENTITIES DESIGN 350
3.3.3.2.9.1.10.1.3 INPUT/OUTPUT 350
3.3.3.2.9.1.10.1.4 LOCAL DATA 350
3.3.3.2.9.1.10.1.5 PROCESS CONTROL 351
3.3.3.2.9.1.10.1.6 PROCESSING 351
3.3.3.2.9.1.10.1.7 UTILIZATION OF OTHER ELEMENTS 351
3.3.3.2.9.1.10.1.8 LIMITATIONS 353
3.3.3.2.9.2 KALMAN UPDATE (BODY) PACKAGE DESIGN (CATALOG #P142-0) 353
3.3.3.2.9.2.1 REQUIREMENTS ALLOCATION 353
3.3.3.2.9.2.2 LOCAL ENTITIES DESIGN 353
3.3.3.2.9.2.3 INPUT/OUTPUT 353
3.3.3.2.9.2.4 LOCAL DATA 355
3.3.3.2.9.2.5 PROCESS CONTROL 355
3.3.3.2.9.2.6 PROCESSING 355

3.3.3.2.9.2.7 UTILIZATION OF OTHER ELEMENTS 356
3.3.3.2.9.2.8 LIMITATIONS 356
3.3.3.2.9.2.9 LLCSC DESIGN 357
3.3.3.2.9.2.10 UNIT DESIGN 357
3.3.3.2.9.2.10.1 UPDATE UNIT DESIGN 357
3.3.3.2.9.2.10.1.1 REQUIREMENTS ALLOCATION 357
3.3.3.2.9.2.10.1.2 LOCAL ENTITIES DESIGN 357
3.3.3.2.9.2.10.1.3 INPUT/OUTPUT 357
3.3.3.2.9.2.10.1.4 LOCAL DATA 358
3.3.3.2.9.2.10.1.5 PROCESS CONTROL 358
3.3.3.2.9.2.10.1.6 PROCESSING 358
3.3.3.2.9.2.10.1.7 UTILIZATION OF OTHER ELEMENTS 359
3.3.3.2.9.2.10.1.8 LIMITATIONS 360
3.3.3.2.10 UNIT DESIGN 360
3.3.3.2.10.1 COMPUTE_KALMAN_GAIN UNIT DESIGN (CATALOG #P138-0) 360
3.3.3.2.10.1.1 REQUIREMENTS ALLOCATION 361
3.3.3.2.10.1.2 LOCAL ENTITIES DESIGN 361
3.3.3.2.10.1.3 INPUT/OUTPUT 361
3.3.3.2.10.1.4 LOCAL DATA 362
3.3.3.2.10.1.5 PROCESS CONTROL 362
3.3.3.2.10.1.6 PROCESSING 362
3.3.3.2.10.1.7 UTILIZATION OF OTHER ELEMENTS 362
3.3.3.2.10.1.8 LIMITATIONS 362
3.3.3.2.10.2 UPDATE_ERROR_COVARIANCE_MATRIX UNIT DESIGN (CATALOG #P139-0) 362
3.3.3.2.10.2.1 REQUIREMENTS ALLOCATION 363
3.3.3.2.10.2.2 LOCAL ENTITIES DESIGN 363
3.3.3.2.10.2.3 INPUT/OUTPUT 363
3.3.3.2.10.2.4 LOCAL DATA 364
3.3.3.2.10.2.5 PROCESS CONTROL 364
3.3.3.2.10.2.6 PROCESSING 364
3.3.3.2.10.2.7 UTILIZATION OF OTHER ELEMENTS 364
3.3.3.2.10.2.8 LIMITATIONS 364
3.3.3.2.10.3 UPDATE_STATE_VECTOR UNIT DESIGN (CATALOG #P140-0) 365
3.3.3.2.10.3.1 REQUIREMENTS ALLOCATION 365
3.3.3.2.10.3.2 LOCAL ENTITIES DESIGN 365
3.3.3.2.10.3.3 INPUT/OUTPUT 365
3.3.3.2.10.3.4 LOCAL DATA 366
3.3.3.2.10.3.5 PROCESS CONTROL 366
3.3.3.2.10.3.6 PROCESSING 366
3.3.3.2.10.3.7 UTILIZATION OF OTHER ELEMENTS 366
3.3.3.2.10.3.8 LIMITATIONS 366
3.3.3.2.10.4 UPDATE_ERROR_COVARIANCE_MATRIX_GENERAL_FORM UNIT DESIGN (CATALOG #P1121-0) 367
3.3.3.2.10.4.1 REQUIREMENTS ALLOCATION 367
3.3.3.2.10.4.2 LOCAL ENTITIES DESIGN 367
3.3.3.2.10.4.3 INPUT/OUTPUT 367
3.3.3.2.10.4.4 LOCAL DATA 369
3.3.3.2.10.4.5 PROCESS CONTROL 369
3.3.3.2.10.4.6 PROCESSING 369
3.3.3.2.10.4.7 UTILIZATION OF OTHER ELEMENTS 370
3.3.3.2.10.4.8 LIMITATIONS 370

3.3.3.3 KALMAN FILTER COMPLICATED H PARTS (BODY) TLCSC (CATALOG #P149-0) 381

3.3.3.3.1 REQUIREMENTS ALLOCATION 381
3.3.3.3.2 LOCAL ENTITIES DESIGN 381
3.3.3.3.3 INPUT/OUTPUT 381
3.3.3.3.4 LOCAL DATA 381
3.3.3.3.5 PROCESS CONTROL 381
3.3.3.3.6 PROCESSING 381
3.3.3.3.7 UTILIZATION OF OTHER ELEMENTS 382
3.3.3.3.8 LIMITATIONS 382
3.3.3.3.9 LLCSC DESIGN 382
3.3.3.3.9.1 SEQUENTIALLY UPDATE COVARIANCE MATRIX AND STATE VECTOR PACKAGE DESIGN (CATALOG #P153-0) 382
3.3.3.3.9.1.1 REQUIREMENTS ALLOCATION 383
3.3.3.3.9.1.2 LOCAL ENTITIES DESIGN 383
3.3.3.3.9.1.3 INPUT/OUTPUT 383
3.3.3.3.9.1.4 LOCAL DATA 385
3.3.3.3.9.1.5 PROCESS CONTROL 385
3.3.3.3.9.1.6 PROCESSING 385
3.3.3.3.9.1.7 UTILIZATION OF OTHER ELEMENTS 386
3.3.3.3.9.1.8 LIMITATIONS 387
3.3.3.3.9.1.9 LLCSC DESIGN 387
3.3.3.3.9.1.10 UNIT DESIGN 387
3.3.3.3.9.1.10.1 UPDATE UNIT DESIGN 387
3.3.3.3.9.1.10.1.1 REQUIREMENTS ALLOCATION 387
3.3.3.3.9.1.10.1.2 LOCAL ENTITIES DESIGN 387
3.3.3.3.9.1.10.1.3 INPUT/OUTPUT 387
3.3.3.3.9.1.10.1.4 LOCAL DATA 388
3.3.3.3.9.1.10.1.5 PROCESS CONTROL 388
3.3.3.3.9.1.10.1.6 PROCESSING 388
3.3.3.3.9.1.10.1.7 UTILIZATION OF OTHER ELEMENTS 389
3.3.3.3.9.1.10.1.8 LIMITATIONS 390
3.3.3.3.9.2 KALMAN UPDATE PACKAGE DESIGN (CATALOG #P155-0) 390
3.3.3.3.9.2.1 REQUIREMENTS ALLOCATION 391
3.3.3.3.9.2.2 LOCAL ENTITIES DESIGN 391
3.3.3.3.9.2.3 INPUT/OUTPUT 391
3.3.3.3.9.2.4 LOCAL DATA 393
3.3.3.3.9.2.5 PROCESS CONTROL 394
3.3.3.3.9.2.6 PROCESSING 394
3.3.3.3.9.2.7 UTILIZATION OF OTHER ELEMENTS 394
3.3.3.3.9.2.8 LIMITATIONS 395
3.3.3.3.9.2.9 LLCSC DESIGN 395
3.3.3.3.9.2.10 UNIT DESIGN 395
3.3.3.3.9.2.10.1 UPDATE UNIT DESIGN 395
3.3.3.3.9.2.10.1.1 REQUIREMENTS ALLOCATION 395
3.3.3.3.9.2.10.1.2 LOCAL ENTITIES DESIGN 395
3.3.3.3.9.2.10.1.3 INPUT/OUTPUT 396
3.3.3.3.9.2.10.1.4 LOCAL DATA 396
3.3.3.3.9.2.10.1.5 PROCESS CONTROL 396
3.3.3.3.9.2.10.1.6 PROCESSING 396
3.3.3.3.9.2.10.1.7 UTILIZATION OF OTHER ELEMENTS 397
3.3.3.3.9.2.10.1.8 LIMITATIONS 399
3.3.3.3.10 UNIT DESIGN 399
3.3.3.3.10.1 COMPUTE_KALMAN_GAIN (BODY) UNIT DESIGN (CATALOG #P150-0) 399
3.3.3.3.10.1.1 REQUIREMENTS ALLOCATION 399
3.3.3.3.10.1.2 LOCAL ENTITIES DESIGN 399
3.3.3.3.10.1.3 INPUT/OUTPUT 399
3.3.3.3.10.1.4 LOCAL DATA 400

3.3.3.3.10.1.5 PROCESS CONTROL 401
3.3.3.3.10.1.6 PROCESSING 401
3.3.3.3.10.1.7 UTILIZATION OF OTHER ELEMENTS 402
3.3.3.3.10.1.8 LIMITATIONS 402
3.3.3.3.10.2 UPDATE_ERROR_COVARIANCE_MATRIX (BODY) UNIT DESIGN (CATALOG #P151-0) 402
3.3.3.3.10.2.1 REQUIREMENTS ALLOCATION 402
3.3.3.3.10.2.2 LOCAL ENTITIES DESIGN 402
3.3.3.3.10.2.3 INPUT/OUTPUT 402
3.3.3.3.10.2.4 LOCAL DATA 403
3.3.3.3.10.2.5 PROCESS CONTROL 404
3.3.3.3.10.2.6 PROCESSING 404
3.3.3.3.10.2.7 UTILIZATION OF OTHER ELEMENTS 404
3.3.3.3.10.2.8 LIMITATIONS 404
3.3.3.3.10.3 UPDATE_STATE_VECTOR UNIT DESIGN (CATALOG #P152-0) 404
3.3.3.3.10.3.1 REQUIREMENTS ALLOCATION 404
3.3.3.3.10.3.2 LOCAL ENTITIES DESIGN 404
3.3.3.3.10.3.3 INPUT/OUTPUT 405
3.3.3.3.10.3.4 LOCAL DATA 406
3.3.3.3.10.3.5 PROCESS CONTROL 406
3.3.3.3.10.3.6 PROCESSING 406
3.3.3.3.10.3.7 UTILIZATION OF OTHER ELEMENTS 407
3.3.3.3.10.3.8 LIMITATIONS 407
3.3.3.3.10.4 UPDATE_ERROR_COVARIANCE_MATRIX_GENERAL_FORM UNIT DESIGN (CATALOG #P1119-0) 407
3.3.3.3.10.4.1 REQUIREMENTS ALLOCATION 407
3.3.3.3.10.4.2 LOCAL ENTITIES DESIGN 407
3.3.3.3.10.4.3 INPUT/OUTPUT 407
3.3.3.3.10.4.4 LOCAL DATA 409
3.3.3.3.10.4.5 PROCESS CONTROL 409
3.3.3.3.10.4.6 PROCESSING 409
3.3.3.3.10.4.7 UTILIZATION OF OTHER ELEMENTS 410
3.3.3.3.10.4.8 LIMITATIONS 410

3.3.4 GUIDANCE AND CONTROL 421

3.3.4.1 WAYPOINT_STEERING (PACKAGE BODY) TLCSC P661 (CATALOG #P106-0) 423
3.3.4.1.1 REQUIREMENTS ALLOCATION 423
3.3.4.1.2 LOCAL ENTITIES DESIGN 423
3.3.4.1.3 INPUT/OUTPUT 423
3.3.4.1.4 LOCAL DATA 423
3.3.4.1.5 PROCESS CONTROL 423
3.3.4.1.6 PROCESSING 424
3.3.4.1.7 UTILIZATION OF OTHER ELEMENTS 424
3.3.4.1.8 LIMITATIONS 424
3.3.4.1.9 LLCSC DESIGN 424
3.3.4.1.9.1 STEERING_VECTOR_OPERATIONS (PACKAGE BODY) PACKAGE DESIGN (CATALOG #P107-0) 425
3.3.4.1.9.1.1 REQUIREMENTS ALLOCATION 425
3.3.4.1.9.1.2 LOCAL ENTITIES DESIGN 425
3.3.4.1.9.1.3 INPUT/OUTPUT 425
3.3.4.1.9.1.4 LOCAL DATA 426
3.3.4.1.9.1.5 PROCESS CONTROL 427
3.3.4.1.9.1.6 PROCESSING 427
3.3.4.1.9.1.7 UTILIZATION OF OTHER ELEMENTS 428

3.3.4.1.9.1.8 LIMITATIONS 428
3.3.4.1.9.1.9 LLCSC DESIGN 429
3.3.4.1.9.1.10 UNIT DESIGN 429
3.3.4.1.9.1.10.1 INITIALIZE (PROCEDURE BODY) UNIT DESIGN 429
3.3.4.1.9.1.10.1.1 REQUIREMENTS ALLOCATION 429
3.3.4.1.9.1.10.1.2 LOCAL ENTITIES DESIGN 429
3.3.4.1.9.1.10.1.3 INPUT/OUTPUT 429
3.3.4.1.9.1.10.1.4 LOCAL DATA 429
3.3.4.1.9.1.10.1.5 PROCESS CONTROL 430
3.3.4.1.9.1.10.1.6 PROCESSING 430
3.3.4.1.9.1.10.1.7 UTILIZATION OF OTHER ELEMENTS 431
3.3.4.1.9.1.10.1.8 LIMITATIONS 433
3.3.4.1.9.1.10.2 UPDATE (PROCEDURE BODY) UNIT DESIGN 433
3.3.4.1.9.1.10.2.1 REQUIREMENTS ALLOCATION 433
3.3.4.1.9.1.10.2.2 LOCAL ENTITIES DESIGN 433
3.3.4.1.9.1.10.2.3 INPUT/OUTPUT 433
3.3.4.1.9.1.10.2.4 LOCAL DATA 433
3.3.4.1.9.1.10.2.5 PROCESS CONTROL 433
3.3.4.1.9.1.10.2.6 PROCESSING 434
3.3.4.1.9.1.10.2.7 UTILIZATION OF OTHER ELEMENTS 434
3.3.4.1.9.1.10.2.8 LIMITATIONS 436
3.3.4.1.9.2 CROSSTRACK_AND_HEADING_ERROR_OPERATIONS (PACKAGE BODY) PACKAGE DESIGN (CATALOG #P109-0) 436
3.3.4.1.9.2.1 REQUIREMENTS ALLOCATION 436
3.3.4.1.9.2.2 LOCAL ENTITIES DESIGN 436
3.3.4.1.9.2.3 INPUT/OUTPUT 436
3.3.4.1.9.2.4 LOCAL DATA 438
3.3.4.1.9.2.5 PROCESS CONTROL 438
3.3.4.1.9.2.6 PROCESSING 438
3.3.4.1.9.2.7 UTILIZATION OF OTHER ELEMENTS 439
3.3.4.1.9.2.8 LIMITATIONS 439
3.3.4.1.9.2.9 LLCSC DESIGN 440
3.3.4.1.9.2.10 UNIT DESIGN 440
3.3.4.1.9.2.10.1 COMPUTE_WHEN_TURNING (PROCEDURE BODY) UNIT DESIGN 440
3.3.4.1.9.2.10.1.1 REQUIREMENTS ALLOCATION 440
3.3.4.1.9.2.10.1.2 LOCAL ENTITIES DESIGN 440
3.3.4.1.9.2.10.1.3 INPUT/OUTPUT 440
3.3.4.1.9.2.10.1.4 LOCAL DATA 441
3.3.4.1.9.2.10.1.5 PROCESS CONTROL 442
3.3.4.1.9.2.10.1.6 PROCESSING 442
3.3.4.1.9.2.10.1.7 UTILIZATION OF OTHER ELEMENTS 443
3.3.4.1.9.2.10.1.8 LIMITATIONS 446
3.3.4.1.9.2.10.2 COMPUTE_WHEN_NOT_TURNING (PROCEDURE BODY) UNIT DESIGN 446
3.3.4.1.9.2.10.2.1 REQUIREMENTS ALLOCATION 446
3.3.4.1.9.2.10.2.2 LOCAL ENTITIES DESIGN 446
3.3.4.1.9.2.10.2.3 INPUT/OUTPUT 446
3.3.4.1.9.2.10.2.4 LOCAL DATA 447
3.3.4.1.9.2.10.2.5 PROCESS CONTROL 447
3.3.4.1.9.2.10.2.6 PROCESSING 447
3.3.4.1.9.2.10.2.7 UTILIZATION OF OTHER ELEMENTS 448
3.3.4.1.9.2.10.2.8 LIMITATIONS 450
3.3.4.1.9.2.10.3 COMPUTE (PROCEDURE BODY) UNIT DESIGN 450
3.3.4.1.9.2.10.3.1 REQUIREMENTS ALLOCATION 450
3.3.4.1.9.2.10.3.2 LOCAL ENTITIES DESIGN 450

3.3.4.1.9.2.10.3.3 INPUT/OUTPUT 450
3.3.4.1.9.2.10.3.4 LOCAL DATA 451
3.3.4.1.9.2.10.3.5 PROCESS CONTROL 451
3.3.4.1.9.2.10.3.6 PROCESSING 451
3.3.4.1.9.2.10.3.7 UTILIZATION OF OTHER ELEMENTS 452
3.3.4.1.9.2.10.3.8 LIMITATIONS 453
3.3.4.1.9.3 TURN_TEST_OPERATIONS (PACKAGE BODY) PACKAGE DESIGN (CATALOG #P112-0) 453
3.3.4.1.9.3.1 REQUIREMENTS ALLOCATION 454
3.3.4.1.9.3.2 LOCAL ENTITIES DESIGN 454
3.3.4.1.9.3.3 INPUT/OUTPUT 454
3.3.4.1.9.3.4 LOCAL DATA 454
3.3.4.1.9.3.5 PROCESS CONTROL 454
3.3.4.1.9.3.6 PROCESSING 454
3.3.4.1.9.3.7 UTILIZATION OF OTHER ELEMENTS 455
3.3.4.1.9.3.8 LIMITATIONS 455
3.3.4.1.9.3.9 LLCSC DESIGN 455
3.3.4.1.9.3.10 UNIT DESIGN 455
3.3.4.1.9.3.10.1 STOP_TEST (FUNCTION BODY) UNIT DESIGN 455
3.3.4.1.9.3.10.1.1 REQUIREMENTS ALLOCATION 455
3.3.4.1.9.3.10.1.2 LOCAL ENTITIES DESIGN 456
3.3.4.1.9.3.10.1.3 INPUT/OUTPUT 456
3.3.4.1.9.3.10.1.4 LOCAL DATA 456
3.3.4.1.9.3.10.1.5 PROCESS CONTROL 456
3.3.4.1.9.3.10.1.6 PROCESSING 456
3.3.4.1.9.3.10.1.7 UTILIZATION OF OTHER ELEMENTS 457
3.3.4.1.9.3.10.1.8 LIMITATIONS 457
3.3.4.1.9.3.10.2 START_TEST (FUNCTION BODY) UNIT DESIGN 457
3.3.4.1.9.3.10.2.1 REQUIREMENTS ALLOCATION 458
3.3.4.1.9.3.10.2.2 LOCAL ENTITIES DESIGN 458
3.3.4.1.9.3.10.2.3 INPUT/OUTPUT 458
3.3.4.1.9.3.10.2.4 LOCAL DATA 458
3.3.4.1.9.3.10.2.5 PROCESS CONTROL 458
3.3.4.1.9.3.10.2.6 PROCESSING 458
3.3.4.1.9.3.10.2.7 UTILIZATION OF OTHER ELEMENTS 459
3.3.4.1.9.3.10.2.8 LIMITATIONS 459
3.3.4.1.9.4 STEERING_VECTOR_OPERATIONS_WITH_ARCSIN (PACKAGE BODY) PACKAGE DESIGN (CATALOG #P1048-0) 460
3.3.4.1.9.4.1 REQUIREMENTS ALLOCATION 460
3.3.4.1.9.4.2 LOCAL ENTITIES DESIGN 460
3.3.4.1.9.4.3 INPUT/OUTPUT 460
3.3.4.1.9.4.4 LOCAL DATA 461
3.3.4.1.9.4.5 PROCESS CONTROL 462
3.3.4.1.9.4.6 PROCESSING 462
3.3.4.1.9.4.7 UTILIZATION OF OTHER ELEMENTS 463
3.3.4.1.9.4.8 LIMITATIONS 463
3.3.4.1.9.4.9 LLCSC DESIGN 463
3.3.4.1.9.4.10 UNIT DESIGN 463
3.3.4.1.9.4.10.1 INITIALIZE (PROCEDURE BODY) UNIT DESIGN 463
3.3.4.1.9.4.10.1.1 REQUIREMENTS ALLOCATION 463
3.3.4.1.9.4.10.1.2 LOCAL ENTITIES DESIGN 464
3.3.4.1.9.4.10.1.3 INPUT/OUTPUT 464
3.3.4.1.9.4.10.1.4 LOCAL DATA 464
3.3.4.1.9.4.10.1.5 PROCESS CONTROL 464
3.3.4.1.9.4.10.1.6 PROCESSING 464
3.3.4.1.9.4.10.1.7 UTILIZATION OF OTHER ELEMENTS 466

3.3.4.1.9.4.10.1.8 LIMITATIONS 467
3.3.4.1.9.4.10.2 UPDATE (PROCEDURE BODY) UNIT DESIGN 467
3.3.4.1.9.4.10.2.1 REQUIREMENTS ALLOCATION 468
3.3.4.1.9.4.10.2.2 LOCAL ENTITIES DESIGN 468
3.3.4.1.9.4.10.2.3 INPUT/OUTPUT 468
3.3.4.1.9.4.10.2.4 LOCAL DATA 468
3.3.4.1.9.4.10.2.5 PROCESS CONTROL 468
3.3.4.1.9.4.10.2.6 PROCESSING 468
3.3.4.1.9.4.10.2.7 UTILIZATION OF OTHER ELEMENTS 469
3.3.4.1.9.4.10.2.8 LIMITATIONS 470
3.3.4.1.10 UNIT DESIGN 470
3.3.4.1.10.1 COMPUTE TURN ANGLE AND DIRECTION (PROCEDURE BODY) UNIT DESIGN (CATALOG #P108-0) 470
3.3.4.1.10.1.1 REQUIREMENTS ALLOCATION 470
3.3.4.1.10.1.2 LOCAL ENTITIES DESIGN 471
3.3.4.1.10.1.3 INPUT/OUTPUT 471
3.3.4.1.10.1.4 LOCAL DATA 472
3.3.4.1.10.1.5 PROCESS CONTROL 472
3.3.4.1.10.1.6 PROCESSING 472
3.3.4.1.10.1.7 UTILIZATION OF OTHER ELEMENTS 473
3.3.4.1.10.1.8 LIMITATIONS 473
3.3.4.1.10.2 DISTANCE TO CURRENT WAYPOINT (FUNCTION BODY) UNIT DESIGN (CATALOG #P110-0) 473
3.3.4.1.10.2.1 REQUIREMENTS ALLOCATION 473
3.3.4.1.10.2.2 LOCAL ENTITIES DESIGN 473
3.3.4.1.10.2.3 INPUT/OUTPUT 474
3.3.4.1.10.2.4 LOCAL DATA 475
3.3.4.1.10.2.5 PROCESS CONTROL 475
3.3.4.1.10.2.6 PROCESSING 475
3.3.4.1.10.2.7 UTILIZATION OF OTHER ELEMENTS 476
3.3.4.1.10.2.8 LIMITATIONS 476
3.3.4.1.10.3 COMPUTE TURNING AND NONTURNING DISTANCES(PROCEDURE BODY) UNIT DESIGN (CATALOG #P111-0) 476
3.3.4.1.10.3.1 REQUIREMENTS ALLOCATION 476
3.3.4.1.10.3.2 LOCAL ENTITIES DESIGN 476
3.3.4.1.10.3.3 INPUT/OUTPUT 476
3.3.4.1.10.3.4 LOCAL DATA 477
3.3.4.1.10.3.5 PROCESS CONTROL 477
3.3.4.1.10.3.6 PROCESSING 477
3.3.4.1.10.3.7 UTILIZATION OF OTHER ELEMENTS 478
3.3.4.1.10.3.8 LIMITATIONS 478
3.3.4.1.10.4 DISTANCE TO CURRENT WAYPOINT WITH ARCSIN (FUNCTION BODY) UNIT DESIGN (CATALOG #P1117-0) 478
3.3.4.1.10.4.1 REQUIREMENTS ALLOCATION 478
3.3.4.1.10.4.2 LOCAL ENTITIES DESIGN 478
3.3.4.1.10.4.3 INPUT/OUTPUT 478
3.3.4.1.10.4.4 LOCAL DATA 480
3.3.4.1.10.4.5 PROCESS CONTROL 480
3.3.4.1.10.4.6 PROCESSING 480
3.3.4.1.10.4.7 UTILIZATION OF OTHER ELEMENTS 481
3.3.4.1.10.4.8 LIMITATIONS 481

3.3.4.2 AUTOPILOT (PACKAGE BODY) TLCSC (CATALOG #P305-0) 505
3.3.4.2.1 REQUIREMENTS ALLOCATION 505
3.3.4.2.2 LOCAL ENTITIES DESIGN 505
3.3.4.2.3 INPUT/OUTPUT 505
3.3.4.2.4 LOCAL DATA 505

3.3.4.2.5 PROCESS CONTROL 505
3.3.4.2.6 PROCESSING 505
3.3.4.2.7 UTILIZATION OF OTHER ELEMENTS 506
3.3.4.2.8 LIMITATIONS 506
3.3.4.2.9 LLCSC DESIGN 506
3.3.4.2.9.1 INTEGRAL_PLUS_PROPORTIONAL_GAIN (PACKAGE BODY) PACKAGE DESIGN (CATALOG #P306-0) 506
3.3.4.2.9.1.1 REQUIREMENTS ALLOCATION 506
3.3.4.2.9.1.2 LOCAL ENTITIES DESIGN 506
3.3.4.2.9.1.3 INPUT/OUTPUT 506
3.3.4.2.9.1.4 LOCAL DATA 507
3.3.4.2.9.1.5 PROCESS CONTROL 507
3.3.4.2.9.1.6 PROCESSING 507
3.3.4.2.9.1.7 UTILIZATION OF OTHER ELEMENTS 508
3.3.4.2.9.1.8 LIMITATIONS 508
3.3.4.2.9.1.9 LLCSC DESIGN 508
3.3.4.2.9.1.10 UNIT DESIGN 508
3.3.4.2.9.1.10.1 INTEGRATE UNIT DESIGN 508
3.3.4.2.9.1.10.1.1 REQUIREMENTS ALLOCATION 508
3.3.4.2.9.1.10.1.2 LOCAL ENTITIES DESIGN 508
3.3.4.2.9.1.10.1.3 INPUT/OUTPUT 508
3.3.4.2.9.1.10.1.4 LOCAL DATA 509
3.3.4.2.9.1.10.1.5 PROCESS CONTROL 509
3.3.4.2.9.1.10.1.6 PROCESSING 509
3.3.4.2.9.1.10.1.7 UTILIZATION OF OTHER ELEMENTS 509
3.3.4.2.9.1.10.1.8 LIMITATIONS 510
3.3.4.2.9.1.10.2 UPDATE_PROPORTIONAL_GAIN UNIT DESIGN 510
3.3.4.2.9.1.10.2.1 REQUIREMENTS ALLOCATION 510
3.3.4.2.9.1.10.2.2 LOCAL ENTITIES DESIGN 510
3.3.4.2.9.1.10.2.3 INPUT/OUTPUT 510
3.3.4.2.9.1.10.2.4 LOCAL DATA 511
3.3.4.2.9.1.10.2.5 PROCESS CONTROL 511
3.3.4.2.9.1.10.2.6 PROCESSING 511
3.3.4.2.9.1.10.2.7 UTILIZATION OF OTHER ELEMENTS 511
3.3.4.2.9.1.10.2.8 LIMITATIONS 512
3.3.4.2.9.2 LATERAL DIRECTIONAL AUTOPILOT (PACKAGE BODY) PACKAGE DESIGN (CATALOG #P308-0) 512
3.3.4.2.9.2.1 REQUIREMENTS ALLOCATION 512
3.3.4.2.9.2.2 LOCAL ENTITIES DESIGN 512
3.3.4.2.9.2.3 INPUT/OUTPUT 512
3.3.4.2.9.2.4 LOCAL DATA 516
3.3.4.2.9.2.5 PROCESS CONTROL 516
3.3.4.2.9.2.6 PROCESSING 516
3.3.4.2.9.2.7 UTILIZATION OF OTHER ELEMENTS 517
3.3.4.2.9.2.8 LIMITATIONS 518
3.3.4.2.9.2.9 LLCSC DESIGN 518
3.3.4.2.9.2.10 UNIT DESIGN 519
3.3.4.2.9.2.10.1 INITIALIZE_LATERAL_DIRECTIONAL_AUTOPILOT UNIT DESIGN 519
3.3.4.2.9.2.10.1.1 REQUIREMENTS ALLOCATION 519
3.3.4.2.9.2.10.1.2 LOCAL ENTITIES DESIGN 519
3.3.4.2.9.2.10.1.3 INPUT/OUTPUT 519
3.3.4.2.9.2.10.1.4 LOCAL DATA 521
3.3.4.2.9.2.10.1.5 PROCESS CONTROL 521
3.3.4.2.9.2.10.1.6 PROCESSING 521
3.3.4.2.9.2.10.1.7 UTILIZATION OF OTHER ELEMENTS 522
3.3.4.2.9.2.10.1.8 LIMITATIONS 523

3.3.4.2.9.2.10.2 COMPUTE_AILERON_RUDDER_COMMANDS(FUNCTION BODY) UNIT DESIGN 524
3.3.4.2.9.2.10.2.1 REQUIREMENTS ALLOCATION 524
3.3.4.2.9.2.10.2.2 LOCAL ENTITIES DESIGN 524
3.3.4.2.9.2.10.2.3 INPUT/OUTPUT 524
3.3.4.2.9.2.10.2.4 LOCAL DATA 526
3.3.4.2.9.2.10.2.5 PROCESS CONTROL 526
3.3.4.2.9.2.10.2.6 PROCESSING 526
3.3.4.2.9.2.10.2.7 UTILIZATION OF OTHER ELEMENTS 528
3.3.4.2.9.2.10.2.8 LIMITATIONS 529
3.3.4.2.9.2.10.3 UPDATE_AILERON_INTEGRATOR_GAIN UNIT DESIGN 529
3.3.4.2.9.2.10.3.1 REQUIREMENTS ALLOCATION 529
3.3.4.2.9.2.10.3.2 LOCAL ENTITIES DESIGN 529
3.3.4.2.9.2.10.3.3 INPUT/OUTPUT 529
3.3.4.2.9.2.10.3.4 LOCAL DATA 530
3.3.4.2.9.2.10.3.5 PROCESS CONTROL 530
3.3.4.2.9.2.10.3.6 PROCESSING 530
3.3.4.2.9.2.10.3.7 UTILIZATION OF OTHER ELEMENTS 530
3.3.4.2.9.2.10.3.8 LIMITATIONS 531
3.3.4.2.9.2.10.4 UPDATE_AILERON_INTEGRATOR_LIMIT UNIT DESIGN 531
3.3.4.2.9.2.10.4.1 REQUIREMENTS ALLOCATION 531
3.3.4.2.9.2.10.4.2 LOCAL ENTITIES DESIGN 531
3.3.4.2.9.2.10.4.3 INPUT/OUTPUT 531
3.3.4.2.9.2.10.4.4 LOCAL DATA 532
3.3.4.2.9.2.10.4.5 PROCESS CONTROL 532
3.3.4.2.9.2.10.4.6 PROCESSING 532
3.3.4.2.9.2.10.4.7 UTILIZATION OF OTHER ELEMENTS 532
3.3.4.2.9.2.10.4.8 LIMITATIONS 533
3.3.4.2.9.2.10.5 UPDATE_ROLL_COMMAND_PROPORTIONAL_GAIN UNIT DESIGN 533
3.3.4.2.9.2.10.5.1 REQUIREMENTS ALLOCATION 533
3.3.4.2.9.2.10.5.2 LOCAL ENTITIES DESIGN 533
3.3.4.2.9.2.10.5.3 INPUT/OUTPUT 533
3.3.4.2.9.2.10.5.4 LOCAL DATA 534
3.3.4.2.9.2.10.5.5 PROCESS CONTROL 534
3.3.4.2.9.2.10.5.6 PROCESSING 534
3.3.4.2.9.2.10.5.7 UTILIZATION OF OTHER ELEMENTS 534
3.3.4.2.9.2.10.5.8 LIMITATIONS 535
3.3.4.2.9.2.10.6 UPDATE_ROLL_RATE_GAIN_FOR_AILERON UNIT DESIGN 535
3.3.4.2.9.2.10.6.1 REQUIREMENTS ALLOCATION 535
3.3.4.2.9.2.10.6.2 LOCAL ENTITIES DESIGN 535
3.3.4.2.9.2.10.6.3 INPUT/OUTPUT 535
3.3.4.2.9.2.10.6.4 LOCAL DATA 535
3.3.4.2.9.2.10.6.5 PROCESS CONTROL 536
3.3.4.2.9.2.10.6.6 PROCESSING 536
3.3.4.2.9.2.10.6.7 UTILIZATION OF OTHER ELEMENTS 536
3.3.4.2.9.2.10.6.8 LIMITATIONS 536
3.3.4.2.9.2.10.7 UPDATE_YAW_RATE_GAIN_FOR_AILERON UNIT DESIGN 536
3.3.4.2.9.2.10.7.1 REQUIREMENTS ALLOCATION 536
3.3.4.2.9.2.10.7.2 LOCAL ENTITIES DESIGN 536
3.3.4.2.9.2.10.7.3 INPUT/OUTPUT 536
3.3.4.2.9.2.10.7.4 LOCAL DATA 537
3.3.4.2.9.2.10.7.5 PROCESS CONTROL 537
3.3.4.2.9.2.10.7.6 PROCESSING 537

3.3.4.2.9.2.10.7.7 UTILIZATION OF OTHER ELEMENTS 537
3.3.4.2.9.2.10.7.8 LIMITATIONS 537
3.3.4.2.9.2.10.8 UPDATE_RUDDER_INTEGRATOR_GAIN UNIT DESIGN 538
3.3.4.2.9.2.10.8.1 REQUIREMENTS ALLOCATION 538
3.3.4.2.9.2.10.8.2 LOCAL ENTITIES DESIGN 538
3.3.4.2.9.2.10.8.3 INPUT/OUTPUT 538
3.3.4.2.9.2.10.8.4 LOCAL DATA 538
3.3.4.2.9.2.10.8.5 PROCESS CONTROL 538
3.3.4.2.9.2.10.8.6 PROCESSING 539
3.3.4.2.9.2.10.8.7 UTILIZATION OF OTHER ELEMENTS 539
3.3.4.2.9.2.10.8.8 LIMITATIONS 539
3.3.4.2.9.2.10.9 UPDATE_RUDDER_INTEGRATOR_LIMIT UNIT DESIGN 539
3.3.4.2.9.2.10.9.1 REQUIREMENTS ALLOCATION 540
3.3.4.2.9.2.10.9.2 LOCAL ENTITIES DESIGN 540
3.3.4.2.9.2.10.9.3 INPUT/OUTPUT 540
3.3.4.2.9.2.10.9.4 LOCAL DATA 540
3.3.4.2.9.2.10.9.5 PROCESS CONTROL 540
3.3.4.2.9.2.10.9.6 PROCESSING 540
3.3.4.2.9.2.10.9.7 UTILIZATION OF OTHER ELEMENTS 541
3.3.4.2.9.2.10.9.8 LIMITATIONS 541
3.3.4.2.9.2.10.10 UPDATE_FEEDBACK_RATE_GAIN_FOR_RUDDER UNIT DESIGN 541
3.3.4.2.9.2.10.10.1 REQUIREMENTS ALLOCATION 541
3.3.4.2.9.2.10.10.2 LOCAL ENTITIES DESIGN 542
3.3.4.2.9.2.10.10.3 INPUT/OUTPUT 542
3.3.4.2.9.2.10.10.4 LOCAL DATA 542
3.3.4.2.9.2.10.10.5 PROCESS CONTROL 542
3.3.4.2.9.2.10.10.6 PROCESSING 542
3.3.4.2.9.2.10.10.7 UTILIZATION OF OTHER ELEMENTS 543
3.3.4.2.9.2.10.10.8 LIMITATIONS 543
3.3.4.2.9.2.10.11 UPDATE_ROLL_RATE_GAIN_FOR_RUDDER UNIT DESIGN 543
3.3.4.2.9.2.10.11.1 REQUIREMENTS ALLOCATION 543
3.3.4.2.9.2.10.11.2 LOCAL ENTITIES DESIGN 543
3.3.4.2.9.2.10.11.3 INPUT/OUTPUT 543
3.3.4.2.9.2.10.11.4 LOCAL DATA 544
3.3.4.2.9.2.10.11.5 PROCESS CONTROL 544
3.3.4.2.9.2.10.11.6 PROCESSING 544
3.3.4.2.9.2.10.11.7 UTILIZATION OF OTHER ELEMENTS 544
3.3.4.2.9.2.10.11.8 LIMITATIONS 544
3.3.4.2.9.2.10.12 UPDATE_ACCELERATION_PROPORTIONAL_GAIN UNIT DESIGN 544
3.3.4.2.9.2.10.12.1 REQUIREMENTS ALLOCATION 544
3.3.4.2.9.2.10.12.2 LOCAL ENTITIES DESIGN 544
3.3.4.2.9.2.10.12.3 INPUT/OUTPUT 544
3.3.4.2.9.2.10.12.4 LOCAL DATA 545
3.3.4.2.9.2.10.12.5 PROCESS CONTROL 545
3.3.4.2.9.2.10.12.6 PROCESSING 545
3.3.4.2.9.2.10.12.7 UTILIZATION OF OTHER ELEMENTS 545
3.3.4.2.9.2.10.12.8 LIMITATIONS 546
3.3.4.2.9.3 PITCH AUTOPILOT PACKAGE DESIGN (CATALOG #P307-0) 546
3.3.4.2.9.3.1 REQUIREMENTS ALLOCATION 546
3.3.4.2.9.3.2 LOCAL ENTITIES DESIGN 546
3.3.4.2.9.3.3 INPUT/OUTPUT 546
3.3.4.2.9.3.4 LOCAL DATA 547
3.3.4.2.9.3.5 PROCESS CONTROL 548

3.3.4.2.9.3.6 PROCESSING 548
3.3.4.2.9.3.7 UTILIZATION OF OTHER ELEMENTS 548
3.3.4.2.9.3.8 LIMITATIONS 549
3.3.4.2.9.3.9 LLCSC DESIGN 549
3.3.4.2.9.3.10 UNIT DESIGN 549
3.3.4.2.9.3.10.1 INITIALIZE_PITCH_AUTOPILOT UNIT DESIGN 549
3.3.4.2.9.3.10.1.1 REQUIREMENTS ALLOCATION 549
3.3.4.2.9.3.10.1.2 LOCAL ENTITIES DESIGN 549
3.3.4.2.9.3.10.1.3 INPUT/OUTPUT 550
3.3.4.2.9.3.10.1.4 LOCAL DATA 550
3.3.4.2.9.3.10.1.5 PROCESS CONTROL 550
3.3.4.2.9.3.10.1.6 PROCESSING 550
3.3.4.2.9.3.10.1.7 UTILIZATION OF OTHER ELEMENTS 551
3.3.4.2.9.3.10.1.8 LIMITATIONS 552
3.3.4.2.9.3.10.2 COMPUTE_ELEVATOR_COMMAND UNIT DESIGN 552
3.3.4.2.9.3.10.2.1 REQUIREMENTS ALLOCATION 552
3.3.4.2.9.3.10.2.2 LOCAL ENTITIES DESIGN 552
3.3.4.2.9.3.10.2.3 INPUT/OUTPUT 552
3.3.4.2.9.3.10.2.4 LOCAL DATA 552
3.3.4.2.9.3.10.2.5 PROCESS CONTROL 553
3.3.4.2.9.3.10.2.6 PROCESSING 553
3.3.4.2.9.3.10.2.7 UTILIZATION OF OTHER ELEMENTS 554
3.3.4.2.9.3.10.2.8 LIMITATIONS 555
3.3.4.2.9.3.10.3 UPDATE_PITCH_RATE_GAIN UNIT DESIGN 555
3.3.4.2.9.3.10.3.1 REQUIREMENTS ALLOCATION 555
3.3.4.2.9.3.10.3.2 LOCAL ENTITIES DESIGN 555
3.3.4.2.9.3.10.3.3 INPUT/OUTPUT 555
3.3.4.2.9.3.10.3.4 LOCAL DATA 556
3.3.4.2.9.3.10.3.5 PROCESS CONTROL 556
3.3.4.2.9.3.10.3.6 PROCESSING 556
3.3.4.2.9.3.10.3.7 UTILIZATION OF OTHER ELEMENTS 556
3.3.4.2.9.3.10.3.8 LIMITATIONS 557
3.3.4.2.9.3.10.4 UPDATE_ACCELERATION_GAIN UNIT DESIGN 557
3.3.4.2.9.3.10.4.1 REQUIREMENTS ALLOCATION 557
3.3.4.2.9.3.10.4.2 LOCAL ENTITIES DESIGN 557
3.3.4.2.9.3.10.4.3 INPUT/OUTPUT 557
3.3.4.2.9.3.10.4.4 LOCAL DATA 558
3.3.4.2.9.3.10.4.5 PROCESS CONTROL 558
3.3.4.2.9.3.10.4.6 PROCESSING 558
3.3.4.2.9.3.10.4.7 UTILIZATION OF OTHER ELEMENTS 558
3.3.4.2.9.3.10.4.8 LIMITATIONS 558
3.3.4.2.9.3.10.5 UPDATE_INTEGRATOR_GAIN UNIT DESIGN 558
3.3.4.2.9.3.10.5.1 REQUIREMENTS ALLOCATION 559
3.3.4.2.9.3.10.5.2 LOCAL ENTITIES DESIGN 559
3.3.4.2.9.3.10.5.3 INPUT/OUTPUT 559
3.3.4.2.9.3.10.5.4 LOCAL DATA 559
3.3.4.2.9.3.10.5.5 PROCESS CONTROL 559
3.3.4.2.9.3.10.5.6 PROCESSING 559
3.3.4.2.9.3.10.5.7 UTILIZATION OF OTHER ELEMENTS 560
3.3.4.2.9.3.10.5.8 LIMITATIONS 560
3.3.4.2.9.3.10.6 UPDATE_INTEGRATOR_LIMIT UNIT DESIGN 560
3.3.4.2.9.3.10.6.1 REQUIREMENTS ALLOCATION 560
3.3.4.2.9.3.10.6.2 LOCAL ENTITIES DESIGN 560
3.3.4.2.9.3.10.6.3 INPUT/OUTPUT 561
3.3.4.2.9.3.10.6.4 LOCAL DATA 561
3.3.4.2.9.3.10.6.5 PROCESS CONTROL 561
3.3.4.2.9.3.10.6.6 PROCESSING 561

3.3.4.2.9.3.10.6.7 UTILIZATION OF OTHER ELEMENTS . . . 561
3.3.4.2.9.3.10.6.8 LIMITATIONS . . . 562
3.3.4.2.9.3.10.7 UPDATE_PROPORTIONAL_GAIN UNIT DESIGN . . . 562
3.3.4.2.9.3.10.7.1 REQUIREMENTS ALLOCATION . . . 562
3.3.4.2.9.3.10.7.2 LOCAL ENTITIES DESIGN . . . 562
3.3.4.2.9.3.10.7.3 INPUT/OUTPUT . . . 562
3.3.4.2.9.3.10.7.4 LOCAL DATA . . . 563
3.3.4.2.9.3.10.7.5 PROCESS CONTROL . . . 563
3.3.4.2.9.3.10.7.6 PROCESSING . . . 563
3.3.4.2.9.3.10.7.7 UTILIZATION OF OTHER ELEMENTS . . . 563
3.3.4.2.9.3.10.7.8 LIMITATIONS . . . 564
3.3.4.2.10 UNIT DESIGN . . . 564

3.3.5 NONGUIDANCE AND CONTROL . . . 575

3.3.5.1 AIR_DATA_PARTS (PACKAGE BODY) TLCSC P671 (CATALOG #P316-0) . . . 577
3.3.5.1.1 REQUIREMENTS ALLOCATION . . . 577
3.3.5.1.2 LOCAL ENTITIES DESIGN . . . 577
3.3.5.1.3 INPUT/OUTPUT . . . 577
3.3.5.1.4 LOCAL DATA . . . 577
3.3.5.1.5 PROCESS CONTROL . . . 577
3.3.5.1.6 PROCESSING . . . 577
3.3.5.1.7 UTILIZATION OF OTHER ELEMENTS . . . 578
3.3.5.1.8 LIMITATIONS . . . 578
3.3.5.1.9 LLCSC DESIGN . . . 578
3.3.5.1.9.1 BAROMETRIC_ALTITUDE_INTEGRATION (PACKAGE BODY) PACKAGE DESIGN (CATALOG #P322-0) . . . 578
3.3.5.1.9.1.1 REQUIREMENTS ALLOCATION . . . 578
3.3.5.1.9.1.2 LOCAL ENTITIES DESIGN . . . 578
3.3.5.1.9.1.3 INPUT/OUTPUT . . . 579
3.3.5.1.9.1.4 LOCAL DATA . . . 580
3.3.5.1.9.1.5 PROCESS CONTROL . . . 580
3.3.5.1.9.1.6 PROCESSING . . . 580
3.3.5.1.9.1.7 UTILIZATION OF OTHER ELEMENTS . . . 580
3.3.5.1.9.1.8 LIMITATIONS . . . 580
3.3.5.1.9.1.9 LLCSC DESIGN . . . 580
3.3.5.1.9.1.10 UNIT DESIGN . . . 580
3.3.5.1.9.1.10.1 COMPUTE_BAROMETRIC_ALTITUDE UNIT DESIGN . . . 580
3.3.5.1.9.1.10.1.1 REQUIREMENTS ALLOCATION . . . 580
3.3.5.1.9.1.10.1.2 LOCAL ENTITIES DESIGN . . . 581
3.3.5.1.9.1.10.1.3 INPUT/OUTPUT . . . 581
3.3.5.1.9.1.10.1.4 LOCAL DATA . . . 581
3.3.5.1.9.1.10.1.5 PROCESS CONTROL . . . 581
3.3.5.1.9.1.10.1.6 PROCESSING . . . 581
3.3.5.1.9.1.10.1.7 UTILIZATION OF OTHER ELEMENTS . . . 582
3.3.5.1.9.1.10.1.8 LIMITATIONS . . . 582
3.3.5.1.10 UNIT DESIGN . . . 582
3.3.5.1.10.1 COMPUTE_OUTSIDE_AIR_TEMPERATURE (FUNCTION BODY) UNIT DESIGN (CATALOG #P317-0) . . . 582
3.3.5.1.10.1.1 REQUIREMENTS ALLOCATION . . . 582
3.3.5.1.10.1.2 LOCAL ENTITIES DESIGN . . . 582
3.3.5.1.10.1.3 INPUT/OUTPUT . . . 582
3.3.5.1.10.1.4 LOCAL DATA . . . 584
3.3.5.1.10.1.5 PROCESS CONTROL . . . 584
3.3.5.1.10.1.6 PROCESSING . . . 584
3.3.5.1.10.1.7 UTILIZATION OF OTHER ELEMENTS . . . 584

3.3.5.1.10.1.8 LIMITATIONS 584
3.3.5.1.10.2 COMPUTE_PRESSURE_RATIO (FUNCTION BODY) UNIT DESIGN (CATALOG #P318-0) 584
3.3.5.1.10.2.1 REQUIREMENTS ALLOCATION 585
3.3.5.1.10.2.2 LOCAL ENTITIES DESIGN 585
3.3.5.1.10.2.3 INPUT/OUTPUT 585
3.3.5.1.10.2.4 LOCAL DATA 586
3.3.5.1.10.2.5 PROCESS CONTROL 586
3.3.5.1.10.2.6 PROCESSING 586
3.3.5.1.10.2.7 UTILIZATION OF OTHER ELEMENTS 586
3.3.5.1.10.2.8 LIMITATIONS 586
3.3.5.1.10.3 COMPUTE_MACH (FUNCTION BODY) UNIT DESIGN (CATALOG #P319-0) 587
3.3.5.1.10.3.1 REQUIREMENTS ALLOCATION 587
3.3.5.1.10.3.2 LOCAL ENTITIES DESIGN 587
3.3.5.1.10.3.3 INPUT/OUTPUT 587
3.3.5.1.10.3.4 LOCAL DATA 588
3.3.5.1.10.3.5 PROCESS CONTROL 588
3.3.5.1.10.3.6 PROCESSING 588
3.3.5.1.10.3.7 UTILIZATION OF OTHER ELEMENTS 588
3.3.5.1.10.3.8 LIMITATIONS 588
3.3.5.1.10.4 COMPUTE_DYNAMIC_PRESSURE (FUNCTION BODY) UNIT DESIGN (CATALOG #P320-0) 589
3.3.5.1.10.4.1 REQUIREMENTS ALLOCATION 589
3.3.5.1.10.4.2 LOCAL ENTITIES DESIGN 589
3.3.5.1.10.4.3 INPUT/OUTPUT 589
3.3.5.1.10.4.4 LOCAL DATA 590
3.3.5.1.10.4.5 PROCESS CONTROL 590
3.3.5.1.10.4.6 PROCESSING 590
3.3.5.1.10.4.7 UTILIZATION OF OTHER ELEMENTS 590
3.3.5.1.10.4.8 LIMITATIONS 590
3.3.5.1.10.5 COMPUTE_SPEED_OF_SOUND (FUNCTION BODY) UNIT DESIGN (CATALOG #P321-0) 590
3.3.5.1.10.5.1 REQUIREMENTS ALLOCATION 591
3.3.5.1.10.5.2 LOCAL ENTITIES DESIGN 591
3.3.5.1.10.5.3 INPUT/OUTPUT 591
3.3.5.1.10.5.4 LOCAL DATA 592
3.3.5.1.10.5.5 PROCESS CONTROL 592
3.3.5.1.10.5.6 PROCESSING 592
3.3.5.1.10.5.7 UTILIZATION OF OTHER ELEMENTS 592
3.3.5.1.10.5.8 LIMITATIONS 592

3.3.5.2 FUEL_CONTROL_PARTS TLCSC P672 (CATALOG #P1096-0) 601
3.3.5.2.1 REQUIREMENTS ALLOCATION 601
3.3.5.2.2 LOCAL ENTITIES DESIGN 601
3.3.5.2.3 INPUT/OUTPUT 601
3.3.5.2.4 LOCAL DATA 601
3.3.5.2.5 PROCESS CONTROL 601
3.3.5.2.6 PROCESSING 601
3.3.5.2.7 UTILIZATION OF OTHER ELEMENTS 601
3.3.5.2.8 LIMITATIONS 602
3.3.5.2.9 LLCSC DESIGN 602
3.3.5.2.9.1 THROTTLE_COMMAND_MANAGER PACKAGE DESIGN 602
3.3.5.2.9.1.1 REQUIREMENTS ALLOCATION 602
3.3.5.2.9.1.2 LOCAL ENTITIES DESIGN 602
3.3.5.2.9.1.3 INPUT/OUTPUT 602
3.3.5.2.9.1.4 LOCAL DATA 604

3.3.5.2.9.1.5 PROCESS CONTROL 604
3.3.5.2.9.1.6 PROCESSING 604
3.3.5.2.9.1.7 UTILIZATION OF OTHER ELEMENTS 606
3.3.5.2.9.1.8 LIMITATIONS 607
3.3.5.2.9.1.9 LLCSC DESIGN 607
3.3.5.2.9.1.10 UNIT DESIGN 607
3.3.5.2.9.1.10.1 COMPUTE_THROTTLE_COMMAND UNIT DESIGN 607
3.3.5.2.9.1.10.1.1 REQUIREMENTS ALLOCATION 607
3.3.5.2.9.1.10.1.2 LOCAL ENTITIES DESIGN 608
3.3.5.2.9.1.10.1.3 INPUT/OUTPUT 608
3.3.5.2.9.1.10.1.4 LOCAL DATA 608
3.3.5.2.9.1.10.1.5 PROCESS CONTROL 608
3.3.5.2.9.1.10.1.6 PROCESSING 608
3.3.5.2.9.1.10.1.7 UTILIZATION OF OTHER ELEMENTS 609
3.3.5.2.9.1.10.1.8 LIMITATIONS 610
3.3.5.2.9.1.10.2 UPDATE_MACH_ERROR_LIMIT UNIT DESIGN 610
3.3.5.2.9.1.10.2.1 REQUIREMENTS ALLOCATION 610
3.3.5.2.9.1.10.2.2 LOCAL ENTITIES DESIGN 610
3.3.5.2.9.1.10.2.3 INPUT/OUTPUT 610
3.3.5.2.9.1.10.2.4 LOCAL DATA 610
3.3.5.2.9.1.10.2.5 PROCESS CONTROL 611
3.3.5.2.9.1.10.2.6 PROCESSING 611
3.3.5.2.9.1.10.2.7 UTILIZATION OF OTHER ELEMENTS 611
3.3.5.2.9.1.10.2.8 LIMITATIONS 611
3.3.5.2.9.1.10.3 UPDATE_MACH_ERROR_INTEGRAL_LIMIT UNIT DESIGN 611
3.3.5.2.9.1.10.3.1 REQUIREMENTS ALLOCATION 612
3.3.5.2.9.1.10.3.2 LOCAL ENTITIES DESIGN 612
3.3.5.2.9.1.10.3.3 INPUT/OUTPUT 612
3.3.5.2.9.1.10.3.4 LOCAL DATA 612
3.3.5.2.9.1.10.3.5 PROCESS CONTROL 612
3.3.5.2.9.1.10.3.6 PROCESSING 612
3.3.5.2.9.1.10.3.7 UTILIZATION OF OTHER ELEMENTS 612
3.3.5.2.9.1.10.3.8 LIMITATIONS 613
3.3.5.2.9.1.10.4 UPDATE_THROTTLE_RATE_LIMIT UNIT DESIGN 613
3.3.5.2.9.1.10.4.1 REQUIREMENTS ALLOCATION 613
3.3.5.2.9.1.10.4.2 LOCAL ENTITIES DESIGN 613
3.3.5.2.9.1.10.4.3 INPUT/OUTPUT 613
3.3.5.2.9.1.10.4.4 LOCAL DATA 614
3.3.5.2.9.1.10.4.5 PROCESS CONTROL 614
3.3.5.2.9.1.10.4.6 PROCESSING 614
3.3.5.2.9.1.10.4.7 UTILIZATION OF OTHER ELEMENTS 614
3.3.5.2.9.1.10.4.8 LIMITATIONS 615
3.3.5.2.9.1.10.5 UPDATE_THROTTLE_COMMAND_LIMITS UNIT DESIGN 615
3.3.5.2.9.1.10.5.1 REQUIREMENTS ALLOCATION 615
3.3.5.2.9.1.10.5.2 LOCAL ENTITIES DESIGN 615
3.3.5.2.9.1.10.5.3 INPUT/OUTPUT 615
3.3.5.2.9.1.10.5.4 LOCAL DATA 615
3.3.5.2.9.1.10.5.5 PROCESS CONTROL 615
3.3.5.2.9.1.10.5.6 PROCESSING 615
3.3.5.2.9.1.10.5.7 UTILIZATION OF OTHER ELEMENTS 616
3.3.5.2.9.1.10.5.8 LIMITATIONS 616
3.3.5.2.9.1.10.6 UPDATE_MACH_ERROR_GAIN UNIT DESIGN 616
3.3.5.2.9.1.10.6.1 REQUIREMENTS ALLOCATION 616
3.3.5.2.9.1.10.6.2 LOCAL ENTITIES DESIGN 616
3.3.5.2.9.1.10.6.3 INPUT/OUTPUT 617
3.3.5.2.9.1.10.6.4 LOCAL DATA 617

3.3.5.2.9.1.10.6.5 PROCESS CONTROL 617
3.3.5.2.9.1.10.6.6 PROCESSING 617
3.3.5.2.9.1.10.6.7 UTILIZATION OF OTHER ELEMENTS 617
3.3.5.2.9.1.10.6.8 LIMITATIONS 618
3.3.5.2.9.1.10.7 UPDATE_THROTTLE_BANDWIDTH UNIT DESIGN 618
3.3.5.2.9.1.10.7.1 REQUIREMENTS ALLOCATION 618
3.3.5.2.9.1.10.7.2 LOCAL ENTITIES DESIGN 618
3.3.5.2.9.1.10.7.3 INPUT/OUTPUT 618
3.3.5.2.9.1.10.7.4 LOCAL DATA 618
3.3.5.2.9.1.10.7.5 PROCESS CONTROL 618
3.3.5.2.9.1.10.7.6 PROCESSING 618
3.3.5.2.9.1.10.7.7 UTILIZATION OF OTHER ELEMENTS 619
3.3.5.2.9.1.10.7.8 LIMITATIONS 619
3.3.5.2.10 UNIT DESIGN 619

3.3.6 MATHEMATICAL 625

3.3.6.1 COORDINATE VECTOR MATRIX ALGEBRA (BODY) TLCSC P681 (CATALOG #P53-0) 627
3.3.6.1.1 REQUIREMENTS ALLOCATION 627
3.3.6.1.2 LOCAL ENTITIES DESIGN 627
3.3.6.1.3 INPUT/OUTPUT 627
3.3.6.1.4 LOCAL DATA 627
3.3.6.1.5 PROCESS CONTROL 627
3.3.6.1.6 PROCESSING 628
3.3.6.1.7 UTILIZATION OF OTHER ELEMENTS 628
3.3.6.1.8 LIMITATIONS 628
3.3.6.1.9 LLCSC DESIGN 628
3.3.6.1.9.1 VECTOR OPERATIONS PACKAGE DESIGN (CATALOG #P54-0) 628
3.3.6.1.9.1.1 REQUIREMENTS ALLOCATION 628
3.3.6.1.9.1.2 LOCAL ENTITIES DESIGN 629
3.3.6.1.9.1.3 INPUT/OUTPUT 629
3.3.6.1.9.1.4 LOCAL DATA 630
3.3.6.1.9.1.5 PROCESS CONTROL 630
3.3.6.1.9.1.6 PROCESSING 630
3.3.6.1.9.1.7 UTILIZATION OF OTHER ELEMENTS 631
3.3.6.1.9.1.8 LIMITATIONS 631
3.3.6.1.9.1.9 LLCSC DESIGN 631
3.3.6.1.9.1.10 UNIT DESIGN 632
3.3.6.1.9.1.10.1 "+" UNIT DESIGN (CATALOG #P693-0) 632
3.3.6.1.9.1.10.1.1 REQUIREMENTS ALLOCATION 632
3.3.6.1.9.1.10.1.2 LOCAL ENTITIES DESIGN 632
3.3.6.1.9.1.10.1.3 INPUT/OUTPUT 632
3.3.6.1.9.1.10.1.4 LOCAL DATA 632
3.3.6.1.9.1.10.1.5 PROCESS CONTROL 632
3.3.6.1.9.1.10.1.6 PROCESSING 632
3.3.6.1.9.1.10.1.7 UTILIZATION OF OTHER ELEMENTS 633
3.3.6.1.9.1.10.1.8 LIMITATIONS 634
3.3.6.1.9.1.10.2 "-" UNIT DESIGN (CATALOG #P694-0) 634
3.3.6.1.9.1.10.2.1 REQUIREMENTS ALLOCATION 634
3.3.6.1.9.1.10.2.2 LOCAL ENTITIES DESIGN 634
3.3.6.1.9.1.10.2.3 INPUT/OUTPUT 634
3.3.6.1.9.1.10.2.4 LOCAL DATA 634
3.3.6.1.9.1.10.2.5 PROCESS CONTROL 635
3.3.6.1.9.1.10.2.6 PROCESSING 635
3.3.6.1.9.1.10.2.7 UTILIZATION OF OTHER ELEMENTS 635

3.3.6.1.9.1.10.2.8 LIMITATIONS 636
3.3.6.1.9.1.10.3 VECTOR_LENGTH UNIT DESIGN (CATALOG #P55-0) 636
3.3.6.1.9.1.10.3.1 REQUIREMENTS ALLOCATION 636
3.3.6.1.9.1.10.3.2 LOCAL ENTITIES DESIGN 637
3.3.6.1.9.1.10.3.3 INPUT/OUTPUT 637
3.3.6.1.9.1.10.3.4 LOCAL DATA 637
3.3.6.1.9.1.10.3.5 PROCESS CONTROL 637
3.3.6.1.9.1.10.3.6 PROCESSING 637
3.3.6.1.9.1.10.3.7 UTILIZATION OF OTHER ELEMENTS 637
3.3.6.1.9.1.10.3.8 LIMITATIONS 638
3.3.6.1.9.1.10.4 DOT_PRODUCT UNIT DESIGN (CATALOG #P56-0) 639
3.3.6.1.9.1.10.4.1 REQUIREMENTS ALLOCATION 639
3.3.6.1.9.1.10.4.2 LOCAL ENTITIES DESIGN 639
3.3.6.1.9.1.10.4.3 INPUT/OUTPUT 639
3.3.6.1.9.1.10.4.4 LOCAL DATA 639
3.3.6.1.9.1.10.4.5 PROCESS CONTROL 639
3.3.6.1.9.1.10.4.6 PROCESSING 639
3.3.6.1.9.1.10.4.7 UTILIZATION OF OTHER ELEMENTS 640
3.3.6.1.9.1.10.4.8 LIMITATIONS 641
3.3.6.1.9.1.10.5 SPARSE RIGHT Z ADD UNIT DESIGN (CATALOG #P57-0) 641
3.3.6.1.9.1.10.5.1 REQUIREMENTS ALLOCATION 641
3.3.6.1.9.1.10.5.2 LOCAL ENTITIES DESIGN 641
3.3.6.1.9.1.10.5.3 INPUT/OUTPUT 641
3.3.6.1.9.1.10.5.4 LOCAL DATA 642
3.3.6.1.9.1.10.5.5 PROCESS CONTROL 642
3.3.6.1.9.1.10.5.6 PROCESSING 642
3.3.6.1.9.1.10.5.7 UTILIZATION OF OTHER ELEMENTS 642
3.3.6.1.9.1.10.5.8 LIMITATIONS 643
3.3.6.1.9.1.10.6 SPARSE RIGHT X ADD UNIT DESIGN (CATALOG #P58-0) 643
3.3.6.1.9.1.10.6.1 REQUIREMENTS ALLOCATION 643
3.3.6.1.9.1.10.6.2 LOCAL ENTITIES DESIGN 644
3.3.6.1.9.1.10.6.3 INPUT/OUTPUT 644
3.3.6.1.9.1.10.6.4 LOCAL DATA 644
3.3.6.1.9.1.10.6.5 PROCESS CONTROL 644
3.3.6.1.9.1.10.6.6 PROCESSING 644
3.3.6.1.9.1.10.6.7 UTILIZATION OF OTHER ELEMENTS 645
3.3.6.1.9.1.10.6.8 LIMITATIONS 646
3.3.6.1.9.1.10.7 SPARSE RIGHT XY SUBTRACT UNIT DESIGN (CATALOG #P59-0) 646
3.3.6.1.9.1.10.7.1 REQUIREMENTS ALLOCATION 646
3.3.6.1.9.1.10.7.2 LOCAL ENTITIES DESIGN 646
3.3.6.1.9.1.10.7.3 INPUT/OUTPUT 646
3.3.6.1.9.1.10.7.4 LOCAL DATA 646
3.3.6.1.9.1.10.7.5 PROCESS CONTROL 646
3.3.6.1.9.1.10.7.6 PROCESSING 647
3.3.6.1.9.1.10.7.7 UTILIZATION OF OTHER ELEMENTS 647
3.3.6.1.9.1.10.7.8 LIMITATIONS 648
3.3.6.1.9.1.10.8 SET_TO_ZERO_VECTOR UNIT DESIGN (CATALOG #P60-0) 648
3.3.6.1.9.1.10.8.1 REQUIREMENTS ALLOCATION 648
3.3.6.1.9.1.10.8.2 LOCAL ENTITIES DESIGN 648
3.3.6.1.9.1.10.8.3 INPUT/OUTPUT 648
3.3.6.1.9.1.10.8.4 LOCAL DATA 648
3.3.6.1.9.1.10.8.5 PROCESS CONTROL 649
3.3.6.1.9.1.10.8.6 PROCESSING 649

3.3.6.1.9.1.10.8.7 UTILIZATION OF OTHER ELEMENTS 649
3.3.6.1.9.1.10.8.8 LIMITATIONS 650
3.3.6.1.9.2 VECTOR_SCALAR_OPERATIONS (BODY) PACKAGE DESIGN (CATALOG #P64-0) 650
3.3.6.1.9.2.1 REQUIREMENTS ALLOCATION 650
3.3.6.1.9.2.2 LOCAL ENTITIES DESIGN 650
3.3.6.1.9.2.3 INPUT/OUTPUT 650
3.3.6.1.9.2.4 LOCAL DATA 650
3.3.6.1.9.2.5 PROCESS CONTROL 651
3.3.6.1.9.2.6 PROCESSING 651
3.3.6.1.9.2.7 UTILIZATION OF OTHER ELEMENTS 651
3.3.6.1.9.2.8 LIMITATIONS 651
3.3.6.1.9.2.9 LLCSC DESIGN 651
3.3.6.1.9.2.10 UNIT DESIGN 651
3.3.6.1.9.2.10.1 "*" UNIT DESIGN (CATALOG #P700-0) 651
3.3.6.1.9.2.10.1.1 REQUIREMENTS ALLOCATION 652
3.3.6.1.9.2.10.1.2 LOCAL ENTITIES DESIGN 652
3.3.6.1.9.2.10.1.3 INPUT/OUTPUT 652
3.3.6.1.9.2.10.1.4 LOCAL DATA 652
3.3.6.1.9.2.10.1.5 PROCESS CONTROL 652
3.3.6.1.9.2.10.1.6 PROCESSING 652
3.3.6.1.9.2.10.1.7 UTILIZATION OF OTHER ELEMENTS 653
3.3.6.1.9.2.10.1.8 LIMITATIONS 653
3.3.6.1.9.2.10.2 "/" UNIT DESIGN (CATALOG #P699-0) 654
3.3.6.1.9.2.10.2.1 REQUIREMENTS ALLOCATION 654
3.3.6.1.9.2.10.2.2 LOCAL ENTITIES DESIGN 654
3.3.6.1.9.2.10.2.3 INPUT/OUTPUT 654
3.3.6.1.9.2.10.2.4 LOCAL DATA 654
3.3.6.1.9.2.10.2.5 PROCESS CONTROL 654
3.3.6.1.9.2.10.2.6 PROCESSING 654
3.3.6.1.9.2.10.2.7 UTILIZATION OF OTHER ELEMENTS 655
3.3.6.1.9.2.10.2.8 LIMITATIONS 656
3.3.6.1.9.2.10.3 SPARSE X VECTOR SCALAR MULTIPLY UNIT DESIGN (CATALOG #P65-0) 656
3.3.6.1.9.2.10.3.1 REQUIREMENTS ALLOCATION 656
3.3.6.1.9.2.10.3.2 LOCAL ENTITIES DESIGN 656
3.3.6.1.9.2.10.3.3 INPUT/OUTPUT 656
3.3.6.1.9.2.10.3.4 LOCAL DATA 656
3.3.6.1.9.2.10.3.5 PROCESS CONTROL 657
3.3.6.1.9.2.10.3.6 PROCESSING 657
3.3.6.1.9.2.10.3.7 UTILIZATION OF OTHER ELEMENTS 657
3.3.6.1.9.2.10.3.8 LIMITATIONS 658
3.3.6.1.9.3 MATRIX OPERATIONS PACKAGE DESIGN (CATALOG #P61-0) 658
3.3.6.1.9.3.1 REQUIREMENTS ALLOCATION 658
3.3.6.1.9.3.2 LOCAL ENTITIES DESIGN 659
3.3.6.1.9.3.3 INPUT/OUTPUT 659
3.3.6.1.9.3.4 LOCAL DATA 659
3.3.6.1.9.3.5 PROCESS CONTROL 659
3.3.6.1.9.3.6 PROCESSING 659
3.3.6.1.9.3.7 UTILIZATION OF OTHER ELEMENTS 660
3.3.6.1.9.3.8 LIMITATIONS 660
3.3.6.1.9.3.9 LLCSC DESIGN 660
3.3.6.1.9.3.10 UNIT DESIGN 660
3.3.6.1.9.3.10.1 "+" (MATRICES + MATRICES) UNIT DESIGN (CATALOG #P695-0) 660
3.3.6.1.9.3.10.1.1 REQUIREMENTS ALLOCATION 661

3.3.6.1.9.3.10.1.2 LOCAL ENTITIES DESIGN 661
3.3.6.1.9.3.10.1.3 INPUT/OUTPUT 661
3.3.6.1.9.3.10.1.4 LOCAL DATA 661
3.3.6.1.9.3.10.1.5 PROCESS CONTROL 661
3.3.6.1.9.3.10.1.6 PROCESSING 661
3.3.6.1.9.3.10.1.7 UTILIZATION OF OTHER ELEMENTS 662
3.3.6.1.9.3.10.1.8 LIMITATIONS 662
3.3.6.1.9.3.10.2 "-" (MATRICES - MATRICES) UNIT DESIGN (CATALOG #P696-0) 662
3.3.6.1.9.3.10.2.1 REQUIREMENTS ALLOCATION 662
3.3.6.1.9.3.10.2.2 LOCAL ENTITIES DESIGN 662
3.3.6.1.9.3.10.2.3 INPUT/OUTPUT 662
3.3.6.1.9.3.10.2.4 LOCAL DATA 663
3.3.6.1.9.3.10.2.5 PROCESS CONTROL 663
3.3.6.1.9.3.10.2.6 PROCESSING 663
3.3.6.1.9.3.10.2.7 UTILIZATION OF OTHER ELEMENTS 664
3.3.6.1.9.3.10.2.8 LIMITATIONS 664
3.3.6.1.9.3.10.3 "+" (MATRICES + ELEMENTS) UNIT DESIGN (CATALOG #P697-0) 664
3.3.6.1.9.3.10.3.1 REQUIREMENTS ALLOCATION 664
3.3.6.1.9.3.10.3.2 LOCAL ENTITIES DESIGN 664
3.3.6.1.9.3.10.3.3 INPUT/OUTPUT 664
3.3.6.1.9.3.10.3.4 LOCAL DATA 664
3.3.6.1.9.3.10.3.5 PROCESS CONTROL 665
3.3.6.1.9.3.10.3.6 PROCESSING 665
3.3.6.1.9.3.10.3.7 UTILIZATION OF OTHER ELEMENTS 665
3.3.6.1.9.3.10.3.8 LIMITATIONS 666
3.3.6.1.9.3.10.4 "-" (MATRICES - ELEMENTS) UNIT DESIGN (CATALOG #P698-0) 666
3.3.6.1.9.3.10.4.1 REQUIREMENTS ALLOCATION 666
3.3.6.1.9.3.10.4.2 LOCAL ENTITIES DESIGN 666
3.3.6.1.9.3.10.4.3 INPUT/OUTPUT 666
3.3.6.1.9.3.10.4.4 LOCAL DATA 666
3.3.6.1.9.3.10.4.5 PROCESS CONTROL 666
3.3.6.1.9.3.10.4.6 PROCESSING 667
3.3.6.1.9.3.10.4.7 UTILIZATION OF OTHER ELEMENTS 667
3.3.6.1.9.3.10.4.8 LIMITATIONS 667
3.3.6.1.9.3.10.5 SET TO IDENTITY MATRIX UNIT DESIGN (CATALOG #P62-0) 667
3.3.6.1.9.3.10.5.1 REQUIREMENTS ALLOCATION 668
3.3.6.1.9.3.10.5.2 LOCAL ENTITIES DESIGN 668
3.3.6.1.9.3.10.5.3 INPUT/OUTPUT 668
3.3.6.1.9.3.10.5.4 LOCAL DATA 668
3.3.6.1.9.3.10.5.5 PROCESS CONTROL 668
3.3.6.1.9.3.10.5.6 PROCESSING 668
3.3.6.1.9.3.10.5.7 UTILIZATION OF OTHER ELEMENTS 668
3.3.6.1.9.3.10.5.8 LIMITATIONS 668
3.3.6.1.9.3.10.6 SET TO ZERO MATRIX UNIT DESIGN (CATALOG #P63-0) 668
3.3.6.1.9.3.10.6.1 REQUIREMENTS ALLOCATION 669
3.3.6.1.9.3.10.6.2 LOCAL ENTITIES DESIGN 669
3.3.6.1.9.3.10.6.3 INPUT/OUTPUT 669
3.3.6.1.9.3.10.6.4 LOCAL DATA 669
3.3.6.1.9.3.10.6.5 PROCESS CONTROL 669
3.3.6.1.9.3.10.6.6 PROCESSING 669
3.3.6.1.9.3.10.6.7 UTILIZATION OF OTHER ELEMENTS 669
3.3.6.1.9.3.10.6.8 LIMITATIONS 669

3.3.6.1.9.4 MATRIX SCALAR OPERATIONS PACKAGE DESIGN (CATALOG #P66-0) . . . 669
3.3.6.1.9.4.1 REQUIREMENTS ALLOCATION . . . 670
3.3.6.1.9.4.2 LOCAL ENTITIES DESIGN . . . 670
3.3.6.1.9.4.3 INPUT/OUTPUT . . . 670
3.3.6.1.9.4.4 LOCAL DATA . . . 670
3.3.6.1.9.4.5 PROCESS CONTROL . . . 670
3.3.6.1.9.4.6 PROCESSING . . . 670
3.3.6.1.9.4.7 UTILIZATION OF OTHER ELEMENTS . . . 671
3.3.6.1.9.4.8 LIMITATIONS . . . 671
3.3.6.1.9.4.9 LLCSC DESIGN . . . 671
3.3.6.1.9.4.10 UNIT DESIGN . . . 671
3.3.6.1.9.4.10.1 "*" UNIT DESIGN (CATALOG #P701-0) . . . 671
3.3.6.1.9.4.10.1.1 REQUIREMENTS ALLOCATION . . . 671
3.3.6.1.9.4.10.1.2 LOCAL ENTITIES DESIGN . . . 671
3.3.6.1.9.4.10.1.3 INPUT/OUTPUT . . . 671
3.3.6.1.9.4.10.1.4 LOCAL DATA . . . 672
3.3.6.1.9.4.10.1.5 PROCESS CONTROL . . . 672
3.3.6.1.9.4.10.1.6 PROCESSING . . . 672
3.3.6.1.9.4.10.1.7 UTILIZATION OF OTHER ELEMENTS . . . 672
3.3.6.1.9.4.10.1.8 LIMITATIONS . . . 673
3.3.6.1.9.4.10.2 "/" UNIT DESIGN (CATALOG #P702-0) . . . 673
3.3.6.1.9.4.10.2.1 REQUIREMENTS ALLOCATION . . . 673
3.3.6.1.9.4.10.2.2 LOCAL ENTITIES DESIGN . . . 673
3.3.6.1.9.4.10.2.3 INPUT/OUTPUT . . . 673
3.3.6.1.9.4.10.2.4 LOCAL DATA . . . 673
3.3.6.1.9.4.10.2.5 PROCESS CONTROL . . . 673
3.3.6.1.9.4.10.2.6 PROCESSING . . . 674
3.3.6.1.9.4.10.2.7 UTILIZATION OF OTHER ELEMENTS . . . 674
3.3.6.1.9.4.10.2.8 LIMITATIONS . . . 674
3.3.6.1.10 UNIT DESIGN . . . 674
3.3.6.1.10.1 CROSS PRODUCT UNIT DESIGN (CATALOG #P67-0) . . . 674
3.3.6.1.10.1.1 REQUIREMENTS ALLOCATION . . . 675
3.3.6.1.10.1.2 LOCAL ENTITIES DESIGN . . . 675
3.3.6.1.10.1.3 INPUT/OUTPUT . . . 675
3.3.6.1.10.1.4 LOCAL DATA . . . 675
3.3.6.1.10.1.5 PROCESS CONTROL . . . 675
3.3.6.1.10.1.6 PROCESSING . . . 676
3.3.6.1.10.1.7 UTILIZATION OF OTHER ELEMENTS . . . 676
3.3.6.1.10.1.8 LIMITATIONS . . . 676
3.3.6.1.10.2 MATRIX VECTOR MULTIPLY UNIT DESIGN (CATALOG #P68-0) . . . 676
3.3.6.1.10.2.1 REQUIREMENTS ALLOCATION . . . 676
3.3.6.1.10.2.2 LOCAL ENTITIES DESIGN . . . 677
3.3.6.1.10.2.3 INPUT/OUTPUT . . . 677
3.3.6.1.10.2.4 LOCAL DATA . . . 677
3.3.6.1.10.2.5 PROCESS CONTROL . . . 677
3.3.6.1.10.2.6 PROCESSING . . . 677
3.3.6.1.10.2.7 UTILIZATION OF OTHER ELEMENTS . . . 678
3.3.6.1.10.2.8 LIMITATIONS . . . 678
3.3.6.1.10.3 MATRIX MATRIX MULTIPLY UNIT DESIGN (CATALOG #P69-0) . . . 678
3.3.6.1.10.3.1 REQUIREMENTS ALLOCATION . . . 678
3.3.6.1.10.3.2 LOCAL ENTITIES DESIGN . . . 679
3.3.6.1.10.3.3 INPUT/OUTPUT . . . 679
3.3.6.1.10.3.4 LOCAL DATA . . . 679
3.3.6.1.10.3.5 PROCESS CONTROL . . . 679

3.3.6.1.10.3.6 PROCESSING 679
3.3.6.1.10.3.7 UTILIZATION OF OTHER ELEMENTS 681
3.3.6.1.10.3.8 LIMITATIONS 681

3.3.6.2 GENERAL_VECTOR_MATRIX_ALGEBRA (BODY) TLCSC P682 (CATALOG #P197-0) 701
3.3.6.2.1 REQUIREMENTS ALLOCATION 701
3.3.6.2.2 LOCAL ENTITIES DESIGN 702
3.3.6.2.3 INPUT/OUTPUT 702
3.3.6.2.4 LOCAL DATA 703
3.3.6.2.5 PROCESS CONTROL 703
3.3.6.2.6 PROCESSING 703
3.3.6.2.7 UTILIZATION OF OTHER ELEMENTS 704
3.3.6.2.8 LIMITATIONS 704
3.3.6.2.9 LLCSC DESIGN 705
3.3.6.2.9.1 VECTOR_OPERATIONS_UNCONSTRAINED PACKAGE DESIGN (CATALOG #P337-0) 705
3.3.6.2.9.1.1 REQUIREMENTS ALLOCATION 705
3.3.6.2.9.1.2 LOCAL ENTITIES DESIGN 705
3.3.6.2.9.1.3 INPUT/OUTPUT 705
3.3.6.2.9.1.4 LOCAL DATA 706
3.3.6.2.9.1.5 PROCESS CONTROL 706
3.3.6.2.9.1.6 PROCESSING 706
3.3.6.2.9.1.7 UTILIZATION OF OTHER ELEMENTS 706
3.3.6.2.9.1.8 LIMITATIONS 706
3.3.6.2.9.1.9 LLCSC DESIGN 706
3.3.6.2.9.1.10 UNIT DESIGN 707
3.3.6.2.9.1.10.1 "+" (VECTOR + VECTORS := VECTORS) UNIT DESIGN (CATALOG #P338-0) 707
3.3.6.2.9.1.10.1.1 REQUIREMENTS ALLOCATION 707
3.3.6.2.9.1.10.1.2 LOCAL ENTITIES DESIGN 707
3.3.6.2.9.1.10.1.3 INPUT/OUTPUT 707
3.3.6.2.9.1.10.1.4 LOCAL DATA 707
3.3.6.2.9.1.10.1.5 PROCESS CONTROL 707
3.3.6.2.9.1.10.1.6 PROCESSING 708
3.3.6.2.9.1.10.1.7 UTILIZATION OF OTHER ELEMENTS 708
3.3.6.2.9.1.10.1.8 LIMITATIONS 709
3.3.6.2.9.1.10.2 "-" (VECTORS - VECTORS := VECTORS) UNIT DESIGN (CATALOG #P339-0) 710
3.3.6.2.9.1.10.2.1 REQUIREMENTS ALLOCATION 710
3.3.6.2.9.1.10.2.2 LOCAL ENTITIES DESIGN 710
3.3.6.2.9.1.10.2.3 INPUT/OUTPUT 710
3.3.6.2.9.1.10.2.4 LOCAL DATA 710
3.3.6.2.9.1.10.2.5 PROCESS CONTROL 710
3.3.6.2.9.1.10.2.6 PROCESSING 710
3.3.6.2.9.1.10.2.7 UTILIZATION OF OTHER ELEMENTS 711
3.3.6.2.9.1.10.2.8 LIMITATIONS 712
3.3.6.2.9.1.10.3 VECTOR_LENGTH UNIT DESIGN (CATALOG #P340-0) 712
3.3.6.2.9.1.10.3.1 REQUIREMENTS ALLOCATION 713
3.3.6.2.9.1.10.3.2 LOCAL ENTITIES DESIGN 713
3.3.6.2.9.1.10.3.3 INPUT/OUTPUT 713
3.3.6.2.9.1.10.3.4 LOCAL DATA 713
3.3.6.2.9.1.10.3.5 PROCESS CONTROL 713
3.3.6.2.9.1.10.3.6 PROCESSING 713
3.3.6.2.9.1.10.3.7 UTILIZATION OF OTHER ELEMENTS 714
3.3.6.2.9.1.10.3.8 LIMITATIONS 715

3.3.6.2.9.1.10.4 DOT PRODUCT UNIT DESIGN (CATALOG #P341-0) 715
3.3.6.2.9.1.10.4.1 REQUIREMENTS ALLOCATION 715
3.3.6.2.9.1.10.4.2 LOCAL ENTITIES DESIGN 715
3.3.6.2.9.1.10.4.3 INPUT/OUTPUT 715
3.3.6.2.9.1.10.4.4 LOCAL DATA 716
3.3.6.2.9.1.10.4.5 PROCESS CONTROL 716
3.3.6.2.9.1.10.4.6 PROCESSING 716
3.3.6.2.9.1.10.4.7 UTILIZATION OF OTHER ELEMENTS 717
3.3.6.2.9.1.10.4.8 LIMITATIONS 718
3.3.6.2.9.2 MATRIX_OPERATIONS_UNCONSTRAINED PACKAGE DESIGN (CATALOG #P347-0) 718
3.3.6.2.9.2.1 REQUIREMENTS ALLOCATION 719
3.3.6.2.9.2.2 LOCAL ENTITIES DESIGN 719
3.3.6.2.9.2.3 INPUT/OUTPUT 719
3.3.6.2.9.2.4 LOCAL DATA 719
3.3.6.2.9.2.5 PROCESS CONTROL 720
3.3.6.2.9.2.6 PROCESSING 720
3.3.6.2.9.2.7 UTILIZATION OF OTHER ELEMENTS 720
3.3.6.2.9.2.8 LIMITATIONS 720
3.3.6.2.9.2.9 LLCSC DESIGN 720
3.3.6.2.9.2.10 UNIT DESIGN 720
3.3.6.2.9.2.10.1 "+" (MATRICES + MATRICES := MATRICES) UNIT DESIGN (CATALOG #P348-0) 720
3.3.6.2.9.2.10.1.1 REQUIREMENTS ALLOCATION 720
3.3.6.2.9.2.10.1.2 LOCAL ENTITIES DESIGN 721
3.3.6.2.9.2.10.1.3 INPUT/OUTPUT 721
3.3.6.2.9.2.10.1.4 LOCAL DATA 721
3.3.6.2.9.2.10.1.5 PROCESS CONTROL 721
3.3.6.2.9.2.10.1.6 PROCESSING 721
3.3.6.2.9.2.10.1.7 UTILIZATION OF OTHER ELEMENTS 722
3.3.6.2.9.2.10.1.8 LIMITATIONS 723
3.3.6.2.9.2.10.2 "-" (MATRICES - MATRICES := MATRICES) UNIT DESIGN (CATALOG #P349-0) 723
3.3.6.2.9.2.10.2.1 REQUIREMENTS ALLOCATION 724
3.3.6.2.9.2.10.2.2 LOCAL ENTITIES DESIGN 724
3.3.6.2.9.2.10.2.3 INPUT/OUTPUT 724
3.3.6.2.9.2.10.2.4 LOCAL DATA 724
3.3.6.2.9.2.10.2.5 PROCESS CONTROL 724
3.3.6.2.9.2.10.2.6 PROCESSING 724
3.3.6.2.9.2.10.2.7 UTILIZATION OF OTHER ELEMENTS 726
3.3.6.2.9.2.10.2.8 LIMITATIONS 726
3.3.6.2.9.2.10.3 "+" (MATRICES + ELEMENTS := MATRICES) UNIT DESIGN (CATALOG #P350-0) 727
3.3.6.2.9.2.10.3.1 REQUIREMENTS ALLOCATION 727
3.3.6.2.9.2.10.3.2 LOCAL ENTITIES DESIGN 727
3.3.6.2.9.2.10.3.3 INPUT/OUTPUT 727
3.3.6.2.9.2.10.3.4 LOCAL DATA 727
3.3.6.2.9.2.10.3.5 PROCESS CONTROL 727
3.3.6.2.9.2.10.3.6 PROCESSING 727
3.3.6.2.9.2.10.3.7 UTILIZATION OF OTHER ELEMENTS 728
3.3.6.2.9.2.10.3.8 LIMITATIONS 729
3.3.6.2.9.2.10.4 "-" (MATRICES - ELEMENTS := MATRICES) UNIT DESIGN (CATALOG #P351-0) 729
3.3.6.2.9.2.10.4.1 REQUIREMENTS ALLOCATION 729
3.3.6.2.9.2.10.4.2 LOCAL ENTITIES DESIGN 729
3.3.6.2.9.2.10.4.3 INPUT/OUTPUT 729
3.3.6.2.9.2.10.4.4 LOCAL DATA 729

3.3.6.2.9.2.10.4.5 PROCESS CONTROL 729
3.3.6.2.9.2.10.4.6 PROCESSING 730
3.3.6.2.9.2.10.4.7 UTILIZATION OF OTHER ELEMENTS 730
3.3.6.2.9.2.10.4.8 LIMITATIONS 731
3.3.6.2.9.2.10.5 SET_TO_IDENTITY_MATRIX UNIT DESIGN (CATALOG #P352-0) 731
3.3.6.2.9.2.10.5.1 REQUIREMENTS ALLOCATION 731
3.3.6.2.9.2.10.5.2 LOCAL ENTITIES DESIGN 731
3.3.6.2.9.2.10.5.3 INPUT/OUTPUT 731
3.3.6.2.9.2.10.5.4 LOCAL DATA 731
3.3.6.2.9.2.10.5.5 PROCESS CONTROL 732
3.3.6.2.9.2.10.5.6 PROCESSING 732
3.3.6.2.9.2.10.5.7 UTILIZATION OF OTHER ELEMENTS 733
3.3.6.2.9.2.10.5.8 LIMITATIONS 733
3.3.6.2.9.2.10.6 SET TO ZERO MATRIX UNIT DESIGN (CATALOG #P353-0) 734
3.3.6.2.9.2.10.6.1 REQUIREMENTS ALLOCATION 734
3.3.6.2.9.2.10.6.2 LOCAL ENTITIES DESIGN 734
3.3.6.2.9.2.10.6.3 INPUT/OUTPUT 734
3.3.6.2.9.2.10.6.4 LOCAL DATA 734
3.3.6.2.9.2.10.6.5 PROCESS CONTROL 734
3.3.6.2.9.2.10.6.6 PROCESSING 734
3.3.6.2.9.2.10.6.7 UTILIZATION OF OTHER ELEMENTS 735
3.3.6.2.9.2.10.6.8 LIMITATIONS 735
3.3.6.2.9.2.10.7 "*" (MATRICES * MATRICES => MATRICES) UNIT DESIGN (CATALOG #P354-0) 735
3.3.6.2.9.2.10.7.1 REQUIREMENTS ALLOCATION 735
3.3.6.2.9.2.10.7.2 LOCAL ENTITIES DESIGN 736
3.3.6.2.9.2.10.7.3 INPUT/OUTPUT 736
3.3.6.2.9.2.10.7.4 LOCAL DATA 736
3.3.6.2.9.2.10.7.5 PROCESS CONTROL 736
3.3.6.2.9.2.10.7.6 PROCESSING 736
3.3.6.2.9.2.10.7.7 UTILIZATION OF OTHER ELEMENTS 738
3.3.6.2.9.2.10.7.8 LIMITATIONS 739
3.3.6.2.9.3 DYNAMICALLY_SPARSE_MATRIX_OPERATIONS_UNCONSTRAINED PACKAGE DESIGN (CATALOG #P362-0) 739
3.3.6.2.9.3.1 REQUIREMENTS ALLOCATION 739
3.3.6.2.9.3.2 LOCAL ENTITIES DESIGN 739
3.3.6.2.9.3.3 INPUT/OUTPUT 739
3.3.6.2.9.3.4 LOCAL DATA 740
3.3.6.2.9.3.5 PROCESS CONTROL 740
3.3.6.2.9.3.6 PROCESSING 740
3.3.6.2.9.3.7 UTILIZATION OF OTHER ELEMENTS 740
3.3.6.2.9.3.8 LIMITATIONS 740
3.3.6.2.9.3.9 LLCSC DESIGN 740
3.3.6.2.9.3.10 UNIT DESIGN 741
3.3.6.2.9.3.10.1 SET_TO_IDENTITY_MATRIX UNIT DESIGN (CATALOG #P363-0) 741
3.3.6.2.9.3.10.1.1 REQUIREMENTS ALLOCATION 741
3.3.6.2.9.3.10.1.2 LOCAL ENTITIES DESIGN 741
3.3.6.2.9.3.10.1.3 INPUT/OUTPUT 741
3.3.6.2.9.3.10.1.4 LOCAL DATA 741
3.3.6.2.9.3.10.1.5 PROCESS CONTROL 741
3.3.6.2.9.3.10.1.6 PROCESSING 741
3.3.6.2.9.3.10.1.7 UTILIZATION OF OTHER ELEMENTS 742
3.3.6.2.9.3.10.1.8 LIMITATIONS 743

3.3.6.2.9.3.10.2 SET_TO_ZERO_MATRIX UNIT DESIGN (CATALOG #P364-0) 743
3.3.6.2.9.3.10.2.1 REQUIREMENTS ALLOCATION 743
3.3.6.2.9.3.10.2.2 LOCAL ENTITIES DESIGN 744
3.3.6.2.9.3.10.2.3 INPUT/OUTPUT 744
3.3.6.2.9.3.10.2.4 LOCAL DATA 744
3.3.6.2.9.3.10.2.5 PROCESS CONTROL 744
3.3.6.2.9.3.10.2.6 PROCESSING 744
3.3.6.2.9.3.10.2.7 UTILIZATION OF OTHER ELEMENTS 744
3.3.6.2.9.3.10.2.8 LIMITATIONS 745
3.3.6.2.9.3.10.3 ADD_TO_IDENTITY UNIT DESIGN (CATALOG #P365-0) 745
3.3.6.2.9.3.10.3.1 REQUIREMENTS ALLOCATION 745
3.3.6.2.9.3.10.3.2 LOCAL ENTITIES DESIGN 745
3.3.6.2.9.3.10.3.3 INPUT/OUTPUT 745
3.3.6.2.9.3.10.3.4 LOCAL DATA 746
3.3.6.2.9.3.10.3.5 PROCESS CONTROL 746
3.3.6.2.9.3.10.3.6 PROCESSING 746
3.3.6.2.9.3.10.3.7 UTILIZATION OF OTHER ELEMENTS 747
3.3.6.2.9.3.10.3.8 LIMITATIONS 748
3.3.6.2.9.3.10.4 SUBTRACT_FROM_IDENTITY UNIT DESIGN (CATALOG #P366-0) 748
3.3.6.2.9.3.10.4.1 REQUIREMENTS ALLOCATION 748
3.3.6.2.9.3.10.4.2 LOCAL ENTITIES DESIGN 748
3.3.6.2.9.3.10.4.3 INPUT/OUTPUT 748
3.3.6.2.9.3.10.4.4 LOCAL DATA 748
3.3.6.2.9.3.10.4.5 PROCESS CONTROL 749
3.3.6.2.9.3.10.4.6 PROCESSING 749
3.3.6.2.9.3.10.4.7 UTILIZATION OF OTHER ELEMENTS 750
3.3.6.2.9.3.10.4.8 LIMITATIONS 751
3.3.6.2.9.3.10.5 "+" UNIT DESIGN (CATALOG #P367-0) 751
3.3.6.2.9.3.10.5.1 REQUIREMENTS ALLOCATION 751
3.3.6.2.9.3.10.5.2 LOCAL ENTITIES DESIGN 751
3.3.6.2.9.3.10.5.3 INPUT/OUTPUT 751
3.3.6.2.9.3.10.5.4 LOCAL DATA 752
3.3.6.2.9.3.10.5.5 PROCESS CONTROL 752
3.3.6.2.9.3.10.5.6 PROCESSING 752
3.3.6.2.9.3.10.5.7 UTILIZATION OF OTHER ELEMENTS 753
3.3.6.2.9.3.10.5.8 LIMITATIONS 754
3.3.6.2.9.3.10.6 "-" UNIT DESIGN (CATALOG #P368-0) 754
3.3.6.2.9.3.10.6.1 REQUIREMENTS ALLOCATION 755
3.3.6.2.9.3.10.6.2 LOCAL ENTITIES DESIGN 755
3.3.6.2.9.3.10.6.3 INPUT/OUTPUT 755
3.3.6.2.9.3.10.6.4 LOCAL DATA 755
3.3.6.2.9.3.10.6.5 PROCESS CONTROL 755
3.3.6.2.9.3.10.6.6 PROCESSING 755
3.3.6.2.9.3.10.6.7 UTILIZATION OF OTHER ELEMENTS 757
3.3.6.2.9.3.10.6.8 LIMITATIONS 758
3.3.6.2.9.4 SYMMETRIC_HALF_STORAGE_MATRIX_OPERATIONS PACKAGE DESIGN (CATALOG #P376-0) 758
3.3.6.2.9.4.1 REQUIREMENTS ALLOCATION 758
3.3.6.2.9.4.2 LOCAL ENTITIES DESIGN 758
3.3.6.2.9.4.3 INPUT/OUTPUT 758
3.3.6.2.9.4.4 LOCAL DATA 759
3.3.6.2.9.4.5 PROCESS CONTROL 760
3.3.6.2.9.4.6 PROCESSING 760
3.3.6.2.9.4.7 UTILIZATION OF OTHER ELEMENTS 762

3.3.6.2.9.4.8 LIMITATIONS 762
3.3.6.2.9.4.9 LLCSC DESIGN 763
3.3.6.2.9.4.10 UNIT DESIGN 763
3.3.6.2.9.4.10.1 SWAP_COL UNIT DESIGN 763
3.3.6.2.9.4.10.1.1 REQUIREMENTS ALLOCATION 763
3.3.6.2.9.4.10.1.2 LOCAL ENTITIES DESIGN 763
3.3.6.2.9.4.10.1.3 INPUT/OUTPUT 763
3.3.6.2.9.4.10.1.4 LOCAL DATA 763
3.3.6.2.9.4.10.1.5 PROCESS CONTROL 763
3.3.6.2.9.4.10.1.6 PROCESSING 764
3.3.6.2.9.4.10.1.7 UTILIZATION OF OTHER ELEMENTS 764
3.3.6.2.9.4.10.1.8 LIMITATIONS 764
3.3.6.2.9.4.10.2 SWAP_ROW UNIT DESIGN 764
3.3.6.2.9.4.10.2.1 REQUIREMENTS ALLOCATION 764
3.3.6.2.9.4.10.2.2 LOCAL ENTITIES DESIGN 764
3.3.6.2.9.4.10.2.3 INPUT/OUTPUT 765
3.3.6.2.9.4.10.2.4 LOCAL DATA 765
3.3.6.2.9.4.10.2.5 PROCESS CONTROL 765
3.3.6.2.9.4.10.2.6 PROCESSING 765
3.3.6.2.9.4.10.2.7 UTILIZATION OF OTHER ELEMENTS 765
3.3.6.2.9.4.10.2.8 LIMITATIONS 766
3.3.6.2.9.4.10.3 INITIALIZE UNIT DESIGN (CATALOG #P379-0) 766
3.3.6.2.9.4.10.3.1 REQUIREMENTS ALLOCATION 766
3.3.6.2.9.4.10.3.2 LOCAL ENTITIES DESIGN 766
3.3.6.2.9.4.10.3.3 INPUT/OUTPUT 766
3.3.6.2.9.4.10.3.4 LOCAL DATA 766
3.3.6.2.9.4.10.3.5 PROCESS CONTROL 766
3.3.6.2.9.4.10.3.6 PROCESSING 766
3.3.6.2.9.4.10.3.7 UTILIZATION OF OTHER ELEMENTS 767
3.3.6.2.9.4.10.3.8 LIMITATIONS 769
3.3.6.2.9.4.10.4 IDENTITY_MATRIX UNIT DESIGN (CATALOG #P380-0) 769
3.3.6.2.9.4.10.4.1 REQUIREMENTS ALLOCATION 769
3.3.6.2.9.4.10.4.2 LOCAL ENTITIES DESIGN 769
3.3.6.2.9.4.10.4.3 INPUT/OUTPUT 769
3.3.6.2.9.4.10.4.4 LOCAL DATA 769
3.3.6.2.9.4.10.4.5 PROCESS CONTROL 769
3.3.6.2.9.4.10.4.6 PROCESSING 769
3.3.6.2.9.4.10.4.7 UTILIZATION OF OTHER ELEMENTS 769
3.3.6.2.9.4.10.4.8 LIMITATIONS 770
3.3.6.2.9.4.10.5 ZERO_MATRIX UNIT DESIGN (CATALOG #P381-0) 770
3.3.6.2.9.4.10.5.1 REQUIREMENTS ALLOCATION 770
3.3.6.2.9.4.10.5.2 LOCAL ENTITIES DESIGN 770
3.3.6.2.9.4.10.5.3 INPUT/OUTPUT 770
3.3.6.2.9.4.10.5.4 LOCAL DATA 771
3.3.6.2.9.4.10.5.5 PROCESS CONTROL 771
3.3.6.2.9.4.10.5.6 PROCESSING 771
3.3.6.2.9.4.10.5.7 UTILIZATION OF OTHER ELEMENTS 771
3.3.6.2.9.4.10.5.8 LIMITATIONS 772
3.3.6.2.9.4.10.6 CHANGE_ELEMENT UNIT DESIGN (CATALOG #P382-0) 772
3.3.6.2.9.4.10.6.1 REQUIREMENTS ALLOCATION 772
3.3.6.2.9.4.10.6.2 LOCAL ENTITIES DESIGN 772
3.3.6.2.9.4.10.6.3 INPUT/OUTPUT 772
3.3.6.2.9.4.10.6.4 LOCAL DATA 772
3.3.6.2.9.4.10.6.5 PROCESS CONTROL 772
3.3.6.2.9.4.10.6.6 PROCESSING 772

3.3.6.2.9.4.10.6.7 UTILIZATION OF OTHER ELEMENTS 773
3.3.6.2.9.4.10.6.8 LIMITATIONS 774
3.3.6.2.9.4.10.7 RETRIEVE_ELEMENT UNIT DESIGN (CATALOG #P383-0) 774
3.3.6.2.9.4.10.7.1 REQUIREMENTS ALLOCATION 774
3.3.6.2.9.4.10.7.2 LOCAL ENTITIES DESIGN 774
3.3.6.2.9.4.10.7.3 INPUT/OUTPUT 774
3.3.6.2.9.4.10.7.4 LOCAL DATA 775
3.3.6.2.9.4.10.7.5 PROCESS CONTROL 775
3.3.6.2.9.4.10.7.6 PROCESSING 775
3.3.6.2.9.4.10.7.7 UTILIZATION OF OTHER ELEMENTS 776
3.3.6.2.9.4.10.7.8 LIMITATIONS 777
3.3.6.2.9.4.10.8 ROW_SLICE UNIT DESIGN (CATALOG #P384-0) 777
3.3.6.2.9.4.10.8.1 REQUIREMENTS ALLOCATION 777
3.3.6.2.9.4.10.8.2 LOCAL ENTITIES DESIGN 777
3.3.6.2.9.4.10.8.3 INPUT/OUTPUT 777
3.3.6.2.9.4.10.8.4 LOCAL DATA 777
3.3.6.2.9.4.10.8.5 PROCESS CONTROL 778
3.3.6.2.9.4.10.8.6 PROCESSING 778
3.3.6.2.9.4.10.8.7 UTILIZATION OF OTHER ELEMENTS 779
3.3.6.2.9.4.10.8.8 LIMITATIONS 780
3.3.6.2.9.4.10.9 COLUMN_SLICE UNIT DESIGN (CATALOG #P385-0) 780
3.3.6.2.9.4.10.9.1 REQUIREMENTS ALLOCATION 780
3.3.6.2.9.4.10.9.2 LOCAL ENTITIES DESIGN 780
3.3.6.2.9.4.10.9.3 INPUT/OUTPUT 780
3.3.6.2.9.4.10.9.4 LOCAL DATA 780
3.3.6.2.9.4.10.9.5 PROCESS CONTROL 781
3.3.6.2.9.4.10.9.6 PROCESSING 781
3.3.6.2.9.4.10.9.7 UTILIZATION OF OTHER ELEMENTS 781
3.3.6.2.9.4.10.9.8 LIMITATIONS 783
3.3.6.2.9.4.10.10 ADD_TO_IDENTITY UNIT DESIGN (CATALOG #P386-0) 783
3.3.6.2.9.4.10.10.1 REQUIREMENTS ALLOCATION 783
3.3.6.2.9.4.10.10.2 LOCAL ENTITIES DESIGN 783
3.3.6.2.9.4.10.10.3 INPUT/OUTPUT 783
3.3.6.2.9.4.10.10.4 LOCAL DATA 783
3.3.6.2.9.4.10.10.5 PROCESS CONTROL 783
3.3.6.2.9.4.10.10.6 PROCESSING 784
3.3.6.2.9.4.10.10.7 UTILIZATION OF OTHER ELEMENTS 784
3.3.6.2.9.4.10.10.8 LIMITATIONS 785
3.3.6.2.9.4.10.11 SUBTRACT_FROM_IDENTITY UNIT DESIGN (CATALOG #P387-0) 786
3.3.6.2.9.4.10.11.1 REQUIREMENTS ALLOCATION 786
3.3.6.2.9.4.10.11.2 LOCAL ENTITIES DESIGN 786
3.3.6.2.9.4.10.11.3 INPUT/OUTPUT 786
3.3.6.2.9.4.10.11.4 LOCAL DATA 786
3.3.6.2.9.4.10.11.5 PROCESS CONTROL 786
3.3.6.2.9.4.10.11.6 PROCESSING 786
3.3.6.2.9.4.10.11.7 UTILIZATION OF OTHER ELEMENTS 787
3.3.6.2.9.4.10.11.8 LIMITATIONS 788
3.3.6.2.9.4.10.12 "+" UNIT DESIGN (CATALOG #P388-0) 789
3.3.6.2.9.4.10.12.1 REQUIREMENTS ALLOCATION 789
3.3.6.2.9.4.10.12.2 LOCAL ENTITIES DESIGN 789
3.3.6.2.9.4.10.12.3 INPUT/OUTPUT 789
3.3.6.2.9.4.10.12.4 LOCAL DATA 789
3.3.6.2.9.4.10.12.5 PROCESS CONTROL 789
3.3.6.2.9.4.10.12.6 PROCESSING 789

3.3.6.2.9.4.10.12.7 UTILIZATION OF OTHER ELEMENTS 790
3.3.6.2.9.4.10.12.8 LIMITATIONS 791
3.3.6.2.9.4.10.13 "-" UNIT DESIGN (CATALOG #P389-0) 791
3.3.6.2.9.4.10.13.1 REQUIREMENTS ALLOCATION 792
3.3.6.2.9.4.10.13.2 LOCAL ENTITIES DESIGN 792
3.3.6.2.9.4.10.13.3 INPUT/OUTPUT 792
3.3.6.2.9.4.10.13.4 LOCAL DATA 792
3.3.6.2.9.4.10.13.5 PROCESS CONTROL 792
3.3.6.2.9.4.10.13.6 PROCESSING 792
3.3.6.2.9.4.10.13.7 UTILIZATION OF OTHER ELEMENTS 793
3.3.6.2.9.4.10.13.8 LIMITATIONS 794
3.3.6.2.9.5 SYMMETRIC_FULL_STORAGE_MATRIX_OPERATIONS_UNCONSTRAINED PACKAGE DESIGN (CATALOG #P390-0) 794
3.3.6.2.9.5.1 REQUIREMENTS ALLOCATION 795
3.3.6.2.9.5.2 LOCAL ENTITIES DESIGN 795
3.3.6.2.9.5.3 INPUT/OUTPUT 795
3.3.6.2.9.5.4 LOCAL DATA 795
3.3.6.2.9.5.5 PROCESS CONTROL 796
3.3.6.2.9.5.6 PROCESSING 796
3.3.6.2.9.5.7 UTILIZATION OF OTHER ELEMENTS 796
3.3.6.2.9.5.8 LIMITATIONS 796
3.3.6.2.9.5.9 LLCSC DESIGN 796
3.3.6.2.9.5.10 UNIT DESIGN 796
3.3.6.2.9.5.10.1 CHANGE_ELEMENT UNIT DESIGN (CATALOG #P391-0) 796
3.3.6.2.9.5.10.1.1 REQUIREMENTS ALLOCATION 796
3.3.6.2.9.5.10.1.2 LOCAL ENTITIES DESIGN 796
3.3.6.2.9.5.10.1.3 INPUT/OUTPUT 796
3.3.6.2.9.5.10.1.4 LOCAL DATA 797
3.3.6.2.9.5.10.1.5 PROCESS CONTROL 797
3.3.6.2.9.5.10.1.6 PROCESSING 797
3.3.6.2.9.5.10.1.7 UTILIZATION OF OTHER ELEMENTS 798
3.3.6.2.9.5.10.1.8 LIMITATIONS 799
3.3.6.2.9.5.10.2 SET TO IDENTITY MATRIX UNIT DESIGN (CATALOG #P392-0) 799
3.3.6.2.9.5.10.2.1 REQUIREMENTS ALLOCATION 799
3.3.6.2.9.5.10.2.2 LOCAL ENTITIES DESIGN 799
3.3.6.2.9.5.10.2.3 INPUT/OUTPUT 799
3.3.6.2.9.5.10.2.4 LOCAL DATA 800
3.3.6.2.9.5.10.2.5 PROCESS CONTROL 800
3.3.6.2.9.5.10.2.6 PROCESSING 800
3.3.6.2.9.5.10.2.7 UTILIZATION OF OTHER ELEMENTS 801
3.3.6.2.9.5.10.2.8 LIMITATIONS 802
3.3.6.2.9.5.10.3 SET TO ZERO MATRIX UNIT DESIGN (CATALOG #P393-0) 802
3.3.6.2.9.5.10.3.1 REQUIREMENTS ALLOCATION 802
3.3.6.2.9.5.10.3.2 LOCAL ENTITIES DESIGN 802
3.3.6.2.9.5.10.3.3 INPUT/OUTPUT 802
3.3.6.2.9.5.10.3.4 LOCAL DATA 803
3.3.6.2.9.5.10.3.5 PROCESS CONTROL 803
3.3.6.2.9.5.10.3.6 PROCESSING 803
3.3.6.2.9.5.10.3.7 UTILIZATION OF OTHER ELEMENTS 803
3.3.6.2.9.5.10.3.8 LIMITATIONS 804
3.3.6.2.9.5.10.4 ADD_TO_IDENTITY UNIT DESIGN (CATALOG #P394-0) 804
3.3.6.2.9.5.10.4.1 REQUIREMENTS ALLOCATION 804
3.3.6.2.9.5.10.4.2 LOCAL ENTITIES DESIGN 804

3.3.6.2.9.5.10.4.3 INPUT/OUTPUT 804
3.3.6.2.9.5.10.4.4 LOCAL DATA 804
3.3.6.2.9.5.10.4.5 PROCESS CONTROL 804
3.3.6.2.9.5.10.4.6 PROCESSING 805
3.3.6.2.9.5.10.4.7 UTILIZATION OF OTHER ELEMENTS 805
3.3.6.2.9.5.10.4.8 LIMITATIONS 806
3.3.6.2.9.5.10.5 SUBTRACT_FROM_IDENTITY UNIT DESIGN (CATALOG #P395-0) 806
3.3.6.2.9.5.10.5.1 REQUIREMENTS ALLOCATION 807
3.3.6.2.9.5.10.5.2 LOCAL ENTITIES DESIGN 807
3.3.6.2.9.5.10.5.3 INPUT/OUTPUT 807
3.3.6.2.9.5.10.5.4 LOCAL DATA 807
3.3.6.2.9.5.10.5.5 PROCESS CONTROL 808
3.3.6.2.9.5.10.5.6 PROCESSING 808
3.3.6.2.9.5.10.5.7 UTILIZATION OF OTHER ELEMENTS 809
3.3.6.2.9.5.10.5.8 LIMITATIONS 810
3.3.6.2.9.5.10.6 "+" UNIT DESIGN (CATALOG #P396-0) 810
3.3.6.2.9.5.10.6.1 REQUIREMENTS ALLOCATION 810
3.3.6.2.9.5.10.6.2 LOCAL ENTITIES DESIGN 811
3.3.6.2.9.5.10.6.3 INPUT/OUTPUT 811
3.3.6.2.9.5.10.6.4 LOCAL DATA 811
3.3.6.2.9.5.10.6.5 PROCESS CONTROL 811
3.3.6.2.9.5.10.6.6 PROCESSING 811
3.3.6.2.9.5.10.6.7 UTILIZATION OF OTHER ELEMENTS 813
3.3.6.2.9.5.10.6.8 LIMITATIONS 814
3.3.6.2.9.5.10.7 "-" UNIT DESIGN (CATALOG #P397-0) 814
3.3.6.2.9.5.10.7.1 REQUIREMENTS ALLOCATION 814
3.3.6.2.9.5.10.7.2 LOCAL ENTITIES DESIGN 814
3.3.6.2.9.5.10.7.3 INPUT/OUTPUT 814
3.3.6.2.9.5.10.7.4 LOCAL DATA 815
3.3.6.2.9.5.10.7.5 PROCESS CONTROL 815
3.3.6.2.9.5.10.7.6 PROCESSING 815
3.3.6.2.9.5.10.7.7 UTILIZATION OF OTHER ELEMENTS 817
3.3.6.2.9.5.10.7.8 LIMITATIONS 818
3.3.6.2.9.6 DIAGONAL_MATRIX_OPERATIONS PACKAGE DESIGN (CATALOG #P408-0) 818
3.3.6.2.9.6.1 REQUIREMENTS ALLOCATION 818
3.3.6.2.9.6.2 LOCAL ENTITIES DESIGN 818
3.3.6.2.9.6.3 INPUT/OUTPUT 818
3.3.6.2.9.6.4 LOCAL DATA 819
3.3.6.2.9.6.5 PROCESS CONTROL 820
3.3.6.2.9.6.6 PROCESSING 820
3.3.6.2.9.6.7 UTILIZATION OF OTHER ELEMENTS 822
3.3.6.2.9.6.8 LIMITATIONS 822
3.3.6.2.9.6.9 LLCSC DESIGN 822
3.3.6.2.9.6.10 UNIT DESIGN 822
3.3.6.2.9.6.10.1 IDENTITY_MATRIX UNIT DESIGN (CATALOG #P409-0) 822
3.3.6.2.9.6.10.1.1 REQUIREMENTS ALLOCATION 822
3.3.6.2.9.6.10.1.2 LOCAL ENTITIES DESIGN 822
3.3.6.2.9.6.10.1.3 INPUT/OUTPUT 823
3.3.6.2.9.6.10.1.4 LOCAL DATA 823
3.3.6.2.9.6.10.1.5 PROCESS CONTROL 823
3.3.6.2.9.6.10.1.6 PROCESSING 823
3.3.6.2.9.6.10.1.7 UTILIZATION OF OTHER ELEMENTS 823
3.3.6.2.9.6.10.1.8 LIMITATIONS 824
3.3.6.2.9.6.10.2 ZERO_MATRIX UNIT DESIGN (CATALOG #P410-0) 824

3.3.6.2.9.6.10.2.1 REQUIREMENTS ALLOCATION 824
3.3.6.2.9.6.10.2.2 LOCAL ENTITIES DESIGN 824
3.3.6.2.9.6.10.2.3 INPUT/OUTPUT 824
3.3.6.2.9.6.10.2.4 LOCAL DATA 824
3.3.6.2.9.6.10.2.5 PROCESS CONTROL 824
3.3.6.2.9.6.10.2.6 PROCESSING 825
3.3.6.2.9.6.10.2.7 UTILIZATION OF OTHER ELEMENTS 825
3.3.6.2.9.6.10.2.8 LIMITATIONS 826
3.3.6.2.9.6.10.3 CHANGE_ELEMENT UNIT DESIGN (CATALOG #P411-0) 826
3.3.6.2.9.6.10.3.1 REQUIREMENTS ALLOCATION 826
3.3.6.2.9.6.10.3.2 LOCAL ENTITIES DESIGN 826
3.3.6.2.9.6.10.3.3 INPUT/OUTPUT 826
3.3.6.2.9.6.10.3.4 LOCAL DATA 826
3.3.6.2.9.6.10.3.5 PROCESS CONTROL 826
3.3.6.2.9.6.10.3.6 PROCESSING 827
3.3.6.2.9.6.10.3.7 UTILIZATION OF OTHER ELEMENTS 827
3.3.6.2.9.6.10.3.8 LIMITATIONS 828
3.3.6.2.9.6.10.4 RETRIEVE_ELEMENT UNIT DESIGN (CATALOG #P412-0) 829
3.3.6.2.9.6.10.4.1 REQUIREMENTS ALLOCATION 829
3.3.6.2.9.6.10.4.2 LOCAL ENTITIES DESIGN 829
3.3.6.2.9.6.10.4.3 INPUT/OUTPUT 829
3.3.6.2.9.6.10.4.4 LOCAL DATA 829
3.3.6.2.9.6.10.4.5 PROCESS CONTROL 829
3.3.6.2.9.6.10.4.6 PROCESSING 829
3.3.6.2.9.6.10.4.7 UTILIZATION OF OTHER ELEMENTS 830
3.3.6.2.9.6.10.4.8 LIMITATIONS 831
3.3.6.2.9.6.10.5 ROW_SLICE UNIT DESIGN (CATALOG #P413-0) 831
3.3.6.2.9.6.10.5.1 REQUIREMENTS ALLOCATION 831
3.3.6.2.9.6.10.5.2 LOCAL ENTITIES DESIGN 831
3.3.6.2.9.6.10.5.3 INPUT/OUTPUT 831
3.3.6.2.9.6.10.5.4 LOCAL DATA 831
3.3.6.2.9.6.10.5.5 PROCESS CONTROL 832
3.3.6.2.9.6.10.5.6 PROCESSING 832
3.3.6.2.9.6.10.5.7 UTILIZATION OF OTHER ELEMENTS 832
3.3.6.2.9.6.10.5.8 LIMITATIONS 834
3.3.6.2.9.6.10.6 COLUMN_SLICE UNIT DESIGN (CATALOG #P414-0) 834
3.3.6.2.9.6.10.6.1 REQUIREMENTS ALLOCATION 834
3.3.6.2.9.6.10.6.2 LOCAL ENTITIES DESIGN 834
3.3.6.2.9.6.10.6.3 INPUT/OUTPUT 834
3.3.6.2.9.6.10.6.4 LOCAL DATA 834
3.3.6.2.9.6.10.6.5 PROCESS CONTROL 834
3.3.6.2.9.6.10.6.6 PROCESSING 835
3.3.6.2.9.6.10.6.7 UTILIZATION OF OTHER ELEMENTS 835
3.3.6.2.9.6.10.6.8 LIMITATIONS 836
3.3.6.2.9.6.10.7 ADD_TO_IDENTITY UNIT DESIGN (CATALOG #P415-0) 836
3.3.6.2.9.6.10.7.1 REQUIREMENTS ALLOCATION 836
3.3.6.2.9.6.10.7.2 LOCAL ENTITIES DESIGN 837
3.3.6.2.9.6.10.7.3 INPUT/OUTPUT 837
3.3.6.2.9.6.10.7.4 LOCAL DATA 837
3.3.6.2.9.6.10.7.5 PROCESS CONTROL 837
3.3.6.2.9.6.10.7.6 PROCESSING 837
3.3.6.2.9.6.10.7.7 UTILIZATION OF OTHER ELEMENTS 838
3.3.6.2.9.6.10.7.8 LIMITATIONS 839

3.3.6.2.9.6.10.8 SUBTRACT_FROM_IDENTITY UNIT DESIGN (CATALOG #P416-0) . . . 839
3.3.6.2.9.6.10.8.1 REQUIREMENTS ALLOCATION . . . 839
3.3.6.2.9.6.10.8.2 LOCAL ENTITIES DESIGN . . . 839
3.3.6.2.9.6.10.8.3 INPUT/OUTPUT . . . 839
3.3.6.2.9.6.10.8.4 LOCAL DATA . . . 839
3.3.6.2.9.6.10.8.5 PROCESS CONTROL . . . 839
3.3.6.2.9.6.10.8.6 PROCESSING . . . 840
3.3.6.2.9.6.10.8.7 UTILIZATION OF OTHER ELEMENTS . . . 840
3.3.6.2.9.6.10.8.8 LIMITATIONS . . . 841
3.3.6.2.9.6.10.9 "+" (DIAGONAL_MATRICES + DIAGONAL_MATRICES => DIAGONAL_MATRICES) UNIT DESIGN (CATALOG #P417-0) . . . 841
3.3.6.2.9.6.10.9.1 REQUIREMENTS ALLOCATION . . . 841
3.3.6.2.9.6.10.9.2 LOCAL ENTITIES DESIGN . . . 841
3.3.6.2.9.6.10.9.3 INPUT/OUTPUT . . . 841
3.3.6.2.9.6.10.9.4 LOCAL DATA . . . 842
3.3.6.2.9.6.10.9.5 PROCESS CONTROL . . . 842
3.3.6.2.9.6.10.9.6 PROCESSING . . . 842
3.3.6.2.9.6.10.9.7 UTILIZATION OF OTHER ELEMENTS . . . 842
3.3.6.2.9.6.10.9.8 LIMITATIONS . . . 843
3.3.6.2.9.6.10.10 "-" (DIAGONAL_MATRICES - DIAGONAL_MATRICES => DIAGONAL_MATRICES) UNIT DESIGN (CATALOG #P418-0) . . . 843
3.3.6.2.9.6.10.10.1 REQUIREMENTS ALLOCATION . . . 843
3.3.6.2.9.6.10.10.2 LOCAL ENTITIES DESIGN . . . 844
3.3.6.2.9.6.10.10.3 INPUT/OUTPUT . . . 844
3.3.6.2.9.6.10.10.4 LOCAL DATA . . . 844
3.3.6.2.9.6.10.10.5 PROCESS CONTROL . . . 844
3.3.6.2.9.6.10.10.6 PROCESSING . . . 844
3.3.6.2.9.6.10.10.7 UTILIZATION OF OTHER ELEMENTS . . . 845
3.3.6.2.9.6.10.10.8 LIMITATIONS . . . 846
3.3.6.2.9.7 VECTOR_SCALAR_OPERATIONS_UNCONSTRAINED PACKAGE DESIGN (CATALOG #P419-0) . . . 846
3.3.6.2.9.7.1 REQUIREMENTS ALLOCATION . . . 846
3.3.6.2.9.7.2 LOCAL ENTITIES DESIGN . . . 846
3.3.6.2.9.7.3 INPUT/OUTPUT . . . 846
3.3.6.2.9.7.4 LOCAL DATA . . . 847
3.3.6.2.9.7.5 PROCESS CONTROL . . . 847
3.3.6.2.9.7.6 PROCESSING . . . 847
3.3.6.2.9.7.7 UTILIZATION OF OTHER ELEMENTS . . . 847
3.3.6.2.9.7.8 LIMITATIONS . . . 848
3.3.6.2.9.7.9 LLCSC DESIGN . . . 848
3.3.6.2.9.7.10 UNIT DESIGN . . . 848
3.3.6.2.9.7.10.1 "*" UNIT DESIGN (CATALOG #P420-0) . . . 848
3.3.6.2.9.7.10.1.1 REQUIREMENTS ALLOCATION . . . 848
3.3.6.2.9.7.10.1.2 LOCAL ENTITIES DESIGN . . . 848
3.3.6.2.9.7.10.1.3 INPUT/OUTPUT . . . 848
3.3.6.2.9.7.10.1.4 LOCAL DATA . . . 848
3.3.6.2.9.7.10.1.5 PROCESS CONTROL . . . 849
3.3.6.2.9.7.10.1.6 PROCESSING . . . 849
3.3.6.2.9.7.10.1.7 UTILIZATION OF OTHER ELEMENTS . . . 849
3.3.6.2.9.7.10.1.8 LIMITATIONS . . . 850
3.3.6.2.9.7.10.2 "/" UNIT DESIGN (CATALOG #P421-0) . . . 850
3.3.6.2.9.7.10.2.1 REQUIREMENTS ALLOCATION . . . 850
3.3.6.2.9.7.10.2.2 LOCAL ENTITIES DESIGN . . . 850
3.3.6.2.9.7.10.2.3 INPUT/OUTPUT . . . 851

3.3.6.2.9.7.10.2.4 LOCAL DATA 851
3.3.6.2.9.7.10.2.5 PROCESS CONTROL 851
3.3.6.2.9.7.10.2.6 PROCESSING 851
3.3.6.2.9.7.10.2.7 UTILIZATION OF OTHER ELEMENTS 852
3.3.6.2.9.7.10.2.8 LIMITATIONS 853
3.3.6.2.9.8 MATRIX_SCALAR_OPERATIONS_UNCONSTRAINED PACKAGE DESIGN (CATALOG #P425-0) 853
3.3.6.2.9.8.1 REQUIREMENTS ALLOCATION 853
3.3.6.2.9.8.2 LOCAL ENTITIES DESIGN 853
3.3.6.2.9.8.3 INPUT/OUTPUT 853
3.3.6.2.9.8.4 LOCAL DATA 854
3.3.6.2.9.8.5 PROCESS CONTROL 854
3.3.6.2.9.8.6 PROCESSING 854
3.3.6.2.9.8.7 UTILIZATION OF OTHER ELEMENTS 855
3.3.6.2.9.8.8 LIMITATIONS 855
3.3.6.2.9.8.9 LLCSC DESIGN 855
3.3.6.2.9.8.10 UNIT DESIGN 855
3.3.6.2.9.8.10.1 "*" UNIT DESIGN (CATALOG #P426-0) 855
3.3.6.2.9.8.10.1.1 REQUIREMENTS ALLOCATION 855
3.3.6.2.9.8.10.1.2 LOCAL ENTITIES DESIGN 855
3.3.6.2.9.8.10.1.3 INPUT/OUTPUT 855
3.3.6.2.9.8.10.1.4 LOCAL DATA 855
3.3.6.2.9.8.10.1.5 PROCESS CONTROL 856
3.3.6.2.9.8.10.1.6 PROCESSING 856
3.3.6.2.9.8.10.1.7 UTILIZATION OF OTHER ELEMENTS 857
3.3.6.2.9.8.10.1.8 LIMITATIONS 858
3.3.6.2.9.8.10.2 "/" UNIT DESIGN (CATALOG #P427-0) 858
3.3.6.2.9.8.10.2.1 REQUIREMENTS ALLOCATION 858
3.3.6.2.9.8.10.2.2 LOCAL ENTITIES DESIGN 858
3.3.6.2.9.8.10.2.3 INPUT/OUTPUT 858
3.3.6.2.9.8.10.2.4 LOCAL DATA 858
3.3.6.2.9.8.10.2.5 PROCESS CONTROL 859
3.3.6.2.9.8.10.2.6 PROCESSING 859
3.3.6.2.9.8.10.2.7 UTILIZATION OF OTHER ELEMENTS 860
3.3.6.2.9.8.10.2.8 LIMITATIONS 861
3.3.6.2.9.9 DIAGONAL_MATRIX_SCALAR_OPERATIONS PACKAGE DESIGN (CATALOG #P431-0) 861
3.3.6.2.9.9.1 REQUIREMENTS ALLOCATION 861
3.3.6.2.9.9.2 LOCAL ENTITIES DESIGN 861
3.3.6.2.9.9.3 INPUT/OUTPUT 861
3.3.6.2.9.9.4 LOCAL DATA 862
3.3.6.2.9.9.5 PROCESS CONTROL 862
3.3.6.2.9.9.6 PROCESSING 862
3.3.6.2.9.9.7 UTILIZATION OF OTHER ELEMENTS 863
3.3.6.2.9.9.8 LIMITATIONS 863
3.3.6.2.9.9.9 LLCSC DESIGN 863
3.3.6.2.9.9.10 UNIT DESIGN 863
3.3.6.2.9.9.10.1 "*" UNIT DESIGN (CATALOG #P432-0) 863
3.3.6.2.9.9.10.1.1 REQUIREMENTS ALLOCATION 863
3.3.6.2.9.9.10.1.2 LOCAL ENTITIES DESIGN 864
3.3.6.2.9.9.10.1.3 INPUT/OUTPUT 864
3.3.6.2.9.9.10.1.4 LOCAL DATA 864
3.3.6.2.9.9.10.1.5 PROCESS CONTROL 864
3.3.6.2.9.9.10.1.6 PROCESSING 864
3.3.6.2.9.9.10.1.7 UTILIZATION OF OTHER ELEMENTS 865
3.3.6.2.9.9.10.1.8 LIMITATIONS 865
3.3.6.2.9.9.10.2 "/" UNIT DESIGN (CATALOG #P433-0) 866

3.3.6.2.9.9.10.2.1 REQUIREMENTS ALLOCATION 866
3.3.6.2.9.9.10.2.2 LOCAL ENTITIES DESIGN 866
3.3.6.2.9.9.10.2.3 INPUT/OUTPUT 866
3.3.6.2.9.9.10.2.4 LOCAL DATA 866
3.3.6.2.9.9.10.2.5 PROCESS CONTROL 866
3.3.6.2.9.9.10.2.6 PROCESSING 866
3.3.6.2.9.9.10.2.7 UTILIZATION OF OTHER ELEMENTS 867
3.3.6.2.9.9.10.2.8 LIMITATIONS 868
3.3.6.2.9.10 MATRIX_MATRIX_MULTIPLY_UNRESTRICTED PACKAGE DESIGN (CATALOG #P439-0) 868
3.3.6.2.9.10.1 REQUIREMENTS ALLOCATION 868
3.3.6.2.9.10.2 LOCAL ENTITIES DESIGN 868
3.3.6.2.9.10.3 INPUT/OUTPUT 869
3.3.6.2.9.10.4 LOCAL DATA 870
3.3.6.2.9.10.5 PROCESS CONTROL 870
3.3.6.2.9.10.6 PROCESSING 870
3.3.6.2.9.10.7 UTILIZATION OF OTHER ELEMENTS 870
3.3.6.2.9.10.8 LIMITATIONS 871
3.3.6.2.9.10.9 LLCSC DESIGN 871
3.3.6.2.9.10.10 UNIT DESIGN 871
3.3.6.2.9.10.10.1 "*" UNIT DESIGN (CATALOG #P440-0) 871
3.3.6.2.9.10.10.1.1 REQUIREMENTS ALLOCATION 871
3.3.6.2.9.10.10.1.2 LOCAL ENTITIES DESIGN 871
3.3.6.2.9.10.10.1.3 INPUT/OUTPUT 871
3.3.6.2.9.10.10.1.4 LOCAL DATA 872
3.3.6.2.9.10.10.1.5 PROCESS CONTROL 872
3.3.6.2.9.10.10.1.6 PROCESSING 872
3.3.6.2.9.10.10.1.7 UTILIZATION OF OTHER ELEMENTS 873
3.3.6.2.9.10.10.1.8 LIMITATIONS 874
3.3.6.2.9.11 MATRIX_VECTOR_MULTIPLY_UNRESTRICTED PACKAGE DESIGN (CATALOG #P434-0) 875
3.3.6.2.9.11.1 REQUIREMENTS ALLOCATION 875
3.3.6.2.9.11.2 LOCAL ENTITIES DESIGN 875
3.3.6.2.9.11.3 INPUT/OUTPUT 875
3.3.6.2.9.11.4 LOCAL DATA 876
3.3.6.2.9.11.5 PROCESS CONTROL 876
3.3.6.2.9.11.6 PROCESSING 877
3.3.6.2.9.11.7 UTILIZATION OF OTHER ELEMENTS 877
3.3.6.2.9.11.8 LIMITATIONS 877
3.3.6.2.9.11.9 LLCSC DESIGN 878
3.3.6.2.9.11.10 UNIT DESIGN 878
3.3.6.2.9.11.10.1 "*" UNIT DESIGN (CATALOG #P435-0) 878
3.3.6.2.9.11.10.1.1 REQUIREMENTS ALLOCATION 878
3.3.6.2.9.11.10.1.2 LOCAL ENTITIES DESIGN 878
3.3.6.2.9.11.10.1.3 INPUT/OUTPUT 878
3.3.6.2.9.11.10.1.4 LOCAL DATA 878
3.3.6.2.9.11.10.1.5 PROCESS CONTROL 879
3.3.6.2.9.11.10.1.6 PROCESSING 879
3.3.6.2.9.11.10.1.7 UTILIZATION OF OTHER ELEMENTS 880
3.3.6.2.9.11.10.1.8 LIMITATIONS 881
3.3.6.2.9.12 VECTOR_VECTOR_TRANSPOSE_MULTIPLY_UNRESTRICTED PACKAGE DESIGN (CATALOG #P442-0) 881
3.3.6.2.9.12.1 REQUIREMENTS ALLOCATION 881
3.3.6.2.9.12.2 LOCAL ENTITIES DESIGN 881
3.3.6.2.9.12.3 INPUT/OUTPUT 881
3.3.6.2.9.12.4 LOCAL DATA 882
3.3.6.2.9.12.5 PROCESS CONTROL 882

3.3.6.2.9.12.6 PROCESSING 883
3.3.6.2.9.12.7 UTILIZATION OF OTHER ELEMENTS 883
3.3.6.2.9.12.8 LIMITATIONS 883
3.3.6.2.9.12.9 LLCSC DESIGN 884
3.3.6.2.9.12.10 UNIT DESIGN 884
3.3.6.2.9.12.10.1 "*" UNIT DESIGN (CATALOG #P443-0) 884
3.3.6.2.9.12.10.1.1 REQUIREMENTS ALLOCATION 884
3.3.6.2.9.12.10.1.2 LOCAL ENTITIES DESIGN 884
3.3.6.2.9.12.10.1.3 INPUT/OUTPUT 884
3.3.6.2.9.12.10.1.4 LOCAL DATA 884
3.3.6.2.9.12.10.1.5 PROCESS CONTROL 885
3.3.6.2.9.12.10.1.6 PROCESSING 885
3.3.6.2.9.12.10.1.7 UTILIZATION OF OTHER ELEMENTS 886
3.3.6.2.9.12.10.1.8 LIMITATIONS 887
3.3.6.2.9.13 MATRIX_MATRIX_TRANSPOSE_MULTIPLY_UNRESTRICTED PACKAGE DESIGN (CATALOG #P445-0) 887
3.3.6.2.9.13.1 REQUIREMENTS ALLOCATION 887
3.3.6.2.9.13.2 LOCAL ENTITIES DESIGN 887
3.3.6.2.9.13.3 INPUT/OUTPUT 887
3.3.6.2.9.13.4 LOCAL DATA 888
3.3.6.2.9.13.5 PROCESS CONTROL 888
3.3.6.2.9.13.6 PROCESSING 889
3.3.6.2.9.13.7 UTILIZATION OF OTHER ELEMENTS 889
3.3.6.2.9.13.8 LIMITATIONS 889
3.3.6.2.9.13.9 LLCSC DESIGN 890
3.3.6.2.9.13.10 UNIT DESIGN 890
3.3.6.2.9.13.10.1 "*" UNIT DESIGN (CATALOG #P446-0) 890
3.3.6.2.9.13.10.1.1 REQUIREMENTS ALLOCATION 890
3.3.6.2.9.13.10.1.2 LOCAL ENTITIES DESIGN 890
3.3.6.2.9.13.10.1.3 INPUT/OUTPUT 890
3.3.6.2.9.13.10.1.4 LOCAL DATA 890
3.3.6.2.9.13.10.1.5 PROCESS CONTROL 891
3.3.6.2.9.13.10.1.6 PROCESSING 891
3.3.6.2.9.13.10.1.7 UTILIZATION OF OTHER ELEMENTS 892
3.3.6.2.9.13.10.1.8 LIMITATIONS 893
3.3.6.2.9.14 DOT_PRODUCT_OPERATIONS_UNRESTRICTED PACKAGE DESIGN (CATALOG #P448-0) 893
3.3.6.2.9.14.1 REQUIREMENTS ALLOCATION 894
3.3.6.2.9.14.2 LOCAL ENTITIES DESIGN 894
3.3.6.2.9.14.3 INPUT/OUTPUT 894
3.3.6.2.9.14.4 LOCAL DATA 895
3.3.6.2.9.14.5 PROCESS CONTROL 895
3.3.6.2.9.14.6 PROCESSING 895
3.3.6.2.9.14.7 UTILIZATION OF OTHER ELEMENTS 895
3.3.6.2.9.14.8 LIMITATIONS 896
3.3.6.2.9.14.9 LLCSC DESIGN 896
3.3.6.2.9.14.10 UNIT DESIGN 896
3.3.6.2.9.14.10.1 DOT_PRODUCT UNIT DESIGN (CATALOG #P449-0) 896
3.3.6.2.9.14.10.1.1 REQUIREMENTS ALLOCATION 896
3.3.6.2.9.14.10.1.2 LOCAL ENTITIES DESIGN 896
3.3.6.2.9.14.10.1.3 INPUT/OUTPUT 896
3.3.6.2.9.14.10.1.4 LOCAL DATA 896
3.3.6.2.9.14.10.1.5 PROCESS CONTROL 897
3.3.6.2.9.14.10.1.6 PROCESSING 897
3.3.6.2.9.14.10.1.7 UTILIZATION OF OTHER ELEMENTS 898
3.3.6.2.9.14.10.1.8 LIMITATIONS 898

3.3.6.2.9.15 DIAGONAL_FULL_MATRIX_ADD_UNRESTRICTED PACKAGE DESIGN (CATALOG #P451-0) 898
3.3.6.2.9.15.1 REQUIREMENTS ALLOCATION 898
3.3.6.2.9.15.2 LOCAL ENTITIES DESIGN 898
3.3.6.2.9.15.3 INPUT/OUTPUT 899
3.3.6.2.9.15.4 LOCAL DATA 899
3.3.6.2.9.15.5 PROCESS CONTROL 899
3.3.6.2.9.15.6 PROCESSING 899
3.3.6.2.9.15.7 UTILIZATION OF OTHER ELEMENTS 900
3.3.6.2.9.15.8 LIMITATIONS 900
3.3.6.2.9.15.9 LLCSC DESIGN 900
3.3.6.2.9.15.10 UNIT DESIGN 901
3.3.6.2.9.15.10.1 "+" UNIT DESIGN (CATALOG #P452-0) 901
3.3.6.2.9.15.10.1.1 REQUIREMENTS ALLOCATION 901
3.3.6.2.9.15.10.1.2 LOCAL ENTITIES DESIGN 901
3.3.6.2.9.15.10.1.3 INPUT/OUTPUT 901
3.3.6.2.9.15.10.1.4 LOCAL DATA 901
3.3.6.2.9.15.10.1.5 PROCESS CONTROL 902
3.3.6.2.9.15.10.1.6 PROCESSING 902
3.3.6.2.9.15.10.1.7 UTILIZATION OF OTHER ELEMENTS 903
3.3.6.2.9.15.10.1.8 LIMITATIONS 904
3.3.6.2.9.16 VECTOR_OPERATIONS_CONSTRAINED PACKAGE DESIGN (CATALOG #P342-0) 904
3.3.6.2.9.16.1 REQUIREMENTS ALLOCATION 904
3.3.6.2.9.16.2 LOCAL ENTITIES DESIGN 905
3.3.6.2.9.16.3 INPUT/OUTPUT 905
3.3.6.2.9.16.4 LOCAL DATA 905
3.3.6.2.9.16.5 PROCESS CONTROL 906
3.3.6.2.9.16.6 PROCESSING 906
3.3.6.2.9.16.7 UTILIZATION OF OTHER ELEMENTS 906
3.3.6.2.9.16.8 LIMITATIONS 906
3.3.6.2.9.16.9 LLCSC DESIGN 906
3.3.6.2.9.16.10 UNIT DESIGN 906
3.3.6.2.9.16.10.1 "+" (VECTOR + VECTORS := VECTORS) UNIT DESIGN (CATALOG #P343-0) 906
3.3.6.2.9.16.10.1.1 REQUIREMENTS ALLOCATION 906
3.3.6.2.9.16.10.1.2 LOCAL ENTITIES DESIGN 906
3.3.6.2.9.16.10.1.3 INPUT/OUTPUT 906
3.3.6.2.9.16.10.1.4 LOCAL DATA 907
3.3.6.2.9.16.10.1.5 PROCESS CONTROL 907
3.3.6.2.9.16.10.1.6 PROCESSING 907
3.3.6.2.9.16.10.1.7 UTILIZATION OF OTHER ELEMENTS 908
3.3.6.2.9.16.10.1.8 LIMITATIONS 908
3.3.6.2.9.16.10.2 "-" (VECTORS - VECTORS := VECTORS) UNIT DESIGN (CATALOG #P344-0) 908
3.3.6.2.9.16.10.2.1 REQUIREMENTS ALLOCATION 908
3.3.6.2.9.16.10.2.2 LOCAL ENTITIES DESIGN 908
3.3.6.2.9.16.10.2.3 INPUT/OUTPUT 909
3.3.6.2.9.16.10.2.4 LOCAL DATA 909
3.3.6.2.9.16.10.2.5 PROCESS CONTROL 909
3.3.6.2.9.16.10.2.6 PROCESSING 909
3.3.6.2.9.16.10.2.7 UTILIZATION OF OTHER ELEMENTS 910
3.3.6.2.9.16.10.2.8 LIMITATIONS 910
3.3.6.2.9.16.10.3 VECTOR_LENGTH UNIT DESIGN (CATALOG #P345-0) 910
3.3.6.2.9.16.10.3.1 REQUIREMENTS ALLOCATION 910
3.3.6.2.9.16.10.3.2 LOCAL ENTITIES DESIGN 911

3.3.6.2.9.16.10.3.3 INPUT/OUTPUT 911
3.3.6.2.9.16.10.3.4 LOCAL DATA 911
3.3.6.2.9.16.10.3.5 PROCESS CONTROL 911
3.3.6.2.9.16.10.3.6 PROCESSING 911
3.3.6.2.9.16.10.3.7 UTILIZATION OF OTHER ELEMENTS 912
3.3.6.2.9.16.10.3.8 LIMITATIONS 913
3.3.6.2.9.16.10.4 DOT PRODUCT UNIT DESIGN (CATALOG #P346-0) 913
3.3.6.2.9.16.10.4.1 REQUIREMENTS ALLOCATION 913
3.3.6.2.9.16.10.4.2 LOCAL ENTITIES DESIGN 913
3.3.6.2.9.16.10.4.3 INPUT/OUTPUT 913
3.3.6.2.9.16.10.4.4 LOCAL DATA 914
3.3.6.2.9.16.10.4.5 PROCESS CONTROL 914
3.3.6.2.9.16.10.4.6 PROCESSING 914
3.3.6.2.9.16.10.4.7 UTILIZATION OF OTHER ELEMENTS 914
3.3.6.2.9.16.10.4.8 LIMITATIONS 915
3.3.6.2.9.17 MATRIX_OPERATIONS_CONSTRAINED PACKAGE DESIGN (CATALOG #P355-0) 915
3.3.6.2.9.17.1 REQUIREMENTS ALLOCATION 916
3.3.6.2.9.17.2 LOCAL ENTITIES DESIGN 916
3.3.6.2.9.17.3 INPUT/OUTPUT 916
3.3.6.2.9.17.4 LOCAL DATA 916
3.3.6.2.9.17.5 PROCESS CONTROL 917
3.3.6.2.9.17.6 PROCESSING 917
3.3.6.2.9.17.7 UTILIZATION OF OTHER ELEMENTS 917
3.3.6.2.9.17.8 LIMITATIONS 917
3.3.6.2.9.17.9 LLCSC DESIGN 917
3.3.6.2.9.17.10 UNIT DESIGN 917
3.3.6.2.9.17.10.1 "+" (MATRICES + MATRICES := MATRICES) UNIT DESIGN (CATALOG #P356-0) 917
3.3.6.2.9.17.10.1.1 REQUIREMENTS ALLOCATION 917
3.3.6.2.9.17.10.1.2 LOCAL ENTITIES DESIGN 917
3.3.6.2.9.17.10.1.3 INPUT/OUTPUT 917
3.3.6.2.9.17.10.1.4 LOCAL DATA 918
3.3.6.2.9.17.10.1.5 PROCESS CONTROL 918
3.3.6.2.9.17.10.1.6 PROCESSING 918
3.3.6.2.9.17.10.1.7 UTILIZATION OF OTHER ELEMENTS 919
3.3.6.2.9.17.10.1.8 LIMITATIONS 919
3.3.6.2.9.17.10.2 "-" (MATRICES - MATRICES := MATRICES) UNIT DESIGN (CATALOG #P357-0) 919
3.3.6.2.9.17.10.2.1 REQUIREMENTS ALLOCATION 919
3.3.6.2.9.17.10.2.2 LOCAL ENTITIES DESIGN 920
3.3.6.2.9.17.10.2.3 INPUT/OUTPUT 920
3.3.6.2.9.17.10.2.4 LOCAL DATA 920
3.3.6.2.9.17.10.2.5 PROCESS CONTROL 920
3.3.6.2.9.17.10.2.6 PROCESSING 920
3.3.6.2.9.17.10.2.7 UTILIZATION OF OTHER ELEMENTS 921
3.3.6.2.9.17.10.2.8 LIMITATIONS 921
3.3.6.2.9.17.10.3 "+" (MATRICES + ELEMENTS := MATRICES) UNIT DESIGN (CATALOG #P358-0) 922
3.3.6.2.9.17.10.3.1 REQUIREMENTS ALLOCATION 922
3.3.6.2.9.17.10.3.2 LOCAL ENTITIES DESIGN 922
3.3.6.2.9.17.10.3.3 INPUT/OUTPUT 922
3.3.6.2.9.17.10.3.4 LOCAL DATA 922
3.3.6.2.9.17.10.3.5 PROCESS CONTROL 922
3.3.6.2.9.17.10.3.6 PROCESSING 922
3.3.6.2.9.17.10.3.7 UTILIZATION OF OTHER ELEMENTS 923
3.3.6.2.9.17.10.3.8 LIMITATIONS 924

3.3.6.2.9.17.10.4 "-" (MATRICES - ELEMENTS := MATRICES) UNIT DESIGN (CATALOG #P359-0) 924
3.3.6.2.9.17.10.4.1 REQUIREMENTS ALLOCATION 924
3.3.6.2.9.17.10.4.2 LOCAL ENTITIES DESIGN 924
3.3.6.2.9.17.10.4.3 INPUT/OUTPUT 924
3.3.6.2.9.17.10.4.4 LOCAL DATA 924
3.3.6.2.9.17.10.4.5 PROCESS CONTROL 924
3.3.6.2.9.17.10.4.6 PROCESSING 925
3.3.6.2.9.17.10.4.7 UTILIZATION OF OTHER ELEMENTS 925
3.3.6.2.9.17.10.4.8 LIMITATIONS 926
3.3.6.2.9.17.10.5 SET TO IDENTITY MATRIX UNIT DESIGN (CATALOG #P360-0) 926
3.3.6.2.9.17.10.5.1 REQUIREMENTS ALLOCATION 926
3.3.6.2.9.17.10.5.2 LOCAL ENTITIES DESIGN 926
3.3.6.2.9.17.10.5.3 INPUT/OUTPUT 926
3.3.6.2.9.17.10.5.4 LOCAL DATA 926
3.3.6.2.9.17.10.5.5 PROCESS CONTROL 927
3.3.6.2.9.17.10.5.6 PROCESSING 927
3.3.6.2.9.17.10.5.7 UTILIZATION OF OTHER ELEMENTS 928
3.3.6.2.9.17.10.5.8 LIMITATIONS 928
3.3.6.2.9.17.10.6 SET TO ZERO MATRIX UNIT DESIGN (CATALOG #P361-0) 929
3.3.6.2.9.17.10.6.1 REQUIREMENTS ALLOCATION 929
3.3.6.2.9.17.10.6.2 LOCAL ENTITIES DESIGN 929
3.3.6.2.9.17.10.6.3 INPUT/OUTPUT 929
3.3.6.2.9.17.10.6.4 LOCAL DATA 929
3.3.6.2.9.17.10.6.5 PROCESS CONTROL 929
3.3.6.2.9.17.10.6.6 PROCESSING 929
3.3.6.2.9.17.10.6.7 UTILIZATION OF OTHER ELEMENTS 930
3.3.6.2.9.17.10.6.8 LIMITATIONS 930
3.3.6.2.9.18 DYNAMICALLY_SPARSE_MATRIX_OPERATIONS_CONSTRAINED PACKAGE DESIGN (CATALOG #P369-0) 930
3.3.6.2.9.18.1 REQUIREMENTS ALLOCATION 930
3.3.6.2.9.18.2 LOCAL ENTITIES DESIGN 931
3.3.6.2.9.18.3 INPUT/OUTPUT 931
3.3.6.2.9.18.4 LOCAL DATA 931
3.3.6.2.9.18.5 PROCESS CONTROL 931
3.3.6.2.9.18.6 PROCESSING 931
3.3.6.2.9.18.7 UTILIZATION OF OTHER ELEMENTS 932
3.3.6.2.9.18.8 LIMITATIONS 932
3.3.6.2.9.18.9 LLCSC DESIGN 932
3.3.6.2.9.18.10 UNIT DESIGN 932
3.3.6.2.9.18.10.1 SET_TO_IDENTITY_MATRIX UNIT DESIGN (CATALOG #P370-0) 932
3.3.6.2.9.18.10.1.1 REQUIREMENTS ALLOCATION 932
3.3.6.2.9.18.10.1.2 LOCAL ENTITIES DESIGN 932
3.3.6.2.9.18.10.1.3 INPUT/OUTPUT 932
3.3.6.2.9.18.10.1.4 LOCAL DATA 932
3.3.6.2.9.18.10.1.5 PROCESS CONTROL 933
3.3.6.2.9.18.10.1.6 PROCESSING 933
3.3.6.2.9.18.10.1.7 UTILIZATION OF OTHER ELEMENTS 934
3.3.6.2.9.18.10.1.8 LIMITATIONS 934
3.3.6.2.9.18.10.2 SET_TO_ZERO_MATRIX UNIT DESIGN (CATALOG #P371-0) 935
3.3.6.2.9.18.10.2.1 REQUIREMENTS ALLOCATION 935
3.3.6.2.9.18.10.2.2 LOCAL ENTITIES DESIGN 935
3.3.6.2.9.18.10.2.3 INPUT/OUTPUT 935

3.3.6.2.9.18.10.2.4 LOCAL DATA 935
3.3.6.2.9.18.10.2.5 PROCESS CONTROL 935
3.3.6.2.9.18.10.2.6 PROCESSING 935
3.3.6.2.9.18.10.2.7 UTILIZATION OF OTHER ELEMENTS 936
3.3.6.2.9.18.10.2.8 LIMITATIONS 936
3.3.6.2.9.18.10.3 ADD_TO_IDENTITY UNIT DESIGN (CATALOG #P372-0) 936
3.3.6.2.9.18.10.3.1 REQUIREMENTS ALLOCATION 936
3.3.6.2.9.18.10.3.2 LOCAL ENTITIES DESIGN 936
3.3.6.2.9.18.10.3.3 INPUT/OUTPUT 937
3.3.6.2.9.18.10.3.4 LOCAL DATA 937
3.3.6.2.9.18.10.3.5 PROCESS CONTROL 937
3.3.6.2.9.18.10.3.6 PROCESSING 937
3.3.6.2.9.18.10.3.7 UTILIZATION OF OTHER ELEMENTS 938
3.3.6.2.9.18.10.3.8 LIMITATIONS 939
3.3.6.2.9.18.10.4 SUBTRACT_FROM_IDENTITY UNIT DESIGN (CATALOG #P373-0) 939
3.3.6.2.9.18.10.4.1 REQUIREMENTS ALLOCATION 939
3.3.6.2.9.18.10.4.2 LOCAL ENTITIES DESIGN 939
3.3.6.2.9.18.10.4.3 INPUT/OUTPUT 939
3.3.6.2.9.18.10.4.4 LOCAL DATA 940
3.3.6.2.9.18.10.4.5 PROCESS CONTROL 940
3.3.6.2.9.18.10.4.6 PROCESSING 940
3.3.6.2.9.18.10.4.7 UTILIZATION OF OTHER ELEMENTS 941
3.3.6.2.9.18.10.4.8 LIMITATIONS 942
3.3.6.2.9.18.10.5 "+" UNIT DESIGN (CATALOG #P374-0) 942
3.3.6.2.9.18.10.5.1 REQUIREMENTS ALLOCATION 942
3.3.6.2.9.18.10.5.2 LOCAL ENTITIES DESIGN 942
3.3.6.2.9.18.10.5.3 INPUT/OUTPUT 942
3.3.6.2.9.18.10.5.4 LOCAL DATA 943
3.3.6.2.9.18.10.5.5 PROCESS CONTROL 943
3.3.6.2.9.18.10.5.6 PROCESSING 943
3.3.6.2.9.18.10.5.7 UTILIZATION OF OTHER ELEMENTS 944
3.3.6.2.9.18.10.5.8 LIMITATIONS 945
3.3.6.2.9.18.10.6 "-" UNIT DESIGN (CATALOG #P375-0) 945
3.3.6.2.9.18.10.6.1 REQUIREMENTS ALLOCATION 945
3.3.6.2.9.18.10.6.2 LOCAL ENTITIES DESIGN 945
3.3.6.2.9.18.10.6.3 INPUT/OUTPUT 945
3.3.6.2.9.18.10.6.4 LOCAL DATA 945
3.3.6.2.9.18.10.6.5 PROCESS CONTROL 946
3.3.6.2.9.18.10.6.6 PROCESSING 946
3.3.6.2.9.18.10.6.7 UTILIZATION OF OTHER ELEMENTS 946
3.3.6.2.9.18.10.6.8 LIMITATIONS 947
3.3.6.2.9.19 SYMMETRIC_FULL_STORAGE_MATRIX_OPERATIONS_CONSTRAINED PACKAGE DESIGN (CATALOG #P398-0) 947
3.3.6.2.9.19.1 REQUIREMENTS ALLOCATION 948
3.3.6.2.9.19.2 LOCAL ENTITIES DESIGN 948
3.3.6.2.9.19.3 INPUT/OUTPUT 948
3.3.6.2.9.19.4 LOCAL DATA 948
3.3.6.2.9.19.5 PROCESS CONTROL 948
3.3.6.2.9.19.6 PROCESSING 949
3.3.6.2.9.19.7 UTILIZATION OF OTHER ELEMENTS 949
3.3.6.2.9.19.8 LIMITATIONS 949
3.3.6.2.9.19.9 LLCSC DESIGN 949
3.3.6.2.9.19.10 UNIT DESIGN 950
3.3.6.2.9.19.10.1 CHANGE_ELEMENT UNIT DESIGN (CATALOG #P399-0) 950

3.3.6.2.9.19.10.1.1 REQUIREMENTS ALLOCATION 950
3.3.6.2.9.19.10.1.2 LOCAL ENTITIES DESIGN 950
3.3.6.2.9.19.10.1.3 INPUT/OUTPUT 950
3.3.6.2.9.19.10.1.4 LOCAL DATA 950
3.3.6.2.9.19.10.1.5 PROCESS CONTROL 950
3.3.6.2.9.19.10.1.6 PROCESSING 950
3.3.6.2.9.19.10.1.7 UTILIZATION OF OTHER ELEMENTS 951
3.3.6.2.9.19.10.1.8 LIMITATIONS 952
3.3.6.2.9.19.10.2 SET TO IDENTITY MATRIX UNIT DESIGN (CATALOG #P400-0) 952
3.3.6.2.9.19.10.2.1 REQUIREMENTS ALLOCATION 952
3.3.6.2.9.19.10.2.2 LOCAL ENTITIES DESIGN 952
3.3.6.2.9.19.10.2.3 INPUT/OUTPUT 952
3.3.6.2.9.19.10.2.4 LOCAL DATA 952
3.3.6.2.9.19.10.2.5 PROCESS CONTROL 952
3.3.6.2.9.19.10.2.6 PROCESSING 953
3.3.6.2.9.19.10.2.7 UTILIZATION OF OTHER ELEMENTS 953
3.3.6.2.9.19.10.2.8 LIMITATIONS 954
3.3.6.2.9.19.10.3 SET TO ZERO MATRIX UNIT DESIGN (CATALOG #P401-0) 954
3.3.6.2.9.19.10.3.1 REQUIREMENTS ALLOCATION 954
3.3.6.2.9.19.10.3.2 LOCAL ENTITIES DESIGN 954
3.3.6.2.9.19.10.3.3 INPUT/OUTPUT 954
3.3.6.2.9.19.10.3.4 LOCAL DATA 955
3.3.6.2.9.19.10.3.5 PROCESS CONTROL 955
3.3.6.2.9.19.10.3.6 PROCESSING 955
3.3.6.2.9.19.10.3.7 UTILIZATION OF OTHER ELEMENTS 955
3.3.6.2.9.19.10.3.8 LIMITATIONS 956
3.3.6.2.9.19.10.4 ADD TO IDENTITY UNIT DESIGN (CATALOG #P402-0) 956
3.3.6.2.9.19.10.4.1 REQUIREMENTS ALLOCATION 956
3.3.6.2.9.19.10.4.2 LOCAL ENTITIES DESIGN 956
3.3.6.2.9.19.10.4.3 INPUT/OUTPUT 956
3.3.6.2.9.19.10.4.4 LOCAL DATA 956
3.3.6.2.9.19.10.4.5 PROCESS CONTROL 956
3.3.6.2.9.19.10.4.6 PROCESSING 957
3.3.6.2.9.19.10.4.7 UTILIZATION OF OTHER ELEMENTS 957
3.3.6.2.9.19.10.4.8 LIMITATIONS 958
3.3.6.2.9.19.10.5 SUBTRACT FROM IDENTITY UNIT DESIGN (CATALOG #P403-0) 958
3.3.6.2.9.19.10.5.1 REQUIREMENTS ALLOCATION 958
3.3.6.2.9.19.10.5.2 LOCAL ENTITIES DESIGN 958
3.3.6.2.9.19.10.5.3 INPUT/OUTPUT 958
3.3.6.2.9.19.10.5.4 LOCAL DATA 959
3.3.6.2.9.19.10.5.5 PROCESS CONTROL 959
3.3.6.2.9.19.10.5.6 PROCESSING 959
3.3.6.2.9.19.10.5.7 UTILIZATION OF OTHER ELEMENTS 960
3.3.6.2.9.19.10.5.8 LIMITATIONS 961
3.3.6.2.9.19.10.6 "+" UNIT DESIGN (CATALOG #P404-0) 961
3.3.6.2.9.19.10.6.1 REQUIREMENTS ALLOCATION 961
3.3.6.2.9.19.10.6.2 LOCAL ENTITIES DESIGN 961
3.3.6.2.9.19.10.6.3 INPUT/OUTPUT 961
3.3.6.2.9.19.10.6.4 LOCAL DATA 962
3.3.6.2.9.19.10.6.5 PROCESS CONTROL 962
3.3.6.2.9.19.10.6.6 PROCESSING 962
3.3.6.2.9.19.10.6.7 UTILIZATION OF OTHER ELEMENTS 963
3.3.6.2.9.19.10.6.8 LIMITATIONS 964

3.3.6.2.9.19.10.7 "-" UNIT DESIGN (CATALOG #P407-0) 964
3.3.6.2.9.19.10.7.1 REQUIREMENTS ALLOCATION 964
3.3.6.2.9.19.10.7.2 LOCAL ENTITIES DESIGN 964
3.3.6.2.9.19.10.7.3 INPUT/OUTPUT 964
3.3.6.2.9.19.10.7.4 LOCAL DATA 964
3.3.6.2.9.19.10.7.5 PROCESS CONTROL 965
3.3.6.2.9.19.10.7.6 PROCESSING 965
3.3.6.2.9.19.10.7.7 UTILIZATION OF OTHER ELEMENTS 966
3.3.6.2.9.19.10.7.8 LIMITATIONS 967
3.3.6.2.9.20 VECTOR_SCALAR_OPERATIONS_CONSTRAINED PACKAGE DESIGN (CATALOG #P422-0) 967
3.3.6.2.9.20.1 REQUIREMENTS ALLOCATION 967
3.3.6.2.9.20.2 LOCAL ENTITIES DESIGN 967
3.3.6.2.9.20.3 INPUT/OUTPUT 967
3.3.6.2.9.20.4 LOCAL DATA 968
3.3.6.2.9.20.5 PROCESS CONTROL 968
3.3.6.2.9.20.6 PROCESSING 968
3.3.6.2.9.20.7 UTILIZATION OF OTHER ELEMENTS 968
3.3.6.2.9.20.8 LIMITATIONS 969
3.3.6.2.9.20.9 LLCSC DESIGN 969
3.3.6.2.9.20.10 UNIT DESIGN 969
3.3.6.2.9.20.10.1 "*" UNIT DESIGN (CATALOG #P423-0) 969
3.3.6.2.9.20.10.1.1 REQUIREMENTS ALLOCATION 969
3.3.6.2.9.20.10.1.2 LOCAL ENTITIES DESIGN 969
3.3.6.2.9.20.10.1.3 INPUT/OUTPUT 969
3.3.6.2.9.20.10.1.4 LOCAL DATA 969
3.3.6.2.9.20.10.1.5 PROCESS CONTROL 970
3.3.6.2.9.20.10.1.6 PROCESSING 970
3.3.6.2.9.20.10.1.7 UTILIZATION OF OTHER ELEMENTS 970
3.3.6.2.9.20.10.1.8 LIMITATIONS 971
3.3.6.2.9.20.10.2 "/" UNIT DESIGN (CATALOG #P424-0) 971
3.3.6.2.9.20.10.2.1 REQUIREMENTS ALLOCATION 971
3.3.6.2.9.20.10.2.2 LOCAL ENTITIES DESIGN 971
3.3.6.2.9.20.10.2.3 INPUT/OUTPUT 971
3.3.6.2.9.20.10.2.4 LOCAL DATA 972
3.3.6.2.9.20.10.2.5 PROCESS CONTROL 972
3.3.6.2.9.20.10.2.6 PROCESSING 972
3.3.6.2.9.20.10.2.7 UTILIZATION OF OTHER ELEMENTS 973
3.3.6.2.9.20.10.2.8 LIMITATIONS 973
3.3.6.2.9.21 MATRIX_SCALAR_OPERATIONS_CONSTRAINED PACKAGE DESIGN (CATALOG #P428-0) 973
3.3.6.2.9.21.1 REQUIREMENTS ALLOCATION 974
3.3.6.2.9.21.2 LOCAL ENTITIES DESIGN 974
3.3.6.2.9.21.3 INPUT/OUTPUT 974
3.3.6.2.9.21.4 LOCAL DATA 975
3.3.6.2.9.21.5 PROCESS CONTROL 975
3.3.6.2.9.21.6 PROCESSING 975
3.3.6.2.9.21.7 UTILIZATION OF OTHER ELEMENTS 975
3.3.6.2.9.21.8 LIMITATIONS 975
3.3.6.2.9.21.9 LLCSC DESIGN 975
3.3.6.2.9.21.10 UNIT DESIGN 975
3.3.6.2.9.21.10.1 "*" UNIT DESIGN (CATALOG #P429-0) 975
3.3.6.2.9.21.10.1.1 REQUIREMENTS ALLOCATION 975
3.3.6.2.9.21.10.1.2 LOCAL ENTITIES DESIGN 975
3.3.6.2.9.21.10.1.3 INPUT/OUTPUT 976
3.3.6.2.9.21.10.1.4 LOCAL DATA 976
3.3.6.2.9.21.10.1.5 PROCESS CONTROL 976

3.3.6.2.9.21.10.1.6 PROCESSING 976
3.3.6.2.9.21.10.1.7 UTILIZATION OF OTHER ELEMENTS 977
3.3.6.2.9.21.10.1.8 LIMITATIONS 978
3.3.6.2.9.21.10.2 "/" UNIT DESIGN (CATALOG #P430-0) 978
3.3.6.2.9.21.10.2.1 REQUIREMENTS ALLOCATION 978
3.3.6.2.9.21.10.2.2 LOCAL ENTITIES DESIGN 978
3.3.6.2.9.21.10.2.3 INPUT/OUTPUT 978
3.3.6.2.9.21.10.2.4 LOCAL DATA 978
3.3.6.2.9.21.10.2.5 PROCESS CONTROL 978
3.3.6.2.9.21.10.2.6 PROCESSING 979
3.3.6.2.9.21.10.2.7 UTILIZATION OF OTHER ELEMENTS 979
3.3.6.2.9.21.10.2.8 LIMITATIONS 980
3.3.6.2.9.22 VECTOR_MATRIX_MULTIPLY_UNRESTRICTED PACKAGE DESIGN (CATALOG #P437-0) 980
3.3.6.2.9.22.1 REQUIREMENTS ALLOCATION 981
3.3.6.2.9.22.2 LOCAL ENTITIES DESIGN 981
3.3.6.2.9.22.3 INPUT/OUTPUT 981
3.3.6.2.9.22.4 LOCAL DATA 982
3.3.6.2.9.22.5 PROCESS CONTROL 983
3.3.6.2.9.22.6 PROCESSING 983
3.3.6.2.9.22.7 UTILIZATION OF OTHER ELEMENTS 984
3.3.6.2.9.22.8 LIMITATIONS 984
3.3.6.2.9.22.9 LLCSC DESIGN 985
3.3.6.2.9.22.10 UNIT DESIGN 985
3.3.6.2.9.23 ABA_TRANS_DYNAM_SPARSE_MATRIX_SQ_MATRIX PACKAGE DESIGN (CATALOG #P1066-0) 985
3.3.6.2.9.23.1 REQUIREMENTS ALLOCATION 985
3.3.6.2.9.23.2 LOCAL ENTITIES DESIGN 985
3.3.6.2.9.23.3 INPUT/OUTPUT 985
3.3.6.2.9.23.4 LOCAL DATA 986
3.3.6.2.9.23.5 PROCESS CONTROL 986
3.3.6.2.9.23.6 PROCESSING 986
3.3.6.2.9.23.7 UTILIZATION OF OTHER ELEMENTS 988
3.3.6.2.9.23.8 LIMITATIONS 988
3.3.6.2.9.23.9 LLCSC DESIGN 989
3.3.6.2.9.23.10 UNIT DESIGN 989
3.3.6.2.9.24 ABA_TRANS_VECTOR_SQ_MATRIX PACKAGE DESIGN (CATALOG #P1068-0) 989
3.3.6.2.9.24.1 REQUIREMENTS ALLOCATION 989
3.3.6.2.9.24.2 LOCAL ENTITIES DESIGN 989
3.3.6.2.9.24.3 INPUT/OUTPUT 990
3.3.6.2.9.24.4 LOCAL DATA 990
3.3.6.2.9.24.5 PROCESS CONTROL 990
3.3.6.2.9.24.6 PROCESSING 990
3.3.6.2.9.24.7 UTILIZATION OF OTHER ELEMENTS 992
3.3.6.2.9.24.8 LIMITATIONS 992
3.3.6.2.9.24.9 LLCSC DESIGN 992
3.3.6.2.9.24.10 UNIT DESIGN 992
3.3.6.2.9.25 ABA_TRANS_VECTOR_SCALAR PACKAGE DESIGN (CATALOG #P1070-0) 992
3.3.6.2.9.25.1 REQUIREMENTS ALLOCATION 993
3.3.6.2.9.25.2 LOCAL ENTITIES DESIGN 993
3.3.6.2.9.25.3 INPUT/OUTPUT 994
3.3.6.2.9.25.4 LOCAL DATA 994
3.3.6.2.9.25.5 PROCESS CONTROL 994
3.3.6.2.9.25.6 PROCESSING 994
3.3.6.2.9.25.7 UTILIZATION OF OTHER ELEMENTS 996

3.3.6.2.9.25.8 LIMITATIONS 997
3.3.6.2.9.25.9 LLCSC DESIGN 997
3.3.6.2.9.25.10 UNIT DESIGN 997
3.3.6.2.9.26 COLUMN_MATRIX_OPERATIONS PACKAGE DESIGN (CATALOG #P1072-0) 997
3.3.6.2.9.26.1 REQUIREMENTS ALLOCATION 997
3.3.6.2.9.26.2 LOCAL ENTITIES DESIGN 997
3.3.6.2.9.26.3 INPUT/OUTPUT 997
3.3.6.2.9.26.4 LOCAL DATA 998
3.3.6.2.9.26.5 PROCESS CONTROL 998
3.3.6.2.9.26.6 PROCESSING 998
3.3.6.2.9.26.7 UTILIZATION OF OTHER ELEMENTS 998
3.3.6.2.9.26.8 LIMITATIONS 998
3.3.6.2.9.26.9 LLCSC DESIGN 998
3.3.6.2.9.26.10 UNIT DESIGN 998
3.3.6.2.9.26.10.1 SET_DIAGONAL_AND_SUBTRACT_FROM_IDENTITY UNIT DESIGN (CATALOG #P1073-0) 998
3.3.6.2.9.26.10.1.1 REQUIREMENTS ALLOCATION 998
3.3.6.2.9.26.10.1.2 LOCAL ENTITIES DESIGN 998
3.3.6.2.9.26.10.1.3 INPUT/OUTPUT 999
3.3.6.2.9.26.10.1.4 LOCAL DATA 999
3.3.6.2.9.26.10.1.5 PROCESS CONTROL 999
3.3.6.2.9.26.10.1.6 PROCESSING 999
3.3.6.2.9.26.10.1.7 UTILIZATION OF OTHER ELEMENTS 1000
3.3.6.2.9.26.10.1.8 LIMITATIONS 1000
3.3.6.2.9.26.10.2 ABA_TRANSPOSE UNIT DESIGN (CATALOG #P1075-0) 1000
3.3.6.2.9.26.10.2.1 REQUIREMENTS ALLOCATION 1000
3.3.6.2.9.26.10.2.2 LOCAL ENTITIES DESIGN 1000
3.3.6.2.9.26.10.2.3 INPUT/OUTPUT 1000
3.3.6.2.9.26.10.2.4 LOCAL DATA 1001
3.3.6.2.9.26.10.2.5 PROCESS CONTROL 1001
3.3.6.2.9.26.10.2.6 PROCESSING 1001
3.3.6.2.9.26.10.2.7 UTILIZATION OF OTHER ELEMENTS 1002
3.3.6.2.9.26.10.2.8 LIMITATIONS 1002
3.3.6.2.9.26.10.3 ABA_SYMM_TRANSPOSE UNIT DESIGN (CATALOG #P1076-0) 1002
3.3.6.2.9.26.10.3.1 REQUIREMENTS ALLOCATION 1002
3.3.6.2.9.26.10.3.2 LOCAL ENTITIES DESIGN 1002
3.3.6.2.9.26.10.3.3 INPUT/OUTPUT 1002
3.3.6.2.9.26.10.3.4 LOCAL DATA 1003
3.3.6.2.9.26.10.3.5 PROCESS CONTROL 1003
3.3.6.2.9.26.10.3.6 PROCESSING 1003
3.3.6.2.9.26.10.3.7 UTILIZATION OF OTHER ELEMENTS 1004
3.3.6.2.9.26.10.3.8 LIMITATIONS 1005
3.3.6.2.10 UNIT DESIGN 1005
3.3.6.2.10.1 MATRIX_MATRIX_MULTIPLY_RESTRICTED UNIT DESIGN (CATALOG #P441-0) 1005
3.3.6.2.10.1.1 REQUIREMENTS ALLOCATION 1005
3.3.6.2.10.1.2 LOCAL ENTITIES DESIGN 1005
3.3.6.2.10.1.3 INPUT/OUTPUT 1005
3.3.6.2.10.1.4 LOCAL DATA 1006
3.3.6.2.10.1.5 PROCESS CONTROL 1007
3.3.6.2.10.1.6 PROCESSING 1007
3.3.6.2.10.1.7 UTILIZATION OF OTHER ELEMENTS 1008
3.3.6.2.10.1.8 LIMITATIONS 1008

3.3.6.2.10.2 MATRIX_VECTOR_MULTIPLY_RESTRICTED UNIT DESIGN (CATALOG #P436-0) 1008
3.3.6.2.10.2.1 REQUIREMENTS ALLOCATION 1008
3.3.6.2.10.2.2 LOCAL ENTITIES DESIGN 1008
3.3.6.2.10.2.3 INPUT/OUTPUT 1008
3.3.6.2.10.2.4 LOCAL DATA 1009
3.3.6.2.10.2.5 PROCESS CONTROL 1010
3.3.6.2.10.2.6 PROCESSING 1010
3.3.6.2.10.2.7 UTILIZATION OF OTHER ELEMENTS 1011
3.3.6.2.10.2.8 LIMITATIONS 1011
3.3.6.2.10.3 VECTOR_VECTOR_TRANSPOSE_MULTIPLY_RESTRICTED UNIT DESIGN (CATALOG #P444-0) 1011
3.3.6.2.10.3.1 REQUIREMENTS ALLOCATION 1011
3.3.6.2.10.3.2 LOCAL ENTITIES DESIGN 1011
3.3.6.2.10.3.3 INPUT/OUTPUT 1011
3.3.6.2.10.3.4 LOCAL DATA 1012
3.3.6.2.10.3.5 PROCESS CONTROL 1013
3.3.6.2.10.3.6 PROCESSING 1013
3.3.6.2.10.3.7 UTILIZATION OF OTHER ELEMENTS 1013
3.3.6.2.10.3.8 LIMITATIONS 1013
3.3.6.2.10.4 MATRIX_MATRIX_TRANSPOSE_MULTIPLY_RESTRICTED UNIT DESIGN (CATALOG #P447-0) 1014
3.3.6.2.10.4.1 REQUIREMENTS ALLOCATION 1014
3.3.6.2.10.4.2 LOCAL ENTITIES DESIGN 1014
3.3.6.2.10.4.3 INPUT/OUTPUT 1014
3.3.6.2.10.4.4 LOCAL DATA 1015
3.3.6.2.10.4.5 PROCESS CONTROL 1015
3.3.6.2.10.4.6 PROCESSING 1015
3.3.6.2.10.4.7 UTILIZATION OF OTHER ELEMENTS 1016
3.3.6.2.10.4.8 LIMITATIONS 1016
3.3.6.2.10.5 DOT_PRODUCT_OPERATIONS_RESTRICTED UNIT DESIGN (CATALOG #P450-0) 1016
3.3.6.2.10.5.1 REQUIREMENTS ALLOCATION 1016
3.3.6.2.10.5.2 LOCAL ENTITIES DESIGN 1016
3.3.6.2.10.5.3 INPUT/OUTPUT 1017
3.3.6.2.10.5.4 LOCAL DATA 1017
3.3.6.2.10.5.5 PROCESS CONTROL 1018
3.3.6.2.10.5.6 PROCESSING 1018
3.3.6.2.10.5.7 UTILIZATION OF OTHER ELEMENTS 1018
3.3.6.2.10.5.8 LIMITATIONS 1018
3.3.6.2.10.6 DIAGONAL_FULL_MATRIX_ADD_RESTRICTED UNIT DESIGN (CATALOG #P453-0) 1019
3.3.6.2.10.6.1 REQUIREMENTS ALLOCATION 1019
3.3.6.2.10.6.2 LOCAL ENTITIES DESIGN 1019
3.3.6.2.10.6.3 INPUT/OUTPUT 1019
3.3.6.2.10.6.4 LOCAL DATA 1020
3.3.6.2.10.6.5 PROCESS CONTROL 1020
3.3.6.2.10.6.6 PROCESSING 1020
3.3.6.2.10.6.7 UTILIZATION OF OTHER ELEMENTS 1021
3.3.6.2.10.6.8 LIMITATIONS 1021
3.3.6.2.10.7 VECTOR_MATRIX_MULTIPLY_RESTRICTED UNIT DESIGN (CATALOG #P438-0) 1021
3.3.6.2.10.7.1 REQUIREMENTS ALLOCATION 1021
3.3.6.2.10.7.2 LOCAL ENTITIES DESIGN 1021
3.3.6.2.10.7.3 INPUT/OUTPUT 1021
3.3.6.2.10.7.4 LOCAL DATA 1022
3.3.6.2.10.7.5 PROCESS CONTROL 1023

3.3.6.2.10.7.6 PROCESSING 1023
3.3.6.2.10.7.7 UTILIZATION OF OTHER ELEMENTS 1023
3.3.6.2.10.7.8 LIMITATIONS 1024

3.3.6.3 STANDARD TRIG (BODY) TLCSC P683 (CATALOG #P2-0) 1. 121
3.3.6.3.1 REQUIREMENTS ALLOCATION 1121
3.3.6.3.2 LOCAL ENTITIES DESIGN 1121
3.3.6.3.3 INPUT/OUTPUT 1122
3.3.6.3.4 LOCAL DATA 1122
3.3.6.3.5 PROCESS CONTROL 1123
3.3.6.3.6 PROCESSING 1123
3.3.6.3.7 UTILIZATION OF OTHER ELEMENTS 1126
3.3.6.3.8 LIMITATIONS 1127
3.3.6.3.9 LLCSC DESIGN 1127
3.3.6.3.10 UNIT DESIGN 1127
3.3.6.3.10.1 SIN (FOR RADIANS, SEMICIRCLES, AND DEGREES) UNIT DESIGN (CATALOG #P3-0 {RADIANS},P538-0 {SEMICIRCLES},P539-0 {DEGREES}) 1127
3.3.6.3.10.1.1 REQUIREMENTS ALLOCATION 1127
3.3.6.3.10.1.2 LOCAL ENTITIES DESIGN 1127
3.3.6.3.10.1.3 INPUT/OUTPUT 1127
3.3.6.3.10.1.4 LOCAL DATA 1127
3.3.6.3.10.1.5 PROCESS CONTROL 1128
3.3.6.3.10.1.6 PROCESSING 1128
3.3.6.3.10.1.7 UTILIZATION OF OTHER ELEMENTS 1128
3.3.6.3.10.1.8 LIMITATIONS 1129
3.3.6.3.10.2 COS (FOR RADIANS, SEMICIRCLES, AND DEGREES) UNIT DESIGN (CATALOG #P4-0 {RADIANS},P540-0 {SEMICIRCLES},P541-0 {DEGREES}) 1129
3.3.6.3.10.2.1 REQUIREMENTS ALLOCATION 1129
3.3.6.3.10.2.2 LOCAL ENTITIES DESIGN 1129
3.3.6.3.10.2.3 INPUT/OUTPUT 1129
3.3.6.3.10.2.4 LOCAL DATA 1129
3.3.6.3.10.2.5 PROCESS CONTROL 1129
3.3.6.3.10.2.6 PROCESSING 1130
3.3.6.3.10.2.7 UTILIZATION OF OTHER ELEMENTS 1130
3.3.6.3.10.2.8 LIMITATIONS 1131
3.3.6.3.10.3 SIN COS (FOR RADIANS, SEMICIRCLES, AND DEGREES) UNIT DESIGN (CATALOG #P5-0 {RADIANS},P542-0 {SEMICIRCLES},P543-0 {DEGREES}) 1131
3.3.6.3.10.3.1 REQUIREMENTS ALLOCATION 1131
3.3.6.3.10.3.2 LOCAL ENTITIES DESIGN 1131
3.3.6.3.10.3.3 INPUT/OUTPUT 1131
3.3.6.3.10.3.4 LOCAL DATA 1132
3.3.6.3.10.3.5 PROCESS CONTROL 1132
3.3.6.3.10.3.6 PROCESSING 1132
3.3.6.3.10.3.7 UTILIZATION OF OTHER ELEMENTS 1135
3.3.6.3.10.3.8 LIMITATIONS 1136
3.3.6.3.10.4 TAN (FOR RADIANS, SEMICIRCLES, AND DEGREES) UNIT DESIGN (CATALOG #P6-0 {RADIANS},P544-0 {SEMICIRCLES},P545-0 {DEGREES}) 1136
3.3.6.3.10.4.1 REQUIREMENTS ALLOCATION 1136
3.3.6.3.10.4.2 LOCAL ENTITIES DESIGN 1136
3.3.6.3.10.4.3 INPUT/OUTPUT 1136
3.3.6.3.10.4.4 LOCAL DATA 1136
3.3.6.3.10.4.5 PROCESS CONTROL 1136

3.3.6.3.10.4.6 PROCESSING 1137
3.3.6.3.10.4.7 UTILIZATION OF OTHER ELEMENTS 1137
3.3.6.3.10.4.8 LIMITATIONS 1138
3.3.6.3.10.5 ARCSIN (FOR RADIANS, SEMICIRCLES, AND DEGREES) UNIT DESIGN (CATALOG #P7-0 {RADIANS},P546-0 {SEMICIRCLES},P547-0 {DEGREES}) 1138
3.3.6.3.10.5.1 REQUIREMENTS ALLOCATION 1138
3.3.6.3.10.5.2 LOCAL ENTITIES DESIGN 1138
3.3.6.3.10.5.3 INPUT/OUTPUT 1138
3.3.6.3.10.5.4 LOCAL DATA 1138
3.3.6.3.10.5.5 PROCESS CONTROL 1138
3.3.6.3.10.5.6 PROCESSING 1138
3.3.6.3.10.5.7 UTILIZATION OF OTHER ELEMENTS 1139
3.3.6.3.10.5.8 LIMITATIONS 1139
3.3.6.3.10.6 ARCCOS (FOR RADIANS, SEMICIRCLES, AND DEGREES) UNIT DESIGN (CATALOG #P8-0 {RADIANS},P548-0 {SEMICIRCLES},P549-0 {DEGREES}) 1139
3.3.6.3.10.6.1 REQUIREMENTS ALLOCATION 1140
3.3.6.3.10.6.2 LOCAL ENTITIES DESIGN 1140
3.3.6.3.10.6.3 INPUT/OUTPUT 1140
3.3.6.3.10.6.4 LOCAL DATA 1140
3.3.6.3.10.6.5 PROCESS CONTROL 1140
3.3.6.3.10.6.6 PROCESSING 1140
3.3.6.3.10.6.7 UTILIZATION OF OTHER ELEMENTS 1141
3.3.6.3.10.6.8 LIMITATIONS 1141
3.3.6.3.10.7 ARCSIN ARCCOS (FOR RADIANS, SEMICIRCLES, AND DEGREES) UNIT DESIGN (CATALOG #P9-0 {RADIANS},P550-0 {SEMICIRCLES},P551-0 {DEGREES}) 1141
3.3.6.3.10.7.1 REQUIREMENTS ALLOCATION 1141
3.3.6.3.10.7.2 LOCAL ENTITIES DESIGN 1142
3.3.6.3.10.7.3 INPUT/OUTPUT 1142
3.3.6.3.10.7.4 LOCAL DATA 1142
3.3.6.3.10.7.5 PROCESS CONTROL 1142
3.3.6.3.10.7.6 PROCESSING 1142
3.3.6.3.10.7.7 UTILIZATION OF OTHER ELEMENTS 1143
3.3.6.3.10.7.8 LIMITATIONS 1144
3.3.6.3.10.8 ARCTAN (FOR RADIANS, SEMICIRCLES, AND DEGREES) UNIT DESIGN (CATALOG #P10-0 {RADIANS},P552-0 {SEMICIRCLES},P553-0 {DEGREES}) 1144
3.3.6.3.10.8.1 REQUIREMENTS ALLOCATION 1145
3.3.6.3.10.8.2 LOCAL ENTITIES DESIGN 1145
3.3.6.3.10.8.3 INPUT/OUTPUT 1145
3.3.6.3.10.8.4 LOCAL DATA 1145
3.3.6.3.10.8.5 PROCESS CONTROL 1145
3.3.6.3.10.8.6 PROCESSING 1145
3.3.6.3.10.8.7 UTILIZATION OF OTHER ELEMENTS 1146
3.3.6.3.10.8.8 LIMITATIONS 1146
3.3.6.3.10.9 ARCTAN2 (FUNCTION BODY) UNIT DESIGN (CATALOG #P554-0) 1146
3.3.6.3.10.9.1 REQUIREMENTS ALLOCATION 1146
3.3.6.3.10.9.2 LOCAL ENTITIES DESIGN 1147
3.3.6.3.10.9.3 INPUT/OUTPUT 1147
3.3.6.3.10.9.4 LOCAL DATA 1148

3.3.6.3.10.9.5 PROCESS CONTROL 1148
3.3.6.3.10.9.6 PROCESSING 1148
3.3.6.3.10.9.7 UTILIZATION OF OTHER ELEMENTS 1149
3.3.6.3.10.9.8 LIMITATIONS 1149

3.3.6.4 GEOMETRIC_OPERATIONS TLCSC P684 (CATALOG #P118-0) 1. 161
3.3.6.4.1 REQUIREMENTS ALLOCATION 1161
3.3.6.4.2 LOCAL ENTITIES DESIGN 1161
3.3.6.4.3 INPUT/OUTPUT 1161
3.3.6.4.4 LOCAL DATA 1161
3.3.6.4.5 PROCESS CONTROL 1161
3.3.6.4.6 PROCESSING 1161
3.3.6.4.7 UTILIZATION OF OTHER ELEMENTS 1162
3.3.6.4.8 LIMITATIONS 1162
3.3.6.4.9 LLCSC DESIGN 1162
3.3.6.4.9.1 GREAT_CIRCLE_ARC_LENGTH PACKAGE DESIGN (CATALOG #P122-0) 1162
3.3.6.4.9.1.1 REQUIREMENTS ALLOCATION 1163
3.3.6.4.9.1.2 LOCAL ENTITIES DESIGN 1163
3.3.6.4.9.1.3 INPUT/OUTPUT 1163
3.3.6.4.9.1.4 LOCAL DATA 1164
3.3.6.4.9.1.5 PROCESS CONTROL 1165
3.3.6.4.9.1.6 PROCESSING 1165
3.3.6.4.9.1.7 UTILIZATION OF OTHER ELEMENTS 1166
3.3.6.4.9.1.8 LIMITATIONS 1168
3.3.6.4.9.1.9 LLCSC DESIGN 1168
3.3.6.4.9.1.10 UNIT DESIGN 1168
3.3.6.4.10 UNIT DESIGN 1168
3.3.6.4.10.1 UNIT_RADIAL_VECTOR UNIT DESIGN (CATALOG #P119-0) 1168
3.3.6.4.10.1.1 REQUIREMENTS ALLOCATION 1168
3.3.6.4.10.1.2 LOCAL ENTITIES DESIGN 1168
3.3.6.4.10.1.3 INPUT/OUTPUT 1168
3.3.6.4.10.1.4 LOCAL DATA 1170
3.3.6.4.10.1.5 PROCESS CONTROL 1170
3.3.6.4.10.1.6 PROCESSING 1170
3.3.6.4.10.1.7 UTILIZATION OF OTHER ELEMENTS 1171
3.3.6.4.10.1.8 LIMITATIONS 1171
3.3.6.4.10.2 UNIT_NORMAL_VECTOR UNIT DESIGN (CATALOG #P120-0) 1171
3.3.6.4.10.2.1 REQUIREMENTS ALLOCATION 1171
3.3.6.4.10.2.2 LOCAL ENTITIES DESIGN 1171
3.3.6.4.10.2.3 INPUT/OUTPUT 1171
3.3.6.4.10.2.4 LOCAL DATA 1172
3.3.6.4.10.2.5 PROCESS CONTROL 1172
3.3.6.4.10.2.6 PROCESSING 1173
3.3.6.4.10.2.7 UTILIZATION OF OTHER ELEMENTS 1173
3.3.6.4.10.2.8 LIMITATIONS 1173
3.3.6.4.10.3 COMPUTE_SEGMENT_AND_UNIT_NORMAL_VECTOR UNIT DESIGN (CATALOG #P121-0) 1173
3.3.6.4.10.3.1 REQUIREMENTS ALLOCATION 1174
3.3.6.4.10.3.2 LOCAL ENTITIES DESIGN 1174
3.3.6.4.10.3.3 INPUT/OUTPUT 1174
3.3.6.4.10.3.4 LOCAL DATA 1175
3.3.6.4.10.3.5 PROCESS CONTROL 1175
3.3.6.4.10.3.6 PROCESSING 1175
3.3.6.4.10.3.7 UTILIZATION OF OTHER ELEMENTS 1176

3.3.6.4.10.3.8 LIMITATIONS 1176
3.3.6.4.10.4 COMPUTE_SEGMENT_AND_UNIT_NORMAL_VECTOR_WITH_ARCSIN UNIT DESIGN (CATALOG #P1050-0) 1176
3.3.6.4.10.4.1 REQUIREMENTS ALLOCATION 1176
3.3.6.4.10.4.2 LOCAL ENTITIES DESIGN 1177
3.3.6.4.10.4.3 INPUT/OUTPUT 1177
3.3.6.4.10.4.4 LOCAL DATA 1178
3.3.6.4.10.4.5 PROCESS CONTROL 1178
3.3.6.4.10.4.6 PROCESSING 1178
3.3.6.4.10.4.7 UTILIZATION OF OTHER ELEMENTS 1179
3.3.6.4.10.4.8 LIMITATIONS 1179

3.3.6.5 DATA_CONVERSION 1189

3.3.6.5.1 UNIT_CONVERSIONS (PACKAGE BODY) TLCSC P851 (CATALOG #P631-0) 1191
3.3.6.5.1.1 REQUIREMENTS ALLOCATION 1193
3.3.6.5.1.2 LOCAL ENTITIES DESIGN 1193
3.3.6.5.1.3 INPUT/OUTPUT 1193
3.3.6.5.1.4 LOCAL DATA 1193
3.3.6.5.1.5 PROCESS CONTROL 1193
3.3.6.5.1.6 PROCESSING 1193
3.3.6.5.1.7 UTILIZATION OF OTHER ELEMENTS 1200
3.3.6.5.1.8 LIMITATIONS 1200
3.3.6.5.1.9 LLCSC DESIGN 1200
3.3.6.5.1.10 UNIT DESIGN 1200

3.3.6.5.2 EXTERNAL_FORM_CONVERSION_TWOS_COMPLEMENT (PACKAGE BODY) TLCSC P852 (CATALOG #P684-0) 1209
3.3.6.5.2.1 REQUIREMENTS ALLOCATION 1209
3.3.6.5.2.2 LOCAL ENTITIES DESIGN 1209
3.3.6.5.2.3 INPUT/OUTPUT 1210
3.3.6.5.2.4 LOCAL DATA 1211
3.3.6.5.2.5 PROCESS CONTROL 1211
3.3.6.5.2.6 PROCESSING 1211
3.3.6.5.2.7 UTILIZATION OF OTHER ELEMENTS 1212
3.3.6.5.2.8 LIMITATIONS 1212
3.3.6.5.2.9 LLCSC DESIGN 1213
3.3.6.5.2.10 UNIT DESIGN 1213
3.3.6.5.2.10.1 SCALE (FUNCTION BODY) UNIT DESIGN (CATALOG #P685-0) 1213
3.3.6.5.2.10.1.1 REQUIREMENTS ALLOCATION 1213
3.3.6.5.2.10.1.2 LOCAL ENTITIES DESIGN 1213
3.3.6.5.2.10.1.3 INPUT/OUTPUT 1213
3.3.6.5.2.10.1.4 LOCAL DATA 1214
3.3.6.5.2.10.1.5 PROCESS CONTROL 1214
3.3.6.5.2.10.1.6 PROCESSING 1214
3.3.6.5.2.10.1.7 UTILIZATION OF OTHER ELEMENTS 1214
3.3.6.5.2.10.1.8 LIMITATIONS 1215
3.3.6.5.2.10.2 UNSCALE UNIT DESIGN (CATALOG #P686-0) 1215
3.3.6.5.2.10.2.1 REQUIREMENTS ALLOCATION 1216
3.3.6.5.2.10.2.2 LOCAL ENTITIES DESIGN 1216
3.3.6.5.2.10.2.3 INPUT/OUTPUT 1216
3.3.6.5.2.10.2.4 LOCAL DATA 1217
3.3.6.5.2.10.2.5 PROCESS CONTROL 1217
3.3.6.5.2.10.2.6 PROCESSING 1217
3.3.6.5.2.10.2.7 UTILIZATION OF OTHER ELEMENTS 1217

3.3.6.5.2.10.2.8 LIMITATIONS 1218

3.3.6.6 SIGNAL_PROCESSING (BODY) TLCSC P686 (BODY) (CATALOG #P81-0) 1221
3.3.6.6.1 REQUIREMENTS ALLOCATION 1221
3.3.6.6.2 LOCAL ENTITIES DESIGN 1221
3.3.6.6.3 INPUT/OUTPUT 1221
3.3.6.6.4 LOCAL DATA 1222
3.3.6.6.5 PROCESS CONTROL 1222
3.3.6.6.6 PROCESSING 1222
3.3.6.6.7 UTILIZATION OF OTHER ELEMENTS 1222
3.3.6.6.8 LIMITATIONS 1223
3.3.6.6.9 LLCSC DESIGN 1223
3.3.6.6.9.1 UPPER_LOWER_LIMITER (PACKAGE BODY) PACKAGE DESIGN (CATALOG #P82-0) 1223
3.3.6.6.9.1.1 REQUIREMENTS ALLOCATION 1223
3.3.6.6.9.1.2 LOCAL ENTITIES DESIGN 1223
3.3.6.6.9.1.3 INPUT/OUTPUT 1223
3.3.6.6.9.1.4 LOCAL DATA 1223
3.3.6.6.9.1.5 PROCESS CONTROL 1224
3.3.6.6.9.1.6 PROCESSING 1224
3.3.6.6.9.1.7 UTILIZATION OF OTHER ELEMENTS 1224
3.3.6.6.9.1.8 LIMITATIONS 1224
3.3.6.6.9.1.9 LLCSC DESIGN 1224
3.3.6.6.9.1.10 UNIT DESIGN 1225
3.3.6.6.9.1.10.1 UPDATE_LIMITS (SUBPROGRAM BODY) UNIT DESIGN 1225
3.3.6.6.9.1.10.1.1 REQUIREMENTS ALLOCATION 1225
3.3.6.6.9.1.10.1.2 LOCAL ENTITIES DESIGN 1225
3.3.6.6.9.1.10.1.3 INPUT/OUTPUT 1225
3.3.6.6.9.1.10.1.4 LOCAL DATA 1226
3.3.6.6.9.1.10.1.5 PROCESS CONTROL 1226
3.3.6.6.9.1.10.1.6 PROCESSING 1226
3.3.6.6.9.1.10.1.7 UTILIZATION OF OTHER ELEMENTS 1226
3.3.6.6.9.1.10.1.8 LIMITATIONS 1226
3.3.6.6.9.1.10.2 LIMIT (SUBPROGRAM BODY) UNIT DESIGN 1226
3.3.6.6.9.1.10.2.1 REQUIREMENTS ALLOCATION 1226
3.3.6.6.9.1.10.2.2 LOCAL ENTITIES DESIGN 1227
3.3.6.6.9.1.10.2.3 INPUT/OUTPUT 1227
3.3.6.6.9.1.10.2.4 LOCAL DATA 1227
3.3.6.6.9.1.10.2.5 PROCESS CONTROL 1228
3.3.6.6.9.1.10.2.6 PROCESSING 1228
3.3.6.6.9.1.10.2.7 UTILIZATION OF OTHER ELEMENTS 1228
3.3.6.6.9.1.10.2.8 LIMITATIONS 1228
3.3.6.6.9.2 UPPER_LIMITER (PACKAGE BODY) PACKAGE DESIGN (CATALOG #P83-0) 1228
3.3.6.6.9.2.1 REQUIREMENTS ALLOCATION 1228
3.3.6.6.9.2.2 LOCAL ENTITIES DESIGN 1229
3.3.6.6.9.2.3 INPUT/OUTPUT 1229
3.3.6.6.9.2.4 LOCAL DATA 1229
3.3.6.6.9.2.5 PROCESS CONTROL 1229
3.3.6.6.9.2.6 PROCESSING 1229
3.3.6.6.9.2.7 UTILIZATION OF OTHER ELEMENTS 1230
3.3.6.6.9.2.8 LIMITATIONS 1230
3.3.6.6.9.2.9 LLCSC DESIGN 1230
3.3.6.6.9.2.10 UNIT DESIGN 1230

3.3.6.6.9.2.10.1 UPDATE_LIMIT (SUBPROGRAM BODY) UNIT DESIGN 1230
3.3.6.6.9.2.10.1.1 REQUIREMENTS ALLOCATION 1230
3.3.6.6.9.2.10.1.2 LOCAL ENTITIES DESIGN 1230
3.3.6.6.9.2.10.1.3 INPUT/OUTPUT 1230
3.3.6.6.9.2.10.1.4 LOCAL DATA 1231
3.3.6.6.9.2.10.1.5 PROCESS CONTROL 1231
3.3.6.6.9.2.10.1.6 PROCESSING 1231
3.3.6.6.9.2.10.1.7 UTILIZATION OF OTHER ELEMENTS 1231
3.3.6.6.9.2.10.1.8 LIMITATIONS 1231
3.3.6.6.9.2.10.2 LIMIT (SUBPROGRAM BODY) UNIT DESIGN 1232
3.3.6.6.9.2.10.2.1 REQUIREMENTS ALLOCATION 1232
3.3.6.6.9.2.10.2.2 LOCAL ENTITIES DESIGN 1232
3.3.6.6.9.2.10.2.3 INPUT/OUTPUT 1232
3.3.6.6.9.2.10.2.4 LOCAL DATA 1232
3.3.6.6.9.2.10.2.5 PROCESS CONTROL 1233
3.3.6.6.9.2.10.2.6 PROCESSING 1233
3.3.6.6.9.2.10.2.7 UTILIZATION OF OTHER ELEMENTS 1233
3.3.6.6.9.2.10.2.8 LIMITATIONS 1233
3.3.6.6.9.3 LOWER_LIMITER (PACKAGE BODY) PACKAGE DESIGN (CATALOG #P84-0) 1233
3.3.6.6.9.3.1 REQUIREMENTS ALLOCATION 1234
3.3.6.6.9.3.2 LOCAL ENTITIES DESIGN 1234
3.3.6.6.9.3.3 INPUT/OUTPUT 1234
3.3.6.6.9.3.4 LOCAL DATA 1234
3.3.6.6.9.3.5 PROCESS CONTROL 1234
3.3.6.6.9.3.6 PROCESSING 1235
3.3.6.6.9.3.7 UTILIZATION OF OTHER ELEMENTS 1235
3.3.6.6.9.3.8 LIMITATIONS 1235
3.3.6.6.9.3.9 LLCSC DESIGN 1235
3.3.6.6.9.3.10 UNIT DESIGN 1235
3.3.6.6.9.3.10.1 UPDATE_LIMIT (SUBPROGRAM BODY) UNIT DESIGN 1235
3.3.6.6.9.3.10.1.1 REQUIREMENTS ALLOCATION 1235
3.3.6.6.9.3.10.1.2 LOCAL ENTITIES DESIGN 1235
3.3.6.6.9.3.10.1.3 INPUT/OUTPUT 1235
3.3.6.6.9.3.10.1.4 LOCAL DATA 1236
3.3.6.6.9.3.10.1.5 PROCESS CONTROL 1236
3.3.6.6.9.3.10.1.6 PROCESSING 1236
3.3.6.6.9.3.10.1.7 UTILIZATION OF OTHER ELEMENTS 1236
3.3.6.6.9.3.10.1.8 LIMITATIONS 1237
3.3.6.6.9.3.10.2 LIMIT (SUBPROGRAM BODY) UNIT DESIGN 1237
3.3.6.6.9.3.10.2.1 REQUIREMENTS ALLOCATION 1237
3.3.6.6.9.3.10.2.2 LOCAL ENTITIES DESIGN 1237
3.3.6.6.9.3.10.2.3 INPUT/OUTPUT 1237
3.3.6.6.9.3.10.2.4 LOCAL DATA 1238
3.3.6.6.9.3.10.2.5 PROCESS CONTROL 1238
3.3.6.6.9.3.10.2.6 PROCESSING 1238
3.3.6.6.9.3.10.2.7 UTILIZATION OF OTHER ELEMENTS 1238
3.3.6.6.9.3.10.2.8 LIMITATIONS 1238
3.3.6.6.9.4 ABSOLUTE_LIMITER (PACKAGE BODY) PACKAGE DESIGN (CATALOG #P85-0) 1238
3.3.6.6.9.4.1 REQUIREMENTS ALLOCATION 1239
3.3.6.6.9.4.2 LOCAL ENTITIES DESIGN 1239
3.3.6.6.9.4.3 INPUT/OUTPUT 1239
3.3.6.6.9.4.4 LOCAL DATA 1239
3.3.6.6.9.4.5 PROCESS CONTROL 1240

3.3.6.6.9.4.6 PROCESSING 1240
3.3.6.6.9.4.7 UTILIZATION OF OTHER ELEMENTS 1240
3.3.6.6.9.4.8 LIMITATIONS 1240
3.3.6.6.9.4.9 LLCSC DESIGN 1240
3.3.6.6.9.4.10 UNIT DESIGN 1240
3.3.6.6.9.4.10.1 UPDATE_LIMIT (SUBPROGRAM BODY) UNIT DESIGN 1240
3.3.6.6.9.4.10.1.1 REQUIREMENTS ALLOCATION 1240
3.3.6.6.9.4.10.1.2 LOCAL ENTITIES DESIGN 1240
3.3.6.6.9.4.10.1.3 INPUT/OUTPUT 1241
3.3.6.6.9.4.10.1.4 LOCAL DATA 1241
3.3.6.6.9.4.10.1.5 PROCESS CONTROL 1241
3.3.6.6.9.4.10.1.6 PROCESSING 1241
3.3.6.6.9.4.10.1.7 UTILIZATION OF OTHER ELEMENTS 1242
3.3.6.6.9.4.10.1.8 LIMITATIONS 1242
3.3.6.6.9.4.10.2 LIMIT (SUBPROGRAM BODY) UNIT DESIGN 1242
3.3.6.6.9.4.10.2.1 REQUIREMENTS ALLOCATION 1242
3.3.6.6.9.4.10.2.2 LOCAL ENTITIES DESIGN 1242
3.3.6.6.9.4.10.2.3 INPUT/OUTPUT 1242
3.3.6.6.9.4.10.2.4 LOCAL DATA 1243
3.3.6.6.9.4.10.2.5 PROCESS CONTROL 1243
3.3.6.6.9.4.10.2.6 PROCESSING 1243
3.3.6.6.9.4.10.2.7 UTILIZATION OF OTHER ELEMENTS 1244
3.3.6.6.9.4.10.2.8 LIMITATIONS 1244
3.3.6.6.9.5 ABSOLUTE_LIMITER_WITH_FLAG (PACKAGE BODY) PACKAGE DESIGN (CATALOG #P86-0) 1244
3.3.6.6.9.5.1 REQUIREMENTS ALLOCATION 1244
3.3.6.6.9.5.2 LOCAL ENTITIES DESIGN 1244
3.3.6.6.9.5.3 INPUT/OUTPUT 1244
3.3.6.6.9.5.4 LOCAL DATA 1245
3.3.6.6.9.5.5 PROCESS CONTROL 1245
3.3.6.6.9.5.6 PROCESSING 1245
3.3.6.6.9.5.7 UTILIZATION OF OTHER ELEMENTS 1245
3.3.6.6.9.5.8 LIMITATIONS 1246
3.3.6.6.9.5.9 LLCSC DESIGN 1246
3.3.6.6.9.5.10 UNIT DESIGN 1246
3.3.6.6.9.5.10.1 UPDATE_LIMIT (SUBPROGRAM BODY) UNIT DESIGN 1246
3.3.6.6.9.5.10.1.1 REQUIREMENTS ALLOCATION 1246
3.3.6.6.9.5.10.1.2 LOCAL ENTITIES DESIGN 1246
3.3.6.6.9.5.10.1.3 INPUT/OUTPUT 1246
3.3.6.6.9.5.10.1.4 LOCAL DATA 1247
3.3.6.6.9.5.10.1.5 PROCESS CONTROL 1247
3.3.6.6.9.5.10.1.6 PROCESSING 1247
3.3.6.6.9.5.10.1.7 UTILIZATION OF OTHER ELEMENTS 1247
3.3.6.6.9.5.10.1.8 LIMITATIONS 1247
3.3.6.6.9.5.10.2 LIMIT (SUBPROGRAM BODY) UNIT DESIGN 1247
3.3.6.6.9.5.10.2.1 REQUIREMENTS ALLOCATION 1247
3.3.6.6.9.5.10.2.2 LOCAL ENTITIES DESIGN 1247
3.3.6.6.9.5.10.2.3 INPUT/OUTPUT 1248
3.3.6.6.9.5.10.2.4 LOCAL DATA 1248
3.3.6.6.9.5.10.2.5 PROCESS CONTROL 1248
3.3.6.6.9.5.10.2.6 PROCESSING 1249
3.3.6.6.9.5.10.2.7 UTILIZATION OF OTHER ELEMENTS 1249
3.3.6.6.9.5.10.2.8 LIMITATIONS 1249
3.3.6.6.9.5.10.3 LIMIT_FLAG_SETTING (SUBPROGRAM BODY) UNIT DESIGN 1249

3.3.6.6.9.5.10.3.1 REQUIREMENTS ALLOCATION 1249
3.3.6.6.9.5.10.3.2 LOCAL ENTITIES DESIGN 1250
3.3.6.6.9.5.10.3.3 INPUT/OUTPUT 1250
3.3.6.6.9.5.10.3.4 LOCAL DATA 1250
3.3.6.6.9.5.10.3.5 PROCESS CONTROL 1250
3.3.6.6.9.5.10.3.6 PROCESSING 1250
3.3.6.6.9.5.10.3.7 UTILIZATION OF OTHER ELEMENTS 1250
3.3.6.6.9.5.10.3.8 LIMITATIONS 1250
3.3.6.6.9.6 GENERAL FIRST ORDER FILTER (PACKAGE BODY) PACKAGE DESIGN (CATALOG #P87-0) 1250
3.3.6.6.9.6.1 REQUIREMENTS ALLOCATION 1251
3.3.6.6.9.6.2 LOCAL ENTITIES DESIGN 1251
3.3.6.6.9.6.3 INPUT/OUTPUT 1251
3.3.6.6.9.6.4 LOCAL DATA 1252
3.3.6.6.9.6.5 PROCESS CONTROL 1252
3.3.6.6.9.6.6 PROCESSING 1252
3.3.6.6.9.6.7 UTILIZATION OF OTHER ELEMENTS 1253
3.3.6.6.9.6.8 LIMITATIONS 1253
3.3.6.6.9.6.9 LLCSC DESIGN 1253
3.3.6.6.9.6.10 UNIT DESIGN 1253
3.3.6.6.9.6.10.1 UPDATE COEFFICIENTS (SUBPROGRAM BODY) UNIT DESIGN 1253
3.3.6.6.9.6.10.1.1 REQUIREMENTS ALLOCATION 1253
3.3.6.6.9.6.10.1.2 LOCAL ENTITIES DESIGN 1253
3.3.6.6.9.6.10.1.3 INPUT/OUTPUT 1253
3.3.6.6.9.6.10.1.4 LOCAL DATA 1254
3.3.6.6.9.6.10.1.5 PROCESS CONTROL 1254
3.3.6.6.9.6.10.1.6 PROCESSING 1254
3.3.6.6.9.6.10.1.7 UTILIZATION OF OTHER ELEMENTS 1254
3.3.6.6.9.6.10.1.8 LIMITATIONS 1254
3.3.6.6.9.6.10.2 FILTER (SUBPROGRAM BODY) UNIT DESIGN 1255
3.3.6.6.9.6.10.2.1 REQUIREMENTS ALLOCATION 1255
3.3.6.6.9.6.10.2.2 LOCAL ENTITIES DESIGN 1255
3.3.6.6.9.6.10.2.3 INPUT/OUTPUT 1255
3.3.6.6.9.6.10.2.4 LOCAL DATA 1256
3.3.6.6.9.6.10.2.5 PROCESS CONTROL 1256
3.3.6.6.9.6.10.2.6 PROCESSING 1256
3.3.6.6.9.6.10.2.7 UTILIZATION OF OTHER ELEMENTS 1256
3.3.6.6.9.6.10.2.8 LIMITATIONS 1256
3.3.6.6.9.6.10.3 REINITIALIZE (SUBPROGRAM BODY) UNIT DESIGN 1257
3.3.6.6.9.6.10.3.1 REQUIREMENTS ALLOCATION 1257
3.3.6.6.9.6.10.3.2 LOCAL ENTITIES DESIGN 1257
3.3.6.6.9.6.10.3.3 INPUT/OUTPUT 1257
3.3.6.6.9.6.10.3.4 LOCAL DATA 1258
3.3.6.6.9.6.10.3.5 PROCESS CONTROL 1258
3.3.6.6.9.6.10.3.6 PROCESSING 1258
3.3.6.6.9.6.10.3.7 UTILIZATION OF OTHER ELEMENTS 1258
3.3.6.6.9.6.10.3.8 LIMITATIONS 1258
3.3.6.6.9.7 TUSTIN LAG FILTER (PACKAGE BODY) PACKAGE DESIGN (CATALOG #P88-0) 1258
3.3.6.6.9.7.1 REQUIREMENTS ALLOCATION 1259
3.3.6.6.9.7.2 LOCAL ENTITIES DESIGN 1259
3.3.6.6.9.7.3 INPUT/OUTPUT 1259
3.3.6.6.9.7.4 LOCAL DATA 1260
3.3.6.6.9.7.5 PROCESS CONTROL 1260
3.3.6.6.9.7.6 PROCESSING 1260

3.3.6.6.9.7.7 UTILIZATION OF OTHER ELEMENTS 1260
3.3.6.6.9.7.8 LIMITATIONS 1260
3.3.6.6.9.7.9 LLCSC DESIGN 1261
3.3.6.6.9.7.10 UNIT DESIGN 1261
3.3.6.6.9.7.10.1 UPDATE_COEFFICIENTS (SUBPROGRAM BODY) UNIT DESIGN 1261
3.3.6.6.9.7.10.1.1 REQUIREMENTS ALLOCATION 1261
3.3.6.6.9.7.10.1.2 LOCAL ENTITIES DESIGN 1261
3.3.6.6.9.7.10.1.3 INPUT/OUTPUT 1261
3.3.6.6.9.7.10.1.4 LOCAL DATA 1262
3.3.6.6.9.7.10.1.5 PROCESS CONTROL 1262
3.3.6.6.9.7.10.1.6 PROCESSING 1262
3.3.6.6.9.7.10.1.7 UTILIZATION OF OTHER ELEMENTS 1262
3.3.6.6.9.7.10.1.8 LIMITATIONS 1262
3.3.6.6.9.7.10.2 FILTER (SUBPROGRAM BODY) UNIT DESIGN 1262
3.3.6.6.9.7.10.2.1 REQUIREMENTS ALLOCATION 1263
3.3.6.6.9.7.10.2.2 LOCAL ENTITIES DESIGN 1263
3.3.6.6.9.7.10.2.3 INPUT/OUTPUT 1263
3.3.6.6.9.7.10.2.4 LOCAL DATA 1264
3.3.6.6.9.7.10.2.5 PROCESS CONTROL 1264
3.3.6.6.9.7.10.2.6 PROCESSING 1264
3.3.6.6.9.7.10.2.7 UTILIZATION OF OTHER ELEMENTS 1265
3.3.6.6.9.7.10.2.8 LIMITATIONS 1265
3.3.6.6.9.7.10.3 REINITIALIZE (SUBPROGRAM BODY) UNIT DESIGN 1265
3.3.6.6.9.7.10.3.1 REQUIREMENTS ALLOCATION 1265
3.3.6.6.9.7.10.3.2 LOCAL ENTITIES DESIGN 1265
3.3.6.6.9.7.10.3.3 INPUT/OUTPUT 1265
3.3.6.6.9.7.10.3.4 LOCAL DATA 1266
3.3.6.6.9.7.10.3.5 PROCESS CONTROL 1266
3.3.6.6.9.7.10.3.6 PROCESSING 1266
3.3.6.6.9.7.10.3.7 UTILIZATION OF OTHER ELEMENTS 1266
3.3.6.6.9.7.10.3.8 LIMITATIONS 1266
3.3.6.6.9.8 TUSTIN_LEAD_LAG_FILTER (PACKAGE BODY) PACKAGE DESIGN (CATALOG #P89-0) 1266
3.3.6.6.9.8.1 REQUIREMENTS ALLOCATION 1267
3.3.6.6.9.8.2 LOCAL ENTITIES DESIGN 1267
3.3.6.6.9.8.3 INPUT/OUTPUT 1267
3.3.6.6.9.8.4 LOCAL DATA 1268
3.3.6.6.9.8.5 PROCESS CONTROL 1268
3.3.6.6.9.8.6 PROCESSING 1268
3.3.6.6.9.8.7 UTILIZATION OF OTHER ELEMENTS 1269
3.3.6.6.9.8.8 LIMITATIONS 1269
3.3.6.6.9.8.9 LLCSC DESIGN 1269
3.3.6.6.9.8.10 UNIT DESIGN 1269
3.3.6.6.9.8.10.1 UPDATE_COEFFICIENTS (SUBPROGRAM BODY) UNIT DESIGN 1269
3.3.6.6.9.8.10.1.1 REQUIREMENTS ALLOCATION 1269
3.3.6.6.9.8.10.1.2 LOCAL ENTITIES DESIGN 1269
3.3.6.6.9.8.10.1.3 INPUT/OUTPUT 1269
3.3.6.6.9.8.10.1.4 LOCAL DATA 1270
3.3.6.6.9.8.10.1.5 PROCESS CONTROL 1270
3.3.6.6.9.8.10.1.6 PROCESSING 1270
3.3.6.6.9.8.10.1.7 UTILIZATION OF OTHER ELEMENTS 1270
3.3.6.6.9.8.10.1.8 LIMITATIONS 1270
3.3.6.6.9.8.10.2 FILTER (SUBPROGRAM BODY) UNIT DESIGN 1270
3.3.6.6.9.8.10.2.1 REQUIREMENTS ALLOCATION 1271

3.3.6.6.9.8.10.2.2 LOCAL ENTITIES DESIGN 1271
3.3.6.6.9.8.10.2.3 INPUT/OUTPUT 1271
3.3.6.6.9.8.10.2.4 LOCAL DATA 1272
3.3.6.6.9.8.10.2.5 PROCESS CONTROL 1272
3.3.6.6.9.8.10.2.6 PROCESSING 1272
3.3.6.6.9.8.10.2.7 UTILIZATION OF OTHER ELEMENTS 1272
3.3.6.6.9.8.10.2.8 LIMITATIONS 1272
3.3.6.6.9.8.10.3 REINITIALIZE (SUBPROGRAM BODY) UNIT DESIGN 1273
3.3.6.6.9.8.10.3.1 REQUIREMENTS ALLOCATION 1273
3.3.6.6.9.8.10.3.2 LOCAL ENTITIES DESIGN 1273
3.3.6.6.9.8.10.3.3 INPUT/OUTPUT 1273
3.3.6.6.9.8.10.3.4 LOCAL DATA 1273
3.3.6.6.9.8.10.3.5 PROCESS CONTROL 1274
3.3.6.6.9.8.10.3.6 PROCESSING 1274
3.3.6.6.9.8.10.3.7 UTILIZATION OF OTHER ELEMENTS 1274
3.3.6.6.9.8.10.3.8 LIMITATIONS 1274
3.3.6.6.9.9 SECOND_ORDER_FILTER (PACKAGE BODY), PACKAGE DESIGN (CATALOG #P90-0) 1274
3.3.6.6.9.9.1 REQUIREMENTS ALLOCATION 1274
3.3.6.6.9.9.2 LOCAL ENTITIES DESIGN 1275
3.3.6.6.9.9.3 INPUT/OUTPUT 1275
3.3.6.6.9.9.4 LOCAL DATA 1276
3.3.6.6.9.9.5 PROCESS CONTROL 1276
3.3.6.6.9.9.6 PROCESSING 1276
3.3.6.6.9.9.7 UTILIZATION OF OTHER ELEMENTS 1277
3.3.6.6.9.9.8 LIMITATIONS 1277
3.3.6.6.9.9.9 LLCSC DESIGN 1277
3.3.6.6.9.9.10 UNIT DESIGN 1277
3.3.6.6.9.9.10.1 REDEFINE_COEFFICIENTS (SUBPROGRAM BODY) UNIT DESIGN 1277
3.3.6.6.9.9.10.1.1 REQUIREMENTS ALLOCATION 1277
3.3.6.6.9.9.10.1.2 LOCAL ENTITIES DESIGN 1277
3.3.6.6.9.9.10.1.3 INPUT/OUTPUT 1277
3.3.6.6.9.9.10.1.4 LOCAL DATA 1278
3.3.6.6.9.9.10.1.5 PROCESS CONTROL 1278
3.3.6.6.9.9.10.1.6 PROCESSING 1278
3.3.6.6.9.9.10.1.7 UTILIZATION OF OTHER ELEMENTS 1279
3.3.6.6.9.9.10.1.8 LIMITATIONS 1279
3.3.6.6.9.9.10.2 FILTER (SUBPROGRAM BODY) UNIT DESIGN 1279
3.3.6.6.9.9.10.2.1 REQUIREMENTS ALLOCATION 1279
3.3.6.6.9.9.10.2.2 LOCAL ENTITIES DESIGN 1279
3.3.6.6.9.9.10.2.3 INPUT/OUTPUT 1279
3.3.6.6.9.9.10.2.4 LOCAL DATA 1280
3.3.6.6.9.9.10.2.5 PROCESS CONTROL 1281
3.3.6.6.9.9.10.2.6 PROCESSING 1281
3.3.6.6.9.9.10.2.7 UTILIZATION OF OTHER ELEMENTS 1281
3.3.6.6.9.9.10.2.8 LIMITATIONS 1281
3.3.6.6.9.9.10.3 REINITIALIZE (SUBPROGRAM BODY) UNIT DESIGN 1281
3.3.6.6.9.9.10.3.1 REQUIREMENTS ALLOCATION 1282
3.3.6.6.9.9.10.3.2 LOCAL ENTITIES DESIGN 1282
3.3.6.6.9.9.10.3.3 INPUT/OUTPUT 1282
3.3.6.6.9.9.10.3.4 LOCAL DATA 1282
3.3.6.6.9.9.10.3.5 PROCESS CONTROL 1282
3.3.6.6.9.9.10.3.6 PROCESSING 1283
3.3.6.6.9.9.10.3.7 UTILIZATION OF OTHER ELEMENTS 1283

3.3.6.6.9.9.10.3.8 LIMITATIONS 1283
3.3.6.6.9.10 TUSTIN INTEGRATOR WITH LIMIT (PACKAGE BODY) PACKAGE DESIGN (CATALOG #P91-0) 1283
3.3.6.6.9.10.1 REQUIREMENTS ALLOCATION 1283
3.3.6.6.9.10.2 LOCAL ENTITIES DESIGN 1283
3.3.6.6.9.10.3 INPUT/OUTPUT 1284
3.3.6.6.9.10.4 LOCAL DATA 1285
3.3.6.6.9.10.5 PROCESS CONTROL 1285
3.3.6.6.9.10.6 PROCESSING 1285
3.3.6.6.9.10.7 UTILIZATION OF OTHER ELEMENTS 1286
3.3.6.6.9.10.8 LIMITATIONS 1287
3.3.6.6.9.10.9 LLCSC DESIGN 1287
3.3.6.6.9.10.10 UNIT DESIGN 1287
3.3.6.6.9.10.10.1 UPDATE_LIMIT (SUBPROGRAM BODY) UNIT DESIGN 1287
3.3.6.6.9.10.10.1.1 REQUIREMENTS ALLOCATION 1287
3.3.6.6.9.10.10.1.2 LOCAL ENTITIES DESIGN 1287
3.3.6.6.9.10.10.1.3 INPUT/OUTPUT 1287
3.3.6.6.9.10.10.1.4 LOCAL DATA 1288
3.3.6.6.9.10.10.1.5 PROCESS CONTROL 1288
3.3.6.6.9.10.10.1.6 PROCESSING 1288
3.3.6.6.9.10.10.1.7 UTILIZATION OF OTHER ELEMENTS 1288
3.3.6.6.9.10.10.1.8 LIMITATIONS 1288
3.3.6.6.9.10.10.2 UPDATE_GAIN (SUBPROGRAM BODY) UNIT DESIGN 1. 288
3.3.6.6.9.10.10.2.1 REQUIREMENTS ALLOCATION 1288
3.3.6.6.9.10.10.2.2 LOCAL ENTITIES DESIGN 1289
3.3.6.6.9.10.10.2.3 INPUT/OUTPUT 1289
3.3.6.6.9.10.10.2.4 LOCAL DATA 1289
3.3.6.6.9.10.10.2.5 PROCESS CONTROL 1289
3.3.6.6.9.10.10.2.6 PROCESSING 1289
3.3.6.6.9.10.10.2.7 UTILIZATION OF OTHER ELEMENTS 1290
3.3.6.6.9.10.10.2.8 LIMITATIONS 1290
3.3.6.6.9.10.10.3 INTEGRATE (SUBPROGRAM BODY) UNIT DESIGN 1. 290
3.3.6.6.9.10.10.3.1 REQUIREMENTS ALLOCATION 1290
3.3.6.6.9.10.10.3.2 LOCAL ENTITIES DESIGN 1290
3.3.6.6.9.10.10.3.3 INPUT/OUTPUT 1290
3.3.6.6.9.10.10.3.4 LOCAL DATA 1291
3.3.6.6.9.10.10.3.5 PROCESS CONTROL 1291
3.3.6.6.9.10.10.3.6 PROCESSING 1291
3.3.6.6.9.10.10.3.7 UTILIZATION OF OTHER ELEMENTS 1292
3.3.6.6.9.10.10.3.8 LIMITATIONS 1292
3.3.6.6.9.10.10.4 RESET (INTEGRATOR_STATE) (SUBPROGRAM BODY) UNIT DESIGN 1292
3.3.6.6.9.10.10.4.1 REQUIREMENTS ALLOCATION 1293
3.3.6.6.9.10.10.4.2 LOCAL ENTITIES DESIGN 1293
3.3.6.6.9.10.10.4.3 INPUT/OUTPUT 1293
3.3.6.6.9.10.10.4.4 LOCAL DATA 1293
3.3.6.6.9.10.10.4.5 PROCESS CONTROL 1293
3.3.6.6.9.10.10.4.6 PROCESSING 1293
3.3.6.6.9.10.10.4.7 UTILIZATION OF OTHER ELEMENTS 1294
3.3.6.6.9.10.10.4.8 LIMITATIONS 1294
3.3.6.6.9.10.10.5 LIMIT_FLAG_SETTING (SUBPROGRAM BODY) UNIT DESIGN 1294
3.3.6.6.9.10.10.5.1 REQUIREMENTS ALLOCATION 1294
3.3.6.6.9.10.10.5.2 LOCAL ENTITIES DESIGN 1295
3.3.6.6.9.10.10.5.3 INPUT/OUTPUT 1295
3.3.6.6.9.10.10.5.4 LOCAL DATA 1295

3.3.6.6.9.10.10.5.5 PROCESS CONTROL 1295
3.3.6.6.9.10.10.5.6 PROCESSING 1295
3.3.6.6.9.10.10.5.7 UTILIZATION OF OTHER ELEMENTS 1295
3.3.6.6.9.10.10.5.8 LIMITATIONS 1295
3.3.6.6.9.11 TUSTIN_INTEGRATOR_WITH_ASYMMETRIC_LIMIT (PACKAGE BODY) PACKAGE DESIGN (CATALOG #P1054-0) 1295
3.3.6.6.9.11.1 REQUIREMENTS ALLOCATION 1296
3.3.6.6.9.11.2 LOCAL ENTITIES DESIGN 1296
3.3.6.6.9.11.3 INPUT/OUTPUT 1296
3.3.6.6.9.11.4 LOCAL DATA 1297
3.3.6.6.9.11.5 PROCESS CONTROL 1298
3.3.6.6.9.11.6 PROCESSING 1298
3.3.6.6.9.11.7 UTILIZATION OF OTHER ELEMENTS 1299
3.3.6.6.9.11.8 LIMITATIONS 1300
3.3.6.6.9.11.9 LLCSC DESIGN 1300
3.3.6.6.9.11.10 UNIT DESIGN 1300
3.3.6.6.9.11.10.1 UPDATE_LIMITS (SUBPROGRAM BODY) UNIT DESIGN 1300
3.3.6.6.9.11.10.1.1 REQUIREMENTS ALLOCATION 1300
3.3.6.6.9.11.10.1.2 LOCAL ENTITIES DESIGN 1300
3.3.6.6.9.11.10.1.3 INPUT/OUTPUT 1300
3.3.6.6.9.11.10.1.4 LOCAL DATA 1301
3.3.6.6.9.11.10.1.5 PROCESS CONTROL 1301
3.3.6.6.9.11.10.1.6 PROCESSING 1301
3.3.6.6.9.11.10.1.7 UTILIZATION OF OTHER ELEMENTS 1301
3.3.6.6.9.11.10.1.8 LIMITATIONS 1301
3.3.6.6.9.11.10.2 UPDATE_GAIN (SUBPROGRAM BODY) UNIT DESIGN 1. 301
3.3.6.6.9.11.10.2.1 REQUIREMENTS ALLOCATION 1301
3.3.6.6.9.11.10.2.2 LOCAL ENTITIES DESIGN 1301
3.3.6.6.9.11.10.2.3 INPUT/OUTPUT 1302
3.3.6.6.9.11.10.2.4 LOCAL DATA 1302
3.3.6.6.9.11.10.2.5 PROCESS CONTROL 1302
3.3.6.6.9.11.10.2.6 PROCESSING 1302
3.3.6.6.9.11.10.2.7 UTILIZATION OF OTHER ELEMENTS 1303
3.3.6.6.9.11.10.2.8 LIMITATIONS 1303
3.3.6.6.9.11.10.3 INTEGRATE (SUBPROGRAM BODY) UNIT DESIGN 1. 303
3.3.6.6.9.11.10.3.1 REQUIREMENTS ALLOCATION 1303
3.3.6.6.9.11.10.3.2 LOCAL ENTITIES DESIGN 1303
3.3.6.6.9.11.10.3.3 INPUT/OUTPUT 1303
3.3.6.6.9.11.10.3.4 LOCAL DATA 1304
3.3.6.6.9.11.10.3.5 PROCESS CONTROL 1304
3.3.6.6.9.11.10.3.6 PROCESSING 1304
3.3.6.6.9.11.10.3.7 UTILIZATION OF OTHER ELEMENTS 1305
3.3.6.6.9.11.10.3.8 LIMITATIONS 1306
3.3.6.6.9.11.10.4 RESET (SUBPROGRAM BODY) UNIT DESIGN 1306
3.3.6.6.9.11.10.4.1 REQUIREMENTS ALLOCATION 1306
3.3.6.6.9.11.10.4.2 LOCAL ENTITIES DESIGN 1306
3.3.6.6.9.11.10.4.3 INPUT/OUTPUT 1306
3.3.6.6.9.11.10.4.4 LOCAL DATA 1306
3.3.6.6.9.11.10.4.5 PROCESS CONTROL 1306
3.3.6.6.9.11.10.4.6 PROCESSING 1307
3.3.6.6.9.11.10.4.7 UTILIZATION OF OTHER ELEMENTS 1307
3.3.6.6.9.11.10.4.8 LIMITATIONS 1307
3.3.6.6.9.11.10.5 LIMIT_FLAG_SETTING (SUBPROGRAM BODY) UNIT DESIGN 1308
3.3.6.6.9.11.10.5.1 REQUIREMENTS ALLOCATION 1308

3.3.6.6.9.11.10.5.2 LOCAL ENTITIES DESIGN 1308
3.3.6.6.9.11.10.5.3 INPUT/OUTPUT 1308
3.3.6.6.9.11.10.5.4 LOCAL DATA 1308
3.3.6.6.9.11.10.5.5 PROCESS CONTROL 1308
3.3.6.6.9.11.10.5.6 PROCESSING 1308
3.3.6.6.9.11.10.5.7 UTILIZATION OF OTHER ELEMENTS 1308
3.3.6.6.9.11.10.5.8 LIMITATIONS 1308
3.3.6.6.10 UNIT DESIGN 1309

3.3.6.7 GENERAL PURPOSE MATH (BODY) TLCSC P687 (CATALOG #P28-0) 1331
3.3.6.7.1 REQUIREMENTS ALLOCATION 1331
3.3.6.7.2 LOCAL ENTITIES DESIGN 1331
3.3.6.7.3 INPUT/OUTPUT 1331
3.3.6.7.4 LOCAL DATA 1331
3.3.6.7.5 PROCESS CONTROL 1332
3.3.6.7.6 PROCESSING 1332
3.3.6.7.7 UTILIZATION OF OTHER ELEMENTS 1333
3.3.6.7.8 LIMITATIONS 1333
3.3.6.7.9 LLCSC DESIGN 1333
3.3.6.7.9.1 LOOKUP_TABLE_EVEN_SPACING (BODY) PACKAGE DESIGN (CATALOG #P29-0) 1333
3.3.6.7.9.1.1 REQUIREMENTS ALLOCATION 1333
3.3.6.7.9.1.2 LOCAL ENTITIES DESIGN 1333
3.3.6.7.9.1.3 INPUT/OUTPUT 1333
3.3.6.7.9.1.4 LOCAL DATA 1334
3.3.6.7.9.1.5 PROCESS CONTROL 1334
3.3.6.7.9.1.6 PROCESSING 1334
3.3.6.7.9.1.7 UTILIZATION OF OTHER ELEMENTS 1334
3.3.6.7.9.1.8 LIMITATIONS 1334
3.3.6.7.9.1.9 LLCSC DESIGN 1335
3.3.6.7.9.1.10 UNIT DESIGN 1335
3.3.6.7.9.1.10.1 INITIALIZE UNIT DESIGN 1335
3.3.6.7.9.1.10.1.1 REQUIREMENTS ALLOCATION 1335
3.3.6.7.9.1.10.1.2 LOCAL ENTITIES DESIGN 1335
3.3.6.7.9.1.10.1.3 INPUT/OUTPUT 1335
3.3.6.7.9.1.10.1.4 LOCAL DATA 1335
3.3.6.7.9.1.10.1.5 PROCESS CONTROL 1335
3.3.6.7.9.1.10.1.6 PROCESSING 1335
3.3.6.7.9.1.10.1.7 UTILIZATION OF OTHER ELEMENTS 1336
3.3.6.7.9.1.10.1.8 LIMITATIONS 1336
3.3.6.7.9.1.10.2 LOOKUP UNIT DESIGN 1336
3.3.6.7.9.1.10.2.1 REQUIREMENTS ALLOCATION 1336
3.3.6.7.9.1.10.2.2 LOCAL ENTITIES DESIGN 1336
3.3.6.7.9.1.10.2.3 INPUT/OUTPUT 1336
3.3.6.7.9.1.10.2.4 LOCAL DATA 1337
3.3.6.7.9.1.10.2.5 PROCESS CONTROL 1337
3.3.6.7.9.1.10.2.6 PROCESSING 1337
3.3.6.7.9.1.10.2.7 UTILIZATION OF OTHER ELEMENTS 1338
3.3.6.7.9.1.10.2.8 LIMITATIONS 1338
3.3.6.7.9.1.10.3 LOOKUP UNIT DESIGN 1338
3.3.6.7.9.1.10.3.1 REQUIREMENTS ALLOCATION 1338
3.3.6.7.9.1.10.3.2 LOCAL ENTITIES DESIGN 1338
3.3.6.7.9.1.10.3.3 INPUT/OUTPUT 1338
3.3.6.7.9.1.10.3.4 LOCAL DATA 1339
3.3.6.7.9.1.10.3.5 PROCESS CONTROL 1339
3.3.6.7.9.1.10.3.6 PROCESSING 1339

3.3.6.7.9.1.10.3.7 UTILIZATION OF OTHER ELEMENTS 1341
3.3.6.7.9.1.10.3.8 LIMITATIONS 1341
3.3.6.7.9.2 LOOKUP_TABLE_UNEVEN_SPACING (BODY) PACKAGE DESIGN (CATALOG #P30-0) 1341
3.3.6.7.9.2.1 REQUIREMENTS ALLOCATION 1341
3.3.6.7.9.2.2 LOCAL ENTITIES DESIGN 1341
3.3.6.7.9.2.3 INPUT/OUTPUT 1341
3.3.6.7.9.2.4 LOCAL DATA 1342
3.3.6.7.9.2.5 PROCESS CONTROL 1342
3.3.6.7.9.2.6 PROCESSING 1342
3.3.6.7.9.2.7 UTILIZATION OF OTHER ELEMENTS 1342
3.3.6.7.9.2.8 LIMITATIONS 1342
3.3.6.7.9.2.9 LLCSC DESIGN 1342
3.3.6.7.9.2.10 UNIT DESIGN 1342
3.3.6.7.9.2.10.1 INITIALIZE UNIT DESIGN 1342
3.3.6.7.9.2.10.1.1 REQUIREMENTS ALLOCATION 1343
3.3.6.7.9.2.10.1.2 LOCAL ENTITIES DESIGN 1343
3.3.6.7.9.2.10.1.3 INPUT/OUTPUT 1343
3.3.6.7.9.2.10.1.4 LOCAL DATA 1343
3.3.6.7.9.2.10.1.5 PROCESS CONTROL 1343
3.3.6.7.9.2.10.1.6 PROCESSING 1343
3.3.6.7.9.2.10.1.7 UTILIZATION OF OTHER ELEMENTS 1344
3.3.6.7.9.2.10.1.8 LIMITATIONS 1344
3.3.6.7.9.2.10.2 LOOKUP UNIT DESIGN 1344
3.3.6.7.9.2.10.2.1 REQUIREMENTS ALLOCATION 1344
3.3.6.7.9.2.10.2.2 LOCAL ENTITIES DESIGN 1344
3.3.6.7.9.2.10.2.3 INPUT/OUTPUT 1344
3.3.6.7.9.2.10.2.4 LOCAL DATA 1344
3.3.6.7.9.2.10.2.5 PROCESS CONTROL 1345
3.3.6.7.9.2.10.2.6 PROCESSING 1345
3.3.6.7.9.2.10.2.7 UTILIZATION OF OTHER ELEMENTS 1346
3.3.6.7.9.2.10.2.8 LIMITATIONS 1346
3.3.6.7.9.2.10.3 LOOKUP UNIT DESIGN 1346
3.3.6.7.9.2.10.3.1 REQUIREMENTS ALLOCATION 1346
3.3.6.7.9.2.10.3.2 LOCAL ENTITIES DESIGN 1346
3.3.6.7.9.2.10.3.3 INPUT/OUTPUT 1346
3.3.6.7.9.2.10.3.4 LOCAL DATA 1347
3.3.6.7.9.2.10.3.5 PROCESS CONTROL 1347
3.3.6.7.9.2.10.3.6 PROCESSING 1347
3.3.6.7.9.2.10.3.7 UTILIZATION OF OTHER ELEMENTS 1348
3.3.6.7.9.2.10.3.8 LIMITATIONS 1348
3.3.6.7.9.3 INCREMENTOR (BODY) PACKAGE DESIGN (CATALOG #P31-0) 1348
3.3.6.7.9.3.1 REQUIREMENTS ALLOCATION 1348
3.3.6.7.9.3.2 LOCAL ENTITIES DESIGN 1349
3.3.6.7.9.3.3 INPUT/OUTPUT 1349
3.3.6.7.9.3.4 LOCAL DATA 1349
3.3.6.7.9.3.5 PROCESS CONTROL 1349
3.3.6.7.9.3.6 PROCESSING 1349
3.3.6.7.9.3.7 UTILIZATION OF OTHER ELEMENTS 1350
3.3.6.7.9.3.8 LIMITATIONS 1350
3.3.6.7.9.3.9 LLCSC DESIGN 1350
3.3.6.7.9.3.10 UNIT DESIGN 1350
3.3.6.7.9.3.10.1 REINITIALIZE UNIT DESIGN 1350
3.3.6.7.9.3.10.1.1 REQUIREMENTS ALLOCATION 1350
3.3.6.7.9.3.10.1.2 LOCAL ENTITIES DESIGN 1350
3.3.6.7.9.3.10.1.3 INPUT/OUTPUT 1350

3.3.6.7.9.3.10.1.4 LOCAL DATA 1351
3.3.6.7.9.3.10.1.5 PROCESS CONTROL 1351
3.3.6.7.9.3.10.1.6 PROCESSING 1351
3.3.6.7.9.3.10.1.7 UTILIZATION OF OTHER ELEMENTS 1351
3.3.6.7.9.3.10.1.8 LIMITATIONS 1351
3.3.6.7.9.3.10.2 INCREMENT UNIT DESIGN 1351
3.3.6.7.9.3.10.2.1 REQUIREMENTS ALLOCATION 1351
3.3.6.7.9.3.10.2.2 LOCAL ENTITIES DESIGN 1351
3.3.6.7.9.3.10.2.3 INPUT/OUTPUT 1351
3.3.6.7.9.3.10.2.4 LOCAL DATA 1352
3.3.6.7.9.3.10.2.5 PROCESS CONTROL 1352
3.3.6.7.9.3.10.2.6 PROCESSING 1352
3.3.6.7.9.3.10.2.7 UTILIZATION OF OTHER ELEMENTS 1352
3.3.6.7.9.3.10.2.8 LIMITATIONS 1353
3.3.6.7.9.4 DECREMENTOR (BODY) PACKAGE DESIGN (CATALOG #P32-0) 1353
3.3.6.7.9.4.1 REQUIREMENTS ALLOCATION 1353
3.3.6.7.9.4.2 LOCAL ENTITIES DESIGN 1353
3.3.6.7.9.4.3 INPUT/OUTPUT 1353
3.3.6.7.9.4.4 LOCAL DATA 1354
3.3.6.7.9.4.5 PROCESS CONTROL 1354
3.3.6.7.9.4.6 PROCESSING 1354
3.3.6.7.9.4.7 UTILIZATION OF OTHER ELEMENTS 1354
3.3.6.7.9.4.8 LIMITATIONS 1355
3.3.6.7.9.4.9 LLCSC DESIGN 1355
3.3.6.7.9.4.10 UNIT DESIGN 1355
3.3.6.7.9.4.10.1 REINITIALIZE UNIT DESIGN 1355
3.3.6.7.9.4.10.1.1 REQUIREMENTS ALLOCATION 1355
3.3.6.7.9.4.10.1.2 LOCAL ENTITIES DESIGN 1355
3.3.6.7.9.4.10.1.3 INPUT/OUTPUT 1355
3.3.6.7.9.4.10.1.4 LOCAL DATA 1355
3.3.6.7.9.4.10.1.5 PROCESS CONTROL 1355
3.3.6.7.9.4.10.1.6 PROCESSING 1355
3.3.6.7.9.4.10.1.7 UTILIZATION OF OTHER ELEMENTS 1356
3.3.6.7.9.4.10.1.8 LIMITATIONS 1356
3.3.6.7.9.4.10.2 DECREMENT UNIT DESIGN 1356
3.3.6.7.9.4.10.2.1 REQUIREMENTS ALLOCATION 1356
3.3.6.7.9.4.10.2.2 LOCAL ENTITIES DESIGN 1357
3.3.6.7.9.4.10.2.3 INPUT/OUTPUT 1357
3.3.6.7.9.4.10.2.4 LOCAL DATA 1357
3.3.6.7.9.4.10.2.5 PROCESS CONTROL 1357
3.3.6.7.9.4.10.2.6 PROCESSING 1357
3.3.6.7.9.4.10.2.7 UTILIZATION OF OTHER ELEMENTS 1357
3.3.6.7.9.4.10.2.8 LIMITATIONS 1357
3.3.6.7.9.5 RUNNING AVERAGE (BODY) PACKAGE DESIGN (CATALOG #P33-0) 1358
3.3.6.7.9.5.1 REQUIREMENTS ALLOCATION 1358
3.3.6.7.9.5.2 LOCAL ENTITIES DESIGN 1358
3.3.6.7.9.5.3 INPUT/OUTPUT 1358
3.3.6.7.9.5.4 LOCAL DATA 1359
3.3.6.7.9.5.5 PROCESS CONTROL 1359
3.3.6.7.9.5.6 PROCESSING 1359
3.3.6.7.9.5.7 UTILIZATION OF OTHER ELEMENTS 1359
3.3.6.7.9.5.8 LIMITATIONS 1359
3.3.6.7.9.5.9 LLCSC DESIGN 1359
3.3.6.7.9.5.10 UNIT DESIGN 1359
3.3.6.7.9.5.10.1 REINITIALIZE UNIT DESIGN 1360

3.3.6.7.9.5.10.1.1 REQUIREMENTS ALLOCATION 1360
3.3.6.7.9.5.10.1.2 LOCAL ENTITIES DESIGN 1360
3.3.6.7.9.5.10.1.3 INPUT/OUTPUT 1360
3.3.6.7.9.5.10.1.4 LOCAL DATA 1360
3.3.6.7.9.5.10.1.5 PROCESS CONTROL 1360
3.3.6.7.9.5.10.1.6 PROCESSING 1360
3.3.6.7.9.5.10.1.7 UTILIZATION OF OTHER ELEMENTS 1361
3.3.6.7.9.5.10.1.8 LIMITATIONS 1361
3.3.6.7.9.5.10.2 REINITIALIZE UNIT DESIGN 1361
3.3.6.7.9.5.10.2.1 REQUIREMENTS ALLOCATION 1361
3.3.6.7.9.5.10.2.2 LOCAL ENTITIES DESIGN 1361
3.3.6.7.9.5.10.2.3 INPUT/OUTPUT 1361
3.3.6.7.9.5.10.2.4 LOCAL DATA 1361
3.3.6.7.9.5.10.2.5 PROCESS CONTROL 1361
3.3.6.7.9.5.10.2.6 PROCESSING 1361
3.3.6.7.9.5.10.2.7 UTILIZATION OF OTHER ELEMENTS 1362
3.3.6.7.9.5.10.2.8 LIMITATIONS 1362
3.3.6.7.9.5.10.3 CURRENT AVERAGE UNIT DESIGN 1362
3.3.6.7.9.5.10.3.1 REQUIREMENTS ALLOCATION 1362
3.3.6.7.9.5.10.3.2 LOCAL ENTITIES DESIGN 1362
3.3.6.7.9.5.10.3.3 INPUT/OUTPUT 1362
3.3.6.7.9.5.10.3.4 LOCAL DATA 1362
3.3.6.7.9.5.10.3.5 PROCESS CONTROL 1362
3.3.6.7.9.5.10.3.6 PROCESSING 1362
3.3.6.7.9.5.10.3.7 UTILIZATION OF OTHER ELEMENTS 1363
3.3.6.7.9.5.10.3.8 LIMITATIONS 1363
3.3.6.7.9.6 ACCUMULATOR PACKAGE DESIGN (CATALOG #P34-0) 1. 363
3.3.6.7.9.6.1 REQUIREMENTS ALLOCATION 1363
3.3.6.7.9.6.2 LOCAL ENTITIES DESIGN 1363
3.3.6.7.9.6.3 INPUT/OUTPUT 1363
3.3.6.7.9.6.4 LOCAL DATA 1364
3.3.6.7.9.6.5 PROCESS CONTROL 1364
3.3.6.7.9.6.6 PROCESSING 1364
3.3.6.7.9.6.7 UTILIZATION OF OTHER ELEMENTS 1364
3.3.6.7.9.6.8 LIMITATIONS 1364
3.3.6.7.9.6.9 LLCSC DESIGN 1365
3.3.6.7.9.6.10 UNIT DESIGN 1365
3.3.6.7.9.6.10.1 REINITIALIZE UNIT DESIGN 1365
3.3.6.7.9.6.10.1.1 REQUIREMENTS ALLOCATION 1365
3.3.6.7.9.6.10.1.2 LOCAL ENTITIES DESIGN 1365
3.3.6.7.9.6.10.1.3 INPUT/OUTPUT 1365
3.3.6.7.9.6.10.1.4 LOCAL DATA 1365
3.3.6.7.9.6.10.1.5 PROCESS CONTROL 1365
3.3.6.7.9.6.10.1.6 PROCESSING 1365
3.3.6.7.9.6.10.1.7 UTILIZATION OF OTHER ELEMENTS 1366
3.3.6.7.9.6.10.1.8 LIMITATIONS 1366
3.3.6.7.9.6.10.2 ACCUMULATE UNIT DESIGN 1366
3.3.6.7.9.6.10.2.1 REQUIREMENTS ALLOCATION 1366
3.3.6.7.9.6.10.2.2 LOCAL ENTITIES DESIGN 1366
3.3.6.7.9.6.10.2.3 INPUT/OUTPUT 1366
3.3.6.7.9.6.10.2.4 LOCAL DATA 1366
3.3.6.7.9.6.10.2.5 PROCESS CONTROL 1366
3.3.6.7.9.6.10.2.6 PROCESSING 1366
3.3.6.7.9.6.10.2.7 UTILIZATION OF OTHER ELEMENTS 1367
3.3.6.7.9.6.10.2.8 LIMITATIONS 1367
3.3.6.7.9.6.10.3 ACCUMULATE UNIT DESIGN 1367
3.3.6.7.9.6.10.3.1 REQUIREMENTS ALLOCATION 1367

3.3.6.7.9.6.10.3.2 LOCAL ENTITIES DESIGN 1367
3.3.6.7.9.6.10.3.3 INPUT/OUTPUT 1367
3.3.6.7.9.6.10.3.4 LOCAL DATA 1367
3.3.6.7.9.6.10.3.5 PROCESS CONTROL 1367
3.3.6.7.9.6.10.3.6 PROCESSING 1368
3.3.6.7.9.6.10.3.7 UTILIZATION OF OTHER ELEMENTS 1368
3.3.6.7.9.6.10.3.8 LIMITATIONS 1368
3.3.6.7.9.6.10.4 RETRIEVE UNIT DESIGN 1368
3.3.6.7.9.6.10.4.1 REQUIREMENTS ALLOCATION 1368
3.3.6.7.9.6.10.4.2 LOCAL ENTITIES DESIGN 1368
3.3.6.7.9.6.10.4.3 INPUT/OUTPUT 1368
3.3.6.7.9.6.10.4.4 LOCAL DATA 1369
3.3.6.7.9.6.10.4.5 PROCESS CONTROL 1369
3.3.6.7.9.6.10.4.6 PROCESSING 1369
3.3.6.7.9.6.10.4.7 UTILIZATION OF OTHER ELEMENTS 1369
3.3.6.7.9.6.10.4.8 LIMITATIONS 1369
3.3.6.7.9.7 CHANGE ACCUMULATOR PACKAGE DESIGN (CATALOG #P36-0) 1369
3.3.6.7.9.7.1 REQUIREMENTS ALLOCATION 1369
3.3.6.7.9.7.2 LOCAL ENTITIES DESIGN 1369
3.3.6.7.9.7.3 INPUT/OUTPUT 1370
3.3.6.7.9.7.4 LOCAL DATA 1370
3.3.6.7.9.7.5 PROCESS CONTROL 1370
3.3.6.7.9.7.6 PROCESSING 1370
3.3.6.7.9.7.7 UTILIZATION OF OTHER ELEMENTS 1371
3.3.6.7.9.7.8 LIMITATIONS 1371
3.3.6.7.9.7.9 LLCSC DESIGN 1371
3.3.6.7.9.7.10 UNIT DESIGN 1371
3.3.6.7.9.7.10.1 REINITIALIZE UNIT DESIGN 1371
3.3.6.7.9.7.10.1.1 REQUIREMENTS ALLOCATION 1371
3.3.6.7.9.7.10.1.2 LOCAL ENTITIES DESIGN 1371
3.3.6.7.9.7.10.1.3 INPUT/OUTPUT 1371
3.3.6.7.9.7.10.1.4 LOCAL DATA 1372
3.3.6.7.9.7.10.1.5 PROCESS CONTROL 1372
3.3.6.7.9.7.10.1.6 PROCESSING 1372
3.3.6.7.9.7.10.1.7 UTILIZATION OF OTHER ELEMENTS 1372
3.3.6.7.9.7.10.1.8 LIMITATIONS 1372
3.3.6.7.9.7.10.2 REINITIALIZE UNIT DESIGN 1372
3.3.6.7.9.7.10.2.1 REQUIREMENTS ALLOCATION 1372
3.3.6.7.9.7.10.2.2 LOCAL ENTITIES DESIGN 1372
3.3.6.7.9.7.10.2.3 INPUT/OUTPUT 1372
3.3.6.7.9.7.10.2.4 LOCAL DATA 1373
3.3.6.7.9.7.10.2.5 PROCESS CONTROL 1373
3.3.6.7.9.7.10.2.6 PROCESSING 1373
3.3.6.7.9.7.10.2.7 UTILIZATION OF OTHER ELEMENTS 1373
3.3.6.7.9.7.10.2.8 LIMITATIONS 1373
3.3.6.7.9.7.10.3 ACCUMULATE CHANGE UNIT DESIGN 1373
3.3.6.7.9.7.10.3.1 REQUIREMENTS ALLOCATION 1373
3.3.6.7.9.7.10.3.2 LOCAL ENTITIES DESIGN 1374
3.3.6.7.9.7.10.3.3 INPUT/OUTPUT 1374
3.3.6.7.9.7.10.3.4 LOCAL DATA 1374
3.3.6.7.9.7.10.3.5 PROCESS CONTROL 1374
3.3.6.7.9.7.10.3.6 PROCESSING 1374
3.3.6.7.9.7.10.3.7 UTILIZATION OF OTHER ELEMENTS 1374
3.3.6.7.9.7.10.3.8 LIMITATIONS 1374
3.3.6.7.9.7.10.4 ACCUMULATE CHANGE UNIT DESIGN 1375
3.3.6.7.9.7.10.4.1 REQUIREMENTS ALLOCATION 1375

3.3.6.7.9.7.10.4.2 LOCAL ENTITIES DESIGN 1375
3.3.6.7.9.7.10.4.3 INPUT/OUTPUT 1375
3.3.6.7.9.7.10.4.4 LOCAL DATA 1375
3.3.6.7.9.7.10.4.5 PROCESS CONTROL 1375
3.3.6.7.9.7.10.4.6 PROCESSING 1375
3.3.6.7.9.7.10.4.7 UTILIZATION OF OTHER ELEMENTS 1376
3.3.6.7.9.7.10.4.8 LIMITATIONS 1376
3.3.6.7.9.7.10.5 RETRIEVE_ACCUMULATION UNIT DESIGN 1376
3.3.6.7.9.7.10.5.1 REQUIREMENTS ALLOCATION 1376
3.3.6.7.9.7.10.5.2 LOCAL ENTITIES DESIGN 1376
3.3.6.7.9.7.10.5.3 INPUT/OUTPUT 1376
3.3.6.7.9.7.10.5.4 LOCAL DATA 1376
3.3.6.7.9.7.10.5.5 PROCESS CONTROL 1376
3.3.6.7.9.7.10.5.6 PROCESSING 1377
3.3.6.7.9.7.10.5.7 UTILIZATION OF OTHER ELEMENTS 1377
3.3.6.7.9.7.10.5.8 LIMITATIONS 1377
3.3.6.7.9.7.10.6 RETRIEVE_PREVIOUS_VALUE UNIT DESIGN 1377
3.3.6.7.9.7.10.6.1 REQUIREMENTS ALLOCATION 1377
3.3.6.7.9.7.10.6.2 LOCAL ENTITIES DESIGN 1377
3.3.6.7.9.7.10.6.3 INPUT/OUTPUT 1377
3.3.6.7.9.7.10.6.4 LOCAL DATA 1377
3.3.6.7.9.7.10.6.5 PROCESS CONTROL 1378
3.3.6.7.9.7.10.6.6 PROCESSING 1378
3.3.6.7.9.7.10.6.7 UTILIZATION OF OTHER ELEMENTS 1378
3.3.6.7.9.7.10.6.8 LIMITATIONS 1378
3.3.6.7.9.8 CHANGE_CALCULATOR PACKAGE DESIGN (CATALOG #P35-0) 1378
3.3.6.7.9.8.1 REQUIREMENTS ALLOCATION 1378
3.3.6.7.9.8.2 LOCAL ENTITIES DESIGN 1378
3.3.6.7.9.8.3 INPUT/OUTPUT 1378
3.3.6.7.9.8.4 LOCAL DATA 1379
3.3.6.7.9.8.5 PROCESS CONTROL 1379
3.3.6.7.9.8.6 PROCESSING 1379
3.3.6.7.9.8.7 UTILIZATION OF OTHER ELEMENTS 1379
3.3.6.7.9.8.8 LIMITATIONS 1379
3.3.6.7.9.8.9 LLCSC DESIGN 1380
3.3.6.7.9.8.10 UNIT DESIGN 1380
3.3.6.7.9.8.10.1 REINITIALIZE UNIT DESIGN 1380
3.3.6.7.9.8.10.1.1 REQUIREMENTS ALLOCATION 1380
3.3.6.7.9.8.10.1.2 LOCAL ENTITIES DESIGN 1380
3.3.6.7.9.8.10.1.3 INPUT/OUTPUT 1380
3.3.6.7.9.8.10.1.4 LOCAL DATA 1380
3.3.6.7.9.8.10.1.5 PROCESS CONTROL 1380
3.3.6.7.9.8.10.1.6 PROCESSING 1380
3.3.6.7.9.8.10.1.7 UTILIZATION OF OTHER ELEMENTS 1381
3.3.6.7.9.8.10.1.8 LIMITATIONS 1381
3.3.6.7.9.8.10.2 CHANGE UNIT DESIGN 1381
3.3.6.7.9.8.10.2.1 REQUIREMENTS ALLOCATION 1381
3.3.6.7.9.8.10.2.2 LOCAL ENTITIES DESIGN 1381
3.3.6.7.9.8.10.2.3 INPUT/OUTPUT 1381
3.3.6.7.9.8.10.2.4 LOCAL DATA 1381
3.3.6.7.9.8.10.2.5 PROCESS CONTROL 1382
3.3.6.7.9.8.10.2.6 PROCESSING 1382
3.3.6.7.9.8.10.2.7 UTILIZATION OF OTHER ELEMENTS 1382
3.3.6.7.9.8.10.2.8 LIMITATIONS 1382
3.3.6.7.9.8.10.3 RETRIEVE_VALUE UNIT DESIGN 1382
3.3.6.7.9.8.10.3.1 REQUIREMENTS ALLOCATION 1382

3.3.6.7.9.8.10.3.2 LOCAL ENTITIES DESIGN 1382
3.3.6.7.9.8.10.3.3 INPUT/OUTPUT 1383
3.3.6.7.9.8.10.3.4 LOCAL DATA 1383
3.3.6.7.9.8.10.3.5 PROCESS CONTROL 1383
3.3.6.7.9.8.10.3.6 PROCESSING 1383
3.3.6.7.9.8.10.3.7 UTILIZATION OF OTHER ELEMENTS 1383
3.3.6.7.9.8.10.3.8 LIMITATIONS 1383
3.3.6.7.9.9 INTEGRATOR (BODY) PACKAGE DESIGN (CATALOG #P37-0) 1383
3.3.6.7.9.9.1 REQUIREMENTS ALLOCATION 1384
3.3.6.7.9.9.2 LOCAL ENTITIES DESIGN 1384
3.3.6.7.9.9.3 INPUT/OUTPUT 1384
3.3.6.7.9.9.4 LOCAL DATA 1384
3.3.6.7.9.9.5 PROCESS CONTROL 1385
3.3.6.7.9.9.6 PROCESSING 1385
3.3.6.7.9.9.7 UTILIZATION OF OTHER ELEMENTS 1385
3.3.6.7.9.9.8 LIMITATIONS 1385
3.3.6.7.9.9.9 LLCSC DESIGN 1385
3.3.6.7.9.9.10 UNIT DESIGN 1385
3.3.6.7.9.9.10.1 REINITIALIZE UNIT DESIGN 1385
3.3.6.7.9.9.10.1.1 REQUIREMENTS ALLOCATION 1385
3.3.6.7.9.9.10.1.2 LOCAL ENTITIES DESIGN 1385
3.3.6.7.9.9.10.1.3 INPUT/OUTPUT 1386
3.3.6.7.9.9.10.1.4 LOCAL DATA 1386
3.3.6.7.9.9.10.1.5 PROCESS CONTROL 1386
3.3.6.7.9.9.10.1.6 PROCESSING 1386
3.3.6.7.9.9.10.1.7 UTILIZATION OF OTHER ELEMENTS 1386
3.3.6.7.9.9.10.1.8 LIMITATIONS 1386
3.3.6.7.9.9.10.2 UPDATE UNIT DESIGN 1386
3.3.6.7.9.9.10.2.1 REQUIREMENTS ALLOCATION 1387
3.3.6.7.9.9.10.2.2 LOCAL ENTITIES DESIGN 1387
3.3.6.7.9.9.10.2.3 INPUT/OUTPUT 1387
3.3.6.7.9.9.10.2.4 LOCAL DATA 1387
3.3.6.7.9.9.10.2.5 PROCESS CONTROL 1387
3.3.6.7.9.9.10.2.6 PROCESSING 1387
3.3.6.7.9.9.10.2.7 UTILIZATION OF OTHER ELEMENTS 1387
3.3.6.7.9.9.10.2.8 LIMITATIONS 1387
3.3.6.7.9.9.10.3 INTEGRATE UNIT DESIGN 1388
3.3.6.7.9.9.10.3.1 REQUIREMENTS ALLOCATION 1388
3.3.6.7.9.9.10.3.2 LOCAL ENTITIES DESIGN 1388
3.3.6.7.9.9.10.3.3 INPUT/OUTPUT 1388
3.3.6.7.9.9.10.3.4 LOCAL DATA 1388
3.3.6.7.9.9.10.3.5 PROCESS CONTROL 1388
3.3.6.7.9.9.10.3.6 PROCESSING 1388
3.3.6.7.9.9.10.3.7 UTILIZATION OF OTHER ELEMENTS 1389
3.3.6.7.9.9.10.3.8 LIMITATIONS 1389
3.3.6.7.9.10 TWO WAY TABLE LOOKUP (BODY) PACKAGE DESIGN (CATALOG #P1078-0) 1389
3.3.6.7.9.10.1 REQUIREMENTS ALLOCATION 1390
3.3.6.7.9.10.2 LOCAL ENTITIES DESIGN 1390
3.3.6.7.9.10.3 INPUT/OUTPUT 1390
3.3.6.7.9.10.4 LOCAL DATA 1390
3.3.6.7.9.10.5 PROCESS CONTROL 1391
3.3.6.7.9.10.6 PROCESSING 1391
3.3.6.7.9.10.7 UTILIZATION OF OTHER ELEMENTS 1394
3.3.6.7.9.10.8 LIMITATIONS 1395
3.3.6.7.9.10.9 LLCSC DESIGN 1395

3.3.6.7.9.10.10 UNIT DESIGN 1395
3.3.6.7.10 UNIT DESIGN 1395
3.3.6.7.10.1 INTERPOLATE_OR_EXTRAPOLATE (BODY) UNIT DESIGN (CATALOG #P39-0) 1395
3.3.6.7.10.1.1 REQUIREMENTS ALLOCATION 1395
3.3.6.7.10.1.2 LOCAL ENTITIES DESIGN 1395
3.3.6.7.10.1.3 INPUT/OUTPUT 1395
3.3.6.7.10.1.4 LOCAL DATA 1396
3.3.6.7.10.1.5 PROCESS CONTROL 1396
3.3.6.7.10.1.6 PROCESSING 1396
3.3.6.7.10.1.7 UTILIZATION OF OTHER ELEMENTS 1397
3.3.6.7.10.1.8 LIMITATIONS 1397
3.3.6.7.10.2 SQUARE_ROOT (BODY) UNIT DESIGN (CATALOG #P40-0) 1397
3.3.6.7.10.2.1 REQUIREMENTS ALLOCATION 1397
3.3.6.7.10.2.2 LOCAL ENTITIES DESIGN 1397
3.3.6.7.10.2.3 INPUT/OUTPUT 1397
3.3.6.7.10.2.4 LOCAL DATA 1398
3.3.6.7.10.2.5 PROCESS CONTROL 1398
3.3.6.7.10.2.6 PROCESSING 1398
3.3.6.7.10.2.7 UTILIZATION OF OTHER ELEMENTS 1398
3.3.6.7.10.2.8 LIMITATIONS 1399
3.3.6.7.10.3 ROOT_SUM_OF_SQUARES (BODY) UNIT DESIGN (CATALOG #P41-0) 1399
3.3.6.7.10.3.1 REQUIREMENTS ALLOCATION 1399
3.3.6.7.10.3.2 LOCAL ENTITIES DESIGN 1399
3.3.6.7.10.3.3 INPUT/OUTPUT 1399
3.3.6.7.10.3.4 LOCAL DATA 1400
3.3.6.7.10.3.5 PROCESS CONTROL 1400
3.3.6.7.10.3.6 PROCESSING 1400
3.3.6.7.10.3.7 UTILIZATION OF OTHER ELEMENTS 1400
3.3.6.7.10.3.8 LIMITATIONS 1400
3.3.6.7.10.4 SIGN (BODY) UNIT DESIGN (CATALOG #P42-0) 1400
3.3.6.7.10.4.1 REQUIREMENTS ALLOCATION 1401
3.3.6.7.10.4.2 LOCAL ENTITIES DESIGN 1401
3.3.6.7.10.4.3 INPUT/OUTPUT 1401
3.3.6.7.10.4.4 LOCAL DATA 1401
3.3.6.7.10.4.5 PROCESS CONTROL 1401
3.3.6.7.10.4.6 PROCESSING 1402
3.3.6.7.10.4.7 UTILIZATION OF OTHER ELEMENTS 1402
3.3.6.7.10.4.8 LIMITATIONS 1402
3.3.6.7.10.5 MEAN_VALUE (BODY) UNIT DESIGN (CATALOG #P43-0) 1402
3.3.6.7.10.5.1 REQUIREMENTS ALLOCATION 1402
3.3.6.7.10.5.2 LOCAL ENTITIES DESIGN 1402
3.3.6.7.10.5.3 INPUT/OUTPUT 1403
3.3.6.7.10.5.4 LOCAL DATA 1403
3.3.6.7.10.5.5 PROCESS CONTROL 1404
3.3.6.7.10.5.6 PROCESSING 1404
3.3.6.7.10.5.7 UTILIZATION OF OTHER ELEMENTS 1404
3.3.6.7.10.5.8 LIMITATIONS 1404
3.3.6.7.10.6 MEAN_ABSOLUTE_DIFFERENCE (BODY) UNIT DESIGN (CATALOG #P44-0) 1404
3.3.6.7.10.6.1 REQUIREMENTS ALLOCATION 1404
3.3.6.7.10.6.2 LOCAL ENTITIES DESIGN 1405
3.3.6.7.10.6.3 INPUT/OUTPUT 1405
3.3.6.7.10.6.4 LOCAL DATA 1405

3.3.6.7.10.6.5 PROCESS CONTROL 1406
3.3.6.7.10.6.6 PROCESSING 1406
3.3.6.7.10.6.7 UTILIZATION OF OTHER ELEMENTS 1407
3.3.6.7.10.6.8 LIMITATIONS 1407

3.3.6.8 POLYNOMIALS (PACKAGE BODY) TLCSC P688 (CATALOG #P722-0) 1435
3.3.6.8.1 REQUIREMENTS ALLOCATION 1435
3.3.6.8.2 LOCAL ENTITIES DESIGN 1435
3.3.6.8.3 INPUT/OUTPUT 1435
3.3.6.8.4 LOCAL DATA 1435
3.3.6.8.5 PROCESS CONTROL 1435
3.3.6.8.6 PROCESSING 1436
3.3.6.8.7 UTILIZATION OF OTHER ELEMENTS 1436
3.3.6.8.8 LIMITATIONS 1436
3.3.6.8.9 LLCSC DESIGN 1436

3.3.6.8.9.1 CHEBYSHEV PACKAGE DESIGN (CATALOG #P723-0) 1437
3.3.6.8.9.1.1 REQUIREMENTS ALLOCATION 1437
3.3.6.8.9.1.2 LOCAL ENTITIES DESIGN 1437
3.3.6.8.9.1.3 INPUT/OUTPUT 1437
3.3.6.8.9.1.4 LOCAL DATA 1437
3.3.6.8.9.1.5 PROCESS CONTROL 1437
3.3.6.8.9.1.6 PROCESSING 1437
3.3 6.8.9.1.7 UTILIZATION OF OTHER ELEMENTS 1437
3.3.6.8.9.1.8 LIMITATIONS 1438
3.3.6.8.9.1.9 LLCSC DESIGN 1438
3.3.6.8.9.1.9.1 CHEBYSHEV RADIAN OPERATIONS PACKAGE DESIGN (CATALOG #P724-0) 1438
3.3.6.8.9.1.9.1.1 REQUIREMENTS ALLOCATION 1438
3.3.6.8.9.1.9.1.2 LOCAL ENTITIES DESIGN 1438
3.3.6.8.9.1.9.1.3 INPUT/OUTPUT 1438
3.3.6.8.9.1.9.1.4 LOCAL DATA 1439
3.3.6.8.9.1.9.1.5 PROCESS CONTROL 1439
3.3.6.8.9.1.9.1.6 PROCESSING 1440
3.3.6.8.9.1.9.1.7 UTILIZATION OF OTHER ELEMENTS 1440
3.3.6.8.9.1.9.1.8 LIMITATIONS 1441
3.3.6.8.9.1.9.1.9 LLCSC DESIGN 1441
3.3.6.8.9.1.9.1.10 UNIT DESIGN 1441
3.3.6.8.9.1.9.2 CHEBYSHEV DEGREE OPERATIONS PACKAGE DESIGN (CATALOG #P1088-0) 1441
3.3.6.8.9.1.9.2.1 REQUIREMENTS ALLOCATION 1441
3.3.6.8.9.1.9.2.2 LOCAL ENTITIES DESIGN 1441
3.3.6.8.9.1.9.2.3 INPUT/OUTPUT 1441
3.3.6.8.9.1.9.2.4 LOCAL DATA 1442
3.3.6.8.9.1.9.2.5 PROCESS CONTROL 1443
3.3.6.8.9.1.9.2.6 PROCESSING 1443
3.3.6.8.9.1.9.2.7 UTILIZATION OF OTHER ELEMENTS 1444
3.3.6.8.9.1.9.2.8 LIMITATIONS 1444
3.3.6.8.9.1.9.2.9 LLCSC DESIGN 1444
3.3.6.8.9.1.9.2.10 UNIT DESIGN 1444
3.3.6.8.9.1.9.3 CHEBYSHEV SEMICIRCLE OPERATIONS PACKAGE DESIGN (CATALOG #P728-0) 1444
3.3.6.8.9.1.9.3.1 REQUIREMENTS ALLOCATION 1445
3.3.6.8.9.1.9.3.2 LOCAL ENTITIES DESIGN 1445
3.3.6.8.9.1.9.3.3 INPUT/OUTPUT 1445
3.3.6.8.9.1.9.3.4 LOCAL DATA 1446

3.3.6.8.9.1.9.3.5 PROCESS CONTROL 1446
3.3.6.8.9.1.9.3.6 PROCESSING 1446
3.3.6.8.9.1.9.3.7 UTILIZATION OF OTHER ELEMENTS 1447
3.3.6.8.9.1.9.3.8 LIMITATIONS 1447
3.3.6.8.9.1.9.3.9 LLCSC DESIGN 1447
3.3.6.8.9.1.9.3.10 UNIT DESIGN 1447
3.3.6.8.9.1.10 UNIT DESIGN 1447

3.3.6.8.9.2 FIKE PACKAGE DESIGN (CATALOG #P734-0) 1449
3.3.6.8.9.2.1 REQUIREMENTS ALLOCATION 1449
3.3.6.8.9.2.2 LOCAL ENTITIES DESIGN 1449
3.3.6.8.9.2.3 INPUT/OUTPUT 1449
3.3.6.8.9.2.4 LOCAL DATA 1449
3.3.6.8.9.2.5 PROCESS CONTROL 1449
3.3.6.8.9.2.6 PROCESSING 1449
3.3.6.8.9.2.7 UTILIZATION OF OTHER ELEMENTS 1449
3.3.6.8.9.2.8 LIMITATIONS 1449
3.3.6.8.9.2.9 LLCSC DESIGN 1450
3.3.6.8.9.2.9.1 FIKE_SEMICIRCLE_OPERATIONS PACKAGE DESIGN (CATALOG #P735-0) 1450
3.3.6.8.9.2.9.1.1 REQUIREMENTS ALLOCATION 1450
3.3.6.8.9.2.9.1.2 LOCAL ENTITIES DESIGN 1450
3.3.6.8.9.2.9.1.3 INPUT/OUTPUT 1450
3.3.6.8.9.2.9.1.4 LOCAL DATA 1451
3.3.6.8.9.2.9.1.5 PROCESS CONTROL 1451
3.3.6.8.9.2.9.1.6 PROCESSING 1451
3.3.6.8.9.2.9.1.7 UTILIZATION OF OTHER ELEMENTS 1453
3.3.6.8.9.2.9.1.8 LIMITATIONS 1453
3.3.6.8.9.2.9.1.9 LLCSC DESIGN 1453
3.3.6.8.9.2.9.1.10 UNIT DESIGN 1453
3.3.6.8.9.2.10 UNIT DESIGN 1453

3.3.6.8.9.3 HART PACKAGE DESIGN (CATALOG #P740-0) 1455
3.3.6.8.9.3.1 REQUIREMENTS ALLOCATION 1455
3.3.6.8.9.3.2 LOCAL ENTITIES DESIGN 1455
3.3.6.8.9.3.3 INPUT/OUTPUT 1455
3.3.6.8.9.3.4 LOCAL DATA 1455
3.3.6.8.9.3.5 PROCESS CONTROL 1455
3.3.6.8.9.3.6 PROCESSING 1455
3.3.6.8.9.3.7 UTILIZATION OF OTHER ELEMENTS 1455
3.3.6.8.9.3.8 LIMITATIONS 1456
3.3.6.8.9.3.9 LLCSC DESIGN 1456
3.3.6.8.9.3.9.1 HART_RADIAN_OPERATIONS PACKAGE DESIGN (CATALOG #P741-0) 1456
3.3.6.8.9.3.9.1.1 REQUIREMENTS ALLOCATION 1456
3.3.6.8.9.3.9.1.2 LOCAL ENTITIES DESIGN 1456
3.3.6.8.9.3.9.1.3 INPUT/OUTPUT 1456
3.3.6.8.9.3.9.1.4 LOCAL DATA 1457
3.3.6.8.9.3.9.1.5 PROCESS CONTROL 1457
3.3.6.8.9.3.9.1.6 PROCESSING 1457
3.3.6.8.9.3.9.1.7 UTILIZATION OF OTHER ELEMENTS 1458
3.3.6.8.9.3.9.1.8 LIMITATIONS 1458
3.3.6.8.9.3.9.1.9 LLCSC DESIGN 1458
3.3.6.8.9.3.9.1.10 UNIT DESIGN 1458
3.3.6.8.9.3.9.2 HART_DEGREE_OPERATIONS PACKAGE DESIGN (CATALOG #P743-0) 1459
3.3.6.8.9.3.9.2.1 REQUIREMENTS ALLOCATION 1459

3.3.6.8.9.3.9.2.2 LOCAL ENTITIES DESIGN 1459
3.3.6.8.9.3.9.2.3 INPUT/OUTPUT 1459
3.3.6.8.9.3.9.2.4 LOCAL DATA 1460
3.3.6.8.9.3.9.2.5 PROCESS CONTROL 1460
3.3.6.8.9.3.9.2.6 PROCESSING 1460
3.3.6.8.9.3.9.2.7 UTILIZATION OF OTHER ELEMENTS 1461
3.3.6.8.9.3.9.2.8 LIMITATIONS 1461
3.3.6.8.9.3.9.2.9 LLCSC DESIGN 1461
3.3.6.8.9.3.9.2.10 UNIT DESIGN 1461
3.3.6.8.9.3.10 UNIT DESIGN 1461

3.3.6.8.9.4 HASTINGS PACKAGE DESIGN (CATALOG #P745-0) 1463
3.3.6.8.9.4.1 REQUIREMENTS ALLOCATION 1463
3.3.6.8.9.4.2 LOCAL ENTITIES DESIGN 1463
3.3.6.8.9.4.3 INPUT/OUTPUT 1463
3.3.6.8.9.4.4 LOCAL DATA 1463
3.3.6.8.9.4.5 PROCESS CONTROL 1463
3.3.6.8.9.4.6 PROCESSING 1463
3.3.6.8.9.4.7 UTILIZATION OF OTHER ELEMENTS 1463
3.3.6.8.9.4.8 LIMITATIONS 1464
3.3.6.8.9.4.9 LLCSC DESIGN 1464
3.3.6.8.9.4.9.1 HASTINGS_RADIAN_OPERATIONS PACKAGE DESIGN (CATALOG #P746-0) 1464
3.3.6.8.9.4.9.1.1 REQUIREMENTS ALLOCATION 1464
3.3.6.8.9.4.9.1.2 LOCAL ENTITIES DESIGN 1464
3.3.6.8.9.4.9.1.3 INPUT/OUTPUT 1464
3.3.6.8.9.4.9.1.4 LOCAL DATA 1465
3.3.6.8.9.4.9.1.5 PROCESS CONTROL 1468
3.3.6.8.9.4.9.1.6 PROCESSING 1468
3.3.6.8.9.4.9.1.7 UTILIZATION OF OTHER ELEMENTS 1474
3.3.6.8.9.4.9.1.8 LIMITATIONS 1474
3.3.6.8.9.4.9.1.9 LLCSC DESIGN 1474
3.3.6.8.9.4.9.1.10 UNIT DESIGN 1474
3.3.6.8.9.4.9.2 HASTINGS_DEGREE_OPERATIONS PACKAGE DESIGN (CATALOG #P759-0) 1474
3.3.6.8.9.4.9.2.1 REQUIREMENTS ALLOCATION 1475
3.3.6.8.9.4.9.2.2 LOCAL ENTITIES DESIGN 1475
3.3.6.8.9.4.9.2.3 INPUT/OUTPUT 1475
3.3.6.8.9.4.9.2.4 LOCAL DATA 1476
3.3.6.8.9.4.9.2.5 PROCESS CONTROL 1476
3.3.6.8.9.4.9.2.6 PROCESSING 1476
3.3.6.8.9.4.9.2.7 UTILIZATION OF OTHER ELEMENTS 1479
3.3.6.8.9.4.9.2.8 LIMITATIONS 1479
3.3.6.8.9.4.9.2.9 LLCSC DESIGN 1479
3.3.6.8.9.4.9.2.10 UNIT DESIGN 1480
3.3.6.8.9.4.10 UNIT DESIGN 1480

3.3.6.8.9.5 MODIFIED_NEWTON_RAPHSON PACKAGE DESIGN (CATALOG #P766-0) 1481
3.3.6.8.9.5.1 REQUIREMENTS ALLOCATION 1481
3.3.6.8.9.5.2 LOCAL ENTITIES DESIGN 1481
3.3.6.8.9.5.3 INPUT/OUTPUT 1481
3.3.6.8.9.5.4 LOCAL DATA 1482
3.3.6.8.9.5.5 PROCESS CONTROL 1482
3.3.6.8.9.5.6 PROCESSING 1482
3.3.6.8.9.5.7 UTILIZATION OF OTHER ELEMENTS 1483
3.3.6.8.9.5.8 LIMITATIONS 1483

3.3.6.8.9.5.9 LLCSC DESIGN 1483
3.3.6.8.9.5.10 UNIT DESIGN 1484

3.3.6.8.9.6 NEWTON RAPHSON PACKAGE DESIGN (CATALOG #P768-0) 1. . . . 485
3.3.6.8.9.6.1 REQUIREMENTS ALLOCATION 1485
3.3.6.8.9.6.2 LOCAL ENTITIES DESIGN 1485
3.3.6.8.9.6.3 INPUT/OUTPUT 1485
3.3.6.8.9.6.4 LOCAL DATA 1486
3.3.6.8.9.6.5 PROCESS CONTROL 1486
3.3.6.8.9.6.6 PROCESSING 1486
3.3.6.8.9.6.7 UTILIZATION OF OTHER ELEMENTS 1487
3.3.6.8.9.6.8 LIMITATIONS 1487
3.3.6.8.9.6.9 LLCSC DESIGN 1488
3.3.6.8.9.6.10 UNIT DESIGN 1488

3.3.6.8.9.7 TAYLOR SERIES PACKAGE DESIGN (CATALOG #P795-0) 1489
3.3.6.8.9.7.1 REQUIREMENTS ALLOCATION 1489
3.3.6.8.9.7.2 LOCAL ENTITIES DESIGN 1489
3.3.6.8.9.7.3 INPUT/OUTPUT 1489
3.3.6.8.9.7.4 LOCAL DATA 1489
3.3.6.8.9.7.5 PROCESS CONTROL 1489
3.3.6.8.9.7.6 PROCESSING 1489
3.3.6.8.9.7.7 UTILIZATION OF OTHER ELEMENTS 1490
3.3.6.8.9.7.8 LIMITATIONS 1490
3.3.6.8.9.7.9 LLCSC DESIGN 1490
3.3.6.8.9.7.9.1 TAYLOR RADIAN OPERATIONS PACKAGE DESIGN (CATALOG #P796-0) 1490
3.3.6.8.9.7.9.1.1 REQUIREMENTS ALLOCATION 1492
3.3.6.8.9.7.9.1.2 LOCAL ENTITIES DESIGN 1492
3.3.6.8.9.7.9.1.3 INPUT/OUTPUT 1492
3.3.6.8.9.7.9.1.4 LOCAL DATA 1493
3.3.6.8.9.7.9.1.5 PROCESS CONTROL 1495
3.3.6.8.9.7.9.1.6 PROCESSING 1495
3.3.6.8.9.7.9.1.7 UTILIZATION OF OTHER ELEMENTS 1516
3.3.6.8.9.7.9.1.8 LIMITATIONS 1517
3.3.6.8.9.7.9.1.9 LLCSC DESIGN 1517
3.3.6.8.9.7.9.1.10 UNIT DESIGN 1517
3.3.6.8.9.7.9.2 TAYLOR DEGREE OPERATIONS PACKAGE DESIGN (CATALOG #P841-0) 1517
3.3.6.8.9.7.9.2.1 REQUIREMENTS ALLOCATION 1518
3.3.6.8.9.7.9.2.2 LOCAL ENTITIES DESIGN 1518
3.3.6.8.9.7.9.2.3 INPUT/OUTPUT 1518
3.3.6.8.9.7.9.2.4 LOCAL DATA 1519
3.3.6.8.9.7.9.2.5 PROCESS CONTROL 1520
3.3.6.8.9.7.9.2.6 PROCESSING 1521
3.3.6.8.9.7.9.2.7 UTILIZATION OF OTHER ELEMENTS 1535
3.3.6.8.9.7.9.2.8 LIMITATIONS 1535
3.3.6.8.9.7.9.2.9 LLCSC DESIGN 1535
3.3.6.8.9.7.9.2.10 UNIT DESIGN 1535
3.3.6.8.9.7.9.3 TAYLOR NATURAL LOG PACKAGE DESIGN (CATALOG #P868-0) 1536
3.3.6.8.9.7.9.3.1 REQUIREMENTS ALLOCATION 1536
3.3.6.8.9.7.9.3.2 LOCAL ENTITIES DESIGN 1536
3.3.6.8.9.7.9.3.3 INPUT/OUTPUT 1536
3.3.6.8.9.7.9.3.4 LOCAL DATA 1537
3.3.6.8.9.7.9.3.5 PROCESS CONTROL 1537
3.3.6.8.9.7.9.3.6 PROCESSING 1537

3.3.6.8.9.7.9.3.7 UTILIZATION OF OTHER ELEMENTS 1539
3.3.6.8.9.7.9.3.8 LIMITATIONS 1539
3.3.6.8.9.7.9.3.9 LLCSC DESIGN 1539
3.3.6.8.9.7.9.3.10 UNIT DESIGN 1540
3.3.6.8.9.7.9.4 TAYLOR_LOG_BASE_N PACKAGE DESIGN (CATALOG #P874-0) 1540
3.3.6.8.9.7.9.4.1 REQUIREMENTS ALLOCATION 1540
3.3.6.8.9.7.9.4.2 LOCAL ENTITIES DESIGN 1540
3.3.6.8.9.7.9.4.3 INPUT/OUTPUT 1540
3.3.6.8.9.7.9.4.4 LOCAL DATA 1541
3.3.6.8.9.7.9.4.5 PROCESS CONTROL 1541
3.3.6.8.9.7.9.4.6 PROCESSING 1541
3.3.6.8.9.7.9.4.7 UTILIZATION OF OTHER ELEMENTS 1543
3.3.6.8.9.7.9.4.8 LIMITATIONS 1543
3.3.6.8.9.7.9.4.9 LLCSC DESIGN 1543
3.3.6.8.9.7.9.4.10 UNIT DESIGN 1543
3.3.6.8.9.7.10 UNIT DESIGN 1543

3.3.6.8.9.8 GENERAL_POLYNOMIAL PACKAGE DESIGN (CATALOG #P738-0) 1545
3.3.6.8.9.8.1 REQUIREMENTS ALLOCATION 1545
3.3.6.8.9.8.2 LOCAL ENTITIES DESIGN 1545
3.3.6.8.9.8.3 INPUT/OUTPUT 1545
3.3.6.8.9.8.4 LOCAL DATA 1546
3.3.6.8.9.8.5 PROCESS CONTROL 1546
3.3.6.8.9.8.6 PROCESSING 1547
3.3.6.8.9.8.7 UTILIZATION OF OTHER ELEMENTS 1547
3.3.6.8.9.8.8 LIMITATIONS 1547
3.3.6.8.9.8.9 LLCSC DESIGN 1547
3.3.6.8.9.8.10 UNIT DESIGN 1547

3.3.6.8.9.9 SYSTEM FUNCTIONS PACKAGE DESIGN (CATALOG #P770-0) 1. . . . 549
3.3.6.8.9.9.1 REQUIREMENTS ALLOCATION 1549
3.3.6.8.9.9.2 LOCAL ENTITIES DESIGN 1549
3.3.6.8.9.9.3 INPUT/OUTPUT 1549
3.3.6.8.9.9.4 LOCAL DATA 1549
3.3.6.8.9.9.5 PROCESS CONTROL 1549
3.3.6.8.9.9.6 PROCESSING 1549
3.3.6.8.9.9.7 UTILIZATION OF OTHER ELEMENTS 1550
3.3.6.8.9.9.8 LIMITATIONS 1550
3.3.6.8.9.9.9 LLCSC DESIGN 1550
3.3.6.8.9.9.9.1 RADIAN OPERATIONS PACKAGE DESIGN (CATALOG #P771-0) 1550
3.3.6.8.9.9.9.1.1 REQUIREMENTS ALLOCATION 1550
3.3.6.8.9.9.9.1.2 LOCAL ENTITIES DESIGN 1550
3.3.6.8.9.9.9.1.3 INPUT/OUTPUT 1550
3.3.6.8.9.9.9.1.4 LOCAL DATA 1551
3.3.6.8.9.9.9.1.5 PROCESS CONTROL 1551
3.3.6.8.9.9.9.1.6 PROCESSING 1551
3.3.6.8.9.9.9.1.7 UTILIZATION OF OTHER ELEMENTS 1552
3.3.6.8.9.9.9.1.8 LIMITATIONS 1552
3.3.6.8.9.9.9.1.9 LLCSC DESIGN 1552
3.3.6.8.9.9.9.1.10 UNIT DESIGN 1552
3.3.6.8.9.9.9.1.10.1 SIN UNIT DESIGN (CATALOG #P772-0) 1552
3.3.6.8.9.9.9.1.10.1.1 REQUIREMENTS ALLOCATION 1552
3.3.6.8.9.9.9.1.10.1.2 LOCAL ENTITIES DESIGN 1552
3.3.6.8.9.9.9.1.10.1.3 INPUT/OUTPUT 1552

3.3.6.8.9.9.9.1.10.1.4 LOCAL DATA 1552
3.3.6.8.9.9.9.1.10.1.5 PROCESS CONTROL 1552
3.3.6.8.9.9.9.1.10.1.6 PROCESSING 1553
3.3.6.8.9.9.9.1.10.1.7 UTILIZATION OF OTHER ELEMENTS 1553
3.3.6.8.9.9.9.1.10.1.8 LIMITATIONS 1554
3.3.6.8.9.9.9.1.10.2 COS UNIT DESIGN (CATALOG #P773-0) 1554
3.3.6.8.9.9.9.1.10.2.1 REQUIREMENTS ALLOCATION 1554
3.3.6.8.9.9.9.1.10.2.2 LOCAL ENTITIES DESIGN 1554
3.3.6.8.9.9.9.1.10.2.3 INPUT/OUTPUT 1554
3.3.6.8.9.9.9.1.10.2.4 LOCAL DATA 1554
3.3.6.8.9.9.9.1.10.2.5 PROCESS CONTROL 1554
3.3.6.8.9.9.9.1.10.2.6 PROCESSING 1555
3.3.6.8.9.9.9.1.10.2.7 UTILIZATION OF OTHER ELEMENTS 1555
3.3.6.8.9.9.9.1.10.2.8 LIMITATIONS 1556
3.3.6.8.9.9.9.1.10.3 TAN UNIT DESIGN (CATALOG #P774-0) 1556
3.3.6.8.9.9.9.1.10.3.1 REQUIREMENTS ALLOCATION 1556
3.3.6.8.9.9.9.1.10.3.2 LOCAL ENTITIES DESIGN 1556
3.3.6.8.9.9.9.1.10.3.3 INPUT/OUTPUT 1556
3.3.6.8.9.9.9.1.10.3.4 LOCAL DATA 1556
3.3.6.8.9.9.9.1.10.3.5 PROCESS CONTROL 1556
3.3.6.8.9.9.9.1.10.3.6 PROCESSING 1557
3.3.6.8.9.9.9.1.10.3.7 UTILIZATION OF OTHER ELEMENTS 1557
3.3.6.8.9.9.9.1.10.3.8 LIMITATIONS 1558
3.3.6.8.9.9.9.1.10.4 ARCSIN UNIT DESIGN (CATALOG #P775-0) 1558
3.3.6.8.9.9.9.1.10.4.1 REQUIREMENTS ALLOCATION 1558
3.3.6.8.9.9.9.1.10.4.2 LOCAL ENTITIES DESIGN 1558
3.3.6.8.9.9.9.1.10.4.3 INPUT/OUTPUT 1558
3.3.6.8.9.9.9.1.10.4.4 LOCAL DATA 1559
3.3.6.8.9.9.9.1.10.4.5 PROCESS CONTROL 1559
3.3.6.8.9.9.9.1.10.4.6 PROCESSING 1559
3.3.6.8.9.9.9.1.10.4.7 UTILIZATION OF OTHER ELEMENTS 1559
3.3.6.8.9.9.9.1.10.4.8 LIMITATIONS 1560
3.3.6.8.9.9.9.1.10.5 ARCCOS UNIT DESIGN (CATALOG #P776-0) 1560
3.3.6.8.9.9.9.1.10.5.1 REQUIREMENTS ALLOCATION 1560
3.3.6.8.9.9.9.1.10.5.2 LOCAL ENTITIES DESIGN 1560
3.3.6.8.9.9.9.1.10.5.3 INPUT/OUTPUT 1560
3.3.6.8.9.9.9.1.10.5.4 LOCAL DATA 1561
3.3.6.8.9.9.9.1.10.5.5 PROCESS CONTROL 1561
3.3.6.8.9.9.9.1.10.5.6 PROCESSING 1561
3.3.6.8.9.9.9.1.10.5.7 UTILIZATION OF OTHER ELEMENTS 1561
3.3.6.8.9.9.9.1.10.5.8 LIMITATIONS 1562
3.3.6.8.9.9.9.1.10.6 ARCTAN UNIT DESIGN (CATALOG #P777-0) 1562
3.3.6.8.9.9.9.1.10.6.1 REQUIREMENTS ALLOCATION 1563
3.3.6.8.9.9.9.1.10.6.2 LOCAL ENTITIES DESIGN 1563
3.3.6.8.9.9.9.1.10.6.3 INPUT/OUTPUT 1563
3.3.6.8.9.9.9.1.10.6.4 LOCAL DATA 1563
3.3.6.8.9.9.9.1.10.6.5 PROCESS CONTROL 1563
3.3.6.8.9.9.9.1.10.6.6 PROCESSING 1563
3.3.6.8.9.9.9.1.10.6.7 UTILIZATION OF OTHER ELEMENTS 1563
3.3.6.8.9.9.9.1.10.6.8 LIMITATIONS 1564
3.3.6.8.9.9.9.2 SEMICIRCLE OPERATIONS PACKAGE DESIGN (CATALOG #P778-0) 1564
3.3.6.8.9.9.9.2.1 REQUIREMENTS ALLOCATION 1565
3.3.6.8.9.9.9.2.2 LOCAL ENTITIES DESIGN 1565
3.3.6.8.9.9.9.2.3 INPUT/OUTPUT 1565
3.3.6.8.9.9.9.2.4 LOCAL DATA 1566
3.3.6.8.9.9.9.2.5 PROCESS CONTROL 1566

3.3.6.8.9.9.9.2.6 PROCESSING 1566
3.3.6.8.9.9.9.2.7 UTILIZATION OF OTHER ELEMENTS 1567
3.3.6.8.9.9.9.2.8 LIMITATIONS 1567
3.3.6.8.9.9.9.2.9 LLCSC DESIGN 1567
3.3.6.8.9.9.9.2.10 UNIT DESIGN 1567
3.3.6.8.9.9.9.2.10.1 SIN UNIT DESIGN (CATALOG #P779-0) 1567
3.3.6.8.9.9.9.2.10.1.1 REQUIREMENTS ALLOCATION 1567
3.3.6.8.9.9.9.2.10.1.2 LOCAL ENTITIES DESIGN 1567
3.3.6.8.9.9.9.2.10.1.3 INPUT/OUTPUT 1567
3.3.6.8.9.9.9.2.10.1.4 LOCAL DATA 1568
3.3.6.8.9.9.9.2.10.1.5 PROCESS CONTROL 1568
3.3.6.8.9.9.9.2.10.1.6 PROCESSING 1568
3.3.6.8.9.9.9.2.10.1.7 UTILIZATION OF OTHER ELEMENTS 1568
3.3.6.8.9.9.9.2.10.1.8 LIMITATIONS 1569
3.3.6.8.9.9.9.2.10.2 COS UNIT DESIGN (CATALOG #P1087-0) 1569
3.3.6.8.9.9.9.2.10.2.1 REQUIREMENTS ALLOCATION 1569
3.3.6.8.9.9.9.2.10.2.2 LOCAL ENTITIES DESIGN 1570
3.3.6.8.9.9.9.2.10.2.3 INPUT/OUTPUT 1570
3.3.6.8.9.9.9.2.10.2.4 LOCAL DATA 1570
3.3.6.8.9.9.9.2.10.2.5 PROCESS CONTROL 1570
3.3.6.8.9.9.9.2.10.2.6 PROCESSING 1570
3.3.6.8.9.9.9.2.10.2.7 UTILIZATION OF OTHER ELEMENTS 1570
3.3.6.8.9.9.9.2.10.2.8 LIMITATIONS 1571
3.3.6.8.9.9.9.2.10.3 TAN UNIT DESIGN (CATALOG #P781-0) 1572
3.3.6.8.9.9.9.2.10.3.1 REQUIREMENTS ALLOCATION 1572
3.3.6.8.9.9.9.2.10.3.2 LOCAL ENTITIES DESIGN 1572
3.3.6.8.9.9.9.2.10.3.3 INPUT/OUTPUT 1572
3.3.6.8.9.9.9.2.10.3.4 LOCAL DATA 1572
3.3.6.8.9.9.9.2.10.3.5 PROCESS CONTROL 1572
3.3.6.8.9.9.9.2.10.3.6 PROCESSING 1572
3.3.6.8.9.9.9.2.10.3.7 UTILIZATION OF OTHER ELEMENTS 1573
3.3.6.8.9.9.9.2.10.3.8 LIMITATIONS 1574
3.3.6.8.9.9.9.2.10.4 ARCSIN UNIT DESIGN (CATALOG #P782-0) 1574
3.3.6.8.9.9.9.2.10.4.1 REQUIREMENTS ALLOCATION 1574
3.3.6.8.9.9.9.2.10.4.2 LOCAL ENTITIES DESIGN 1574
3.3.6.8.9.9.9.2.10.4.3 INPUT/OUTPUT 1574
3.3.6.8.9.9.9.2.10.4.4 LOCAL DATA 1574
3.3.6.8.9.9.9.2.10.4.5 PROCESS CONTROL 1574
3.3.6.8.9.9.9.2.10.4.6 PROCESSING 1575
3.3.6.8.9.9.9.2.10.4.7 UTILIZATION OF OTHER ELEMENTS 1575
3.3.6.8.9.9.9.2.10.4.8 LIMITATIONS 1576
3.3.6.8.9.9.9.2.10.5 ARCCOS UNIT DESIGN (CATALOG #P783-0) 1576
3.3.6.8.9.9.9.2.10.5.1 REQUIREMENTS ALLOCATION 1577
3.3.6.8.9.9.9.2.10.5.2 LOCAL ENTITIES DESIGN 1577
3.3.6.8.9.9.9.2.10.5.3 INPUT/OUTPUT 1577
3.3.6.8.9.9.9.2.10.5.4 LOCAL DATA 1577
3.3.6.8.9.9.9.2.10.5.5 PROCESS CONTROL 1577
3.3.6.8.9.9.9.2.10.5.6 PROCESSING 1577
3.3.6.8.9.9.9.2.10.5.7 UTILIZATION OF OTHER ELEMENTS 1577
3.3.6.8.9.9.9.2.10.5.8 LIMITATIONS 1579
3.3.6.8.9.9.9.2.10.6 ARCTAN UNIT DESIGN (CATALOG #P784-0) 1579
3.3.6.8.9.9.9.2.10.6.1 REQUIREMENTS ALLOCATION 1579
3.3.6.8.9.9.9.2.10.6.2 LOCAL ENTITIES DESIGN 1579
3.3.6.8.9.9.9.2.10.6.3 INPUT/OUTPUT 1579
3.3.6.8.9.9.9.2.10.6.4 LOCAL DATA 1580
3.3.6.8.9.9.9.2.10.6.5 PROCESS CONTROL 1580
3.3.6.8.9.9.9.2.10.6.6 PROCESSING 1580

3.3.6.8.9.9.9.2.10.6.7 UTILIZATION OF OTHER ELEMENTS 1580
3.3.6.8.9.9.9.2.10.6.8 LIMITATIONS 1581
3.3.6.8.9.9.9.3 DEGREE OPERATIONS PACKAGE DESIGN (CATALOG #P785-0) 1581
3.3.6.8.9.9.9.3.1 REQUIREMENTS ALLOCATION 1582
3.3.6.8.9.9.9.3.2 LOCAL ENTITIES DESIGN 1582
3.3.6.8.9.9.9.3.3 INPUT/OUTPUT 1582
3.3.6.8.9.9.9.3.4 LOCAL DATA 1582
3.3.6.8.9.9.9.3.5 PROCESS CONTROL 1583
3.3.6.8.9.9.9.3.6 PROCESSING 1583
3.3.6.8.9.9.9.3.7 UTILIZATION OF OTHER ELEMENTS 1583
3.3.6.8.9.9.9.3.8 LIMITATIONS 1583
3.3.6.8.9.9.9.3.9 LLCSC DESIGN 1583
3.3.6.8.9.9.9.3.10 UNIT DESIGN 1584
3.3.6.8.9.9.9.3.10.1 SIN UNIT DESIGN (CATALOG #P786-0) 1584
3.3.6.8.9.9.9.3.10.1.1 REQUIREMENTS ALLOCATION 1584
3.3.6.8.9.9.9.3.10.1.2 LOCAL ENTITIES DESIGN 1584
3.3.6.8.9.9.9.3.10.1.3 INPUT/OUTPUT 1584
3.3.6.8.9.9.9.3.10.1.4 LOCAL DATA 1584
3.3.6.8.9.9.9.3.10.1.5 PROCESS CONTROL 1584
3.3.6.8.9.9.9.3.10.1.6 PROCESSING 1584
3.3.6.8.9.9.9.3.10.1.7 UTILIZATION OF OTHER ELEMENTS 1585
3.3.6.8.9.9.9.3.10.1.8 LIMITATIONS 1585
3.3.6.8.9.9.9.3.10.2 COS UNIT DESIGN (CATALOG #P787-0) 1586
3.3.6.8.9.9.9.3.10.2.1 REQUIREMENTS ALLOCATION 1586
3.3.6.8.9.9.9.3.10.2.2 LOCAL ENTITIES DESIGN 1586
3.3.6.8.9.9.9.3.10.2.3 INPUT/OUTPUT 1586
3.3.6.8.9.9.9.3.10.2.4 LOCAL DATA 1586
3.3.6.8.9.9.9.3.10.2.5 PROCESS CONTROL 1586
3.3.6.8.9.9.9.3.10.2.6 PROCESSING 1586
3.3.6.8.9.9.9.3.10.2.7 UTILIZATION OF OTHER ELEMENTS 1587
3.3.6.8.9.9.9.3.10.2.8 LIMITATIONS 1588
3.3.6.8.9.9.9.3.10.3 TAN UNIT DESIGN (CATALOG #P788-0) 1588
3.3.6.8.9.9.9.3.10.3.1 REQUIREMENTS ALLOCATION 1588
3.3.6.8.9.9.9.3.10.3.2 LOCAL ENTITIES DESIGN 1588
3.3.6.8.9.9.9.3.10.3.3 INPUT/OUTPUT 1588
3.3.6.8.9.9.9.3.10.3.4 LOCAL DATA 1588
3.3.6.8.9.9.9.3.10.3.5 PROCESS CONTROL 1588
3.3.6.8.9.9.9.3.10.3.6 PROCESSING 1588
3.3.6.8.9.9.9.3.10.3.7 UTILIZATION OF OTHER ELEMENTS 1589
3.3.6.8.9.9.9.3.10.3.8 LIMITATIONS 1590
3.3.6.8.9.9.9.3.10.4 ARCSIN UNIT DESIGN (CATALOG #P789-0) 1590
3.3.6.8.9.9.9.3.10.4.1 REQUIREMENTS ALLOCATION 1590
3.3.6.8.9.9.9.3.10.4.2 LOCAL ENTITIES DESIGN 1590
3.3.6.8.9.9.9.3.10.4.3 INPUT/OUTPUT 1590
3.3.6.8.9.9.9.3.10.4.4 LOCAL DATA 1590
3.3.6.8.9.9.9.3.10.4.5 PROCESS CONTROL 1591
3.3.6.8.9.9.9.3.10.4.6 PROCESSING 1591
3.3.6.8.9.9.9.3.10.4.7 UTILIZATION OF OTHER ELEMENTS 1591
3.3.6.8.9.9.9.3.10.4.8 LIMITATIONS 1592
3.3.6.8.9.9.9.3.10.5 ARCCOS UNIT DESIGN (CATALOG #P790-0) 1592
3.3.6.8.9.9.9.3.10.5.1 REQUIREMENTS ALLOCATION 1592
3.3.6.8.9.9.9.3.10.5.2 LOCAL ENTITIES DESIGN 1592
3.3.6.8.9.9.9.3.10.5.3 INPUT/OUTPUT 1592
3.3.6.8.9.9.9.3.10.5.4 LOCAL DATA 1593
3.3.6.8.9.9.9.3.10.5.5 PROCESS CONTROL 1593
3.3.6.8.9.9.9.3.10.5.6 PROCESSING 1593

3.3.6.8.9.9.9.3.10.5.7 UTILIZATION OF OTHER ELEMENTS 1593
3.3.6.8.9.9.9.3.10.5.8 LIMITATIONS 1594
3.3.6.8.9.9.9.3.10.6 ARCTAN UNIT DESIGN (CATALOG #P791-0) 1594
3.3.6.8.9.9.9.3.10.6.1 REQUIREMENTS ALLOCATION 1595
3.3.6.8.9.9.9.3.10.6.2 LOCAL ENTITIES DESIGN 1595
3.3.6.8.9.9.9.3.10.6.3 INPUT/OUTPUT 1595
3.3.6.8.9.9.9.3.10.6.4 LOCAL DATA 1595
3.3.6.8.9.9.9.3.10.6.5 PROCESS CONTROL 1595
3.3.6.8.9.9.9.3.10.6.6 PROCESSING 1595
3.3.6.8.9.9.9.3.10.6.7 UTILIZATION OF OTHER ELEMENTS 1595
3.3.6.8.9.9.9.3.10.6.8 LIMITATIONS 1596
3.3.6.8.9.9.9.4 SQUARE ROOT PACKAGE DESIGN (CATALOG #P792-0) 1. . . . 596
3.3.6.8.9.9.9.4.1 REQUIREMENTS ALLOCATION 1597
3.3.6.8.9.9.9.4.2 LOCAL ENTITIES DESIGN 1597
3.3.6.8.9.9.9.4.3 INPUT/OUTPUT 1597
3.3.6.8.9.9.9.4.4 LOCAL DATA 1597
3.3.6.8.9.9.9.4.5 PROCESS CONTROL 1597
3.3.6.8.9.9.9.4.6 PROCESSING 1598
3.3.6.8.9.9.9.4.7 UTILIZATION OF OTHER ELEMENTS 1598
3.3.6.8.9.9.9.4.8 LIMITATIONS 1598
3.3.6.8.9.9.9.4.9 LLCSC DESIGN 1598
3.3.6.8.9.9.9.4.10 UNIT DESIGN 1598
3.3.6.8.9.9.9.4.10.1 SQRT UNIT DESIGN 1598
3.3.6.8.9.9.9.4.10.1.1 REQUIREMENTS ALLOCATION 1598
3.3.6.8.9.9.9.4.10.1.2 LOCAL ENTITIES DESIGN 1599
3.3.6.8.9.9.9.4.10.1.3 INPUT/OUTPUT 1599
3.3.6.8.9.9.9.4.10.1.4 LOCAL DATA 1599
3.3.6.8.9.9.9.4.10.1.5 PROCESS CONTROL 1599
3.3.6.8.9.9.9.4.10.1.6 PROCESSING 1599
3.3.6.8.9.9.9.4.10.1.7 UTILIZATION OF OTHER ELEMENTS 1599
3.3.6.8.9.9.9.4.10.1.8 LIMITATIONS 1600
3.3.6.8.9.9.9.5 BASE 10 LOGARITHM PACKAGE DESIGN (CATALOG #P793-0) 1600
3.3.6.8.9.9.9.5.1 REQUIREMENTS ALLOCATION 1601
3.3.6.8.9.9.9.5.2 LOCAL ENTITIES DESIGN 1601
3.3.6.8.9.9.9.5.3 INPUT/OUTPUT 1601
3.3.6.8.9.9.9.5.4 LOCAL DATA 1601
3.3.6.8.9.9.9.5.5 PROCESS CONTROL 1601
3.3.6.8.9.9.9.5.6 PROCESSING 1601
3.3.6.8.9.9.9.5.7 UTILIZATION OF OTHER ELEMENTS 1602
3.3.6.8.9.9.9.5.8 LIMITATIONS 1602
3.3.6.8.9.9.9.5.9 LLCSC DESIGN 1602
3.3.6.8.9.9.9.5.10 UNIT DESIGN 1602
3.3.6.8.9.9.9.5.10.1 LOG 10 UNIT DESIGN 1602
3.3.6.8.9.9.9.5.10.1.1 REQUIREMENTS ALLOCATION 1602
3.3.6.8.9.9.9.5.10.1.2 LOCAL ENTITIES DESIGN 1602
3.3.6.8.9.9.9.5.10.1.3 INPUT/OUTPUT 1602
3.3.6.8.9.9.9.5.10.1.4 LOCAL DATA 1603
3.3.6.8.9.9.9.5.10.1.5 PROCESS CONTROL 1603
3.3.6.8.9.9.9.5.10.1.6 PROCESSING 1603
3.3.6.8.9.9.9.5.10.1.7 UTILIZATION OF OTHER ELEMENTS 1603
3.3.6.8.9.9.9.5.10.1.8 LIMITATIONS 1604
3.3.6.8.9.9.9.6 BASE N LOGARITHM PACKAGE DESIGN (CATALOG #P794-0) 1604
3.3.6.8.9.9.9.6.1 REQUIREMENTS ALLOCATION 1604
3.3.6.8.9.9.9.6.2 LOCAL ENTITIES DESIGN 1604
3.3.6.8.9.9.9.6.3 INPUT/OUTPUT 1605

3.3.6.8.9.9.9.6.4 LOCAL DATA 1606
3.3.6.8.9.9.9.6.5 PROCESS CONTROL 1606
3.3.6.8.9.9.9.6.6 PROCESSING 1606
3.3.6.8.9.9.9.6.7 UTILIZATION OF OTHER ELEMENTS 1607
3.3.6.8.9.9.9.6.8 LIMITATIONS 1607
3.3.6.8.9.9.9.6.9 LLCSC DESIGN 1607
3.3.6.8.9.9.9.6.10 UNIT DESIGN 1607
3.3.6.8.9.9.9.6.10.1 LOG_N UNIT DESIGN 1607
3.3.6.8.9.9.9.6.10.1.1 REQUIREMENTS ALLOCATION 1607
3.3.6.8.9.9.9.6.10.1.2 LOCAL ENTITIES DESIGN 1607
3.3.6.8.9.9.9.6.10.1.3 INPUT/OUTPUT 1607
3.3.6.8.9.9.9.6.10.1.4 LOCAL DATA 1608
3.3.6.8.9.9.9.6.10.1.5 PROCESS CONTROL 1608
3.3.6.8.9.9.9.6.10.1.6 PROCESSING 1608
3.3.6.8.9.9.9.6.10.1.7 UTILIZATION OF OTHER ELEMENTS 1609
3.3.6.8.9.9.9.6.10.1.8 LIMITATIONS 1609
3.3.6.8.9.9.10 UNIT DESIGN 1610

3.3.6.8.9.10 CONTINUED_FRACTIONS PACKAGE DESIGN (CATALOG #P730-0) 1611
3.3.6.8.9.10.1 REQUIREMENTS ALLOCATION 1611
3.3.6.8.9.10.2 LOCAL ENTITIES DESIGN 1611
3.3.6.8.9.10.3 INPUT/OUTPUT 1611
3.3.6.8.9.10.4 LOCAL DATA 1611
3.3.6.8.9.10.5 PROCESS CONTROL 1611
3.3.6.8.9.10.6 PROCESSING 1611
3.3.6.8.9.10.7 UTILIZATION OF OTHER ELEMENTS 1611
3.3.6.8.9.10.8 LIMITATIONS 1611
3.3.6.8.9.10.9 LLCSC DESIGN 1612
3.3.6.8.9.10.9.1 CONTINUED_RADIAN_OPERATIONS PACKAGE DESIGN (CATALOG #P731-0) 1612
3.3.6.8.9.10.9.1.1 REQUIREMENTS ALLOCATION 1612
3.3.6.8.9.10.9.1.2 LOCAL ENTITIES DESIGN 1612
3.3.6.8.9.10.9.1.3 INPUT/OUTPUT 1612
3.3.6.8.9.10.9.1.4 LOCAL DATA 1613
3.3.6.8.9.10.9.1.5 PROCESS CONTROL 1613
3.3.6.8.9.10.9.1.6 PROCESSING 1613
3.3.6.8.9.10.9.1.7 UTILIZATION OF OTHER ELEMENTS 1614
3.3.6.8.9.10.9.1.8 LIMITATIONS 1614
3.3.6.8.9.10.9.1.9 LLCSC DESIGN 1614
3.3.6.8.9.10.9.1.10 UNIT DESIGN 1614
3.3.6.8.9.10.10 UNIT DESIGN 1614

3.3.6.8.9.11 CODY_WAITE PACKAGE DESIGN (CATALOG #P880-0) 1615
3.3.6.8.9.11.1 REQUIREMENTS ALLOCATION 1615
3.3.6.8.9.11.2 LOCAL ENTITIES DESIGN 1615
3.3.6.8.9.11.3 INPUT/OUTPUT 1615
3.3.6.8.9.11.4 LOCAL DATA 1615
3.3.6.8.9.11.5 PROCESS CONTROL 1615
3.3.6.8.9.11.6 PROCESSING 1615
3.3.6.8.9.11.7 UTILIZATION OF OTHER ELEMENTS 1615
3.3.6.8.9.11.8 LIMITATIONS 1616
3.3.6.8.9.11.9 LLCSC DESIGN 1616
3.3.6.8.9.11.9.1 CODY_NATURAL_LOG PACKAGE DESIGN (CATALOG #P881-0) 1616
3.3.6.8.9.11.9.1.1 REQUIREMENTS ALLOCATION 1616
3.3.6.8.9.11.9.1.2 LOCAL ENTITIES DESIGN 1616

3.3.6.8.9.11.9.1.3 INPUT/OUTPUT 1616
3.3.6.8.9.11.9.1.4 LOCAL DATA 1617
3.3.6.8.9.11.9.1.5 PROCESS CONTROL 1617
3.3.6.8.9.11.9.1.6 PROCESSING 1617
3.3.6.8.9.11.9.1.7 UTILIZATION OF OTHER ELEMENTS 1619
3.3.6.8.9.11.9.1.8 LIMITATIONS 1619
3.3.6.8.9.11.9.1.9 LLCSC DESIGN 1619
3.3.6.8.9.11.9.1.10 UNIT DESIGN 1619
3.3.6.8.9.11.9.2 CODY_LOG_BASE_N PACKAGE DESIGN (CATALOG #P883-0) 1620
3.3.6.8.9.11.9.2.1 REQUIREMENTS ALLOCATION 1620
3.3.6.8.9.11.9.2.2 LOCAL ENTITIES DESIGN 1620
3.3.6.8.9.11.9.2.3 INPUT/OUTPUT 1620
3.3.6.8.9.11.9.2.4 LOCAL DATA 1621
3.3.6.8.9.11.9.2.5 PROCESS CONTROL 1621
3.3.6.8.9.11.9.2.6 PROCESSING 1621
3.3.6.8.9.11.9.2.7 UTILIZATION OF OTHER ELEMENTS 1622
3.3.6.8.9.11.9.2.8 LIMITATIONS 1622
3.3.6.8.9.11.9.2.9 LLCSC DESIGN 1622
3.3.6.8.9.11.9.2.10 UNIT DESIGN 1622
3.3.6.8.9.11.10 UNIT DESIGN 1622

3.3.6.8.9.12 REDUCTION_OPERATIONS PACKAGE DESIGN (CATALOG #P1080-0) 1623
3.3.6.8.9.12.1 REQUIREMENTS ALLOCATION 1623
3.3.6.8.9.12.2 LOCAL ENTITIES DESIGN 1623
3.3.6.8.9.12.3 INPUT/OUTPUT 1623
3.3.6.8.9.12.4 LOCAL DATA 1624
3.3.6.8.9.12.5 PROCESS CONTROL 1624
3.3.6.8.9.12.6 PROCESSING 1624
3.3.6.8.9.12.7 UTILIZATION OF OTHER ELEMENTS 1624
3.3.6.8.9.12.8 LIMITATIONS 1624
3.3.6.8.9.12.9 LLCSC DESIGN 1624
3.3.6.8.9.12.10 UNIT DESIGN 1624
3.3.6.8.9.12.10.1 SINE_REDUCTION UNIT DESIGN (CATALOG #P1082-0) 1. . . . 624
3.3.6.8.9.12.10.1.1 REQUIREMENTS ALLOCATION 1624
3.3.6.8.9.12.10.1.2 LOCAL ENTITIES DESIGN 1624
3.3.6.8.9.12.10.1.3 INPUT/OUTPUT 1625
3.3.6.8.9.12.10.1.4 LOCAL DATA 1625
3.3.6.8.9.12.10.1.5 PROCESS CONTROL 1625
3.3.6.8.9.12.10.1.6 PROCESSING 1625
3.3.6.8.9.12.10.1.7 UTILIZATION OF OTHER ELEMENTS 1625
3.3.6.8.9.12.10.1.8 LIMITATIONS 1625
3.3.6.8.9.12.10.2 COSINE_REDUCTION UNIT DESIGN (CATALOG #P1084-0) 1625
3.3.6.8.9.12.10.2.1 REQUIREMENTS ALLOCATION 1626
3.3.6.8.9.12.10.2.2 LOCAL ENTITIES DESIGN 1626
3.3.6.8.9.12.10.2.3 INPUT/OUTPUT 1626
3.3.6.8.9.12.10.2.4 LOCAL DATA 1626
3.3.6.8.9.12.10.2.5 PROCESS CONTROL 1626
3.3.6.8.9.12.10.2.6 PROCESSING 1626
3.3.6.8.9.12.10.2.7 UTILIZATION OF OTHER ELEMENTS 1626
3.3.6.8.9.12.10.2.8 LIMITATIONS 1626

3.3.6.8.10 UNIT DESIGN 1627

3.3.6.9 QUATERNION OPERATIONS (PACKAGE BODY) TLCSC (CATALOG #P127-0) 1715
3.3.6.9.1 REQUIREMENTS ALLOCATION 1715
3.3.6.9.2 LOCAL ENTITIES DESIGN 1715
3.3.6.9.3 INPUT/OUTPUT 1715
3.3.6.9.4 LOCAL DATA 1715
3.3.6.9.5 PROCESS CONTROL 1715
3.3.6.9.6 PROCESSING 1715
3.3.6.9.7 UTILIZATION OF OTHER ELEMENTS 1716
3.3.6.9.8 LIMITATIONS 1716
3.3.6.9.9 LLCSC DESIGN 1716
3.3.6.9.10 UNIT DESIGN 1716
3.3.6.9.10.1 QUATERNION_COMPUTED_FROM_EULER_ANGLES (FUNCTION BODY) UNIT DESIGN (CATALOG #P129-0) 1716
3.3.6.9.10.1.1 REQUIREMENTS ALLOCATION 1716
3.3.6.9.10.1.2 LOCAL ENTITIES DESIGN 1716
3.3.6.9.10.1.3 INPUT/OUTPUT 1716
3.3.6.9.10.1.4 LOCAL DATA 1718
3.3.6.9.10.1.5 PROCESS CONTROL 1719
3.3.6.9.10.1.6 PROCESSING 1719
3.3.6.9.10.1.7 UTILIZATION OF OTHER ELEMENTS 1719
3.3.6.9.10.1.8 LIMITATIONS 1720
3.3.6.9.10.2 NORMALIZED_QUATERNION (FUNCTION BODY) UNIT DESIGN (CATALOG #P130-0) 1720
3.3.6.9.10.2.1 REQUIREMENTS ALLOCATION 1720
3.3.6.9.10.2.2 LOCAL ENTITIES DESIGN 1720
3.3.6.9.10.2.3 INPUT/OUTPUT 1720
3.3.6.9.10.2.4 LOCAL DATA 1720
3.3.6.9.10.2.5 PROCESS CONTROL 1721
3.3.6.9.10.2.6 PROCESSING 1721
3.3.6.9.10.2.7 UTILIZATION OF OTHER ELEMENTS 1721
3.3.6.9.10.2.8 LIMITATIONS 1723
3.3.6.9.10.3 "*" (FUNCTION BODY) UNIT DESIGN (CATALOG #P126-0) 1723
3.3.6.9.10.3.1 REQUIREMENTS ALLOCATION 1723
3.3.6.9.10.3.2 LOCAL ENTITIES DESIGN 1723
3.3.6.9.10.3.3 INPUT/OUTPUT 1723
3.3.6.9.10.3.4 LOCAL DATA 1723
3.3.6.9.10.3.5 PROCESS CONTROL 1724
3.3.6.9.10.3.6 PROCESSING 1724
3.3.6.9.10.3.7 UTILIZATION OF OTHER ELEMENTS 1724
3.3.6.9.10.3.8 LIMITATIONS 1724

3.3.7 ABSTRACT MECHANISMS 1729

3.3.7.1 ABSTRACT_DATA_STRUCTURES TLCSC P691 (CATALOG #P330-0) 1731
3.3.7.1.1 REQUIREMENTS ALLOCATION 1731
3.3.7.1.2 LOCAL ENTITIES DESIGN 1731
3.3.7.1.3 INPUT/OUTPUT 1731
3.3.7.1.4 LOCAL DATA 1731
3.3.7.1.5 PROCESS CONTROL 1732
3.3.7.1.6 PROCESSING 1732
3.3.7.1.7 UTILIZATION OF OTHER ELEMENTS 1732
3.3.7.1.8 LIMITATIONS 1732
3.3.7.1.9 LLCSC DESIGN 1732

3.3.7.1.9.1 AVAILABLE_SPACE_LIST_OPERATIONS PACKAGE DESIGN 1732
3.3.7.1.9.1.1 REQUIREMENTS ALLOCATION 1733
3.3.7.1.9.1.2 LOCAL ENTITIES DESIGN 1733
3.3.7.1.9.1.3 INPUT/OUTPUT 1733
3.3.7.1.9.1.4 LOCAL DATA 1734
3.3.7.1.9.1.5 PROCESS CONTROL 1734
3.3.7.1.9.1.6 PROCESSING 1734
3.3.7.1.9.1.7 UTILIZATION OF OTHER ELEMENTS 1734
3.3.7.1.9.1.8 LIMITATIONS 1734
3.3.7.1.9.1.9 LLCSC DESIGN 1735
3.3.7.1.9.1.10 UNIT DESIGN 1735
3.3.7.1.9.2 AVAILABLE_SPACE_LIST_OPERATIONS PACKAGE DESIGN 1735
3.3.7.1.9.2.1 REQUIREMENTS ALLOCATION 1735
3.3.7.1.9.2.2 LOCAL ENTITIES DESIGN 1735
3.3.7.1.9.2.3 INPUT/OUTPUT 1735
3.3.7.1.9.2.4 LOCAL DATA 1736
3.3.7.1.9.2.5 PROCESS CONTROL 1736
3.3.7.1.9.2.6 PROCESSING 1737
3.3.7.1.9.2.7 UTILIZATION OF OTHER ELEMENTS 1737
3.3.7.1.9.2.8 LIMITATIONS 1737
3.3.7.1.9.2.9 LLCSC DESIGN 1737
3.3.7.1.9.2.10 UNIT DESIGN 1737
3.3.7.1.9.2.10.1 NEW_NODE UNIT DESIGN 1737
3.3.7.1.9.2.10.1.1 REQUIREMENTS ALLOCATION 1737
3.3.7.1.9.2.10.1.2 LOCAL ENTITIES DESIGN 1737
3.3.7.1.9.2.10.1.3 INPUT/OUTPUT 1737
3.3.7.1.9.2.10.1.4 LOCAL DATA 1737
3.3.7.1.9.2.10.1.5 PROCESS CONTROL 1738
3.3.7.1.9.2.10.1.6 PROCESSING 1738
3.3.7.1.9.2.10.1.7 UTILIZATION OF OTHER ELEMENTS 1739
3.3.7.1.9.2.10.1.8 LIMITATIONS 1740
3.3.7.1.9.2.10.2 SAVE_NODE UNIT DESIGN 1740
3.3.7.1.9.2.10.2.1 REQUIREMENTS ALLOCATION 1740
3.3.7.1.9.2.10.2.2 LOCAL ENTITIES DESIGN 1740
3.3.7.1.9.2.10.2.3 INPUT/OUTPUT 1740
3.3.7.1.9.2.10.2.4 LOCAL DATA 1740
3.3.7.1.9.2.10.2.5 PROCESS CONTROL 1740
3.3.7.1.9.2.10.2.6 PROCESSING 1740
3.3.7.1.9.2.10.2.7 UTILIZATION OF OTHER ELEMENTS 1741
3.3.7.1.9.2.10.2.8 LIMITATIONS 1742
3.3.7.1.9.2.10.3 SAVE_LIST UNIT DESIGN 1742
3.3.7.1.9.2.10.3.1 REQUIREMENTS ALLOCATION 1742
3.3.7.1.9.2.10.3.2 LOCAL ENTITIES DESIGN 1742
3.3.7.1.9.2.10.3.3 INPUT/OUTPUT 1742
3.3.7.1.9.2.10.3.4 LOCAL DATA 1742
3.3.7.1.9.2.10.3.5 PROCESS CONTROL 1743
3.3.7.1.9.2.10.3.6 PROCESSING 1743
3.3.7.1.9.2.10.3.7 UTILIZATION OF OTHER ELEMENTS 1743
3.3.7.1.9.2.10.3.8 LIMITATIONS 1744
3.3.7.1.9.3 BOUNDED_FIFO_BUFFER PACKAGE DESIGN (CATALOG #P331-0) 1744
3.3.7.1.9.3.1 REQUIREMENTS ALLOCATION 1744
3.3.7.1.9.3.2 LOCAL ENTITIES DESIGN 1744
3.3.7.1.9.3.3 INPUT/OUTPUT 1744
3.3.7.1.9.3.4 LOCAL DATA 1745

3.3.7.1.9.3.5 PROCESS CONTROL 1745
3.3.7.1.9.3.6 PROCESSING 1745
3.3.7.1.9.3.7 UTILIZATION OF OTHER ELEMENTS 1745
3.3.7.1.9.3.8 LIMITATIONS 1746
3.3.7.1.9.3.9 LLCSC DESIGN 1747
3.3.7.1.9.3.10 UNIT DESIGN 1747
3.3.7.1.9.3.10.1 CLEAR_BUFFER UNIT DESIGN 1747
3.3.7.1.9.3.10.1.1 REQUIREMENTS ALLOCATION 1747
3.3.7.1.9.3.10.1.2 LOCAL ENTITIES DESIGN 1747
3.3.7.1.9.3.10.1.3 INPUT/OUTPUT 1747
3.3.7.1.9.3.10.1.4 LOCAL DATA 1747
3.3.7.1.9.3.10.1.5 PROCESS CONTROL 1747
3.3.7.1.9.3.10.1.6 PROCESSING 1747
3.3.7.1.9.3.10.1.7 UTILIZATION OF OTHER ELEMENTS 1748
3.3.7.1.9.3.10.1.8 LIMITATIONS 1748
3.3.7.1.9.3.10.2 ADD_ELEMENT UNIT DESIGN 1748
3.3.7.1.9.3.10.2.1 REQUIREMENTS ALLOCATION 1748
3.3.7.1.9.3.10.2.2 LOCAL ENTITIES DESIGN 1748
3.3.7.1.9.3.10.2.3 INPUT/OUTPUT 1748
3.3.7.1.9.3.10.2.4 LOCAL DATA 1749
3.3.7.1.9.3.10.2.5 PROCESS CONTROL 1749
3.3.7.1.9.3.10.2.6 PROCESSING 1749
3.3.7.1.9.3.10.2.7 UTILIZATION OF OTHER ELEMENTS 1750
3.3.7.1.9.3.10.2.8 LIMITATIONS 1751
3.3.7.1.9.3.10.3 RETRIEVE_ELEMENT UNIT DESIGN 1751
3.3.7.1.9.3.10.3.1 REQUIREMENTS ALLOCATION 1751
3.3.7.1.9.3.10.3.2 LOCAL ENTITIES DESIGN 1751
3.3.7.1.9.3.10.3.3 INPUT/OUTPUT 1751
3.3.7.1.9.3.10.3.4 LOCAL DATA 1752
3.3.7.1.9.3.10.3.5 PROCESS CONTROL 1752
3.3.7.1.9.3.10.3.6 PROCESSING 1752
3.3.7.1.9.3.10.3.7 UTILIZATION OF OTHER ELEMENTS 1752
3.3.7.1.9.3.10.3.8 LIMITATIONS 1753
3.3.7.1.9.3.10.4 PEEK UNIT DESIGN 1754
3.3.7.1.9.3.10.4.1 REQUIREMENTS ALLOCATION 1754
3.3.7.1.9.3.10.4.2 LOCAL ENTITIES DESIGN 1754
3.3.7.1.9.3.10.4.3 INPUT/OUTPUT 1754
3.3.7.1.9.3.10.4.4 LOCAL DATA 1754
3.3.7.1.9.3.10.4.5 PROCESS CONTROL 1754
3.3.7.1.9.3.10.4.6 PROCESSING 1754
3.3.7.1.9.3.10.4.7 UTILIZATION OF OTHER ELEMENTS 1755
3.3.7.1.9.3.10.4.8 LIMITATIONS 1756
3.3.7.1.9.3.10.5 BUFFER_STATUS UNIT DESIGN 1756
3.3.7.1.9.3.10.5.1 REQUIREMENTS ALLOCATION 1756
3.3.7.1.9.3.10.5.2 LOCAL ENTITIES DESIGN 1757
3.3.7.1.9.3.10.5.3 INPUT/OUTPUT 1757
3.3.7.1.9.3.10.5.4 LOCAL DATA 1757
3.3.7.1.9.3.10.5.5 PROCESS CONTROL 1757
3.3.7.1.9.3.10.5.6 PROCESSING 1757
3.3.7.1.9.3.10.5.7 UTILIZATION OF OTHER ELEMENTS 1758
3.3.7.1.9.3.10.5.8 LIMITATIONS 1759
3.3.7.1.9.3.10.6 BUFFER_LENGTH UNIT DESIGN 1759
3.3.7.1.9.3.10.6.1 REQUIREMENTS ALLOCATION 1759
3.3.7.1.9.3.10.6.2 LOCAL ENTITIES DESIGN 1759
3.3.7.1.9.3.10.6.3 INPUT/OUTPUT 1759
3.3.7.1.9.3.10.6.4 LOCAL DATA 1759
3.3.7.1.9.3.10.6.5 PROCESS CONTROL 1759

3.3.7.1.9.3.10.6.6 PROCESSING 1759
3.3.7.1.9.3.10.6.7 UTILIZATION OF OTHER ELEMENTS 1760
3.3.7.1.9.3.10.6.8 LIMITATIONS 1760
3.3.7.1.9.4 UNBOUNDED_FIFO_BUFFER PACKAGE DESIGN (CATALOG #P332-0) 1760
3.3.7.1.9.4.1 REQUIREMENTS ALLOCATION 1761
3.3.7.1.9.4.2 LOCAL ENTITIES DESIGN 1761
3.3.7.1.9.4.3 INPUT/OUTPUT 1762
3.3.7.1.9.4.4 LOCAL DATA 1762
3.3.7.1.9.4.5 PROCESS CONTROL 1762
3.3.7.1.9.4.6 PROCESSING 1762
3.3.7.1.9.4.7 UTILIZATION OF OTHER ELEMENTS 1764
3.3.7.1.9.4.8 LIMITATIONS 1765
3.3.7.1.9.4.9 LLCSC DESIGN 1765
3.3.7.1.9.4.10 UNIT DESIGN 1766
3.3.7.1.9.4.10.1 INITIALIZE_BUFFER UNIT DESIGN 1766
3.3.7.1.9.4.10.1.1 REQUIREMENTS ALLOCATION 1766
3.3.7.1.9.4.10.1.2 LOCAL ENTITIES DESIGN 1766
3.3.7.1.9.4.10.1.3 INPUT/OUTPUT 1766
3.3.7.1.9.4.10.1.4 LOCAL DATA 1766
3.3.7.1.9.4.10.1.5 PROCESS CONTROL 1766
3.3.7.1.9.4.10.1.6 PROCESSING 1766
3.3.7.1.9.4.10.1.7 UTILIZATION OF OTHER ELEMENTS 1767
3.3.7.1.9.4.10.1.8 LIMITATIONS 1768
3.3.7.1.9.4.10.2 CLEAR_BUFFER UNIT DESIGN 1768
3.3.7.1.9.4.10.2.1 REQUIREMENTS ALLOCATION 1768
3.3.7.1.9.4.10.2.2 LOCAL ENTITIES DESIGN 1768
3.3.7.1.9.4.10.2.3 INPUT/OUTPUT 1768
3.3.7.1.9.4.10.2.4 LOCAL DATA 1768
3.3.7.1.9.4.10.2.5 PROCESS CONTROL 1769
3.3.7.1.9.4.10.2.6 PROCESSING 1769
3.3.7.1.9.4.10.2.7 UTILIZATION OF OTHER ELEMENTS 1769
3.3.7.1.9.4.10.2.8 LIMITATIONS 1770
3.3.7.1.9.4.10.3 FREE_MEMORY UNIT DESIGN 1770
3.3.7.1.9.4.10.3.1 REQUIREMENTS ALLOCATION 1771
3.3.7.1.9.4.10.3.2 LOCAL ENTITIES DESIGN 1771
3.3.7.1.9.4.10.3.3 INPUT/OUTPUT 1771
3.3.7.1.9.4.10.3.4 LOCAL DATA 1771
3.3.7.1.9.4.10.3.5 PROCESS CONTROL 1771
3.3.7.1.9.4.10.3.6 PROCESSING 1771
3.3.7.1.9.4.10.3.7 UTILIZATION OF OTHER ELEMENTS 1772
3.3.7.1.9.4.10.3.8 LIMITATIONS 1773
3.3.7.1.9.4.10.4 ADD_ELEMENT UNIT DESIGN 1773
3.3.7.1.9.4.10.4.1 REQUIREMENTS ALLOCATION 1773
3.3.7.1.9.4.10.4.2 LOCAL ENTITIES DESIGN 1773
3.3.7.1.9.4.10.4.3 INPUT/OUTPUT 1774
3.3.7.1.9.4.10.4.4 LOCAL DATA 1774
3.3.7.1.9.4.10.4.5 PROCESS CONTROL 1774
3.3.7.1.9.4.10.4.6 PROCESSING 1774
3.3.7.1.9.4.10.4.7 UTILIZATION OF OTHER ELEMENTS 1775
3.3.7.1.9.4.10.4.8 LIMITATIONS 1776
3.3.7.1.9.4.10.5 RETRIEVE_ELEMENT UNIT DESIGN 1776
3.3.7.1.9.4.10.5.1 REQUIREMENTS ALLOCATION 1776
3.3.7.1.9.4.10.5.2 LOCAL ENTITIES DESIGN 1776
3.3.7.1.9.4.10.5.3 INPUT/OUTPUT 1777
3.3.7.1.9.4.10.5.4 LOCAL DATA 1777
3.3.7.1.9.4.10.5.5 PROCESS CONTROL 1777

3.3.7.1.9.4.10.5.6 PROCESSING 1777
3.3.7.1.9.4.10.5.7 UTILIZATION OF OTHER ELEMENTS 1778
3.3.7.1.9.4.10.5.8 LIMITATIONS 1779
3.3.7.1.9.4.10.6 PEEK UNIT DESIGN 1779
3.3.7.1.9.4.10.6.1 REQUIREMENTS ALLOCATION 1779
3.3.7.1.9.4.10.6.2 LOCAL ENTITIES DESIGN 1780
3.3.7.1.9.4.10.6.3 INPUT/OUTPUT 1780
3.3.7.1.9.4.10.6.4 LOCAL DATA 1780
3.3.7.1.9.4.10.6.5 PROCESS CONTROL 1780
3.3.7.1.9.4.10.6.6 PROCESSING 1780
3.3.7.1.9.4.10.6.7 UTILIZATION OF OTHER ELEMENTS 1781
3.3.7.1.9.4.10.6.8 LIMITATIONS 1781
3.3.7.1.9.4.10.7 BUFFER_STATUS UNIT DESIGN 1782
3.3.7.1.9.4.10.7.1 REQUIREMENTS ALLOCATION 1782
3.3.7.1.9.4.10.7.2 LOCAL ENTITIES DESIGN 1782
3.3.7.1.9.4.10.7.3 INPUT/OUTPUT 1782
3.3.7.1.9.4.10.7.4 LOCAL DATA 1782
3.3.7.1.9.4.10.7.5 PROCESS CONTROL 1782
3.3.7.1.9.4.10.7.6 PROCESSING 1782
3.3.7.1.9.4.10.7.7 UTILIZATION OF OTHER ELEMENTS 1783
3.3.7.1.9.4.10.7.8 LIMITATIONS 1784
3.3.7.1.9.4.10.8 BUFFER_LENGTH UNIT DESIGN 1784
3.3.7.1.9.4.10.8.1 REQUIREMENTS ALLOCATION 1784
3.3.7.1.9.4.10.8.2 LOCAL ENTITIES DESIGN 1784
3.3.7.1.9.4.10.8.3 INPUT/OUTPUT 1784
3.3.7.1.9.4.10.8.4 LOCAL DATA 1784
3.3.7.1.9.4.10.8.5 PROCESS CONTROL 1784
3.3.7.1.9.4.10.8.6 PROCESSING 1785
3.3.7.1.9.4.10.8.7 UTILIZATION OF OTHER ELEMENTS 1785
3.3.7.1.9.4.10.8.8 LIMITATIONS 1786
3.3.7.1.9.4.10.9 DOT_NEXT UNIT DESIGN 1786
3.3.7.1.9.4.10.9.1 REQUIREMENTS ALLOCATION 1786
3.3.7.1.9.4.10.9.2 LOCAL ENTITIES DESIGN 1786
3.3.7.1.9.4.10.9.3 INPUT/OUTPUT 1786
3.3.7.1.9.4.10.9.4 LOCAL DATA 1786
3.3.7.1.9.4.10.9.5 PROCESS CONTROL 1787
3.3.7.1.9.4.10.9.6 PROCESSING 1787
3.3.7.1.9.4.10.9.7 UTILIZATION OF OTHER ELEMENTS 1787
3.3.7.1.9.4.10.9.8 LIMITATIONS 1787
3.3.7.1.9.4.10.10 SET_NEXT UNIT DESIGN 1787
3.3.7.1.9.4.10.10.1 REQUIREMENTS ALLOCATION 1787
3.3.7.1.9.4.10.10.2 LOCAL ENTITIES DESIGN 1788
3.3.7.1.9.4.10.10.3 INPUT/OUTPUT 1788
3.3.7.1.9.4.10.10.4 LOCAL DATA 1788
3.3.7.1.9.4.10.10.5 PROCESS CONTROL 1788
3.3.7.1.9.4.10.10.6 PROCESSING 1788
3.3.7.1.9.4.10.10.7 UTILIZATION OF OTHER ELEMENTS 1788
3.3.7.1.9.4.10.10.8 LIMITATIONS 1789
3.3.7.1.9.5 NONBLOCKING_CIRCULAR_BUFFER PACKAGE DESIGN (CATALOG #P333-0) 1789
3.3.7.1.9.5.1 REQUIREMENTS ALLOCATION 1789
3.3.7.1.9.5.2 LOCAL ENTITIES DESIGN 1789
3.3.7.1.9.5.3 INPUT/OUTPUT 1789
3.3.7.1.9.5.4 LOCAL DATA 1790
3.3.7.1.9.5.5 PROCESS CONTROL 1790
3.3.7.1.9.5.6 PROCESSING 1790
3.3.7.1.9.5.7 UTILIZATION OF OTHER ELEMENTS 1790

3.3.7.1.9.5.8 LIMITATIONS 1791
3.3.7.1.9.5.9 LLCSC DESIGN 1791
3.3.7.1.9.5.10 UNIT DESIGN 1792
3.3.7.1.9.5.10.1 CLEAR_BUFFER UNIT DESIGN 1792
3.3.7.1.9.5.10.1.1 REQUIREMENTS ALLOCATION 1792
3.3.7.1.9.5.10.1.2 LOCAL ENTITIES DESIGN 1792
3.3.7.1.9.5.10.1.3 INPUT/OUTPUT 1792
3.3.7.1.9.5.10.1.4 LOCAL DATA 1792
3.3.7.1.9.5.10.1.5 PROCESS CONTROL 1792
3.3.7.1.9.5.10.1.6 PROCESSING 1792
3.3.7.1.9.5.10.1.7 UTILIZATION OF OTHER ELEMENTS 1793
3.3.7.1.9.5.10.1.8 LIMITATIONS 1793
3.3.7.1.9.5.10.2 ADD_ELEMENT UNIT DESIGN 1793
3.3.7.1.9.5.10.2.1 REQUIREMENTS ALLOCATION 1794
3.3.7.1.9.5.10.2.2 LOCAL ENTITIES DESIGN 1794
3.3.7.1.9.5.10.2.3 INPUT/OUTPUT 1794
3.3.7.1.9.5.10.2.4 LOCAL DATA 1794
3.3.7.1.9.5.10.2.5 PROCESS CONTROL 1794
3.3.7.1.9.5.10.2.6 PROCESSING 1794
3.3.7.1.9.5.10.2.7 UTILIZATION OF OTHER ELEMENTS 1795
3.3.7.1.9.5.10.2.8 LIMITATIONS 1796
3.3.7.1.9.5.10.3 RETRIEVE_ELEMENT UNIT DESIGN 1796
3.3.7.1.9.5.10.3.1 REQUIREMENTS ALLOCATION 1796
3.3.7.1.9.5.10.3.2 LOCAL ENTITIES DESIGN 1796
3.3.7.1.9.5.10.3.3 INPUT/OUTPUT 1797
3.3.7.1.9.5.10.3.4 LOCAL DATA 1797
3.3.7.1.9.5.10.3.5 PROCESS CONTROL 1797
3.3.7.1.9.5.10.3.6 PROCESSING 1797
3.3.7.1.9.5.10.3.7 UTILIZATION OF OTHER ELEMENTS 1798
3.3.7.1.9.5.10.3.8 LIMITATIONS 1799
3.3.7.1.9.5.10.4 PEEK UNIT DESIGN 1799
3.3.7.1.9.5.10.4.1 REQUIREMENTS ALLOCATION 1799
3.3.7.1.9.5.10.4.2 LOCAL ENTITIES DESIGN 1799
3.3.7.1.9.5.10.4.3 INPUT/OUTPUT 1799
3.3.7.1.9.5.10.4.4 LOCAL DATA 1800
3.3.7.1.9.5.10.4.5 PROCESS CONTROL 1800
3.3.7.1.9.5.10.4.6 PROCESSING 1800
3.3.7.1.9.5.10.4.7 UTILIZATION OF OTHER ELEMENTS 1801
3.3.7.1.9.5.10.4.8 LIMITATIONS 1802
3.3.7.1.9.5.10.5 BUFFER_STATUS UNIT DESIGN 1802
3.3.7.1.9.5.10.5.1 REQUIREMENTS ALLOCATION 1802
3.3.7.1.9.5.10.5.2 LOCAL ENTITIES DESIGN 1802
3.3.7.1.9.5.10.5.3 INPUT/OUTPUT 1803
3.3.7.1.9.5.10.5.4 LOCAL DATA 1803
3.3.7.1.9.5.10.5.5 PROCESS CONTROL 1803
3.3.7.1.9.5.10.5.6 PROCESSING 1803
3.3.7.1.9.5.10.5.7 UTILIZATION OF OTHER ELEMENTS 1804
3.3.7.1.9.5.10.5.8 LIMITATIONS 1804
3.3.7.1.9.5.10.6 BUFFER_LENGTH UNIT DESIGN 1805
3.3.7.1.9.5.10.6.1 REQUIREMENTS ALLOCATION 1805
3.3.7.1.9.5.10.6.2 LOCAL ENTITIES DESIGN 1805
3.3.7.1.9.5.10.6.3 INPUT/OUTPUT 1805
3.3.7.1.9.5.10.6.4 LOCAL DATA 1805
3.3.7.1.9.5.10.6.5 PROCESS CONTROL 1805
3.3.7.1.9.5.10.6.6 PROCESSING 1805
3.3.7.1.9.5.10.6.7 UTILIZATION OF OTHER ELEMENTS 1805
3.3.7.1.9.5.10.6.8 LIMITATIONS 1806

3.3.7.1.9.6 UNBOUNDED_PRIORITY_QUEUE PACKAGE DESIGN (CATALOG #P334-0) 1806
3.3.7.1.9.6.1 REQUIREMENTS ALLOCATION 1807
3.3.7.1.9.6.2 LOCAL ENTITIES DESIGN 1807
3.3.7.1.9.6.3 INPUT/OUTPUT 1807
3.3.7.1.9.6.4 LOCAL DATA 1808
3.3.7.1.9.6.5 PROCESS CONTROL 1809
3.3.7.1.9.6.6 PROCESSING 1809
3.3.7.1.9.6.7 UTILIZATION OF OTHER ELEMENTS 1810
3.3.7.1.9.6.8 LIMITATIONS 1812
3.3.7.1.9.6.9 LLCSC DESIGN 1812
3.3.7.1.9.6.10 UNIT DESIGN 1812
3.3.7.1.9.6.10.1 INITIALIZE UNIT DESIGN 1812
3.3.7.1.9.6.10.1.1 REQUIREMENTS ALLOCATION 1812
3.3.7.1.9.6.10.1.2 LOCAL ENTITIES DESIGN 1812
3.3.7.1.9.6.10.1.3 INPUT/OUTPUT 1812
3.3.7.1.9.6.10.1.4 LOCAL DATA 1812
3.3.7.1.9.6.10.1.5 PROCESS CONTROL 1813
3.3.7.1.9.6.10.1.6 PROCESSING 1813
3.3.7.1.9.6.10.1.7 UTILIZATION OF OTHER ELEMENTS 1813
3.3.7.1.9.6.10.1.8 LIMITATIONS 1814
3.3.7.1.9.6.10.2 CLEAR_QUEUE UNIT DESIGN 1814
3.3.7.1.9.6.10.2.1 REQUIREMENTS ALLOCATION 1814
3.3.7.1.9.6.10.2.2 LOCAL ENTITIES DESIGN 1815
3.3.7.1.9.6.10.2.3 INPUT/OUTPUT 1815
3.3.7.1.9.6.10.2.4 LOCAL DATA 1815
3.3.7.1.9.6.10.2.5 PROCESS CONTROL 1815
3.3.7.1.9.6.10.2.6 PROCESSING 1815
3.3.7.1.9.6.10.2.7 UTILIZATION OF OTHER ELEMENTS 1816
3.3.7.1.9.6.10.2.8 LIMITATIONS 1817
3.3.7.1.9.6.10.3 FREE_MEMORY UNIT DESIGN 1817
3.3.7.1.9.6.10.3.1 REQUIREMENTS ALLOCATION 1817
3.3.7.1.9.6.10.3.2 LOCAL ENTITIES DESIGN 1817
3.3.7.1.9.6.10.3.3 INPUT/OUTPUT 1817
3.3.7.1.9.6.10.3.4 LOCAL DATA 1817
3.3.7.1.9.6.10.3.5 PROCESS CONTROL 1818
3.3.7.1.9.6.10.3.6 PROCESSING 1818
3.3.7.1.9.6.10.3.7 UTILIZATION OF OTHER ELEMENTS 1818
3.3.7.1.9.6.10.3.8 LIMITATIONS 1819
3.3.7.1.9.6.10.4 ADD_ELEMENT UNIT DESIGN 1819
3.3.7.1.9.6.10.4.1 REQUIREMENTS ALLOCATION 1820
3.3.7.1.9.6.10.4.2 LOCAL ENTITIES DESIGN 1820
3.3.7.1.9.6.10.4.3 INPUT/OUTPUT 1820
3.3.7.1.9.6.10.4.4 LOCAL DATA 1820
3.3.7.1.9.6.10.4.5 PROCESS CONTROL 1820
3.3.7.1.9.6.10.4.6 PROCESSING 1820
3.3.7.1.9.6.10.4.7 UTILIZATION OF OTHER ELEMENTS 1821
3.3.7.1.9.6.10.4.8 LIMITATIONS 1823
3.3.7.1.9.6.10.5 RETRIEVE_ELEMENT UNIT DESIGN 1823
3.3.7.1.9.6.10.5.1 REQUIREMENTS ALLOCATION 1823
3.3.7.1.9.6.10.5.2 LOCAL ENTITIES DESIGN 1823
3.3.7.1.9.6.10.5.3 INPUT/OUTPUT 1823
3.3.7.1.9.6.10.5.4 LOCAL DATA 1824
3.3.7.1.9.6.10.5.5 PROCESS CONTROL 1824
3.3.7.1.9.6.10.5.6 PROCESSING 1824
3.3.7.1.9.6.10.5.7 UTILIZATION OF OTHER ELEMENTS 1825
3.3.7.1.9.6.10.5.8 LIMITATIONS 1826

3.3.7.1.9.6.10.6 PEEK UNIT DESIGN 1826
3.3.7.1.9.6.10.6.1 REQUIREMENTS ALLOCATION 1826
3.3.7.1.9.6.10.6.2 LOCAL ENTITIES DESIGN 1826
3.3.7.1.9.6.10.6.3 INPUT/OUTPUT 1826
3.3.7.1.9.6.10.6.4 LOCAL DATA 1827
3.3.7.1.9.6.10.6.5 PROCESS CONTROL 1827
3.3.7.1.9.6.10.6.6 PROCESSING 1827
3.3.7.1.9.6.10.6.7 UTILIZATION OF OTHER ELEMENTS 1827
3.3.7.1.9.6.10.6.8 LIMITATIONS 1828
3.3.7.1.9.6.10.7 QUEUE_STATUS UNIT DESIGN 1828
3.3.7.1.9.6.10.7.1 REQUIREMENTS ALLOCATION 1829
3.3.7.1.9.6.10.7.2 LOCAL ENTITIES DESIGN 1829
3.3.7.1.9.6.10.7.3 INPUT/OUTPUT 1829
3.3.7.1.9.6.10.7.4 LOCAL DATA 1829
3.3.7.1.9.6.10.7.5 PROCESS CONTROL 1829
3.3.7.1.9.6.10.7.6 PROCESSING 1829
3.3.7.1.9.6.10.7.7 UTILIZATION OF OTHER ELEMENTS 1830
3.3.7.1.9.6.10.7.8 LIMITATIONS 1830
3.3.7.1.9.6.10.8 QUEUE_LENGTH UNIT DESIGN 1830
3.3.7.1.9.6.10.8.1 REQUIREMENTS ALLOCATION 1830
3.3.7.1.9.6.10.8.2 LOCAL ENTITIES DESIGN 1831
3.3.7.1.9.6.10.8.3 INPUT/OUTPUT 1831
3.3.7.1.9.6.10.8.4 LOCAL DATA 1831
3.3.7.1.9.6.10.8.5 PROCESS CONTROL 1831
3.3.7.1.9.6.10.8.6 PROCESSING 1831
3.3.7.1.9.6.10.8.7 UTILIZATION OF OTHER ELEMENTS 1832
3.3.7.1.9.6.10.8.8 LIMITATIONS 1832
3.3.7.1.9.6.10.9 DOT_NEXT UNIT DESIGN 1832
3.3.7.1.9.6.10.9.1 REQUIREMENTS ALLOCATION 1833
3.3.7.1.9.6.10.9.2 LOCAL ENTITIES DESIGN 1833
3.3.7.1.9.6.10.9.3 INPUT/OUTPUT 1833
3.3.7.1.9.6.10.9.4 LOCAL DATA 1833
3.3.7.1.9.6.10.9.5 PROCESS CONTROL 1833
3.3.7.1.9.6.10.9.6 PROCESSING 1833
3.3.7.1.9.6.10.9.7 UTILIZATION OF OTHER ELEMENTS 1833
3.3.7.1.9.6.10.9.8 LIMITATIONS 1834
3.3.7.1.9.6.10.10 SET_NEXT UNIT DESIGN 1834
3.3.7.1.9.6.10.10.1 REQUIREMENTS ALLOCATION 1834
3.3.7.1.9.6.10.10.2 LOCAL ENTITIES DESIGN 1834
3.3.7.1.9.6.10.10.3 INPUT/OUTPUT 1834
3.3.7.1.9.6.10.10.4 LOCAL DATA 1834
3.3.7.1.9.6.10.10.5 PROCESS CONTROL 1834
3.3.7.1.9.6.10.10.6 PROCESSING 1835
3.3.7.1.9.6.10.10.7 UTILIZATION OF OTHER ELEMENTS 1835
3.3.7.1.9.6.10.10.8 LIMITATIONS 1835
3.3.7.1.9.7 BOUNDED_STACK PACKAGE DESIGN (CATALOG #P335-0) 1835
3.3.7.1.9.7.1 REQUIREMENTS ALLOCATION 1835
3.3.7.1.9.7.2 LOCAL ENTITIES DESIGN 1836
3.3.7.1.9.7.3 INPUT/OUTPUT 1836
3.3.7.1.9.7.4 LOCAL DATA 1836
3.3.7.1.9.7.5 PROCESS CONTROL 1836
3.3.7.1.9.7.6 PROCESSING 1836
3.3.7.1.9.7.7 UTILIZATION OF OTHER ELEMENTS 1837
3.3.7.1.9.7.8 LIMITATIONS 1838
3.3.7.1.9.7.9 LLCSC DESIGN 1838
3.3.7.1.9.7.10 UNIT DESIGN 1838

3.3.7.1.9.7.10.1 CLEAR_STACK UNIT DESIGN 1838
3.3.7.1.9.7.10.1.1 REQUIREMENTS ALLOCATION 1838
3.3.7.1.9.7.10.1.2 LOCAL ENTITIES DESIGN 1838
3.3.7.1.9.7.10.1.3 INPUT/OUTPUT 1838
3.3.7.1.9.7.10.1.4 LOCAL DATA 1838
3.3.7.1.9.7.10.1.5 PROCESS CONTROL 1839
3.3.7.1.9.7.10.1.6 PROCESSING 1839
3.3.7.1.9.7.10.1.7 UTILIZATION OF OTHER ELEMENTS 1839
3.3.7.1.9.7.10.1.8 LIMITATIONS 1839
3.3.7.1.9.7.10.2 ADD_ELEMENT UNIT DESIGN 1840
3.3.7.1.9.7.10.2.1 REQUIREMENTS ALLOCATION 1840
3.3.7.1.9.7.10.2.2 LOCAL ENTITIES DESIGN 1840
3.3.7.1.9.7.10.2.3 INPUT/OUTPUT 1840
3.3.7.1.9.7.10.2.4 LOCAL DATA 1840
3.3.7.1.9.7.10.2.5 PROCESS CONTROL 1840
3.3.7.1.9.7.10.2.6 PROCESSING 1840
3.3.7.1.9.7.10.2.7 UTILIZATION OF OTHER ELEMENTS 1841
3.3.7.1.9.7.10.2.8 LIMITATIONS 1842
3.3.7.1.9.7.10.3 RETRIEVE_ELEMENT UNIT DESIGN 1842
3.3.7.1.9.7.10.3.1 REQUIREMENTS ALLOCATION 1842
3.3.7.1.9.7.10.3.2 LOCAL ENTITIES DESIGN 1842
3.3.7.1.9.7.10.3.3 INPUT/OUTPUT 1843
3.3.7.1.9.7.10.3.4 LOCAL DATA 1843
3.3.7.1.9.7.10.3.5 PROCESS CONTROL 1843
3.3.7.1.9.7.10.3.6 PROCESSING 1843
3.3.7.1.9.7.10.3.7 UTILIZATION OF OTHER ELEMENTS 1844
3.3.7.1.9.7.10.3.8 LIMITATIONS 1845
3.3.7.1.9.7.10.4 PEEK UNIT DESIGN 1845
3.3.7.1.9.7.10.4.1 REQUIREMENTS ALLOCATION 1845
3.3.7.1.9.7.10.4.2 LOCAL ENTITIES DESIGN 1845
3.3.7.1.9.7.10.4.3 INPUT/OUTPUT 1845
3.3.7.1.9.7.10.4.4 LOCAL DATA 1846
3.3.7.1.9.7.10.4.5 PROCESS CONTROL 1846
3.3.7.1.9.7.10.4.6 PROCESSING 1846
3.3.7.1.9.7.10.4.7 UTILIZATION OF OTHER ELEMENTS 1846
3.3.7.1.9.7.10.4.8 LIMITATIONS 1848
3.3.7.1.9.7.10.5 STACK_STATUS UNIT DESIGN 1848
3.3.7.1.9.7.10.5.1 REQUIREMENTS ALLOCATION 1848
3.3.7.1.9.7.10.5.2 LOCAL ENTITIES DESIGN 1848
3.3.7.1.9.7.10.5.3 INPUT/OUTPUT 1848
3.3.7.1.9.7.10.5.4 LOCAL DATA 1848
3.3.7.1.9.7.10.5.5 PROCESS CONTROL 1849
3.3.7.1.9.7.10.5.6 PROCESSING 1849
3.3.7.1.9.7.10.5.7 UTILIZATION OF OTHER ELEMENTS 1849
3.3.7.1.9.7.10.5.8 LIMITATIONS 1850
3.3.7.1.9.7.10.6 STACK_LENGTH UNIT DESIGN 1850
3.3.7.1.9.7.10.6.1 REQUIREMENTS ALLOCATION 1850
3.3.7.1.9.7.10.6.2 LOCAL ENTITIES DESIGN 1850
3.3.7.1.9.7.10.6.3 INPUT/OUTPUT 1851
3.3.7.1.9.7.10.6.4 LOCAL DATA 1851
3.3.7.1.9.7.10.6.5 PROCESS CONTROL 1851
3.3.7.1.9.7.10.6.6 PROCESSING 1851
3.3.7.1.9.7.10.6.7 UTILIZATION OF OTHER ELEMENTS 1851
3.3.7.1.9.7.10.6.8 LIMITATIONS 1852
3.3.7.1.9.8 UNBOUNDED_STACK PACKAGE DESIGN (CATALOG #P336-0) 1852
3.3.7.1.9.8.1 REQUIREMENTS ALLOCATION 1852

3.3.7.1.9.8.2 LOCAL ENTITIES DESIGN 1852
3.3.7.1.9.8.3 INPUT/OUTPUT 1853
3.3.7.1.9.8.4 LOCAL DATA 1854
3.3.7.1.9.8.5 PROCESS CONTROL 1854
3.3.7.1.9.8.6 PROCESSING 1854
3.3.7.1.9.8.7 UTILIZATION OF OTHER ELEMENTS 1855
3.3.7.1.9.8.8 LIMITATIONS 1857
3.3.7.1.9.8.9 LLCSC DESIGN 1857
3.3.7.1.9.8.10 UNIT DESIGN 1857
3.3.7.1.9.8.10.1 INITIALIZE UNIT DESIGN 1857
3.3.7.1.9.8.10.1.1 REQUIREMENTS ALLOCATION 1857
3.3.7.1.9.8.10.1.2 LOCAL ENTITIES DESIGN 1857
3.3.7.1.9.8.10.1.3 INPUT/OUTPUT 1857
3.3.7.1.9.8.10.1.4 LOCAL DATA 1858
3.3.7.1.9.8.10.1.5 PROCESS CONTROL 1858
3.3.7.1.9.8.10.1.6 PROCESSING 1858
3.3.7.1.9.8.10.1.7 UTILIZATION OF OTHER ELEMENTS 1859
3.3.7.1.9.8.10.1.8 LIMITATIONS 1859
3.3.7.1.9.8.10.2 CLEAR_STACK UNIT DESIGN 1860
3.3.7.1.9.8.10.2.1 REQUIREMENTS ALLOCATION 1860
3.3.7.1.9.8.10.2.2 LOCAL ENTITIES DESIGN 1860
3.3.7.1.9.8.10.2.3 INPUT/OUTPUT 1860
3.3.7.1.9.8.10.2.4 LOCAL DATA 1860
3.3.7.1.9.8.10.2.5 PROCESS CONTROL 1861
3.3.7.1.9.8.10.2.6 PROCESSING 1861
3.3.7.1.9.8.10.2.7 UTILIZATION OF OTHER ELEMENTS 1861
3.3.7.1.9.8.10.2.8 LIMITATIONS 1862
3.3.7.1.9.8.10.3 FREE_MEMORY UNIT DESIGN 1863
3.3.7.1.9.8.10.3.1 REQUIREMENTS ALLOCATION 1863
3.3.7.1.9.8.10.3.2 LOCAL ENTITIES DESIGN 1863
3.3.7.1.9.8.10.3.3 INPUT/OUTPUT 1863
3.3.7.1.9.8.10.3.4 LOCAL DATA 1863
3.3.7.1.9.8.10.3.5 PROCESS CONTROL 1863
3.3.7.1.9.8.10.3.6 PROCESSING 1863
3.3.7.1.9.8.10.3.7 UTILIZATION OF OTHER ELEMENTS 1864
3.3.7.1.9.8.10.3.8 LIMITATIONS 1864
3.3.7.1.9.8.10.4 ADD_ELEMENT UNIT DESIGN 1865
3.3.7.1.9.8.10.4.1 REQUIREMENTS ALLOCATION 1865
3.3.7.1.9.8.10.4.2 LOCAL ENTITIES DESIGN 1865
3.3.7.1.9.8.10.4.3 INPUT/OUTPUT 1865
3.3.7.1.9.8.10.4.4 LOCAL DATA 1865
3.3.7.1.9.8.10.4.5 PROCESS CONTROL 1865
3.3.7.1.9.8.10.4.6 PROCESSING 1865
3.3.7.1.9.8.10.4.7 UTILIZATION OF OTHER ELEMENTS 1866
3.3.7.1.9.8.10.4.8 LIMITATIONS 1867
3.3.7.1.9.8.10.5 RETRIEVE_ELEMENT UNIT DESIGN 1867
3.3.7.1.9.8.10.5.1 REQUIREMENTS ALLOCATION 1868
3.3.7.1.9.8.10.5.2 LOCAL ENTITIES DESIGN 1868
3.3.7.1.9.8.10.5.3 INPUT/OUTPUT 1868
3.3.7.1.9.8.10.5.4 LOCAL DATA 1868
3.3.7.1.9.8.10.5.5 PROCESS CONTROL 1868
3.3.7.1.9.8.10.5.6 PROCESSING 1868
3.3.7.1.9.8.10.5.7 UTILIZATION OF OTHER ELEMENTS 1869
3.3.7.1.9.8.10.5.8 LIMITATIONS 1870
3.3.7.1.9.8.10.6 PEEK UNIT DESIGN 1870
3.3.7.1.9.8.10.6.1 REQUIREMENTS ALLOCATION 1871
3.3.7.1.9.8.10.6.2 LOCAL ENTITIES DESIGN 1871

3.3.7.1.9.8.10.6.3 INPUT/OUTPUT 1871
3.3.7.1.9.8.10.6.4 LOCAL DATA 1871
3.3.7.1.9.8.10.6.5 PROCESS CONTROL 1871
3.3.7.1.9.8.10.6.6 PROCESSING 1871
3.3.7.1.9.8.10.6.7 UTILIZATION OF OTHER ELEMENTS 1872
3.3.7.1.9.8.10.6.8 LIMITATIONS 1873
3.3.7.1.9.8.10.7 STACK_STATUS UNIT DESIGN 1873
3.3.7.1.9.8.10.7.1 REQUIREMENTS ALLOCATION 1873
3.3.7.1.9.8.10.7.2 LOCAL ENTITIES DESIGN 1873
3.3.7.1.9.8.10.7.3 INPUT/OUTPUT 1873
3.3.7.1.9.8.10.7.4 LOCAL DATA 1874
3.3.7.1.9.8.10.7.5 PROCESS CONTROL 1874
3.3.7.1.9.8.10.7.6 PROCESSING 1874
3.3.7.1.9.8.10.7.7 UTILIZATION OF OTHER ELEMENTS 1875
3.3.7.1.9.8.10.7.8 LIMITATIONS 1875
3.3.7.1.9.8.10.8 STACK_LENGTH UNIT DESIGN 1875
3.3.7.1.9.8.10.8.1 REQUIREMENTS ALLOCATION 1875
3.3.7.1.9.8.10.8.2 LOCAL ENTITIES DESIGN 1875
3.3.7.1.9.8.10.8.3 INPUT/OUTPUT 1876
3.3.7.1.9.8.10.8.4 LOCAL DATA 1876
3.3.7.1.9.8.10.8.5 PROCESS CONTROL 1876
3.3.7.1.9.8.10.8.6 PROCESSING 1876
3.3.7.1.9.8.10.8.7 UTILIZATION OF OTHER ELEMENTS 1876
3.3.7.1.9.8.10.8.8 LIMITATIONS 1877
3.3.7.1.9.8.10.9 DOT_NEXT UNIT DESIGN 1877
3.3.7.1.9.8.10.9.1 REQUIREMENTS ALLOCATION 1877
3.3.7.1.9.8.10.9.2 LOCAL ENTITIES DESIGN 1877
3.3.7.1.9.8.10.9.3 INPUT/OUTPUT 1878
3.3.7.1.9.8.10.9.4 LOCAL DATA 1878
3.3.7.1.9.8.10.9.5 PROCESS CONTROL 1878
3.3.7.1.9.8.10.9.6 PROCESSING 1878
3.3.7.1.9.8.10.9.7 UTILIZATION OF OTHER ELEMENTS 1878
3.3.7.1.9.8.10.9.8 LIMITATIONS 1879
3.3.7.1.9.8.10.10 SET_NEXT UNIT DESIGN 1879
3.3.7.1.9.8.10.10.1 REQUIREMENTS ALLOCATION 1879
3.3.7.1.9.8.10.10.2 LOCAL ENTITIES DESIGN 1879
3.3.7.1.9.8.10.10.3 INPUT/OUTPUT 1879
3.3.7.1.9.8.10.10.4 LOCAL DATA 1879
3.3.7.1.9.8.10.10.5 PROCESS CONTROL 1879
3.3.7.1.9.8.10.10.6 PROCESSING 1879
3.3.7.1.9.8.10.10.7 UTILIZATION OF OTHER ELEMENTS 1880
3.3.7.1.9.8.10.10.8 LIMITATIONS 1880
3.3.7.1.10 UNIT DESIGN 1880

3.3.8 GENERAL UTILITIES 1915

3.3.8.1 GENERAL_UTILITIES TLCSC P361 (CATALOG #P267-0) 1917
3.3.8.1.1 REQUIREMENTS ALLOCATION 1917
3.3.8.1.2 LOCAL ENTITIES DESIGN 1917
3.3.8.1.3 INPUT/OUTPUT 1917
3.3.8.1.4 LOCAL DATA 1917
3.3.8.1.5 PROCESS CONTROL 1917
3.3.8.1.6 PROCESSING 1917
3.3.8.1.7 UTILIZATION OF OTHER ELEMENTS 1917
3.3.8.1.8 LIMITATIONS 1917
3.3.8.1.9 LLCSC DESIGN 1918
3.3.8.1.10 UNIT DESIGN 1918

3.3.8.1.10.1 INSTRUCTION_SET_TEST UNIT DESIGN (CATALOG #P268-0) 1918
3.3.8.1.10.1.1 REQUIREMENTS ALLOCATION 1918
3.3.8.1.10.1.2 LOCAL ENTITIES DESIGN 1918
3.3.8.1.10.1.3 INPUT/OUTPUT 1918
3.3.8.1.10.1.4 LOCAL DATA 1919
3.3.8.1.10.1.5 PROCESS CONTROL 1919
3.3.8.1.10.1.6 PROCESSING 1919
3.3.8.1.10.1.7 UTILIZATION OF OTHER ELEMENTS 1919
3.3.8.1.10.1.8 LIMITATIONS 1919

3.3.8.2 COMMUNICATION_PARTS TLCSC P602 (CATALOG #P691-0) 1923
3.3.8.2.1 REQUIREMENTS ALLOCATION 1923
3.3.8.2.2 LOCAL ENTITIES DESIGN 1923
3.3.8.2.3 INPUT/OUTPUT 1923
3.3.8.2.4 LOCAL DATA 1923
3.3.8.2.5 PROCESS CONTROL 1923
3.3.8.2.6 PROCESSING 1923
3.3.8.2.7 UTILIZATION OF OTHER ELEMENTS 1923
3.3.8.2.8 LIMITATIONS 1923
3.3.8.2.9 LLCSC DESIGN 1924
3.3.8.2.9.1 UPDATE_EXCLUSION PACKAGE DESIGN (CATALOG #P692-0) 1924
3.3.8.2.9.1.1 REQUIREMENTS ALLOCATION 1924
3.3.8.2.9.1.2 LOCAL ENTITIES DESIGN 1924
3.3.8.2.9.1.3 INPUT/OUTPUT 1924
3.3.8.2.9.1.4 LOCAL DATA 1925
3.3.8.2.9.1.5 PROCESS CONTROL 1925
3.3.8.2.9.1.6 PROCESSING 1925
3.3.8.2.9.1.7 UTILIZATION OF OTHER ELEMENTS 1927
3.3.8.2.9.1.8 LIMITATIONS 1927
3.3.8.2.9.1.9 LLCSC DESIGN 1928
3.3.8.2.9.1.10 UNIT DESIGN 1928
3.3.8.2.10 UNIT DESIGN 1928

3.3.9 EQUIPMENT INTERFACES 1933

3.3.9.1 CLOCK_HANDLER TLCSC P634 (CATALOG #P270-0) 1935
3.3.9.1.1 REQUIREMENTS ALLOCATION 1935
3.3.9.1.2 LOCAL ENTITIES DESIGN 1935
3.3.9.1.3 INPUT/OUTPUT 1935
3.3.9.1.4 LOCAL DATA 1935
3.3.9.1.5 PROCESS CONTROL 1936
3.3.9.1.6 PROCESSING 1936
3.3.9.1.7 UTILIZATION OF OTHER ELEMENTS 1936
3.3.9.1.8 LIMITATIONS 1937
3.3.9.1.9 LLCSC DESIGN 1937
3.3.9.1.10 UNIT DESIGN 1937
3.3.9.1.10.1 CURRENT_TIME (FUNCTION BODY) UNIT DESIGN 1937
3.3.9.1.10.1.1 REQUIREMENTS ALLOCATION 1937
3.3.9.1.10.1.2 LOCAL ENTITIES DESIGN 1937
3.3.9.1.10.1.3 INPUT/OUTPUT 1937
3.3.9.1.10.1.4 LOCAL DATA 1937
3.3.9.1.10.1.5 PROCESS CONTROL 1937
3.3.9.1.10.1.6 PROCESSING 1938
3.3.9.1.10.1.7 UTILIZATION OF OTHER ELEMENTS 1938
3.3.9.1.10.1.8 LIMITATIONS 1939

3.3.9.1.10.2 CONVERTED_TIME (FUNCTION BODY) UNIT DESIGN 1939
3.3.9.1.10.2.1 REQUIREMENTS ALLOCATION 1939
3.3.9.1.10.2.2 LOCAL ENTITIES DESIGN 1939
3.3.9.1.10.2.3 INPUT/OUTPUT 1939
3.3.9.1.10.2.4 LOCAL DATA 1939
3.3.9.1.10.2.5 PROCESS CONTROL 1940
3.3.9.1.10.2.6 PROCESSING 1940
3.3.9.1.10.2.7 UTILIZATION OF OTHER ELEMENTS 1940
3.3.9.1.10.2.8 LIMITATIONS 1941
3.3.9.1.10.3 RESET_CLOCK (PROCEDURE BODY) UNIT DESIGN 1941
3.3.9.1.10.3.1 REQUIREMENTS ALLOCATION 1941
3.3.9.1.10.3.2 LOCAL ENTITIES DESIGN 1941
3.3.9.1.10.3.3 INPUT/OUTPUT 1941
3.3.9.1.10.3.4 LOCAL DATA 1941
3.3.9.1.10.3.5 PROCESS CONTROL 1941
3.3.9.1.10.3.6 PROCESSING 1941
3.3.9.1.10.3.7 UTILIZATION OF OTHER ELEMENTS 1942
3.3.9.1.10.3.8 LIMITATIONS 1942
3.3.9.1.10.4 SYNCHRONIZE_CLOCK (PROCEDURE BODY) UNIT DESIGN 1943
3.3.9.1.10.4.1 REQUIREMENTS ALLOCATION 1943
3.3.9.1.10.4.2 LOCAL ENTITIES DESIGN 1943
3.3.9.1.10.4.3 INPUT/OUTPUT 1943
3.3.9.1.10.4.4 LOCAL DATA 1943
3.3.9.1.10.4.5 PROCESS CONTROL 1943
3.3.9.1.10.4.6 PROCESSING 1943
3.3.9.1.10.4.7 UTILIZATION OF OTHER ELEMENTS 1944
3.3.9.1.10.4.8 LIMITATIONS 1944
3.3.9.1.10.5 ELAPSED_TIME (FUNCTION BODY) UNIT DESIGN 1944
3.3.9.1.10.5.1 REQUIREMENTS ALLOCATION 1944
3.3.9.1.10.5.2 LOCAL ENTITIES DESIGN 1945
3.3.9.1.10.5.3 INPUT/OUTPUT 1945
3.3.9.1.10.5.4 LOCAL DATA 1945
3.3.9.1.10.5.5 PROCESS CONTROL 1945
3.3.9.1.10.5.6 PROCESSING 1945
3.3.9.1.10.5.7 UTILIZATION OF OTHER ELEMENTS 1946
3.3.9.1.10.5.8 LIMITATIONS 1946

4 (NOT USED) 1951

5 (NOT USED) 1953

6 NOTES 1955

APPENDIX I MODIFICATIONS TO DI-MCCR-80031

I.1 REQUIREMENTS FOR DOCUMENTING DESIGN OF REUSABLE PARTS I-1
I.1.1 PROBLEMS IN USING DI-MCCR-80031 . . . . . I-1
I.1.2 DESIGN CODE HEADER FOR DETAILED DESIGN I-1

SOFTWARE DETAILED DESIGN DOCUMENT FOR THE MISSILE SOFTWARE PARTS OF THE COMMON ADA MISSILE PACKAGES (CAMP) PROJECT

1 SCOPE

1.1 IDENTIFICATION

This Software Detailed Design Document describes the detailed design for the CSCI identified as the Common Ada Missile Packages (CAMP) Missile Software Parts.

The CAMP-1 program was a twelve-month feasibility study which the McDonnell Douglas Astronautics Company performed under contract to the U.S. Air Force Armament Laboratory (AFATL). This project had two primary objectives:

(1) To determine the feasibility of developing reusable missile software components written in Ada; and,

(2) To determine the feasibility of developing an automated or semi-automated missile software generation system.

The CAMP-2 program was a twenty-four-month study which the McDonnell Douglas Astronautics Company performed under contract to the U.S. Air Force Armament Laboratory (AFATL). The primary objectives of this project were:

(1) Prepare the detailed design for those parts receiving top-level design under the CAMP-1 program.

(2) Show that reusable software parts can result in significant productivity and quality improvements.

(3) Implement an 11th missile application using CAMP parts.

The detailed design of this CSCI meets the requirements which resulted from the domain analysis performed during CAMP-1. Although software parts contained in this CSCI are used by the software generator, there are no direct software or hardware interfaces between the two CSCI's.

1.2 PURPOSE

The Missile Software Parts constitute a set of common software identifed under the domain analysis of the CAMP study. These software parts are grouped into Top-Level Computer Software Components (TLCSC's) and are divided into the categories listed below.

a. Data Types - TLCSC's which provide data types used in other TLCSC's or in a user application:

- o Basic Data Types TLCSC
- o Kalman Filter Data Types TLCSC
- o Autopilot Data Types TLCSC

b. Navigation - TLCSC's which provide the basic functionality of a navigation sybsystem.

- o Common Navigation Parts TLCSC
- o North Pointing Navigation Parts TLCSC
- o Wander Angle Navigation Parts TLCSC
- o Direction Cosine Matrix Operations TLCSC

c. Kalman Filter - TLCSC's which provide common Kalman Filter functions.

- o Common Kalman Filter TLCSC
- o Compact-H Kalman Filter TLCSC
- o Complicated-H Kalman Filter TLCSC

d. Guidance and Control - TLCSC's which provide the basic functionality of a guidance and control subsystem.

- o Waypoint Steering TLCSC
- o Autopilot TLCSC

e. Nonguidance Control - TLCSC's which provide the basic functionality of a control subsystem for operations outside of the guidance area.

- o Air Data TLCSC
- o Fuel Control TLCSC

f. Mathematical - TLCSC's which provide a variety of useful mathematical functions such as coordinate and matrix algebra, trigonometric, and signal processing functions.

- o Coordinate Algebra TLCSC
- o Matrix Algebra TLCSC
- o Geometric TLCSC
- o Trigonometry TLCSC
- o Unit Conversion TLCSC
- o External Form Conversion TLCSC
- o Signal Processing TLCSC
- o General Purpose Math TLCSC
- o Polynomial Math TLCSC
- o Quaternion Math TLCSC

g. Abstract Mechanisms - TLCSC's which provide abstract data structures and processes.

- o Data Structures TLCSC

h. General Utilities - TLCSC's which provide other functions needed for missile or other weapons system operation.

- o General Utilities TLCSC
- o Communication Parts TLCSC

i. Equipment Interfaces - TLCSC's which provide functions needed to interface with the hardware:

- o Clock Handler TLCSC

The following lists the TLCSC's identified in the CAMP Top-Level Design Document for which no detailed design exists. Detailed design is not required for these parts either because they deal exclusively with data constants or because they are schematic parts.

a. Data constants - TLCSC's which provide data constants used in a typical application:

- o WGS-72 Engineering TLCSC
- o WGS-72 Metric TLCSC
- o WGS-72 Unitless TLCSC
- o Universal Constants TLCSC
- o Conversion Factors TLCSC

b. Equipment Interfaces - TLCSC's which provide standard interfaces to specific hardware components or to general classes of hardware.

- o Radar Altimeter Interface TLCSC's
- o Data Bus Handler TLCSC

c. Abstract Mechanisms - TLCSC's which provide abstract data structures and processes.

- o Abstract Processes TLCSC (schematic part)

d. Process Management - TLCSC's which provide control of concurrent processes and communication.

- o Asynchronous Control TLCSC (schematic part)

1.3 INTRODUCTION

This Software Detailed Design Document (DDD) provides the detailed design requirements of common missile software parts as stated in the Software Requirements Specification developed under Contract F08635-84-C-0280 of the CAMP Program.

The following figure shows the organization of TLCSC's requiring detailed design and the structure for the major section of this document, Section 3.3 (Detailed Design).

DATA TYPES 3.3.1
Basic Data Types TLCSC 3.3.1.1
Kalman Filter Data Types TLCSC 3.3.1.2
Autopilot Data Types TLCSC 3.3.1.3

NAVIGATION 3.3.2
Common Navigation Parts TLCSC 3.3.2.1
North Pointing Navigation Parts TLCSC 3.3.2.2
Wander Angle Navigation Parts TLCSC 3.3.2.3
Direction Cosine Matrix Operations TLCSC 3.3.2.4

KALMAN FILTER 3.3.3
Common Kalman Filter Parts TLCSC 3.3.3.1
Compact-H Kalman Filter Parts TLCSC 3.3.3.2
Complicated-H Kalman Filter Parts TLCSC 3.3.3.3

GUIDANCE AND CONTROL 3.3.4
Waypoint Steering TLCSC 3.3.4.1
Autopilot TLCSC 3.3.4.2

NONGUIDANCE AND CONTROL 3.3.5
Air Data TLCSC 3.3.5.1
Fuel Control TLCSC 3.3.5.2

MATHEMATICAL 3.3.6
Coordinate Vector/Matrix Algebra TLCSC 3.3.6.1
General Vector/Matrix Algebra TLCSC 3.3.6.2
Trigonometric TLCSC 3.3.6.3
Geometric Algebra TLCSC 3.3.6.4
Data Conversion 3.3.6.5
Unit conversion TLCSC 3.3.6.5.1
External form conversion TLCSC 3.3.6.5.2
Signal Processing TLCSC 3.3.6.6
General Purpose Math TLCSC 3.3.6.7
Polynomial TLCSC 3.3.6.8
Quaternion Operations TLCSC 3.3.6.9

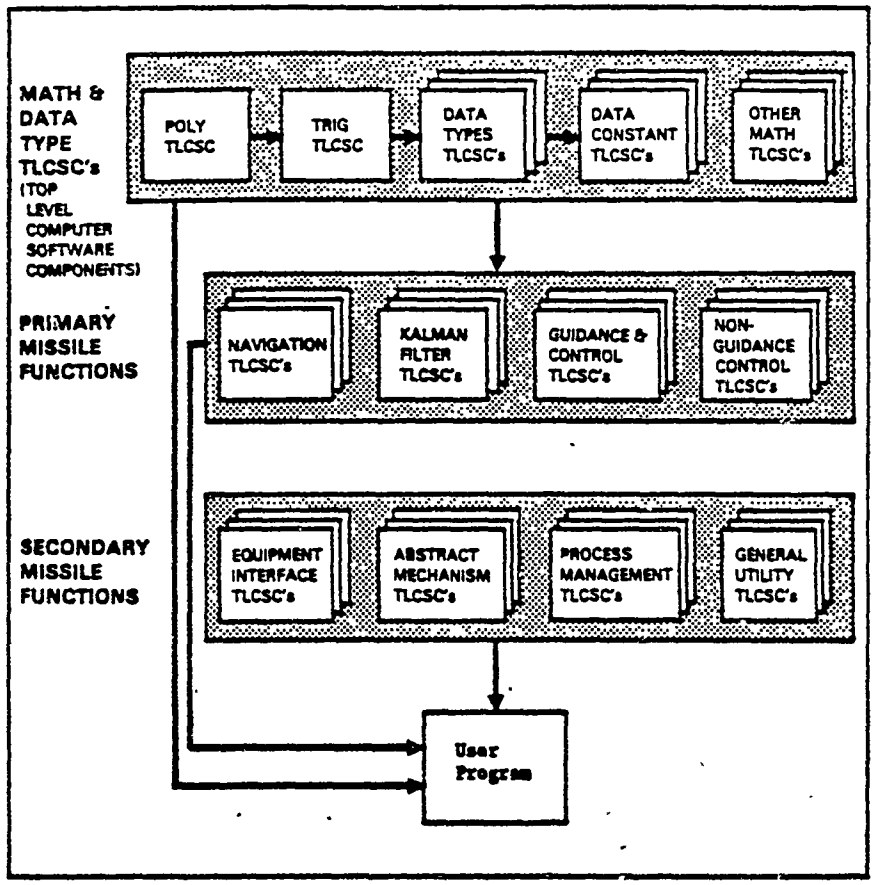
ABSTRACT MECHANISMS 3.3.7
Abstract Data Structures TLCSC 3.3.7.1

GENERAL UTILITIES 3.3.8
General Utilities TLCSC 3.3.8.1
Communication TLCSC 3.3.8.2

EQUIPMENT INTERFACES 3.3.9
Clock Handler TLCSC 3.3.9.1

ORGANIZATION OF CAMP TLCSC'S

The following figure illustrates the top-level architecture and dependencies of the TLCSC's. The top section of this figure shows the mathematical and data type packages. These TLCSC's provide the data types and mathematical operations to a user program or to other TLCSC's within the design, as shown by the arrows. At the center of the figure are the primary missile TLCSC's: Navigation, Kalman Filter, Guidance and Control, and Non-guidance Control. These TLCSC's require data types and mathematical operators from other TLCSC's and provide operations to the user program. On the right of the figure are secondary missile TLCSC's. These also provide operations to a user program, as illustrated by the arrows.



ARCHITECTURAL DEPENDENCIES OF TLCSC'S

(This page intentionally left blank.)

2 APPLICABLE DOCUMENTS

2.1 GOVERNMENT DOCUMENTS

The following documents of the exact issue shown form a part of the specification to the extent described herein. In the event of conflict between the documents referenced herein and the contents of this document, the contents of this document shall be considered a superseding requirement.

STANDARDS

MILITARY

MIL-STD-490	Specification Practices (4 June 1985)
MIL-STD-1815A	Reference Manual for the Ada Programming Language (17 February 1983)
DOD-STD-2167	Defense System Software Development (4 June 1985)

OTHER PUBLICATIONS

CAMP Final Technical Report, Volume 1, 2, & 3 (4 September 1985)

Ada Missile Parts Engineering Expert System Software Requirements Specification (4 September 1985)

Ada Missile Parts Engineering Expert System Software Top-Level Design Document (4 September 1985)

Ada Missile Parts Engineering Expert System Software Detailed Design Document (Draft 30 August 1986)

CAMP Missile Software Parts Requirements Specification (4 September 1985)

CAMP Missile Software Parts Software Top-Level Design Document (4 September 1985)

World Geodetic Survey, 1972 (WGS72). Defense Mapping Agency, DMA 72-1

2.2 NON-GOVERNMENT DOCUMENTS

The following documents of the exact issue shown form a part of the specification to the extend specified herein. In the event of conflict between the documents referenced herein and the contents of this specification, the contents of this specification shall be considered a superseding requirement.

STANDARDS

Ada Design Language Reference Manual ((c) 1983, McDonnell Douglas Astronautics Company)

OTHER PUBLICATIONS:

VAX Ada Programmer's Run-Time Reference Manual (February, 1985, Digital Equipment Corporation)

VAX Ada Language Reference Manual (February, 1985, Digital Equipment Corporation)

3 REQUIREMENTS

3.1 INTERFACE DESIGN

There are no interfaces between this CSCI and other CSCI's or CI's. Interfaces between TLCSC's are described under Section 3.3 (Detailed Design).

3.2 GLOBAL DATA AND DATA TYPES

Global data and data types are provided by TLCSC's in the categories of data constants and data types. The specific global objects and types declared in these TLCSC's are discussed in sections 3.6.1 of the Software Top-Level Design Document (Data Constants) and 3.3.1 of this document (Data Types).

(This page intentionally left blank.)

3.3 DETAILED DESIGN

3.3.1 DATA TYPES

(This page intentionally left blank.)

3.3.1.1 BASIC DATA TYPES (BODY) TLCSC P621 (CATALOG #P1093-0)

This part, which is designed as an Ada package, defines simple data types and special operators for these data types.

The decomposition for this part is the same as that shown in the Top-Level Design Document.

3.3.1.1.1 REQUIREMENTS ALLOCATION

This part meets CAMP requirement R021.

3.3.1.1.2 LOCAL ENTITIES DESIGN

None.

3.3.1.1.3 INPUT/OUTPUT

None.

3.3.1.1.4 LOCAL DATA

None.

3.3.1.1.5 PROCESS CONTROL

Not applicable.

3.3.1.1.6 PROCESSING

The following describes the processing performed by this part:

```
with Conversion_Factors;
package body Basic_Data_Types is

   package CF renames Conversion_Factors;
   use Trig;

-- ----------------------------------------
-- --Define Operators for Basic Data Types-
-- ----------------------------------------

-- --Acceleration * Interval ==> Velocity

   function "*" (Left  :  Feet_per_Second_Squared;
                 Right : Seconds)
                return Feet_Per_Second is
      Temp : Feet_Per_Second;
   begin
      Temp := Feet_Per_Second (Seconds (Left) * Right);
      return Temp;
```

```
   end "*";

   function "*" (Left  :  Meters_per_Second_Squared;
                 Right : Seconds)
                return Meters_Per_Second is
      Temp : Meters_Per_Second;
   begin
      Temp := Meters_Per_Second (Seconds (Left) * Right);
      return Temp;
   end "*";

-- --Acceleration * Sin Cos Ratio ==> Acceleration

   function "*" (Left  : Feet_per_Second_Squared;
                 Right : Trig.Sin_Cos_Ratio) return Feet_per_Second_Squared is
   begin
      return Left * Feet_per_Second_Squared (Right);
   end "*";

   function "*" (Left  : Meters_per_Second_Squared;
                 Right : Trig.Sin_Cos_Ratio)
                return Meters_per_Second_Squared is
   begin
      return Left * Meters_per_Second_Squared (Right);
   end "*";

   function "*" (Left  : Gees;
                 Right : Trig.Sin_Cos_Ratio) return Gees is
   begin
      return Left * Gees (Right);
   end "*";

-- --Acceleration * Tan Ratio ==> Acceleration

   function "*" (Left  : Feet_per_Second_Squared;
                 Right : Trig.Tan_Ratio) return Feet_per_Second_Squared is
   begin
      return Left * Feet_per_Second_Squared (Right);
   end "*";

   function "*" (Left  : Meters_per_Second_Squared;
                 Right : Trig.Tan_Ratio) return Meters_per_Second_Squared is
   begin
      return Left * Meters_per_Second_Squared (Right);
   end "*";

   function "*" (Left  : Gees; Right : Trig.Tan_Ratio) return Gees is
   begin
      return Left * Gees (Right);
   end "*";


-- --Angular_Velocity * Angular_Velocity ==> Angular_Velocity Squared

   function "*" (Left  : Radians_per_Second;
                 Right : Radians_per_Second)
                return Radians_Squared_per_Second_Squared is
```

```
      Temp : Radians_per_Second;
   begin
      Temp := Left * Right;
      return Radians_Squared_per_Second_Squared (Temp);
   end "*";

   function "*" (Left  : Degrees_per_Second;
                 Right : Degrees_per_Second)
                return Degrees_Squared_per_Second_Squared is
      Temp : Degrees_per_Second;
   begin
      Temp := Left * Right;
      return Degrees_Squared_per_Second_Squared (Temp);
   end "*";

   function "*" (Left  : Semicircles_per_Second;
                 Right : Semicircles_per_Second)
                return Semicircles_Squared_per_Second_Squared is
      Temp : Semicircles_per_Second;
   begin
      Temp := Left * Right;
      return Semicircles_Squared_per_Second_Squared (Temp);
   end "*";

-- --Angular Velocity / Sin Cos Ratio ==> Angular Velocity

   function "/" (Left  : Radians_Per_Second;
                 Right : Trig.Sin_Cos_Ratio)
                return Radians_Per_Second is
   begin
      return Left / Radians_Per_Second (Right);
   end "/";

-- --Angular Velocity * Tan_Ratio ==> Angular Velocity

   function "*" (Left  : Radians_Per_Second;
                 Right : Trig.Tan_Ratio)
                return Radians_Per_Second is
   begin
      return Left * Radians_Per_Second (Right);
   end "*";

-- --Angular_Velocity * Time_Interval ==> Angle

   function "*" (Left  : Degrees_Per_Second;
                 Right : Seconds) return Trig.Degrees is
   begin
      return Trig.Degrees (Right * Seconds (Left));
   end "*";

   function "*" (Left  : Radians_Per_Second;
                 Right : Seconds) return Trig.Radians is
   begin
      return Trig.Radians (Right * Seconds (Left));
   end "*";

-- --Angular Velocity * Time Interval ==> Earth Position
```

```
   function "*" (Left  : Radians_Per_Second;
                 Right : Seconds)
                return Earth_Position_Radians is
   begin
      return Earth_Position_Radians (Seconds (Left) * Right);
   end "*";

-- --Angular_Velocity * Velocity ==> Acceleration

   function "*" (Left  : Radians_per_Second;
                 Right : Meters_Per_Second)
                return Meters_Per_Second_Squared is
      Temp : Meters_Per_Second;
   begin
      Temp := Meters_Per_Second (Left) * Right;
      return Meters_Per_Second_Squared (Temp);
   end "*";

   function "*" (Left  : Radians_per_Second;
                 Right : Feet_Per_Second)
                return Feet_Per_Second_Squared is
      Temp : Feet_Per_Second;
   begin
      Temp := Feet_Per_Second (Left) * Right;
      return Feet_Per_Second_Squared (Temp);
   end "*";


-- --Distance * 1/Distance ==> Sin_Cos_Ratio

   function "*" (Left  : Feet;
                 Right : Inverse_Feet) return Trig.Sin_Cos_Ratio is
   begin
      return Trig.Sin_Cos_Ratio (Left * Feet (Right));
   end "*";

   function "*" (Left  : Meters;
                 Right : Inverse_Meters) return Trig.Sin_Cos_Ratio is
   begin
      return Trig.Sin_Cos_Ratio (Left * Meters (Right));
   end "*";

-- --Distance * 1/Distance ==> Tan_Ratio

   function "*" (Left  : Feet;
                 Right : Inverse_Feet) return Trig.Tan_Ratio is
   begin
      return Trig.Tan_Ratio (Left * Feet (Right));
   end "*";

   function "*" (Left  : Meters;
                 Right : Inverse_Meters) return Trig.Tan_Ratio is
   begin
      return Trig.Tan_Ratio (Left * Meters (Right));
   end "*";
```

```
-- --Distance * INTEGER ==> Distance

   function "*" (Left  : Feet;   Right : INTEGER) return Feet is
   begin
      return Left * Feet (Right);
   end "*";

   function "*" (Left  : Meters; Right : INTEGER) return Meters is
   begin
      return Left * Meters (Right);
   end "*";

-- --Distance * Tan_Ratio => Distance

   function "*" (Left  : Feet;
                 Right : Trig.Tan_Ratio) return Feet is
   begin
      return Left * Feet (Right);
   end "*";

   function "*" (Left  : Meters;
                 Right : Trig.Tan_Ratio) return Meters is
   begin
      return Left * Meters (Right);
   end "*";

-- --Distance * Sin_Cos_Ratio ==> Distance

   function "*" (Left  : Feet;   Right : Trig.Sin_Cos_Ratio) return Feet is
   begin
      return Left * Feet (Right);
   end "*";

   function "*" (Left  : Meters; Right : Trig.Sin_Cos_Ratio) return Meters is
   begin
      return Left * Meters (Right);
   end "*";

-- --Distance * Velocity => Tan_Ratio;

   function "*" (Left  : Feet;
                 Right : Feet_Per_Second) return Trig.Tan_Ratio is
      Temp : Feet;
   begin
      Temp := Left * Feet (Right);
      return Trig.Tan_Ratio (Temp);
   end "*";

   function "*" (Left  : Meters;
                 Right : Meters_Per_Second) return Trig.Tan_Ratio is
      Temp : Meters;
   begin
      Temp := Left * Meters (Right);
      return Trig.Tan_Ratio (Temp);
   end "*";
```

```
-- --INTEGER * Sin_Cos_Ratio ==> Sin_Cos_Ratio

   function "*" (Left  : INTEGER;
                 Right : Trig.Sin_Cos_Ratio) return Trig.Sin_Cos_Ratio is
   begin
      return Trig.Sin_Cos_Ratio (Left) * Right;
   end "*";


-- --Inverse_Distances * Tan_Ratio ==> Inverse_Distances

   function "*" (Left  : Inverse_Feet;
                 Right : Trig.Tan_Ratio) return Inverse_Feet is
   begin
      return Left * Inverse_Feet(Right);
   end "*";

   function "*" (Left  : Inverse_Meters;
                 Right : Trig.Tan_Ratio) return Inverse_Meters is
   begin
      return Left * Inverse_Meters(Right);
   end "*";


-- --Sin_Cos_Ratio * Angular Velocity ==> Angular Velocity

   function "*" (Left  : Trig.Sin_Cos_Ratio;
                 Right : Radians_Per_Second)
              return Radians_Per_Second is
   begin
      return Radians_Per_Second (Left) * Right;
   end "*";

-- --Sin_Cos_Ratio * Distance ==> Distance

   function "*" (Left  : Trig.Sin_Cos_Ratio; Right : Feet)   return Feet is
   begin
      return Feet (Left) * Right;
   end "*";

   function "*" (Left  : Trig.Sin_Cos_Ratio; Right : Meters) return Meters is
   begin
      return Meters (Left) * Right;
   end "*";

-- --Sin Cos Ratio * Tan Ratio => Tan Ratio
   function "*" (Left  : Trig.Sin_Cos_Ratio;
                 Right : Trig.Tan_Ratio)
                return Trig.Tan_Ratio is
   begin
      return Trig.Tan_Ratio (Left) * Right;
   end "*";

-- --Sin_Cos_Ratio * Velocity ==> Velocity

   function "*" (Left  : Trig.Sin_Cos_Ratio;
                 Right : Feet_Per_Second) return Feet_Per_Second is
```

```
   begin
      return Feet_Per_Second (Left) * Right;
   end "*";

   function "*" (Left  : Trig.Sin_Cos_Ratio;
                 Right : Meters_Per_Second) return Meters_Per_Second is
   begin
      return Meters_Per_Second (Left) * Right;
   end "*";


-- --Tan Ratio * Sin Cos Ratio => Tan Ratio
   function "*" (Left  : Trig.Tan_Ratio;
                 Right : Trig.Sin_Cos_Ratio)
                return Trig.Tan_Ratio is
   begin
      return Left * Trig.Tan_Ratio (Right);
   end "*";


-- --Velocity * Inverse_Distances ==> Angular_Velocity

   function "*" (Left  : Feet_per_Second;
                 Right : Inverse_Feet) return Radians_per_Second is
   begin
      return Radians_per_Second(Left) * Radians_per_Second(Right);
   end "*";

   function "*" (Left  : Meters_per_Second;
                 Right : Inverse_Meters) return Radians_per_Second is
   begin
      return Radians_per_Second(Left) * Radians_per_Second(Right);
   end "*";

-- --Velocity * Sin Cos Ratio ==> Velocity

   function "*" (Left  : Feet_per_Second;
                 Right : Trig.Sin_Cos_Ratio) return Feet_per_Second is
   begin
      return Left * Feet_per_Second (Right);
   end "*";

   function "*" (Left  : Meters_per_Second;
                 Right : Trig.Sin_Cos_Ratio) return Meters_per_Second  is
   begin
      return Left * Meters_per_Second (Right);
   end "*";

-- --Velocity * Velocity ==> Velocity Squared

   function "*" (Left  : Feet_per_Second;
                 Right : Feet_per_Second)
                return Feet_Squared_per_Second_Squared is
      Temp : Feet_per_Second;
   begin
      Temp := Left * Right;
      return Feet_Squared_per_Second_Squared (Temp);
```

```
   end "*";

   function "*" (Left  : Meters_per_Second;
                 Right : Meters_per_Second)
                return Meters_Squared_per_Second_Squared is
      Temp : Meters_per_Second;
   begin
      Temp := Left * Right;
      return Meters_Squared_per_Second_Squared (Temp);
   end "*";

-- --Velocity * Interval ==> Distance

   function "*" (Left  : Feet_per_Second;   Right : Seconds) return Feet is
      Temp : Feet_per_Second;
   begin
      Temp := Left * Feet_per_Second (Right);
      return Feet (Temp);
   end "*";

   function "*" (Left  : Meters_per_Second;
                 Right : Seconds) return Meters is
      Temp : Meters_per_Second;
   begin
      Temp := Left * Meters_per_Second (Right);
      return Meters (Temp);
   end "*";


-- --Times Operators to Support DCM TLCSC

   function "*" (Left  : Radians_Per_Second;
                 Right : Trig.Sin_Cos_Ratio) return Radians_Per_Second is
   begin
      return Left * Radians_Per_Second (Right);
   end "*";

   function "*" (Left  : Radians_Per_Second;
                 Right : Seconds) return Trig.Sin_Cos_Ratio is
   begin
      return Trig.Sin_Cos_Ratio (Seconds (Left) * Right);
   end "*";

   function "*" (Left  : Trig.Sin_Cos_Ratio;
                 Right : Trig.Sin_Cos_Ratio) return Trig.Radians is
      Temp : Trig.Sin_Cos_Ratio;
   begin
      Temp := Left * Right;
      return Trig.Radians (Temp);
   end "*";

   function "*" (Left  : Trig.Sin_Cos_Ratio;
                 Right : Trig.Radians) return Trig.Sin_Cos_Ratio is
   begin
      return Left * Trig.Sin_Cos_Ratio (Right);
   end "*";
```

```
-- --Acceleration / Velocity ==> Angular Rate

   function "/" (Left  : Feet_per_Second_Squared;
                 Right : Feet_per_Second) return Radians_per_Second is
   begin
      return Radians_per_Second (Left / Feet_per_Second_Squared (Right));
   end "/";

   function "/" (Left  : Meters_per_Second_Squared;
                 Right : Meters_per_Second) return Radians_per_Second is
   begin
      return Radians_per_Second (Left / Meters_per_Second_Squared (Right));
   end "/";

   function "/" (Left  : Gees;
                 Right : Feet_per_Second) return Radians_per_Second is
   begin
      return Radians_per_Second ((CF.Feet_per_Sec2_per_Gee * Left)
                                 / Gees (Right));
   end "/";

   function "/" (Left  : Gees;
                 Right : Meters_per_Second) return Radians_per_Second is
   begin
      return Radians_per_Second (Gees (CF.Meters_per_Sec2_per_Gee) * Left
                                 / Gees (Right));
   end "/";

   function "/" (Left  : Feet_per_Second_Squared;
                 Right : Feet_per_Second) return Degrees_per_Second is
      Answer : Feet_per_Second_Squared;
   begin
      Answer := Left / Feet_per_Second_Squared (Right)
                * Feet_per_Second_Squared (CF.Degrees_per_Radian);
      return Degrees_per_Second (Answer);
   end "/";

   function "/" (Left  : Meters_per_Second_Squared;
                 Right : Meters_per_Second) return Degrees_per_Second is
      Answer : Meters_per_Second_Squared;
   begin
      Answer := Left / Meters_per_Second_Squared (Right)
                * Meters_per_Second_Squared (CF.Degrees_per_Radian);
      return Degrees_per_Second (Answer);
   end "/";

   function "/" (Left  : Gees;
                 Right : Feet_per_Second) return Degrees_per_Second is
      Answer : Gees;
   begin
      Answer := Gees (CF.Feet_per_Sec2_per_Gee) * Left
                / Gees (Right) * Gees (CF.Degrees_per_Radian);
      return Degrees_per_Second (Answer);
   end "/";

   function "/" (Left  : Gees;
```

```
                Right : Meters_per_Second) return Degrees_per_Second is
      Answer : Gees;
   begin
      Answer := Gees(CF.Meters_per_Sec2_per_Gee) * Left
                / Gees (Right) * Gees (CF.Degrees_per_Radian);
      return Degrees_per_Second (Answer);
   end "/";


-- --Angle / Seconds ==> Angular Rate

   function "/" (Left  : Trig.Radians;
                 Right : Seconds)
                return Radians_per_Second is
   begin
      return Radians_Per_Second (Seconds (Left) / Right);
   end "/";

   function "/" (Left  : Trig.Degrees;
                 Right : Seconds)
                return Degrees_Per_Second is
   begin
      return Degrees_Per_Second (Seconds (Left) / Right);
   end "/";

   function "/" (Left  : Trig.Semicircles;
                 Right : Seconds)
                return Semicircles_Per_Second is
   begin
      return Semicircles_Per_Second (Seconds (Left) / Right);
   end "/";


-- --Distance / Sin Cos Ratio ==> Distance

   function "/" (Left  : Feet;   Right : Trig.Sin_Cos_Ratio) return Feet is
   begin
      return Left / Feet (Right);
   end "/";

   function "/" (Left  : Meters; Right : Trig.Sin_Cos_Ratio) return Meters is
   begin
      return Left / Meters (Right);
   end "/";

-- --Distance / Tan Ratio ==> Distance

   function "/" (Left  : Feet;   Right : Trig.Tan_Ratio) return Feet is
   begin
      return Left / Feet (Right);
   end "/";

   function "/" (Left  : Meters; Right : Trig.Tan_Ratio) return Meters is
   begin
      return Left / Meters (Right);
   end "/";
```

```
-- --Distance / Distance ==> Sin_Cos_Ratio

   function "/" (Left  : Feet;
                 Right : Feet)   return Trig.Sin_Cos_Ratio is
      Temp : Feet;
   begin
      Temp := Left / Right;
      return Trig.Sin_Cos_Ratio (Temp);
   end "/";

   function "/" (Left  : Meters;
                 Right : Meters) return Trig.Sin_Cos_Ratio is
      Temp : Meters;
   begin
      Temp := Left / Right;
      return Trig.Sin_Cos_Ratio (Temp);
   end "/";


-- --Tan_Ratio := Sin_Cos_Ratio / Sin_Cos_Ratio

   function "/" (Left  : Trig.Sin_Cos_Ratio;
                 Right : Trig.Sin_Cos_Ratio) return
                Trig.Tan_Ratio is
   begin
      return Trig.Tan_Ratio (Left) / Trig.Tan_Ratio (Right);
   end "/";

-- --Tan_Ratio := Sin_Cos_Ratio / Tan_Ratio

   function "/" (Left  : Trig.Sin_Cos_Ratio;
                 Right : Trig.Tan_Ratio) return Trig.Tan_Ratio is
   begin
      return Trig.Tan_Ratio (Left) / Right;
   end "/";


-- --Velocity / Velocity ==> Sin_Cos_Ratio

   function "/" (Left  : Feet_per_Second;
                 Right : Feet_per_Second) return Trig.Sin_Cos_Ratio is
      Temp : Feet_per_Second;
   begin
      Temp := Left / Right;
      return Trig.Sin_Cos_Ratio (Temp);
   end "/";

   function "/" (Left  : Meters_per_Second;
                 Right : Meters_per_Second)
                return Trig.Sin_Cos_Ratio is
      Temp : Meters_per_Second;
   begin
      Temp := Left / Right;
      return Trig.Sin_Cos_Ratio (Temp);
   end "/";

-- --Velocity / Velocity ==> Tan_Ratio
```

```
   function "/" (Left  : Feet_per_Second;
                 Right : Feet_Per_Second) return Trig.Tan_Ratio is
      Temp : Feet_Per_Second;
   begin
      Temp := Left / Right;
      return Trig.Tan_Ratio (Temp);
   end "/";

   function "/" (Left  : Meters_per_Second;
                 Right : Meters_Per_Second) return Trig.Tan_Ratio is
      Temp : Meters_Per_Second;
   begin
      Temp := Left / Right;
      return Trig.Tan_Ratio (Temp);
   end "/";

-- --Velocity / Time ==> Acceleration
   function "/" (Left  : Feet_per_Second;
                 Right : Seconds)
                return Feet_Per_Second_Squared is
   begin
      return Feet_Per_Second_Squared (Seconds (Left) / Right);
   end "/";

   function "/" (Left  : Meters_per_Second;
                 Right : Seconds)
                return Meters_Per_Second_Squared is
   begin
      return Meters_Per_Second_Squared (Seconds (Left) / Right);
   end "/";

-- --Velocity / Distance ==> Angular Rate

   function "/" (Left  : Feet_per_Second;
                 Right : Feet)
                return Radians_per_Second is
      Temp : Feet;
   begin
      Temp :=  Feet(Left) / Right;
      return Radians_Per_Second (Temp);
   end "/";

   function "/" (Left  : Meters_per_Second;
                 Right : Meters)
                return Radians_per_Second is
      Temp : Meters;
   begin
      Temp :=  Meters(Left) / Right;
      return Radians_Per_Second (Temp);
   end "/";


-- --Tan_Ratio := Tan_Ratio + Sin_Cos_Ratio

   function "+" (Left  : Trig.Tan_Ratio;
                 Right : Trig.Sin_Cos_Ratio) return Trig.Tan_Ratio is
```

```
   begin
      return Left + Trig.Tan_Ratio (Right);
   end "+";

end Basic_Data_Types;
```

3.3.1.1.7 UTILIZATION OF OTHER ELEMENTS

The following library units are with'd by this part:
1. SYSTEM (a predefined part of the Ada environment)
2. Standard_Trig (P689)
3. Conversion_Factors (P614)

3.3.1.1.8 LIMITATIONS

None.

3.3.1.1.9 LLCSC DESIGN

None.

3.3.1.1.10 UNIT DESIGN

None.

(This page left intentionally blank.)

```
with Conversion_Factors;
package body Basic_Data_Types is

   package Cf renames Conversion_Factors;
   use Trig;

-- ----------------------------------------
-- --Define Operators for Basic Data Types-
-- ----------------------------------------

-- --Acceleration * Interval ==> Velocity

   function "*" (Left  :  Feet_Per_Second_Squared;
                 Right : Seconds)
                return Feet_Per_Second is
      Temp : Feet_Per_Second;
   begin
      Temp := Feet_Per_Second (Seconds (Left) * Right);
      return Temp;
   end "*";

   function "*" (Left  :  Meters_Per_Second_Squared;
                 Right : Seconds)
                return Meters_Per_Second is
      Temp : Meters_Per_Second;
   begin
      Temp := Meters_Per_Second (Seconds (Left) * Right);
      return Temp;
   end "*";

-- --Acceleration * Sin Cos Ratio ==> Acceleration

   function "*" (Left  : Feet_Per_Second_Squared;
                 Right : Trig.Sin_Cos_Ratio) return Feet_Per_Second_Squared is
   begin
      return Left * Feet_Per_Second_Squared (Right);
   end "*";

   function "*" (Left  : Meters_Per_Second_Squared;
                 Right : Trig.Sin_Cos_Ratio)
                return Meters_Per_Second_Squared is
   begin
      return Left * Meters_Per_Second_Squared (Right);
   end "*";

   function "*" (Left  : Gees;
                 Right : Trig.Sin_Cos_Ratio) return Gees is
   begin
      return Left * Gees (Right);
   end "*";

-- --Acceleration * Tan Ratio ==> Acceleration

   function "*" (Left  : Feet_Per_Second_Squared;
                 Right : Trig.Tan_Ratio) return Feet_Per_Second_Squared is
   begin
      return Left * Feet_Per_Second_Squared (Right);
```

```
   end "*";

   function "*" (Left  : Meters_Per_Second_Squared;
                 Right : Trig.Tan_Ratio) return Meters_Per_Second_Squared is
   begin
      return Left * Meters_Per_Second_Squared (Right);
   end "*";

   function "*" (Left  : Gees; Right : Trig.Tan_Ratio) return Gees is
   begin
      return Left * Gees (Right);
   end "*";


-- --Angular_Velocity * Angular_Velocity ==> Angular_Velocity Squared

   function "*" (Left  : Radians_Per_Second;
                 Right : Radians_Per_Second)
                return Radians_Squared_Per_Second_Squared is
      Temp : Radians_Per_Second;
   begin
      Temp := Left * Right;
      return Radians_Squared_Per_Second_Squared (Temp);
   end "*";

   function "*" (Left  : Degrees_Per_Second;
                 Right : Degrees_Per_Second)
                return Degrees_Squared_Per_Second_Squared is
      Temp : Degrees_Per_Second;
   begin
      Temp := Left * Right;
      return Degrees_Squared_Per_Second_Squared (Temp);
   end "*";

   function "*" (Left  : Semicircles_Per_Second;
                 Right : Semicircles_Per_Second)
                return Semicircles_Squared_Per_Second_Squared is
      Temp : Semicircles_Per_Second;
   begin
      Temp := Left * Right;
      return Semicircles_Squared_Per_Second_Squared (Temp);
   end "*";

-- --Angular Velocity / Sin Cos Ratio ==> Angular Velocity

   function "/" (Left  : Radians_Per_Second;
                 Right : Trig.Sin_Cos_Ratio)
                return Radians_Per_Second is
   begin
      return Left / Radians_Per_Second (Right);
   end "/";

-- --Angular Velocity * Tan_Ratio ==> Angular Velocity

   function "*" (Left  : Radians_Per_Second;
                 Right : Trig.Tan_Ratio)
                return Radians_Per_Second is
```

```
   begin
      return Left * Radians_Per_Second (Right);
   end "*";

-- --Angular_Velocity * Time_Interval ==> Angle

   function "*" (Left  : Degrees_Per_Second;
                 Right : Seconds) return Trig.Degrees is
   begin
      return Trig.Degrees (Right * Seconds (Left));
   end "*";

   function "*" (Left  : Radians_Per_Second;
                 Right : Seconds) return Trig.Radians is
   begin
      return Trig.Radians (Right * Seconds (Left));
   end "*";

-- --Angular Velocity * Time Interval ==> Earth Position

   function "*" (Left  : Radians_Per_Second;
                 Right : Seconds)
                return Earth_Position_Radians is
   begin
      return Earth_Position_Radians (Seconds (Left) * Right);
   end "*";

-- --Angular_Velocity * Velocity ==> Acceleration

   function "*" (Left  : Radians_Per_Second;
                 Right : Meters_Per_Second)
                return Meters_Per_Second_Squared is
      Temp : Meters_Per_Second;
   begin
      Temp := Meters_Per_Second (Left) * Right;
      return Meters_Per_Second_Squared (Temp);
   end "*";

   function "*" (Left  : Radians_Per_Second;
                 Right : Feet_Per_Second)
                return Feet_Per_Second_Squared is
      Temp : Feet_Per_Second;
   begin
      Temp := Feet_Per_Second (Left) * Right;
      return Feet_Per_Second_Squared (Temp);
   end "*";


-- --Distance * 1/Distance ==> Sin_Cos_Ratio

   function "*" (Left  : Feet;
                 Right : Inverse_Feet) return Trig.Sin_Cos_Ratio is
   begin
      return Trig.Sin_Cos_Ratio (Left * Feet (Right));
   end "*";

   function "*" (Left  : Meters;
```

```
                Right : Inverse_Meters) return Trig.Sin_Cos_Ratio is
   begin
      return Trig.Sin_Cos_Ratio (Left * Meters (Right));
   end "*";

-- --Distance * 1/Distance ==> Tan_Ratio

   function "*" (Left  : Feet;
                 Right : Inverse_Feet) return Trig.Tan_Ratio is
   begin
      return Trig.Tan_Ratio (Left * Feet (Right));
   end "*";

   function "*" (Left  : Meters;
                 Right : Inverse_Meters) return Trig.Tan_Ratio is
   begin
      return Trig.Tan_Ratio (Left * Meters (Right));
   end "*";

-- --Distance * INTEGER ==> Distance

   function "*" (Left  : Feet;   Right : INTEGER) return Feet is
   begin
      return Left * Feet (Right);
   end "*";

   function "*" (Left  : Meters; Right : INTEGER) return Meters is
   begin
      return Left * Meters (Right);
   end "*";

-- --Distance * Tan_Ratio => Distance

   function "*" (Left  : Feet;
                 Right : Trig.Tan_Ratio) return Feet is
   begin
      return Left * Feet (Right);
   end "*";

   function "*" (Left  : Meters;
                 Right : Trig.Tan_Ratio) return Meters is
   begin
      return Left * Meters (Right);
   end "*";

-- --Distance * Sin_Cos_Ratio ==> Distance

   function "*" (Left  : Feet;   Right : Trig.Sin_Cos_Ratio) return Feet is
   begin
      return Left * Feet (Right);
   end "*";

   function "*" (Left  : Meters; Right : Trig.Sin_Cos_Ratio) return Meters is
   begin
      return Left * Meters (Right);
   end "*";
```

```
-- --Distance * Velocity => Tan_Ratio;

   function "*" (Left  : Feet;
                 Right : Feet_Per_Second) return Trig.Tan_Ratio is
      Temp : Feet;
   begin
      Temp := Left * Feet (Right);
      return Trig.Tan_Ratio (Temp);
   end "*";

   function "*" (Left  : Meters;
                 Right : Meters_Per_Second) return Trig.Tan_Ratio is
      Temp : Meters;
   begin
      Temp := Left * Meters (Right);
      return Trig.Tan_Ratio (Temp);
   end "*";


-- --INTEGER * Sin_Cos_Ratio ==> Sin_Cos_Ratio

   function "*" (Left  : INTEGER;
                 Right : Trig.Sin_Cos_Ratio) return Trig.Sin_Cos_Ratio is
   begin
      return Trig.Sin_Cos_Ratio (Left) * Right;
   end "*";


-- --Inverse_Distances * Tan_Ratio ==> Inverse_Distances

   function "*" (Left  : Inverse_Feet;
                 Right : Trig.Tan_Ratio) return Inverse_Feet is
   begin
      return Left * Inverse_Feet(Right);
   end "*";

   function "*" (Left  : Inverse_Meters;
                 Right : Trig.Tan_Ratio) return Inverse_Meters is
   begin
      return Left * Inverse_Meters(Right);
   end "*";


-- --Sin_Cos_Ratio * Angular Velocity ==> Angular Velocity

   function "*" (Left  : Trig.Sin_Cos_Ratio;
                 Right : Radians_Per_Second)
                return Radians_Per_Second is
   begin
      return Radians_Per_Second (Left) * Right;
   end "*";

-- --Sin_Cos_Ratio * Distance ==> Distance

   function "*" (Left  : Trig.Sin_Cos_Ratio; Right : Feet)   return Feet is
   begin
      return Feet (Left) * Right;
```

```
   end "*";

   function "*" (Left  : Trig.Sin_Cos_Ratio; Right : Meters) return Meters is
   begin
      return Meters (Left) * Right;
   end "*";

-- --Sin Cos Ratio * Tan Ratio => Tan Ratio
   function "*" (Left  : Trig.Sin_Cos_Ratio;
                 Right : Trig.Tan_Ratio)
                return Trig.Tan_Ratio is
   begin
      return Trig.Tan_Ratio (Left) * Right;
   end "*";

-- --Sin_Cos_Ratio * Velocity ==> Velocity

   function "*" (Left  : Trig.Sin_Cos_Ratio;
                 Right : Feet_Per_Second) return Feet_Per_Second is
   begin
      return Feet_Per_Second (Left) * Right;
   end "*";

   function "*" (Left  : Trig.Sin_Cos_Ratio;
                 Right : Meters_Per_Second) return Meters_Per_Second is
   begin
      return Meters_Per_Second (Left) * Right;
   end "*";


-- --Tan Ratio * Sin Cos Ratio => Tan Ratio
   function "*" (Left  : Trig.Tan_Ratio;
                 Right : Trig.Sin_Cos_Ratio)
                return Trig.Tan_Ratio is
   begin
      return Left * Trig.Tan_Ratio (Right);
   end "*";


-- --Velocity * Inverse_Distances ==> Angular_Velocity

   function "*" (Left  : Feet_Per_Second;
                 Right : Inverse_Feet) return Radians_Per_Second is
   begin
      return Radians_Per_Second(Left) * Radians_Per_Second(Right);
   end "*";

   function "*" (Left  : Meters_Per_Second;
                 Right : Inverse_Meters) return Radians_Per_Second is
   begin
      return Radians_Per_Second(Left) * Radians_Per_Second(Right);
   end "*";

-- --Velocity * Sin Cos Ratio ==> Velocity

   function "*" (Left  : Feet_Per_Second;
                 Right : Trig.Sin_Cos_Ratio) return Feet_Per_Second is
```

```
   begin
      return Left * Feet_Per_Second (Right);
   end "*";

   function "*" (Left  : Meters_Per_Second;
                 Right : Trig.Sin_Cos_Ratio) return Meters_Per_Second  is
   begin
      return Left * Meters_Per_Second (Right);
   end "*";

-- --Velocity * Velocity ==> Velocity Squared

   function "*" (Left  : Feet_Per_Second;
                 Right : Feet_Per_Second)
                return Feet_Squared_Per_Second_Squared is
      Temp : Feet_Per_Second;
   begin
      Temp := Left * Right;
      return Feet_Squared_Per_Second_Squared (Temp);
   end "*";

   function "*" (Left  : Meters_Per_Second;
                 Right : Meters_Per_Second)
                return Meters_Squared_Per_Second_Squared is
      Temp : Meters_Per_Second;
   begin
      Temp := Left * Right;
      return Meters_Squared_Per_Second_Squared (Temp);
   end "*";

-- --Velocity * Interval ==> Distance

   function "*" (Left  : Feet_Per_Second;   Right : Seconds) return Feet is
      Temp : Feet_Per_Second;
   begin
      Temp := Left * Feet_Per_Second (Right);
      return Feet (Temp);
   end "*";

   function "*" (Left  : Meters_Per_Second;
                 Right : Seconds) return Meters is
      Temp : Meters_Per_Second;
   begin
      Temp := Left * Meters_Per_Second (Right);
      return Meters (Temp);
   end "*";


-- --Times Operators to Support DCM TLCSC

   function "*" (Left  : Radians_Per_Second;
                 Right : Trig.Sin_Cos_Ratio) return Radians_Per_Second is
   begin
      return Left * Radians_Per_Second (Right);
   end "*";

   function "*" (Left  : Radians_Per_Second;
```

```
                Right : Seconds) return Trig.Sin_Cos_Ratio is
   begin
      return Trig.Sin_Cos_Ratio (Seconds (Left) * Right);
   end "*";

   function "*" (Left  : Trig.Sin_Cos_Ratio;
                 Right : Trig.Sin_Cos_Ratio) return Trig.Radians is
      Temp : Trig.Sin_Cos_Ratio;
   begin
      Temp := Left * Right;
      return Trig.Radians (Temp);
   end "*";

   function "*" (Left  : Trig.Sin_Cos_Ratio;
                 Right : Trig.Radians) return Trig.Sin_Cos_Ratio is
   begin
      return Left * Trig.Sin_Cos_Ratio (Right);
   end "*";


-- --Acceleration / Velocity ==> Angular Rate

   function "/" (Left  : Feet_Per_Second_Squared;
                 Right : Feet_Per_Second) return Radians_Per_Second is
   begin
      return Radians_Per_Second (Left / Feet_Per_Second_Squared (Right));
   end "/";

   function "/" (Left  : Meters_Per_Second_Squared;
                 Right : Meters_Per_Second) return Radians_Per_Second is
   begin
      return Radians_Per_Second (Left / Meters_Per_Second_Squared (Right));
   end "/";

   function "/" (Left  : Gees;
                 Right : Feet_Per_Second) return Radians_Per_Second is
   begin
      return Radians_Per_Second ((Cf.Feet_Per_Sec2_Per_Gee * Left)
                                 / Gees (Right));
   end "/";

   function "/" (Left  : Gees;
                 Right : Meters_Per_Second) return Radians_Per_Second is
   begin
      return Radians_Per_Second (Gees (Cf.Meters_Per_Sec2_Per_Gee) * Left
                                 / Gees (Right));
   end "/";

   function "/" (Left  : Feet_Per_Second_Squared;
                 Right : Feet_Per_Second) return Degrees_Per_Second is
      Answer : Feet_Per_Second_Squared;
   begin
      Answer := Left / Feet_Per_Second_Squared (Right)
                * Feet_Per_Second_Squared (Cf.Degrees_Per_Radian);
      return Degrees_Per_Second (Answer);
   end "/";
```

```
   function "/" (Left  : Meters_Per_Second_Squared;
                 Right : Meters_Per_Second) return Degrees_Per_Second is
      Answer : Meters_Per_Second_Squared;
   begin
      Answer := Left / Meters_Per_Second_Squared (Right)
                * Meters_Per_Second_Squared (Cf.Degrees_Per_Radian);
      return Degrees_Per_Second (Answer);
   end "/";

   function "/" (Left  : Gees;
                 Right : Feet_Per_Second) return Degrees_Per_Second is
      Answer : Gees;
   begin
      Answer := Gees (Cf.Feet_Per_Sec2_Per_Gee) * Left
                / Gees (Right) * Gees (Cf.Degrees_Per_Radian);
      return Degrees_Per_Second (Answer);
   end "/";

   function "/" (Left  : Gees;
                 Right : Meters_Per_Second) return Degrees_Per_Second is
      Answer : Gees;
   begin
      Answer := Gees(Cf.Meters_Per_Sec2_Per_Gee) * Left
                / Gees (Right) * Gees (Cf.Degrees_Per_Radian);
      return Degrees_Per_Second (Answer);
   end "/";

-- --Angle / Seconds ==> Angular Rate

   function "/" (Left  : Trig.Radians;
                 Right : Seconds)
                return Radians_Per_Second is
   begin
      return Radians_Per_Second (Seconds (Left) / Right);
   end "/";

   function "/" (Left  : Trig.Degrees;
                 Right : Seconds)
                return Degrees_Per_Second is
   begin
      return Degrees_Per_Second (Seconds (Left) / Right);
   end "/";

   function "/" (Left  : Trig.Semicircles;
                 Right : Seconds)
                return Semicircles_Per_Second is
   begin
      return Semicircles_Per_Second (Seconds (Left) / Right);
   end "/";

-- --Distance / Sin Cos Ratio ==> Distance

   function "/" (Left  : Feet;   Right : Trig.Sin_Cos_Ratio) return Feet is
   begin
      return Left / Feet (Right);
```

```
   end "/";

   function "/" (Left  : Meters; Right : Trig.Sin_Cos_Ratio) return Meters is
   begin
      return Left / Meters (Right);
   end "/";

-- --Distance / Tan Ratio ==> Distance

   function "/" (Left  : Feet;   Right : Trig.Tan_Ratio) return Feet is
   begin
      return Left / Feet (Right);
   end "/";

   function "/" (Left  : Meters; Right : Trig.Tan_Ratio) return Meters is
   begin
      return Left / Meters (Right);
   end "/";

-- --Distance / Distance ==> Sin_Cos_Ratio

   function "/" (Left  : Feet;
                 Right : Feet)   return Trig.Sin_Cos_Ratio is
      Temp : Feet;
   begin
      Temp := Left / Right;
      return Trig.Sin_Cos_Ratio (Temp);
   end "/";

   function "/" (Left  : Meters;
                 Right : Meters) return Trig.Sin_Cos_Ratio is
      Temp : Meters;
   begin
      Temp := Left / Right;
      return Trig.Sin_Cos_Ratio (Temp);
   end "/";


-- --Tan_Ratio := Sin_Cos_Ratio / Sin_Cos_Ratio

   function "/" (Left  : Trig.Sin_Cos_Ratio;
                 Right : Trig.Sin_Cos_Ratio) return
                Trig.Tan_Ratio is
   begin
      return Trig.Tan_Ratio (Left) / Trig.Tan_Ratio (Right);
   end "/";

-- --Tan_Ratio := Sin_Cos_Ratio / Tan_Ratio

   function "/" (Left  : Trig.Sin_Cos_Ratio;
                 Right : Trig.Tan_Ratio) return Trig.Tan_Ratio is
   begin
      return Trig.Tan_Ratio (Left) / Right;
   end "/";


-- --Velocity / Velocity ==> Sin_Cos_Ratio
```

```
   function "/" (Left  : Feet_Per_Second;
                 Right : Feet_Per_Second) return Trig.Sin_Cos_Ratio is
      Temp : Feet_Per_Second;
   begin
      Temp := Left / Right;
      return Trig.Sin_Cos_Ratio (Temp);
   end "/";

   function "/" (Left  : Meters_Per_Second;
                 Right : Meters_Per_Second)
                return Trig.Sin_Cos_Ratio is
      Temp : Meters_Per_Second;
   begin
      Temp := Left / Right;
      return Trig.Sin_Cos_Ratio (Temp);
   end "/";

-- --Velocity / Velocity ==> Tan_Ratio

   function "/" (Left  : Feet_Per_Second;
                 Right : Feet_Per_Second) return Trig.Tan_Ratio is
      Temp : Feet_Per_Second;
   begin
      Temp := Left / Right;
      return Trig.Tan_Ratio (Temp);
   end "/";

   function "/" (Left  : Meters_Per_Second;
                 Right : Meters_Per_Second) return Trig.Tan_Ratio is
      Temp : Meters_Per_Second;
   begin
      Temp := Left / Right;
      return Trig.Tan_Ratio (Temp);
   end "/";

-- --Velocity / Time ==> Acceleration
   function "/" (Left  : Feet_Per_Second;
                 Right : Seconds)
                return Feet_Per_Second_Squared is
   begin
      return Feet_Per_Second_Squared (Seconds (Left) / Right);
   end "/";

   function "/" (Left  : Meters_Per_Second;
                 Right : Seconds)
                return Meters_Per_Second_Squared is
   begin
      return Meters_Per_Second_Squared (Seconds (Left) / Right);
   end "/";

-- --Velocity / Distance ==> Angular Rate

   function "/" (Left  : Feet_Per_Second;
                 Right : Feet)
                return Radians_Per_Second is
      Temp : Feet;
```

```
   begin
      Temp :=  Feet(Left) / Right;
      return Radians_Per_Second (Temp);
   end "/";

   function "/" (Left  : Meters_Per_Second;
                 Right : Meters)
                return Radians_Per_Second is
      Temp : Meters;
   begin
      Temp :=  Meters(Left) / Right;
      return Radians_Per_Second (Temp);
   end "/";


-- --Tan_Ratio := Tan_Ratio + Sin_Cos_Ratio

   function "+" (Left  : Trig.Tan_Ratio;
                 Right : Trig.Sin_Cos_Ratio) return Trig.Tan_Ratio is
   begin
      return Left + Trig.Tan_Ratio (Right);
   end "+";

end Basic_Data_Types;
```

3.3.1.2 KALMAN FILTER DATA TYPES (BODY) TLCSC P622 (CATALOG #P158-0)

This part, which is designed as an Ada generic package, defines data types (and operators on those data types) used in a Kalman Filter.

The decomposition for this part is the same as that shown in the Top-Level Design Document.

3.3.1.2.1 REQUIREMENTS ALLOCATION

This part meets CAMP requirement R023

3.3.1.2.2 LOCAL ENTITIES DESIGN

None.

3.3.1.2.3 INPUT/OUTPUT

GENERIC PARAMETERS:

Data types:

The following table describes the generic formal types required by this part:

Name	Base Type	Description
State_Indices	Discrete type	Indexes the matrices which are a function of the _# of states
Measurement_ Indices	Discrete type	Indexes the matrices which are a function of the _# of measurements
Kalman_Filter _Elements	floating point type	Elements making up the matrices of the Kalman Filter

3.3.1.2.4 LOCAL DATA

None.

3.3.1.2.5 PROCESS CONTROL

Not applicable.

3.3.1.2.6 PROCESSING

The following describes the processing performed by this part:

package body Kalman_Filter_Data_Types is

```
-- ------------------------
-- --NAME:  Active_H_Vector
-- ------------------------

   function Active_H_Vector (Source : M_By_N_Statically_Sparse.Matrices;
                             Row    : Measurement_Indices)
                            return N_By_1.Vectors is
      Work_Vector : N_By_1.Vectors;
   begin
      for I in State_Indices loop
         Work_Vector (I) := Source (Row, I);
      end loop;
      return Work_Vector;
   end Active_H_Vector;

-- ----------------
-- --NAME:  Element
-- ----------------

   function Element (Source : N_By_N_Symmetric.Matrices;
                     Row    : State_Indices;
                     Column : State_Indices)
                    return Kalman_Filter_Elements is
   begin
      return Source (Row, Column);
   end Element;

-- ------------
-- --NAME:  Row
-- ------------

   function Row (Source : N_By_N_Symmetric.Matrices;
                 Row    : State_Indices)
                return N_By_1.Vectors is
      Work_Vector : N_By_1.Vectors;
   begin
      for I in State_Indices loop
         Work_Vector (I) := Source (Row,I);
      end loop;
      return Work_Vector;
   end Row;

end Kalman_Filter_Data_Types;
```

3.3.1.2.7 UTILIZATION OF OTHER ELEMENTS

UTILIZATION OF OTHER ELEMENTS IN TOP LEVEL COMPONENT:

The following tables describe the elements used by this part but defined elsewhere in this top level component:

Data types:

The following table summarizes the types required by this part and defined in ancestral units:

Name	Element Type	Index Types	Matrix Type
M_By_N_Statically _Sparse	Kalman_Filter_Elements	Measurement _Indices, State_Indices	Sparse
N_By_N_Diagonal	Kalman_Filter_Elements	State_Indices, State_Indices	Diagonal
N_By_N_Symmetric	Kalman_Filter_Elements	State_Indices, State_Indices	Symmetric
N_By_1 (Vector)	Kalman_Filter_Elements	State_Indices	Vector

3.3.1.2.8 LIMITATIONS

None.

3.3.1.2.9 LLCSC DESIGN

None.

3.3.1.2.10 UNIT DESIGN

None.

(This page left intentionally blank.)

```
package body Kalman_Filter_Data_Types is

-- ------------------------
-- --NAME: Active_H_Vector
-- ------------------------

   function Active_H_Vector (Source : M_By_N_Statically_Sparse.Matrices;
                             Row    : Measurement_Indices)
                            return N_By_1.Vectors is
      Work_Vector : N_By_1.Vectors;
   begin
      for I in State_Indices loop
         Work_Vector (I) := Source (Row, I);
      end loop;
      return Work_Vector;
   end Active_H_Vector;

-- ----------------
-- --NAME: Element
-- ----------------

   function Element (Source : N_By_N_Symmetric.Matrices;
                     Row    : State_Indices;
                     Column : State_Indices)
                    return Kalman_Filter_Elements is
   begin
      return Source (Row, Column);
   end Element;

-- ------------
-- --NAME: Row
-- ------------

   function Row (Source : N_By_N_Symmetric.Matrices;
                 Row    : State_Indices)
                return N_By_1.Vectors is
      Work_Vector : N_By_1.Vectors;
   begin
      for I in State_Indices loop
         Work_Vector (I) := Source (Row,I);
      end loop;
      return Work_Vector;
   end Row;

end Kalman_Filter_Data_Types;
```

(This page left intentionally blank.)

3.3.1.3 AUTOPILOT DATA TYPES (BODY) TLCSC P623 (CATALOG #P93-0)

This part, which is designed as an Ada package, defines simple data types and special operators for use in the Autopilot Package.

The decomposition for this part is the same as that shown in the Top-Level Design Document.

3.3.1.3.1 REQUIREMENTS ALLOCATION

This part meets CAMP requirement R163.

3.3.1.3.2 LOCAL ENTITIES DESIGN

None.

3.3.1.3.3 INPUT/OUTPUT

None.

3.3.1.3.4 LOCAL DATA

None.

3.3.1.3.5 PROCESS CONTROL

Not applicable.

3.3.1.3.6 PROCESSING

The following describes the processing performed by this part:

```
package body Autopilot_Data_Types is


-------------------------------------------------
-- --Define Operators for Autopilot Data Types-
-------------------------------------------------

-----------------------------------------------------------
-- --Acceleration Feedback * Gain ==> Fin_Deflections-
-----------------------------------------------------------

   function "*" (Left  : Acceleration_Feedbacks_FPS2;
                 Right : FPS2_to_Degrees_Gain)
                 return Fin_Deflections_Degrees is
   begin
      return Fin_Deflections_Degrees
                (Left * Acceleration_Feedbacks_FPS2 (Right));
   end "*";
```

```
   function "*" (Left  : Acceleration_Feedbacks_MPS2;
                 Right : MPS2_to_Degrees_Gain)
                 return Fin_Deflections_Degrees is
   begin
      return Fin_Deflections_Degrees
                (Left * Acceleration_Feedbacks_MPS2 (Right));
   end "*";

   function "*" (Left  : Acceleration_Feedbacks_Gees;
                 Right : Gees_to_Degrees_Gain)
                 return Fin_Deflections_Degrees is
   begin
      return Fin_Deflections_Degrees
             (Left * Acceleration_Feedbacks_Gees (Right));
   end "*";


   function "*" (Left  : Acceleration_Feedbacks_FPS2;
                 Right : FPS2_to_Radians_Gain)
                 return Fin_Deflections_Radians is
   begin
      return Fin_Deflections_Radians
                (Left * Acceleration_Feedbacks_FPS2 (Right));
   end "*";

   function "*" (Left  : Acceleration_Feedbacks_MPS2;
                 Right : MPS2_to_Radians_Gain)
                 return Fin_Deflections_Radians is
   begin
      return Fin_Deflections_Radians
                (Left * Acceleration_Feedbacks_MPS2 (Right));
   end "*";

   function "*" (Left  : Acceleration_Feedbacks_Gees;
                 Right : Gees_to_Radians_Gain)
                 return Fin_Deflections_Radians is
   begin
      return Fin_Deflections_Radians
                (Left * Acceleration_Feedbacks_Gees (Right));
   end "*";

-------------------------------------------------------
-- --Acceleration Commands * Gain ==> Fin Deflections-
-------------------------------------------------------

   function "*" (Left  : Acceleration_Commands_FPS2;
                 Right : FPS2_to_Degrees_Gain)
                 return Fin_Deflections_Degrees is
   begin
      return Fin_Deflections_Degrees
                (Left * Acceleration_Commands_FPS2 (Right));
   end "*";

   function "*" (Left  : Acceleration_Commands_MPS2;
                 Right : MPS2_to_Degrees_Gain)
                 return Fin_Deflections_Degrees is
```

```
   begin
      return Fin_Deflections_Degrees
                (Left * Acceleration_Commands_MPS2 (Right));
   end "*";

   function "*" (Left  : Acceleration_Commands_Gees;
                 Right : Gees_to_Degrees_Gain)
                 return Fin_Deflections_Degrees is
   begin
      return Fin_Deflections_Degrees
                (Left * Acceleration_Commands_Gees (Right));
   end "*";

   function "*" (Left  : Acceleration_Commands_FPS2;
                 Right : FPS2_to_Radians_Gain)
                 return Fin_Deflections_Radians is
   begin
      return Fin_Deflections_Radians
                (Left * Acceleration_Commands_FPS2 (Right));
   end "*";

   function "*" (Left  : Acceleration_Commands_MPS2;
                 Right : MPS2_to_Radians_Gain)
                 return Fin_Deflections_Radians is
   begin
      return Fin_Deflections_Radians
                (Left * Acceleration_Commands_MPS2 (Right));
   end "*";

   function "*" (Left  : Acceleration_Commands_Gees;
                 Right : Gees_to_Radians_Gain)
                 return Fin_Deflections_Radians is
   begin
      return Fin_Deflections_Radians
                (Left * Acceleration_Commands_Gees (Right));
   end "*";

-----------------------------------------------
-- --Feedback Rates * Gain ==> Fin Deflections-
-----------------------------------------------

-- --Ambiguity in the meaning of the "*" operator has required changing the
-- --standard order for explicit type conversion.  The bodies of the next two
-- --functions perform explicit type conversion on the "Left" operand to avoid
-- --ambiguity with subprograms derived from the type declarations of Feedback
-- --_Rates_Degrees and Feedback_Rates_Radians.

   function "*" (Left  : Feedback_Rates_Degrees;
                 Right : DPS_to_Degrees_Gain)
                 return Fin_Deflections_Degrees is
   begin
      return Fin_Deflections_Degrees
                (DPS_to_Degrees_Gain (Left) * Right);
   end "*";

   function "*" (Left  : Feedback_Rates_Radians;
                 Right : RPS_to_Radians_Gain)
```

```
                return Fin_Deflections_Radians is
   begin
      return Fin_Deflections_Radians
                (RPS_to_Radians_Gain (Left) * Right);
   end "*";

-------------------------------------
-- --Roll Commands * Gain ==> Angle-
-------------------------------------

   function "*" (Left  : Roll_Commands_Degrees;
                 Right : Gain_in_Degrees)
                 return Fin_Deflections_Degrees is
   begin
      return Fin_Deflections_Degrees
                (Left * Roll_Commands_Degrees (Right));
   end "*";

   function "*" (Left  : Roll_Commands_Radians;
                 Right : Gain_in_Radians)
                 return Fin_Deflections_Radians is
   begin
      return Fin_Deflections_Radians
                (Left * Roll_Commands_Radians (Right));
   end "*";

-------------------------------------------------------------------------------
-- --Acceleration Commands + Acceleration Feedbacks ==> Acceleration Commands-
-------------------------------------------------------------------------------

   function "+" (Left  : Acceleration_Commands_FPS2;
                 Right : Acceleration_Feedbacks_FPS2)
                 return Acceleration_Commands_FPS2 is
   begin
      return Left + Acceleration_Commands_FPS2 (Right);
   end "+";

   function "+" (Left  : Acceleration_Commands_MPS2;
                 Right : Acceleration_Feedbacks_MPS2)
                 return Acceleration_Commands_MPS2 is
   begin
      return Left + Acceleration_Commands_MPS2 (Right);
   end "+";

   function "+" (Left  : Acceleration_Commands_Gees;
                 Right : Acceleration_Feedbacks_Gees)
                 return Acceleration_Commands_Gees is
   begin
      return Left + Acceleration_Commands_Gees (Right);
   end "+";

-------------------------------------------------------------------------------
-- --Acceleration Commands - Acceleration Feedbacks ==> Acceleration Commands-
-------------------------------------------------------------------------------

   function "-" (Left  : Acceleration_Commands_FPS2;
                 Right : Acceleration_Feedbacks_FPS2)
```

```
                    return Acceleration_Commands_FPS2 is
   begin
      return Left - Acceleration_Commands_FPS2 (Right);
   end "-";

   function "-" (Left  : Acceleration_Commands_MPS2;
                 Right : Acceleration_Feedbacks_MPS2)
                 return Acceleration_Commands_MPS2 is
   begin
      return Left - Acceleration_Commands_MPS2 (Right);
   end "-";

   function "-" (Left  : Acceleration_Commands_Gees;
                 Right : Acceleration_Feedbacks_Gees)
                 return Acceleration_Commands_Gees is
   begin
      return Left - Acceleration_Commands_Gees (Right);
   end "-";

-------------------------------------------------------------
-- --Roll Commands - Missile Attitudes ==> Roll Commands-
-------------------------------------------------------------

   function "-" (Left  : Roll_Commands_Degrees;
                 Right : Missile_Attitudes_Degrees)
                 return Roll_Commands_Degrees is
   begin
      return Left - Roll_Commands_Degrees (Right);
   end "-";

   function "-" (Left  : Roll_Commands_Radians;
                 Right : Missile_Attitudes_Radians)
                 return Roll_Commands_Radians is
   begin
      return Left - Roll_Commands_Radians (Right);
   end "-";

end Autopilot_Data_Types;
```

3.3.1.3.7 UTILIZATION OF OTHER ELEMENTS

None.

3.3.1.3.8 LIMITATIONS

None.

3.3.1.3.9 LLCSC DESIGN

None.

3.3.1.3.10 UNIT DESIGN

None.

```
package body Autopilot_Data_Types is


-------------------------------------------------
-- --Define Operators for Autopilot Data Types-
-------------------------------------------------

-----------------------------------------------------------
-- --Acceleration Feedback * Gain ==> Fin_Deflections-
-----------------------------------------------------------

   function "*" (Left  : Acceleration_Feedbacks_Fps2;
                 Right : Fps2_To_Degrees_Gain)
                 return Fin_Deflections_Degrees is
   begin
      return Fin_Deflections_Degrees
                (Left * Acceleration_Feedbacks_Fps2 (Right));
   end "*";

   function "*" (Left  : Acceleration_Feedbacks_Mps2;
                 Right : Mps2_To_Degrees_Gain)
                 return Fin_Deflections_Degrees is
   begin
      return Fin_Deflections_Degrees
                (Left * Acceleration_Feedbacks_Mps2 (Right));
   end "*";

   function "*" (Left  : Acceleration_Feedbacks_Gees;
                 Right : Gees_To_Degrees_Gain)
                 return Fin_Deflections_Degrees is
   begin
      return Fin_Deflections_Degrees
             (Left * Acceleration_Feedbacks_Gees (Right));
   end "*";


   function "*" (Left  : Acceleration_Feedbacks_Fps2;
                 Right : Fps2_To_Radians_Gain)
                 return Fin_Deflections_Radians is
   begin
      return Fin_Deflections_Radians
                (Left * Acceleration_Feedbacks_Fps2 (Right));
   end "*";

   function "*" (Left  : Acceleration_Feedbacks_Mps2;
                 Right : Mps2_To_Radians_Gain)
                 return Fin_Deflections_Radians is
   begin
      return Fin_Deflections_Radians
                (Left * Acceleration_Feedbacks_Mps2 (Right));
   end "*";

   function "*" (Left  : Acceleration_Feedbacks_Gees;
                 Right : Gees_To_Radians_Gain)
                 return Fin_Deflections_Radians is
   begin
```

```
      return Fin_Deflections_Radians
                (Left * Acceleration_Feedbacks_Gees (Right));
   end "*";

-------------------------------------------------------
-- --Acceleration Commands * Gain ==> Fin Deflections-
-------------------------------------------------------

   function "*" (Left  : Acceleration_Commands_Fps2;
                 Right : Fps2_To_Degrees_Gain)
                 return Fin_Deflections_Degrees is
   begin
      return Fin_Deflections_Degrees
                (Left * Acceleration_Commands_Fps2 (Right));
   end "*";

   function "*" (Left  : Acceleration_Commands_Mps2;
                 Right : Mps2_To_Degrees_Gain)
                 return Fin_Deflections_Degrees is
   begin
      return Fin_Deflections_Degrees
                (Left * Acceleration_Commands_Mps2 (Right));
   end "*";

   function "*" (Left  : Acceleration_Commands_Gees;
                 Right : Gees_To_Degrees_Gain)
                 return Fin_Deflections_Degrees is
   begin
      return Fin_Deflections_Degrees
                (Left * Acceleration_Commands_Gees (Right));
   end "*";

   function "*" (Left  : Acceleration_Commands_Fps2;
                 Right : Fps2_To_Radians_Gain)
                 return Fin_Deflections_Radians is
   begin
      return Fin_Deflections_Radians
                (Left * Acceleration_Commands_Fps2 (Right));
   end "*";

   function "*" (Left  : Acceleration_Commands_Mps2;
                 Right : Mps2_To_Radians_Gain)
                 return Fin_Deflections_Radians is
   begin
      return Fin_Deflections_Radians
                (Left * Acceleration_Commands_Mps2 (Right));
   end "*";

   function "*" (Left  : Acceleration_Commands_Gees;
                 Right : Gees_To_Radians_Gain)
                 return Fin_Deflections_Radians is
   begin
      return Fin_Deflections_Radians
                (Left * Acceleration_Commands_Gees (Right));
   end "*";

-----------------------------------------------
```

```
-- --Feedback Rates * Gain ==> Fin Deflections-
-----------------------------------------------

-- --Ambiguity in the meaning of the "*" operator has required changing the
-- --standard order for explicit type conversion. The bodies of the next two
-- --functions perform explicit type conversion on the "Left" operand to avoid
-- --ambiguity with subprograms derived from the type declarations of Feedback
-- --_Rates_Degrees and Feedback_Rates_Radians.

   function "*" (Left  : Feedback_Rates_Degrees;
                 Right : Dps_To_Degrees_Gain)
                 return Fin_Deflections_Degrees is
   begin
      return Fin_Deflections_Degrees
                (Dps_To_Degrees_Gain (Left) * Right);
   end "*";

   function "*" (Left  : Feedback_Rates_Radians;
                 Right : Rps_To_Radians_Gain)
                 return Fin_Deflections_Radians is
   begin
      return Fin_Deflections_Radians
                (Rps_To_Radians_Gain (Left) * Right);
   end "*";

-------------------------------------
-- --Roll Commands * Gain ==> Angle-
-------------------------------------

   function "*" (Left  : Roll_Commands_Degrees;
                 Right : Gain_In_Degrees)
                 return Fin_Deflections_Degrees is
   begin
      return Fin_Deflections_Degrees
                (Left * Roll_Commands_Degrees (Right));
   end "*";

   function "*" (Left  : Roll_Commands_Radians;
                 Right : Gain_In_Radians)
                 return Fin_Deflections_Radians is
   begin
      return Fin_Deflections_Radians
                (Left * Roll_Commands_Radians (Right));
   end "*";

-------------------------------------------------------------------------------
-- --Acceleration Commands + Acceleration Feedbacks ==> Acceleration Commands-
-------------------------------------------------------------------------------

   function "+" (Left  : Acceleration_Commands_Fps2;
                 Right : Acceleration_Feedbacks_Fps2)
                 return Acceleration_Commands_Fps2 is
   begin
      return Left + Acceleration_Commands_Fps2 (Right);
   end "+";

   function "+" (Left  : Acceleration_Commands_Mps2;
```

```
                Right : Acceleration_Feedbacks_Mps2)
                return Acceleration_Commands_Mps2 is
   begin
      return Left + Acceleration_Commands_Mps2 (Right);
   end "+";

   function "+" (Left  : Acceleration_Commands_Gees;
                Right : Acceleration_Feedbacks_Gees)
                return Acceleration_Commands_Gees is
   begin
      return Left + Acceleration_Commands_Gees (Right);
   end "+";

-------------------------------------------------------------------------------
-- --Acceleration Commands - Acceleration Feedbacks ==> Acceleration Commands-
-------------------------------------------------------------------------------

   function "-" (Left  : Acceleration_Commands_Fps2;
                Right : Acceleration_Feedbacks_Fps2)
                return Acceleration_Commands_Fps2 is
   begin
      return Left - Acceleration_Commands_Fps2 (Right);
   end "-";

   function "-" (Left  : Acceleration_Commands_Mps2;
                Right : Acceleration_Feedbacks_Mps2)
                return Acceleration_Commands_Mps2 is
   begin
      return Left - Acceleration_Commands_Mps2 (Right);
   end "-";

   function "-" (Left  : Acceleration_Commands_Gees;
                Right : Acceleration_Feedbacks_Gees)
                return Acceleration_Commands_Gees is
   begin
      return Left - Acceleration_Commands_Gees (Right);
   end "-";

-------------------------------------------------------------
-- --Roll Commands - Missile Attitudes ==> Roll Commands-
-------------------------------------------------------------

   function "-" (Left  : Roll_Commands_Degrees;
                Right : Missile_Attitudes_Degrees)
                return Roll_Commands_Degrees is
   begin
      return Left - Roll_Commands_Degrees (Right);
   end "-";

   function "-" (Left  : Roll_Commands_Radians;
                Right : Missile_Attitudes_Radians)
                return Roll_Commands_Radians is
   begin
      return Left - Roll_Commands_Radians (Right);
   end "-";

end Autopilot_Data_Types;
```

3.3.2 NAVIGATION

(This page intentionally left blank.)

3.3.2.1 COMMON_NAVIGATION_PARTS (PACKAGE BODY) TLCSC P001 (CATALOG #P208-0)

This part, which is designed as an Ada package, contains specifications for all CAMP parts which can be used in either a Wander Azimuth or a North Pointing navigation coordinate system environment.

The decomposition for this part is the same as that shown in the Top-Level Design Document.

3.3.2.1.1 REQUIREMENTS ALLOCATION

The following chart summarizes the allocation of CAMP requirements to this part:

Name	Requirement Number
Altitude_Integration	R002
Compute_Ground_Velocity	R003
Compute_Gravitational_Acceleration_Lat_In	R005
Compute_Gravitational_Acceleration_Sin_Lat_In	R006
Compute_Heading	R009
Update_Velocity	R010
Compute_Scalar_Velocity	R039
Compute_Rotation_Increments	R157

3.3.2.1.2 LOCAL ENTITIES DESIGN

None.

3.3.2.1.3 INPUT/OUTPUT

None.

3.3.2.1.4 LOCAL DATA

None.

3.3.2.1.5 PROCESS CONTROL

Not applicable.

3.3.2.1.6 PROCESSING

The following describes the processing performed by this part:

```
package body Common_Navigation_Parts is

   package body Altitude_Integration is separate;
```

```
   function Compute_Ground_Velocity
               (East_Velocity  : Velocities;
                North_Velocity : Velocities) return Velocities is separate;

   function Compute_Gravitational_Acceleration_Lat_In
               (Current_Altitude : Distances;
                Current_Latitude : Earth_Positions)
               return Accelerations is separate;

   function Compute_Gravitational_Acceleration_Sin_Lat_In
               (Current_Altitude     : Distances;
                Sin_Current_Latitude : Sin_Cos_Ratio)
               return Accelerations is separate;

   function Compute_Heading (East_Velocity  : Velocities;
                             North_Velocity : Velocities)
                            return Headings is separate;

   package body Update_Velocity is separate;

   function Scalar_Velocity (Velocity_In : Velocity_Vectors)
                            return Velocities is separate;

   function Compute_Rotation_Increments
               (Platform_Rotation_Rate : Angular_Velocity_Vectors;
                Delta_Time             : Intervals)
               return Angle_Vectors is separate;

end Common_Navigation_Parts;
```

3.3.2.1.7 UTILIZATION OF OTHER ELEMENTS

None.

3.3.2.1.8 LIMITATIONS

None.

3.3.2.1.9 LLCSC DESIGN

3.3.2.1.9.1 ALTITUDE_INTEGRATION PACKAGE DESIGN (CATALOG #P209-0)

This package updates the altitude using trapezoidal integration of the earth-relative vertical velocity. It has the ability to reinitialize the previous values of the vertical velocity and the altitude, and the ability to calculate a new value for altitude.

The decomposition for this part is the same as that shown in the Top-Level Design Document.

3.3.2.1.9.1.1 REQUIREMENTS ALLOCATION

This part meets CAMP requirement R002.

3.3.2.1.9.1.2 LOCAL ENTITIES DESIGN

None.

3.3.2.1.9.1.3 INPUT/OUTPUT

GENERIC PARAMETERS:

The following generic parameters were previously defined for this part's package specification:

Data types:

The following table describes the generic formal types required by this part:

Name	Type	Description
Distances	floating point type	Data type used to define distance measurements
Intervals	floating point type	Data type used to define time measurements
Velocities	floating point type	Data type used to define velocity measurements

Data objects:

The following table describes the generic formal objects required by this part:

Name	Type	Value	Description
Default_Delta_Time	Intervals	N/A	Default time over which integration is to be performed
Initial_Vertical_Velocity	Velocities	N/A	Initial vertical velocity to be used by the Integrate function
Initial_Altitude	Distances	N/A	Initial altitude to be used by Integrate function

Subprograms:

The following table describes the generic formal subroutines required by this part:

Name	Type	Description
"*"	function	Multiplication operator used to define the operation: Velocities * Intervals => Distances

3.3.2.1.9.1.4 LOCAL DATA

Data objects:

The following table describes the data objects maintained by this part:

Name	Type	Value	Description
Previous_Vertical_Velocity	Velocities	N/A	Previous value of the vertical velocity
Previous_Altitude	Distances	N/A	Previously calculated altitude

3.3.2.1.9.1.5 PROCESS CONTROL

Not applicable.

3.3.2.1.9.1.6 PROCESSING

The following describes the processing performed by this part:

```
separate (Common_Navigation_Parts)
package body Altitude_Integration is

-- ---------------------
-- --local declarations-
-- ---------------------

   Previous_Vertical_Velocity : Velocities := Initial_Vertical_Velocity;
   Previous_Altitude          : Distances  := Initial_Altitude;

end Altitude_Integration;
```

3.3.2.1.9.1.7 UTILIZATION OF OTHER ELEMENTS

None.

3.3.2.1.9.1.8 LIMITATIONS

None.

3.3.2.1.9.1.9 LLCSC DESIGN

None.

3.3.2.1.9.1.10 UNIT DESIGN

3.3.2.1.9.1.10.1 REINITIALIZE UNIT DESIGN

This procedure reinitializes the previous values for vertical velocity and altitude. These values are required by the Integrate function.

3.3.2.1.9.1.10.1.1 REQUIREMENTS ALLOCATION

This part meets CAMP requirement R002.

3.3.2.1.9.1.10.1.2 LOCAL ENTITIES DESIGN

None.

3.3.2.1.9.1.10.1.3 INPUT/OUTPUT

FORMAL PARAMETERS:

The following table describes this part's formal parameters:

Name	Type	Value	Description
New_Vertical_Velocity	Velocity	In	Value to be used to reinitialize previous vertical velocity
New_Altitude	Distance	In	Value to be used to reinitialize previous altitude

3.3.2.1.9.1.10.1.4 LOCAL DATA

None.

3.3.2.1.9.1.10.1.5 PROCESS CONTROL

Not applicable.

3.3.2.1.9.1.10.1.6 PROCESSING

The following describes the processing performed by this part:

```
procedure Reinitialize (New_Vertical_Velocity : in Velocities;
                        New_Altitude          : in Distances) is

begin
```

```
    Previous_Altitude          := New_Altitude;
    Previous_Vertical_Velocity := New_Vertical_Velocity;

  end Reinitialize;
```

3.3.2.1.9.1.10.1.7 UTILIZATION OF OTHER ELEMENTS

UTILIZATION OF ANCESTRAL ELEMENTS:

The following tables describe the elements used by this part but defined in one or more ancestral units:

Data types:

The following table summarizes the types required by this part and defined as generic parameters to the enclosing package body:

Name	Type	Description
Distances	floating point type	Data type used to define distance measurements
Intervals	floating point type	Data type used to define time measurements
Velocities	floating point type	Data type used to define velocity measurements

Data objects:

The following table describes the data objects required by this part and located in the package body of Altitude_Integration:

Name	Type	Value	Description
Previous_Vertical_Velocity	Velocities	N/A	Previous value of the vertical velocity
Previous_Altitude	Distances	N/A	Previously calculated altitude

3.3.2.1.9.1.10.1.8 LIMITATIONS

None.

3.3.2.1.9.1.10.2 INTEGRATE UNIT DESIGN

This procedure calculates the new altitude using trapezoidal integration of the earth-relative vertical velocity. It also updates previous velocity and altitude variables.

3.3.2.1.9.1.10.2.1 REQUIREMENTS ALLOCATION

This part meets CAMP requirement R002.

3.3.2.1.9.1.10.2.2 LOCAL ENTITIES DESIGN

None.

3.3.2.1.9.1.10.2.3 INPUT/OUTPUT

FORMAL PARAMETERS:

The following table describes this part's formal parameters:

Name	Type	Value	Description
Vertical_Velocity	Velocity	N/A	Current earth-relative vertical velocity of the missile
Delta_Time	Interval	N/A	Time interval over which the integration is to take place

3.3.2.1.9.1.10.2.4 LOCAL DATA

Data objects:

The following table describes the data objects maintained local to this part:

Name	Type	Value	Description
Altitude	Distances	N/A	Altitude being calculated

3.3.2.1.9.1.10.2.5 PROCESS CONTROL

Not applicable.

3.3.2.1.9.1.10.2.6 PROCESSING

The following describes the processing performed by this part:

```
   function Integrate (Vertical_Velocity : Velocities;
                       Delta_Time        : Intervals := Default_Delta_Time)
                      return Distances is

--    --------------------
--    --local declarations
--    --------------------

      Altitude : Distances;
```

```
-- -------------------------
-- --begin function Integrate
-- -------------------------

   begin

      Altitude := Previous_Altitude +
                  ( (Vertical_Velocity + Previous_Vertical_Velocity) *
                    (0.5 * Delta_Time) );

      Previous_Vertical_Velocity := Vertical_Velocity;
      Previous_Altitude          := Altitude;

      return Altitude;

   end Integrate;
```

3.3.2.1.9.1.10.2.7 UTILIZATION OF OTHER ELEMENTS

UTILIZATION OF ANCESTRAL ELEMENTS:

The following tables describe the elements used by this part but defined in one or more ancestral units:

Data types:

The following table summarizes the types required by this part and defined as generic parameters to the enclosing package body:

Name	Type	Description
Distances	floating point type	Data type used to define distance measurements
Intervals	floating point type	Data type used to define time measurements
Velocities	floating point type	Data type used to define velocity measurements

Data objects:

The following table describes the data objects required by this part and located in the package body of Altitude_Integration:

Name	Type	Value	Description
Previous_Vertical_Velocity	Velocities	N/A	Previous value of the vertical velocity
Previous_Altitude	Distances	N/A	Previously calculated altitude

3.3.2.1.9.1.10.2.8 LIMITATIONS

None.

3.3.2.1.9.2 UPDATE_VELOCITY PACKAGE DESIGN (CATALOG #P214-0)

This package contains the subroutines required to compute the current velocity (in vector form).

It consists of reinitialize, a procedure which reinitializes the value of the Velocity vector, and Update, a function which computes the current velocity vector given the velocity change vector, the time interval over which the velocity changes were accumulated, Coriolis acceleration vector, and acceleration due to gravity.

The computations performed by the update function are as follows:

```
New Velocity := Previous Velocity + Delta Velocity -
                ( (Coriolis Acceleration -
                   Gravitational Acceleration) * Delta Time)
```

The decomposition for this part is the same as that shown in the Top-Level Design Document.

3.3.2.1.9.2.1 REQUIREMENTS ALLOCATION

This part meets CAMP requirement R010.

3.3.2.1.9.2.2 LOCAL ENTITIES DESIGN

None.

3.3.2.1.9.2.3 INPUT/OUTPUT

GENERIC PARAMETERS:

This part's generic parameters were previously defined in the package specification of Common_Navigation_Parts:

Data types:

The following table describes the generic formal types required by this part:

Name	Type	Description
Velocity_Vectors	private	Contains the east, north, and vertical components of the missile's current velocity
Intervals	floating point type	Data type used to define time measurements
Accelerations	floating point type	Data type used to define acceleration measurements
Indices	scalar type	Used to dimension Acceleration_Vectors
Acceleration_Vectors	array	Array of Accelerations

Data objects:

The following table describes the generic formal objects required by this part:

Name	Type	Value	Description
Default_ Gravity	Acceleration_ Vectors	N/A	Vector defining acceleration due to gravity (positive in the up direction)
Initial_ Velocity	Velocity_ Vectors	N/A	Initial velocity values used by Update function

Subprograms:

The following table describes the generic formal subroutines (operators) required by this part:

Name	Type	Description
"*"	function	Vector-scalar multiplication operator defining the operation: Acceleration_Vectors * Intervals => Velocity_Vectors
Sparse_Right_ XY_Subtract	function	Subtraction operator operating on acceleration vectors where the assumption is made that the third element of the right input vector equals 0
"+"	function	Vector-vector addition operator defining the operation: Velocity_Vectors + Velocity_Vectors => Velocity_Vectors
"-"	function	Vector-vector subtraction operator defining the operation: Velocity_Vectors - Velocity_Vectors => Velocity_Vectors

3.3.2.1.9.2.4 LOCAL DATA

Data objects:

The following table describes the data objects maintained by this part:

Name	Type	Value	Description
Previous_Velocity	Velocity_Vector	N/A	Velocity vector calculated during previous update operation

3.3.2.1.9.2.5 PROCESS CONTROL

Not applicable.

3.3.2.1.9.2.6 PROCESSING

The following describes the processing performed by this part:

```
separate (Common_Navigation_Parts)
package body Update_Velocity is

-- ------------------
-- --local variables-
-- ------------------

   Previous_Velocity : Velocity_Vectors := Initial_Velocity;

end Update_Velocity;
```

3.3.2.1.9.2.7 UTILIZATION OF OTHER ELEMENTS

None.

3.3.2.1.9.2.8 LIMITATIONS

None.

3.3.2.1.9.2.9 LLCSC DESIGN

None.

3.3.2.1.9.2.10 UNIT DESIGN

3.3.2.1.9.2.10.1 REINITIALIZE UNIT DESIGN

This procedure reinitializes the previous velocity vector required by the update function.

3.3.2.1.9.2.10.1.1 REQUIREMENTS ALLOCATION

This part meets CAMP requirement R010.

3.3.2.1.9.2.10.1.2 LOCAL ENTITIES DESIGN

None.

3.3.2.1.9.2.10.1.3 INPUT/OUTPUT

FORMAL PARAMETERS:

The following table describes this part's formal parameters:

Name	Type	Value	Description
New_Velocity	Velocity_Vectors	In	New value for the previous_ velocity_vector

3.3.2.1.9.2.10.1.4 LOCAL DATA

None.

3.3.2.1.9.2.10.1.5 PROCESS CONTROL

Not applicable.

3.3.2.1.9.2.10.1.6 PROCESSING

The following describes the processing performed by this part:

```
procedure Reinitialize (New_Velocity : in Velocity_Vectors) is
begin
   Previous_Velocity := New_Velocity;
end Reinitialize;
```

3.3.2.1.9.2.10.1.7 UTILIZATION OF OTHER ELEMENTS

UTILIZATION OF ANCESTRAL ELEMENTS:

The following tables describe the elements used by this part but defined in one or more ancestral units:

Data types:

The following table summarizes the types required by this part and defined as generic parameters to the enclosing package body:

Name	Type	Description
Velocity_Vectors	private	Contains the east, north, and vertical components of the missile's current velocity

Data objects:

The following table summarizes the objects required by this part and defined in the enclosing package body:

Name	Type	Value	Description
Previous_Velocity	Velocity_Vectors	N/A	Velocity vector calculated during previous update operation

3.3.2.1.9.2.10.1.8 LIMITATIONS

None.

3.3.2.1.9.2.10.2 UPDATE UNIT DESIGN

This function computes the current velocity vector given the velocity change vector, the time interval over which the velocity changes were accumulated, the coriolis acceleration vector, and the acceleration due to gravity.

The computations performed are as follows:

```
New Velocity := Previous Velocity + Delta Velocity -
                ( (Coriolis Acceleration -
                   Gravitational Acceleration) * Delta Time)
```

3.3.2.1.9.2.10.2.1 REQUIREMENTS ALLOCATION

This part meets CAMP requirement R010.

3.3.2.1.9.2.10.2.2 LOCAL ENTITIES DESIGN

None.

3.3.2.1.9.2.10.2.3 INPUT/OUTPUT

FORMAL PARAMETERS:

The following table describes this part's formal parameters:

Name	Type	Value	Description
Nominal_Delta_Velocity	Velocity_Vectors	N/A	Contains the east, north, and vertical components of the change in the missile's velocity
Coriolis_Acceleration	Acceleration_Vectors	N/A	Contains the east, north, and vertical components of the coriolis acceleration
Delta_Time	Intervals	N/A	Time interval over which the velocity changes occurred

3.3.2.1.9.2.10.2.4 LOCAL DATA

None.

3.3.2.1.9.2.10.2.5 PROCESS CONTROL

Not applicable.

3.3.2.1.9.2.10.2.6 PROCESSING

The following describes the processing performed by this part:

```
   function Update (Nominal_Delta_Velocity : Velocity_Vectors;
                    Coriolis_Acceleration  : Acceleration_Vectors;
                    Delta_Time             : Intervals;
                    Gravity                : Acceleration_Vectors
                                               := Default_Gravity)
                   return Velocity_Vectors is

--    ---------------------------
--    --declaration of variables-
--    ---------------------------

      Current_Velocity  : Velocity_Vectors;
      Temp_A_V          : Acceleration_Vectors;
      Temp_V_V          : Velocity_Vectors;

-- ------------------------------
-- --beginning of function Update
```

```
-- ------------------------------

   begin

      Temp_A_V := Sparse_Right_XY_Subtract(Coriolis_Acceleration, Gravity);
      Temp_V_V := Temp_A_V * Delta_Time;

      Current_Velocity  := Previous_Velocity +
                           Nominal_Delta_Velocity -
                           Temp_V_V;
      Previous_Velocity := Current_Velocity;

      return Current_Velocity;

   end Update;
```

3.3.2.1.9.2.10.2.7 UTILIZATION OF OTHER ELEMENTS

UTILIZATION OF ANCESTRAL ELEMENTS:

The following tables describe the elements used by this part but defined in one or more ancestral units:

Data types:

The following table summarizes the types required by this part and defined as generic parameters to the enclosing package body:

Name	Type	Description
Velocity_Vectors	private	Contains the east, north, and vertical components of the missile's current velocity
Intervals	floating point type	Data type used to define time measurements
Accelerations	floating point type	Data type used to define acceleration measurements
Indices	scalar type	Used to dimension Acceleration_Vector
Acceleration_Vectors	array	Array of Acceleration

Data objects:

The following table summarizes the objects required by this part and defined as generic parameters to the enclosing package:

Name	Type	Value	Description
Default_ Gravity	Acceleration_ Vectors	N/A	Vector defining acceleration due to gravity (positive in the up direction)

The following table summarizes the objects required by this part and defined in the enclosing package body:

Name	Type	Value	Description
Previous_Velocity	Velocity_Vectors	N/A	Velocity vector calculated during previous update operation

Subprograms and task entries:

The following table summarizes the subroutines and task entries required by this part and defined as generic parameters to the enclosing package:

Name	Type	Description
"*"	function	Vector-scalar multiplication operator defining the operation: Acceleration_Vectors * Intervals => Velocity_Vectors
Sparse_Right_XY_Subtract	function	Subtraction operator operating on acceleration vectors where the assumption is made that the third element of the right input vector equals 0
"+"	function	Vector-vector addition operator defining the operation: Velocity_Vectors + Velocity_Vectors => Velocity_Vectors
"-"	function	Vector-vector subtraction operator defining the operation: Velocity_Vectors - Velocity_Vectors => Velocity_Vectors

3.3.2.1.9.2.10.2.8 LIMITATIONS

None.

3.3.2.1.9.2.10.3 CURRENT_VELOCITY UNIT DESIGN

This function returns the previous velocity (i.e. the current velocity).

3.3.2.1.9.2.10.3.1 REQUIREMENTS ALLOCATION

N/A

3.3.2.1.9.2.10.3.2 LOCAL ENTITIES DESIGN

None.

3.3.2.1.9.2.10.3.3 INPUT/OUTPUT

None.

3.3.2.1.9.2.10.3.4 LOCAL DATA

None.

3.3.2.1.9.2.10.3.5 PROCESS CONTROL

Not applicable.

3.3.2.1.9.2.10.3.6 PROCESSING

The following describes the processing performed by this part:

```
function Current_Velocity return Velocity_Vectors is
begin
   return Previous_Velocity;
end Current_Velocity;
```

3.3.2.1.9.2.10.3.7 UTILIZATION OF OTHER ELEMENTS

UTILIZATION OF OTHER ELEMENTS IN TOP-LEVEL COMPONENT:

The following tables describe the elements used by this part but defined elsewhere in this top-level component:

Data types:

The following table summarizes the types required by this part and defined as generic parameters to the enclosing package body:

Name	Type	Description
Velocity_Vectors	private	Contains the east, north, and vertical components of the missile's current velocity

Data objects:

The following table summarizes the objects required by this part and defined in the enclosing package body:

Name	Type	Value	Description
Previous_Velocity	Velocity_Vectors	N/A	Velocity vector calculated during previous update operation

3.3.2.1.9.2.10.3.8 LIMITATIONS

None.

3.3.2.1.10 UNIT DESIGN

3.3.2.1.10.1 COMPUTE_GROUND_VELOCITY UNIT DESIGN (CATALOG #P210-0)

This function calculates the ground (i.e. horizontal) velocity of the missile from the east and north components of the missile velocity.

The ground velocity is as computed as follows:

```
GVel := Sqrt(EVel**2 + NVel **2)

where EVel = East Velocity
      NVel = North Velocity
```

3.3.2.1.10.1.1 REQUIREMENTS ALLOCATION

This part meets CAMP requirement R003.

3.3.2.1.10.1.2 LOCAL ENTITIES DESIGN

None.

3.3.2.1.10.1.3 INPUT/OUTPUT

GENERIC PARAMETERS:

The following generic parameters were previously defined in the package specification of Common_Navigation_Parts:

Data types:

The following table describes the generic formal types required by this part:

Name	Type	Description
Velocities	floating point type	Data type used to define velocity measurements
Velocity_Squared	floating point type	Data type resulting from multiply two objects of type velocity

Subprograms:

The following table describes the generic formal subroutines required by this part:

Name	Type	Description
"*"	function	Multiplication operator defining the operation: Velocities * Velocities => Velocity_Squared
Sqrt	function	Square root function

FORMAL PARAMETERS:

The following table describes this part's formal parameters:

Name	Type	Mode	Description
East_Velocity	Velocities	In	Current east velocity of missile
North_Velocity	Velocities	In	Current north velocity of missile

3.3.2.1.10.1.4 LOCAL DATA

None.

3.3.2.1.10.1.5 PROCESS CONTROL

Not applicable.

3.3.2.1.10.1.6 PROCESSING

The following describes the processing performed by this part:

```
separate (Common_Navigation_Parts)
function Compute_Ground_Velocity
            (East_Velocity  : Velocities;
             North_Velocity : Velocities) return Velocities is

begin

   return Sqrt(East_Velocity  * East_Velocity +
               North_Velocity * North_Velocity);
```

```
end Compute_Ground_Velocity;
```

3.3.2.1.10.1.7 UTILIZATION OF OTHER ELEMENTS

None.

3.3.2.1.10.1.8 LIMITATIONS

None.

3.3.2.1.10.2 COMPUTE_GRAVITATIONAL_ACCELERATION_LAT_IN UNIT DESIGN (CATALOG #P211-0)

The part computes the vertical acceleration due to gravity given the missile's current altitude and latitude. The calculations performed are as follows:

```
Grav := NGrav * (1 - (2*Alt/EER) + (GCF * Sin(Lat)**2) )

where NGrav = Nominal Gravity
      GCF   = Gravity Coefficient
      EER   = Earth Equatorial Radius
```

3.3.2.1.10.2.1 REQUIREMENTS ALLOCATION

This part meets CAMP requirement R005.

3.3.2.1.10.2.2 LOCAL ENTITIES DESIGN

None.

3.3.2.1.10.2.3 INPUT/OUTPUT

GENERIC PARAMETERS:

This part's generic parameters were previously defined in the package specification for Common_Navigation_Parts.

Data types:

The following table describes the generic formal types required by this part:

Name	Type	Description
Accelerations	floating point type	Data type used to define acceleration measurements
Distances	floating point type	Data type used to define distance measurements
Earth_Positions	floating point type	Data type used to define latitude and longitude measurements
Inverse_ Distances	floating point type	Data type used to define 1/distance measurements
Real	floating point type	Data type of Gravity_Coefficient
Sin_Cos_Ratio	floating point type	Data type used to define the results of a Sin calculation

Data objects:

The following table describes the generic formal objects required by this part:

Name	Type	Value	Description
Nominal_Gravity	Accelerations	N/A	Value of absolute mean normal gravity
One_Over_Earth_ Radius	Distances	N/A	Value for 1/earth equatorial radius
Gravity_Coefficient	Real	N/A	Coefficient of.gravity

Subprograms:

The following table describes the generic formal subroutines required by this part:

Name	Type	Description
"*"	function	Multiplication operator defining the operation: Accelerations * Real => Accelerations
"*"	function	Multiplication operator defining the operation: Sin_Cos_Ratio * Real => Real
"*"	function	Division operator defining the operation: Distances * Inverse_Distances => Real
Sin	function	Sine function

FORMAL PARAMETERS:

The following table describes this part's formal parameters:

Name	Type	Mode	Description
Current_Altitude	Distances	In	Current altitude of the missile
Current_Latitude	Earth_Positions	In	Current latitude of the missile

3.3.2.1.10.2.4 LOCAL DATA

Data objects:

The following table describes the data objects maintained by this part:

Name	Type	Value	Description
Sin_of_Lat	Sin_Cos_Ratio	N/A	Sine of Current_Latitude

3.3.2.1.10.2.5 PROCESS CONTROL

Not applicable.

3.3.2.1.10.2.6 PROCESSING

The following describes the processing performed by this part:

```
separate (Common_Navigation_Parts)
function Compute_Gravitational_Acceleration_Lat_In
            (Current_Altitude : Distances;
             Current_Latitude : Earth_Positions) return Accelerations is

-- ---------------------------
-- --declaration of variables-
-- ---------------------------

   Sin_of_Lat : Sin_Cos_Ratio;

------------------------------------------------------------
--begin function Compute_Gravitational_Acceleration_Lat_In
------------------------------------------------------------

begin

   Sin_of_Lat := Sin(Current_Latitude);
   return Nominal_Gravity *
         (1.0 -
          (Current_Altitude+Current_Altitude) * One_Over_Earth_Radius  +
          Sin_of_Lat * Sin_of_Lat * Gravity_Coefficient
          );

end Compute_Gravitational_Acceleration_Lat_In;
```

3.3.2.1.10.2.7 UTILIZATION OF OTHER ELEMENTS

None.

3.3.2.1.10.2.8 LIMITATIONS

None.

3.3.2.1.10.3 COMPUTE_GRAVITATIONAL_ACCELERATION_SIN_LAT_IN UNIT DESIGN (CATALOG #P212-0)

Given the missile's current altitude and the sine of the current latitude, this function computes the vertical acceleration due to gravity. The computations are performed as follows:

```
Grav := NGrav * (1 - (2 * Alt/EER) + (GCF * SinLat**2)))
```

where NGrav = Nominal Gravity
GCF = Gravity Coefficient
EER = Earth Equatorial Radius

3.3.2.1.10.3.1 REQUIREMENTS ALLOCATION

This part meets CAMP requirement R006.

3.3.2.1.10.3.2 LOCAL ENTITIES DESIGN

None.

3.3.2.1.10.3.3 INPUT/OUTPUT

GENERIC PARAMETERS:

This part's generic parameters were previously defined in the package specification of Common_Navigation_Parts:

Data types:

The following table describes the generic formal types required by this part:

Name	Type	Description
Accelerations	floating point type	Data type used to define acceleration measurements
Distances	floating point type	Data type used to define distance measurements
Inverse_Distances	floating point type	Data type used to define 1/distance measurements
Real	floating point type	Data type of Gravity_Coefficient
Sin_Cos_Ratio	floating point type	Data type used to define the results of a Sin calculation

Data objects:

The following table describes the generic formal objects required by this part:

Name	Type	Value	Description
Nominal_Gravity	Accelerations	N/A	Value of absolute mean normal gravity
One_Over_Earth_Radius	Distances	N/A	Value for 1/earth equatorial radius
Gravity_Coefficient	Real	N/A	Coefficient of gravity

Subprograms:

The following table describes the generic formal subroutines required by this part:

Name	Type	Description
"*"	function	Multiplication operator defining the operation: Accelerations * Real => Accelerations
"*"	function	Multiplication operator defining the operation: Sin_Cos_Ratio * Real => Real
"*"	function	Division operator defining the operation: Distances * Inverse_Distances => Real

FORMAL PARAMETERS:

The following table describes this part's formal parameters:

Name	Type	Mode	Description
Current_Altitude	Distances	In	Current altitude of the missile
Sin_Current_Latitude	Sin_Cos_Ratio	In	Sine of the current latitude of the missile

3.3.2.1.10.3.4 LOCAL DATA

None.

3.3.2.1.10.3.5 PROCESS CONTROL

Not applicable.

3.3.2.1.10.3.6 PROCESSING

The following describes the processing performed by this part:

```
separate (Common_Navigation_Parts)
function Compute_Gravitational_Acceleration_Sin_Lat_In
            (Current_Altitude     : Distances;
             Sin_Current_Latitude : Sin_Cos_Ratio)
            return Accelerations is

begin

   return Nominal_Gravity *
          (1.0 -
           (Current_Altitude+Current_Altitude) * One_Over_Earth_Radius +
           Sin_Current_Latitude * Sin_Current_Latitude *
             Gravity_Coefficient
           );

end Compute_Gravitational_Acceleration_Sin_Lat_In;
```

3.3.2.1.10.3.7 UTILIZATION OF OTHER ELEMENTS

None.

3.3.2.1.10.3.8 LIMITATIONS

None.

3.3.2.1.10.4 COMPUTE_HEADING UNIT DESIGN (CATALOG #P213-0)

This function computes the missile's current heading given the missile's current east and north velocities.

The computations are performed as follows:

Hdg := Arctan(EVel / NVel)

where EVel = East velocity
NVel = North velocity

3.3.2.1.10.4.1 REQUIREMENTS ALLOCATION

This part meets CAMP requirement R009.

3.3.2.1.10.4.2 LOCAL ENTITIES DESIGN

None.

3.3.2.1.10.4.3 INPUT/OUTPUT

GENERIC PARAMETERS:

This part's generic parameters were previously defined in the package specification of Common_Navigation_Parts.

Data types:

The following table describes the generic formal types required by this part:

Name	Type	Description
Velocities	floating point type	Data type defining velocity measurements
Headings	floating point type	Data type defining angular measurement

Subprograms:

The following table describes the generic formal subroutines required by this part:

Name	Type	Description
Arctan	function	Arctangent function

FORMAL PARAMETERS:

The following table describes this part's formal parameters:

Name	Type	Mode	Description
East_Velocity	Velocities	In	Current east velocity of missile
North_Velocity	Velocities	In	Current north velocity of missile

3.3.2.1.10.4.4 LOCAL DATA

None.

3.3.2.1.10.4.5 PROCESS CONTROL

Not applicable.

3.3.2.1.10.4.6 PROCESSING

The following describes the processing performed by this part:

```
separate (Common_Navigation_Parts)
function Compute_Heading (East_Velocity  : Velocities;
                          North_Velocity : Velocities) return Headings is

begin

-- --operation performed will be y/x (want East/North)
   return Arctan2 (Y => East_Velocity,
                   X => North_Velocity);

end Compute_Heading;
```

3.3.2.1.10.4.7 UTILIZATION OF OTHER ELEMENTS

None.

3.3.2.1.10.4.8 LIMITATIONS

None.

3.3.2.1.10.5 SCALAR_VELOCITY UNIT DESIGN (CATALOG #P215-0)

This function computes the scalar velocity given a velocity vector.

3.3.2.1.10.5.1 REQUIREMENTS ALLOCATION

This part meets CAMP requirement R039.

3.3.2.1.10.5.2 LOCAL ENTITIES DESIGN

None.

3.3.2.1.10.5.3 INPUT/OUTPUT

GENERIC PARAMETERS:

The following generic parameters were previously defined when this part was specified.

Data types:

The following table describes the generic formal types required by this part:

Name	Type	Description
Velocities	floating point type	Data type defining velocity measurements
Velocity_Vectors	private	Data type containing a velocity vector values

Subprograms:

The following table describes the generic formal subroutines required by this part:

Name	Type	Description
Vector_Length	function	Function used to calculate the length of a vector

FORMAL PARAMETERS:

The following table describes this part's formal parameters:

Name	Type	Mode	Description
Velocity_In	Velocity_Vectors	In	Missile's current velocity vector

3.3.2.1.10.5.4 LOCAL DATA

None.

3.3.2.1.10.5.5 PROCESS CONTROL

Not applicable.

3.3.2.1.10.5.6 PROCESSING

The following describes the processing performed by this part:

```
separate (Common_Navigation_Parts)
function Scalar_Velocity (Velocity_In : Velocity_Vectors)
                                  return Velocities is
begin
```

```
   return Vector_Length(Velocity_In);

end Scalar_Velocity;
```

3.3.2.1.10.5.7 UTILIZATION OF OTHER ELEMENTS

None.

3.3.2.1.10.5.8 LIMITATIONS

None.

3.3.2.1.10.6 COMPUTE_ROTATION_INCREMENTS (FUNCTION BODY) UNIT DESIGN (CATALOG #P216-0)

This function computes the rotation rate increments of the navigation coordinate system with respect to inertial space.

3.3.2.1.10.6.1 REQUIREMENTS ALLOCATION

This part meets CAMP requirement R157.

3.3.2.1.10.6.2 LOCAL ENTITIES DESIGN

None.

3.3.2.1.10.6.3 INPUT/OUTPUT

GENERIC PARAMETERS:

The following generic parameters were previously defined when this part was specified.

Data types:

The following table describes the generic formal types required by this part:

Name	Type	Description
Angle_Vectors	private type	Vector containing the east, north, vertical components of the rotation rate increments
Angular_Velocity_Vectors	private type	Vector containing the east, north, and vertical components of the total platform rate
Intervals	floating point type	Data type defining time measurements

Subprograms:

The following table describes the generic formal subroutines (operators) required by this part:

Name	Left Input Type	Right Input Type	Result Type
"*"	Angular_Velocity_Vectors	Intervals	Angle_Vectors

FORMAL PARAMETERS:

The following table describes the formal parameters of this part:

Name	Type	Mode	Description
Platform_ Rotation_Rate	Angular_ Velocity_Vectors	In	Contains the nominal components of the total rotation rate
Delta_Time	Intervals	In	Time interval over which the increments are to be calculated

3.3.2.1.10.6.4 LOCAL DATA

None.

3.3.2.1.10.6.5 PROCESS CONTROL

Not applicable.

3.3.2.1.10.6.6 PROCESSING

The following describes the processing performed by this part:

```
separate (Common_Navigation_Parts)
function Compute_Rotation_Increments
            (Platform_Rotation_Rate : Angular_Velocity_Vectors;
             Delta_Time             : Intervals)
            return Angle_Vectors is
begin

   return Platform_Rotation_Rate * Delta_Time;

end Compute_Rotation_Increments;
```

3.3.2.1.10.6.7 UTILIZATION OF OTHER ELEMENTS

None.

3.3.2.1.10.6.8 LIMITATIONS

None.

(This page left intentionally blank.)

```
package body Common_Navigation_Parts is

   package body Altitude_Integration is separate;

   function Compute_Ground_Velocity
               (East_Velocity  : Velocities;
                North_Velocity : Velocities) return Velocities is separate;

   function Compute_Gravitational_Acceleration_Lat_In
               (Current_Altitude : Distances;
                Current_Latitude : Earth_Positions)
               return Accelerations is separate;

   function Compute_Gravitational_Acceleration_Sin_Lat_In
               (Current_Altitude     : Distances;
                Sin_Current_Latitude : Sin_Cos_Ratio)
               return Accelerations is separate;

   function Compute_Heading (East_Velocity  : Velocities;
                             North_Velocity : Velocities)
                            return Headings is separate;

   package body Update_Velocity is separate;

   function Scalar_Velocity (Velocity_In : Velocity_Vectors)
                            return Velocities is separate;

   function Compute_Rotation_Increments
               (Platform_Rotation_Rate : Angular_Velocity_Vectors;
                Delta_Time             : Intervals)
               return Angle_Vectors is separate;

end Common_Navigation_Parts;
```

```
separate (Common_Navigation_Parts)
package body Altitude_Integration is

-- ---------------------
-- --local declarations-
-- ---------------------

   Previous_Vertical_Velocity : Velocities := Initial_Vertical_Velocity;
   Previous_Altitude          : Distances  := Initial_Altitude;

pragma PAGE;
   procedure Reinitialize (New_Vertical_Velocity : in Velocities;
                           New_Altitude          : in Distances) is

   begin

      Previous_Altitude          := New_Altitude;
      Previous_Vertical_Velocity := New_Vertical_Velocity;

    end Reinitialize;

pragma PAGE;
   function Integrate (Vertical_Velocity : Velocities;
                       Delta_Time        : Intervals := Default_Delta_Time)
                      return Distances is

--    --------------------
--    --local declarations
--    --------------------

      Altitude : Distances;

-- --------------------------
-- --begin function Integrate
-- --------------------------

   begin

      Altitude := Previous_Altitude +
                  ( (Vertical_Velocity + Previous_Vertical_Velocity) *
                    (0.5 * Delta_Time) );

      Previous_Vertical_Velocity := Vertical_Velocity;
      Previous_Altitude          := Altitude;

      return Altitude;

   end Integrate;

end Altitude_Integration;
```

```
separate (Common_Navigation_Parts)
function Compute_Ground_Velocity
            (East_Velocity  : Velocities;
             North_Velocity : Velocities) return Velocities is

begin

   return Sqrt(East_Velocity  * East_Velocity +
               North_Velocity * North_Velocity);

end Compute_Ground_Velocity;
```

```
separate (Common_Navigation_Parts)
function Compute_Gravitational_Acceleration_Lat_In
            (Current_Altitude : Distances;
             Current_Latitude : Earth_Positions) return Accelerations is

-- ---------------------------
-- --declaration of variables-
-- ---------------------------

   Sin_Of_Lat : Sin_Cos_Ratio;

--------------------------------------------------------------
--begin function Compute_Gravitational_Acceleration_Lat_In
--------------------------------------------------------------

begin

   Sin_Of_Lat := Sin(Current_Latitude);
   return Nominal_Gravity *
          (1.0 -
           (Current_Altitude+Current_Altitude) * One_Over_Earth_Radius  +
           Sin_Of_Lat * Sin_Of_Lat * Gravity_Coefficient
           );

end Compute_Gravitational_Acceleration_Lat_In;
```

```
separate (Common_Navigation_Parts)
function Compute_Gravitational_Acceleration_Sin_Lat_In
           (Current_Altitude     : Distances;
            Sin_Current_Latitude : Sin_Cos_Ratio)
           return Accelerations is

begin

   return Nominal_Gravity *
          (1.0 -
           (Current_Altitude+Current_Altitude) * One_Over_Earth_Radius +
           Sin_Current_Latitude * Sin_Current_Latitude *
             Gravity_Coefficient
           );

end Compute_Gravitational_Acceleration_Sin_Lat_In;
```

```
separate (Common_Navigation_Parts)
function Compute_Heading (East_Velocity  : Velocities;
                          North_Velocity : Velocities) return Headings is

begin

-- --operation performed will be y/x (want East/North)
   return Arctan2 (Y => East_Velocity,
                   X => North_Velocity);

end Compute_Heading;
```

```
separate (Common_Navigation_Parts)
package body Update_Velocity is

-- ------------------
-- --local variables-
-- ------------------

   Previous_Velocity : Velocity_Vectors := Initial_Velocity;

pragma PAGE;
   procedure Reinitialize (New_Velocity : in Velocity_Vectors) is

   begin

      Previous_Velocity := New_Velocity;

   end Reinitialize;

pragma PAGE;
   function Update (Nominal_Delta_Velocity : Velocity_Vectors;
                    Coriolis_Acceleration  : Acceleration_Vectors;
                    Delta_Time             : Intervals;
                    Gravity                : Acceleration_Vectors
                                               := Default_Gravity)
                   return Velocity_Vectors is

--    ----------------------------
--    --declaration of variables-
--    ----------------------------

      Current_Velocity  : Velocity_Vectors;
      Temp_A_V          : Acceleration_Vectors;
      Temp_V_V          : Velocity_Vectors;

-- ------------------------------
-- --beginning of function Update
-- ------------------------------

   begin

      Temp_A_V := Sparse_Right_Xy_Subtract(Coriolis_Acceleration, Gravity);
      Temp_V_V := Temp_A_V * Delta_Time;

      Current_Velocity  := Previous_Velocity +
                           Nominal_Delta_Velocity -
                           Temp_V_V;
      Previous_Velocity := Current_Velocity;

      return Current_Velocity;

   end Update;

pragma PAGE;
   function Current_Velocity return Velocity_Vectors is

   begin
```

```
      return Previous_Velocity;
   end Current_Velocity;
end Update_Velocity;
```

```
separate (Common_Navigation_Parts)
function Scalar_Velocity (Velocity_In : Velocity_Vectors)
                                  return Velocities is
begin

   return Vector_Length(Velocity_In);

end Scalar_Velocity;
```

```
separate (Common_Navigation_Parts)
function Compute_Rotation_Increments
            (Platform_Rotation_Rate : Angular_Velocity_Vectors;
             Delta_Time             : Intervals)
            return Angle_Vectors is
begin

   return Platform_Rotation_Rate * Delta_Time;

end Compute_Rotation_Increments;
```

3.3.2.2 NORTH POINTING NAVIGATION PARTS (BODY) TLCSC P003 (CATALOG #P257-0)

This part, which is a package body, contains all the CAMP parts which can be used in a north pointing navigation coordinate system environment.

The decomposition for this part is the same as that shown in the Top-Level Design Document.

3.3.2.2.1 REQUIREMENTS ALLOCATION

The following chart describes the allocation of requirements to the LLCSC's in this TLCSC.

Name	Requirements Allocation
Compute_Coriolis_Acceleration	R008
Radius_of_Curvature	R036
Total_Platform_Rotation_Rates	R012
Earth_Rotation_Rate	R014
Earth_Relative_Rotation_Rates	R026
Latitude_Integration	R027
Longitude_Integration	R031

3.3.2.2.2 LOCAL ENTITIES DESIGN

None.

3.3.2.2.3 INPUT/OUTPUT

None.

3.3.2.2.4 LOCAL DATA

None.

3.3.2.2.5 PROCESS CONTROL

Not applicable.

3.3.2.2.6 PROCESSING

The following describes the processing performed by this part:

```
package body North_Pointing_Navigation_Parts is

end North_Pointing_Navigation_Parts;
```

3.3.2.2.7 UTILIZATION OF OTHER ELEMENTS

None.

3.3.2.2.8 LIMITATIONS

None.

3.3.2.2.9 LLCSC DESIGN

3.3.2.2.9.1 RADIUS_OF_CURVATURE PACKAGE DESIGN (CATALOG #P259-0)

This LLCSC contains a function which computes the radius of curvature vector given the current altitude and current latitude.

The decomposition for this part is the same as that shown in the Top-Level Design Document.

3.3.2.2.9.1.1 REQUIREMENTS ALLOCATION

This part meets CAMP requirement R036.

3.3.2.2.9.1.2 LOCAL ENTITIES DESIGN

None.

3.3.2.2.9.1.3 INPUT/OUTPUT

GENERIC PARAMETERS:

The following generic parameters were previously defined in the package specification of North_Pointing_Navigation.

Data types:

The following table summarizes the generic formal types required by this part:

Name	Type	Description
Earth_Positions	floating point type	Data type of latitude and longitude values
Distances	floating point type	Data type of distance measurements
Inverse_ Distances	floating point type	Data type of 1/distance measurements
Indices	scalar type	Used to index arrays
Distance_Vectors	array	Array of distances
Real	floating point type	Used to define Earth_Flattening_Coefficient
Sin_Cos_Ratio	floating point type	Data type of results from a sin/cos function

Data objects:

The following table summarizes the generic formal objects required by this part:

Name	Type	Description
Earth_Radius	Distances	Radius of the earth
Earth_Flattening_ Coefficient	Real	Coefficient of earth flattening
One_Over_Earth_ Radius	Inverse_ Distances	Value of 1/radius of the earth

Subprograms:

The following table summarizes the generic formal subroutines required by this part:

Name	Type	Description
"*"	function	Defines the operation: Real * Sin_Cos_Ratio := Real
"*"	function	Defines the operation: Distances * Inverse_Distances := Real
"*"	function	Defines the operation: Distances * Real := Distances
"/"	function	Defines the operation: Distances / Real := Distances
Sin_Cos	procedure	Procedure returning the sine and cosine of an input earth position

FORMAL PARAMETERS:

The following table describes the formal parameters to the unit contained in this part:

Name	Type	Mode	Description
Latitude	Earth_Positions	In	Current latitude of the missile
Altitude	Distances	In	Current altitude of the missile

3.3.2.2.9.1.4 LOCAL DATA

Data objects:

The following table describes the data objects maintained by this part:

Name	Type	Description
E	Indices	Index to first element in vector
N	Indices	Index to second element in vector
V	Indices	Index to last element in vector

The following table describes the data objects maintained by the unit contained in this package:

Name	Type	Description
Cos_Of_Lat, Sin_Of_Lat	Sin_Cos_Ratio	Cosine and sine of input latitude
Flat_x_Sin_Squared	Sin_Cos_Ratio	Used for intermediate calculations
Long_Demoninator, Short_Denominator	Real	Used for intermediate calculations
Radius_Of_ Curvature	Distance_Vectors	Vector being calculated and returned

3.3.2.2.9.1.5 PROCESS CONTROL

Not applicable.

3.3.2.2.9.1.6 PROCESSING

The following describes the processing performed by this part:

```
   package body Radius_of_Curvature is

--    --------------------
--    --local declarations
--    --------------------

      E                   : constant Indices := Indices'FIRST;
      N                   : constant Indices := Indices'SUCC(E);
      V                   : constant Indices := Indices'LAST;
```

```
--    -----------
--    --unit body
--    -----------

      function Compute (Latitude : Earth_Positions;
                        Altitude : Distances)
                       return Distance_Vectors is

--       ---------------------------
--       --declaration of variables-
--       ---------------------------

         Cos_of_Lat          : Sin_Cos_Ratio;
         Flat_x_Sin_Squared  : Real;
         Long_Denominator    : Real;
         Radius_of_Curvature : Distance_Vectors;
         Short_Denominator   : Real;
         Sin_of_Lat          : Sin_Cos_Ratio;

--    ------------------------
--    --begin function Compute
--    ------------------------

      begin

         Sin_Cos(Latitude, Sin_of_Lat, Cos_of_Lat);

         Flat_x_Sin_Squared := Earth_Flattening_Coefficient *
                               (Sin_of_Lat * Sin_of_Lat);

         Short_Denominator := 1.0 -
                              Altitude * One_Over_Earth_Radius -
                              Flat_x_Sin_Squared;

         Long_Denominator  := Short_Denominator +
                              2.0 * Earth_Flattening_Coefficient *
                                (Cos_of_Lat*Cos_of_Lat);

         Radius_of_Curvature(E) := Earth_Radius / Short_Denominator;
         Radius_of_Curvature(N) := Earth_Radius / Long_Denominator;
         Radius_of_Curvature(V) := Earth_Radius * (1.0 - Flat_x_Sin_Squared) +
                                   Altitude;

         return Radius_of_Curvature;

      end Compute;

   end Radius_of_Curvature;
```

3.3.2.2.9.1.7 UTILIZATION OF OTHER ELEMENTS

None.

3.3.2.2.9.1.8 LIMITATIONS

None.

3.3.2.2.9.1.9 LLCSC DESIGN

None.

3.3.2.2.9.1.10 UNIT DESIGN

None.

3.3.2.2.9.2 EARTH_ROTATION_RATE PACKAGE DESIGN (CATALOG #P261-0)

This LLCSC contains a function which computes the rotation rate of the Earth given the current latitude.

The calculations performed are as follows:

```
Omega(E) := 0.0
Omega(N) := Earth_Angular_Velocity * cos(Lat)
Omega(U) := Earth_Angular_Velocity * sin(Lat)
```

The decomposition for this part is the same as that shown in the Top-Level Design Document.

3.3.2.2.9.2.1 REQUIREMENTS ALLOCATION

This part meets CAMP requirement R014.

3.3.2.2.9.2.2 LOCAL ENTITIES DESIGN

None.

3.3.2.2.9.2.3 INPUT/OUTPUT

GENERIC PARAMETERS:

The following generic parameters were previously defined in the package specification for North_Pointing_Navigation_Parts:

Data types:

The following table describes the generic formal types required by this part:

Name	Type	Description
Angular_ Velocities	floating point type	Data type defining angular velocity measurements
Indices	discrete	Index into arrays
Angular_ Velocity_Vectors	array	Array of angular velocities
Earth_Positions	floating point type	Data type defining longitude/latitude measurements
Sin_Cos_Ratio	floating point type	Data type defining results of sin/cos functions

Data objects:

The following table describes the generic formal objects required by this part:

Name	Type	Value	Description
Earth_Rate	Angular_ Velocities	N/A	Rate of rotation of the earth

Subprograms:

The following table describes the generic formal subroutines required by this part:

Name	Type	Description
"*"	function	Multiplication operator defining the operation: Angular_Velocities * Sin_Cos_Ratio := Angular_Velocities
Sin_Cos	procedure	Returns the sine and cosine of an input earth position

FORMAL PARAMETERS:

The following table describes the formal parameters to the unit in this package:

Name	Type	Mode	Description
Latitude	Earth_Positions	In	Current latitude of the missile

3.3.2.2.9.2.4 LOCAL DATA

Data objects:

wl

The following table describes the data objects maintained by this part:

Name	Type	Description
E, N, U	Indices	Indices into 1st, 2nd, and last elements of vector

The following table describes the data objects maintained local to the unit in this package:

Name	Type	Description
Cos_Of_Lat	Sin_Cos_Ratio	Cosine of input latitude
Rotation_Rate	Angular_Velocity_Vectors	Value being calculated and returned
Sin_Of_Lat	Sin_Cos_Ratio	Sine of input latitude

3.3.2.2.9.2.5 PROCESS CONTROL

Not applicable.

3.3.2.2.9.2.6 PROCESSING

The following describes the processing performed by this part:

```
   package body Earth_Rotation_Rate is

--    ---------------------------
--    --declaration of variables-
--    ---------------------------

      E : constant Indices := Indices'FIRST;
      N : constant Indices := Indices'SUCC(E);
      V : constant Indices := Indices'LAST;


--    -----------
--    --unit body
--    -----------

      function Compute (Latitude : Earth_Positions)
                        return Angular_Velocity_Vectors is

--       ---------------------------
--       --declaration of variables-
--       ---------------------------

         Cos_of_Lat    : Sin_Cos_Ratio;
         Rotation_Rate : Angular_Velocity_Vectors;
         Sin_of_Lat    : Sin_Cos_Ratio;
```

```
--    ------------------------
--    --beginning of function-
--    ------------------------

      begin

         Sin_Cos(Latitude, Sin_of_Lat, Cos_of_Lat);

         Rotation_Rate(E) := 0.0;
         Rotation_Rate(N) := Earth_Rate * Cos_of_Lat;
         Rotation_Rate(V) := Earth_Rate * Sin_of_Lat;

         return Rotation_Rate;

      end Compute;

   end Earth_Rotation_Rate;
```

3.3.2.2.9.2.7 UTILIZATION OF OTHER ELEMENTS

None.

3.3.2.2.9.2.8 LIMITATIONS

None.

3.3.2.2.9.2.9 LLCSC DESIGN

None.

3.3.2.2.9.2.10 UNIT DESIGN

None.

3.3.2.2.9.3 EARTH_RELATIVE_NAVIGATION_ROTATION_RATE PACKAGE DESIGN (CATALOG #P262-0)

This package contains a function which computes the rotation rate of the navigation coordinate system with respect to the Earth.

The calculations performed are as follows:

```
Rho(E) := - Vel(N) / Radius_of_Curvature(N)
Rho(N) :=   Vel(E) / Radius_of_Curvature(E)
Rho(V) :=   Rho(N) * tan(lat)
```

where Vel = the missile's current velocity
Lat = latitude of the missile

The decomposition for this part is the same as that shown in the Top-Level Design Document.

3.3.2.2.9.3.1 REQUIREMENTS ALLOCATION

This part meets CAMP requirement R026.

3.3.2.2.9.3.2 LOCAL ENTITIES DESIGN

None.

3.3.2.2.9.3.3 INPUT/OUTPUT

GENERIC PARAMETERS:

The following generic parameters were previously defined in the package specification of North_Pointing_Navigation_Parts:

Data types:

The following table describes the generic formal types required by this part:

Name	Type	Description
Indices	floating point type	Used to dimension imported vector types
Angular_Velocities	floating point type	Data type of angular velocity objects
Angular_Velocity_Vectors	array	Array of angular velocities
Distances	floating point type	Data type of distance measurements
Distance_Vectors	array	Array of distances
Earth_Positions	floating point type	Data type of longitude/latitude objects
Tan_Ratio	floating point type	Data type of results of tangent function
Velocities	floating point type	Data type of velocity measurements
Velocity_Vectors	array	Array of velocities

Subprograms:

The following table describes the generic formal subroutines required by this part:

Name	Type	Description
"*"	function	Multiplication operator defining the operation: Angular_Velocities * Tan_Ratio := Angular_Velocities
"/"	function	Division operator defining the operation: Velocities / Distances := Angular_Velocities
Tan	function	Tangent function

FORMAL PARAMETERS:

The following table describes the formal parameters to the unit contained in this part:

Name	Type	Mode	Description
Missile_Velocity	Velocity_Vectors	In	Missile's current velocity vector
Radius_of_Curvature	Distance_Vectors	In	East and north components of the radius of curvature
Latitude	Earth_Positions	In	Current latitude of the missile

3.3.2.2.9.3.4 LOCAL DATA

Data objects:

The following table describes the data objects maintained by this part:

Name	Type	Description
E, N, U	Indices	Indices into 1st, 2nd, and last positions of a vector

The following table describes the data objects maintained local to the unit contained in this part:

Name	Type	Description
Rho	Angular_Velocity_Vectors	Vector being calculated and returned

3.3.2.2.9.3.5 PROCESS CONTROL

Not applicable.

3.3.2.2.9.3.6 PROCESSING

The following describes the processing performed by this part:

```
   package body Earth_Relative_Navigation_Rotation_Rate is

--    ----------------------------
--    --declaration of variables-
--    ----------------------------
```

```
      E   : constant Indices := Indices'FIRST;
      N   : constant Indices := Indices'SUCC(E);
      V   : constant Indices := Indices'LAST;


--    -----------
--    --unit body
--    -----------

      function Compute (Missile_Velocity    : Velocity_Vectors;
                        Radius_of_Curvature : Distance_Vectors;
                        Latitude            : Earth_Positions)
                       return Angular_Velocity_Vectors is

--       --------------------------
--       --declaration of variables-
--       --------------------------

         Rho : Angular_Velocity_Vectors;

--    ------------------------
--    --beginning of function-
--    ------------------------

      begin

         Rho(e) := - Missile_Velocity(n) / Radius_of_Curvature(n);
         Rho(n) := Missile_Velocity(e) / Radius_of_Curvature(e);
         Rho(v) := Rho(n) * Tan(Latitude);

         return Rho;

      end Compute;

   end Earth_Relative_Navigation_Rotation_Rate;
```

3.3.2.2.9.3.7 UTILIZATION OF OTHER ELEMENTS

None.

3.3.2.2.9.3.8 LIMITATIONS

None.

3.3.2.2.9.3.9 LLCSC DESIGN

None.

3.3.2.2.9.3.10 UNIT DESIGN

None.

3.3.2.2.9.4 LATITUDE INTEGRATION PACKAGE DESIGN (CATALOG #P263-0)

This LLCSC, which is a package, calculates the missile's current latitude using trapezoidal integration of the east component of the Earth-relative rotation rate of the navigation coordinate system. This part can be used when a north pointing, local level coordinate system is being used for navigation.

The decomposition for this part is the same as that shown in the Top-Level Design Document.

3.3.2.2.9.4.1 REQUIREMENTS ALLOCATION

This part meets CAMP requirement R027.

3.3.2.2.9.4.2 LOCAL ENTITIES DESIGN

None.

3.3.2.2.9.4.3 INPUT/OUTPUT

GENERIC PARAMETERS:

Previously defined in package specification for North_Pointing_Navigation_-Parts.

Data types:

The following table describes the generic formal types required by this part:

Name	Type	Description
Angular_Velocities	floating point type	Data type of angular velocity objects
Earth_Positions	floating point type	Data type of latitude and longitude values
Intervals	floating point type	Data type of time values

Data objects:

The following table describes the generic formal objects required by this part:

Name	Type	Value	Description
Default_Delta	Intervals	N/A	Default time over which integration is to take place
Initial_East_Rho	Angular_Velocities	N/A	Initial value of the East component of earth-relative rotation
Initial_Latitude	Earth_Position	N/A	Initial value of the latitude

Subprograms:

The following table describes the generic formal subroutines required by this part:

Name	Type	Description
"*"	function	Multiplication operator defining the operation: Angular_Velocities * Intervals := Earth_Positions

3.3.2.2.9.4.4 LOCAL DATA

Data objects:

The following table describes the data objects maintained local to this part:

Name	Type	Value	Description
Previous_East_Rho	Angular_Velocities	N/A	Previous value of the East component of the Earth-relative rotation
Previous_Latitude	Earth_Positions	N/A	Previous value of the latitude

3.3.2.2.9.4.5 PROCESS CONTROL

Not applicable.

3.3.2.2.9.4.6 PROCESSING

The following describes the processing performed by this part:

```
   package body Latitude_Integration is

--    --------------------
--    --local declarations
--    --------------------

      Previous_East_Rho : Angular_Velocities := Initial_East_Rho;
      Previous_Latitude : Earth_Positions    := Initial_Latitude;

   end Latitude_Integration;
```

3.3.2.2.9.4.7 UTILIZATION OF OTHER ELEMENTS

None.

3.3.2.2.9.4.8 LIMITATIONS

None.

3.3.2.2.9.4.9 LLCSC DESIGN

None.

3.3.2.2.9.4.10 UNIT DESIGN

3.3.2.2.9.4.10.1 REINITIALIZE UNIT DESIGN

This unit, which is a procedure, reinitializes the previous values of the east component of the earth-relative rotation and latitude.

3.3.2.2.9.4.10.1.1 REQUIREMENTS ALLOCATION

This unit partially meets CAMP requirement R027.

3.3.2.2.9.4.10.1.2 LOCAL ENTITIES DESIGN

None.

3.3.2.2.9.4.10.1.3 INPUT/OUTPUT

FORMAL PARAMETERS:

The following table describes this part's formal parameters:

Name	Type	Mode	Description
East_Rho	Angular_Velocities	In	Initial value for previous_east_rho
Latitude	Earth_Positions	In	Initial value for previous_latitude

3.3.2.2.9.4.10.1.4 LOCAL DATA

None.

3.3.2.2.9.4.10.1.5 PROCESS CONTROL

Not applicable.

3.3.2.2.9.4.10.1.6 PROCESSING

The following describes the processing performed by this part:

```
procedure Reinitialize (East_Rho : in Angular_Velocities;
                        Latitude : in Earth_Positions) is

begin

   Previous_East_Rho := East_Rho;
   Previous_Latitude := Latitude;

end Reinitialize;
```

3.3.2.2.9.4.10.1.7 UTILIZATION OF OTHER ELEMENTS

UTILIZATION OF ANCESTRAL ELEMENTS:

The following tables describe the elements used by this part but defined in one or more ancestral units:

Data types:

The following table summarizes the types required by this part and defined as generic formal types in the package specification for Latitude_Integration:

Name	Type	Description
Angular_ Velocities	floating point type	Data type of angular velocity objects
Earth_Positions	floating point type	Data type of latitude and longitude values
Intervals	floating point type	Data type of time values

Data objects:

The following table describes the data objects required by this part and maintained local to the Latitude_Integration package.

Name	Type	Value	Description
Previous_ East_Rho	Angular_ Velocities	N/A	Previous value of the East component of the Earth-relative rotation
Previous_ Latitude	Earth_ Positions	N/A	Previous value of the latitude

3.3.2.2.9.4.10.1.8 LIMITATIONS

None.

3.3.2.2.9.4.10.2 INTEGRATE UNIT DESIGN

This unit, which is a function, calculates a new latitude through trapezoidal integration.

3.3.2.2.9.4.10.2.1 REQUIREMENTS ALLOCATION

This unit partially meets CAMP requirement R027.

3.3.2.2.9.4.10.2.2 LOCAL ENTITIES DESIGN

None.

3.3.2.2.9.4.10.2.3 INPUT/OUTPUT

FORMAL PARAMETERS:

The following table describes this part's formal parameters:

Name	Type	Mode	Description
East_Rho	Angular_Velocities	In	Current value of the east component of the earth-relative rotation rate of the navigation coordinate system
Delta_Time	Intervals	In	Time interval over which the integration is to take place

3.3.2.2.9.4.10.2.4 LOCAL DATA

None.

3.3.2.2.9.4.10.2.5 PROCESS CONTROL

Not applicable.

3.3.2.2.9.4.10.2.6 PROCESSING

The following describes the processing performed by this part:

```
function Integrate (East_Rho   : Angular_Velocities;
                    Delta_Time : Intervals := Default_Delta_Time)
                   return Earth_Positions is
```

```
      Latitude : Earth_Positions;

   begin

      Latitude := Previous_Latitude +
                  ((East_Rho - Previous_East_Rho) * (0.5 * Delta_Time) );

      Previous_Latitude := Latitude;
      Previous_East_Rho := East_Rho;

      return Latitude;

   end Integrate;
```

3.3.2.2.9.4.10.2.7 UTILIZATION OF OTHER ELEMENTS

UTILIZATION OF ANCESTRAL ELEMENTS:

The following tables describe the elements used by this part but defined in one or more ancestral units:

Data types:

The following table summarizes the types required by this part and defined as generic formal types in the package specification for Latitude_Integration:

Name	Type	Description
Angular_Velocities	floating point type	Data type of angular velocity objects
Earth_Positions	floating point type	Data type of latitude and longitude values
Intervals	floating point type	Data type of time values

Data objects:

The following table describes the data objects required by this part and maintained local to the Latitude_Integration package.

Name	Type	Value	Description
Previous_East_Rho	Angular_Velocities	N/A	Previous value of the East component of the Earth-relative rotation
Previous_Latitude	Earth_Positions	N/A	Previous value of the latitude

Subprograms and task entries:

The following describes the subprograms required by this part and defined as generic formal subprograms to the Latitude_Integration package:

Name	Type	Description
"*"	function	Multiplication operator defining the operation: Angular_Velocities * Intervals := Earth_Positions

3.3.2.2.9.4.10.2.8 LIMITATIONS

None.

3.3.2.2.9.5 LONGITUDE INTEGRATION PACKAGE DESIGN (CATALOG #P264-0)

This LLCSC, which is a package, calculates the missile's current longitude using trapezoidal integration. This part can be used when a north pointing local level coordinate system is being used for navigation.

The decomposition for this part is the same as that shown in the Top-Level Design Document.

3.3.2.2.9.5.1 REQUIREMENTS ALLOCATION

This part meets CAMP requirement R031.

3.3.2.2.9.5.2 LOCAL ENTITIES DESIGN

None.

3.3.2.2.9.5.3 INPUT/OUTPUT

GENERIC PARAMETERS:

Previously defined in package specification for North_Pointing_Navigation_Parts

Data types:

The following table describes the generic formal types required by this part:

Name	Type	Description
Angular_Velocities	floating point type	Data type of angular velocity objects
Earth_Positions	floating point type	Data type of latitude/longitude values
Intervals	floating point type	Data type of time values
Sin_Cos_Ratio	floating point type	Data type of values return by sin/cos functions

Data objects:

The following table describes the generic formal objects required by this part:

Name	Type	Value	Description
Default_Delta	Intervals	N/A	Default time over which integration is to take place
Initial_North_Rho	Angular_Velocities	N/A	Initial value of the North component of the earth-relative rotation rate of the navigation coordinate system
Initial_Latitude	Earth_Positions	N/A	Initial value of the latitude
Initial_Longitude	Earth_Positions	N/A	Initial value of the longitude

Subprograms:

The following table describes the generic formal subroutines required by this part:

Name	Type	Description
"/"	function	Division operator defining the operation: Angular_Velocities / Sin_Cos_Ratio := Angular_Velocities
"*"	function	Multiplication operator defining the operation: Angular_Velocities * Intervals := Earth_Positions
Cos	function	Cosine function

3.3.2.2.9.5.4 LOCAL DATA

Data objects:

The following table describes the data objects maintained local to this part:

Name	Type	Description
Previous_Integration_Variable	Angular_Velocities	Previous value of the integration variable
Previous_Longitude	Earth_Positions	Previous longitude value .

3.3.2.2.9.5.5 PROCESS CONTROL

Not applicable.

3.3.2.2.9.5.6 PROCESSING

The following describes the processing performed by this part:

```
   package body Longitude_Integration is

--    ------------------
--    --local variables-
--    ------------------

      Previous_Integration_Variable : Angular_Velocities
                                      := Initial_North_Rho /
                                         Cos(Initial_Latitude);

      Previous_Longitude            : Earth_Positions := Initial_Longitude;

   end Longitude_Integration;
```

3.3.2.2.9.5.7 UTILIZATION OF OTHER ELEMENTS

None.

3.3.2.2.9.5.8 LIMITATIONS

None.

3.3.2.2.9.5.9 LLCSC DESIGN

None.

3.3.2.2.9.5.10 UNIT DESIGN

3.3.2.2.9.5.10.1 REINITIALIZE UNIT DESIGN

This unit, which is a procedure, reinitializes the previous integration variable and the previous longitude.

3.3.2.2.9.5.10.1.1 REQUIREMENTS ALLOCATION

This part partially meets CAMP requirement R031.

3.3.2.2.9.5.10.1.2 LOCAL ENTITIES DESIGN

None.

3.3.2.2.9.5.10.1.3 INPUT/OUTPUT

FORMAL PARAMETERS:

The following table describes this part's formal parameters:

Name	Type	Mode	Description
North_Rho	Angular_ Velocities	In	Initial value of the North component of the earth-relative rotation rate of the navigation coordinate system
Latitude	Earth_Positions	In	Initial value of the latitude
Longitude	Earth_Positions	In	Initial value of the longitude

3.3.2.2.9.5.10.1.4 LOCAL DATA

None.

3.3.2.2.9.5.10.1.5 PROCESS CONTROL

Not applicable.

3.3.2.2.9.5.10.1.6 PROCESSING

The following describes the processing performed by this part:

```
procedure Reinitialize (North_rho : in Angular_Velocities;
                        Latitude  : in Earth_Positions;
                        Longitude : in Earth_Positions) is

begin

   Previous_Integration_Variable := North_Rho / Cos(Latitude);
   Previous_Longitude            := Longitude;

end Reinitialize;
```

3.3.2.2.9.5.10.1.7 UTILIZATION OF OTHER ELEMENTS

UTILIZATION OF ANCESTRAL ELEMENTS:

The following tables describe the elements used by this part but defined in one or more ancestral units:

Data types:

The following table summarizes the types required by this part and defined as generic formal parameters to the Longitude_Integration package:

Name	Type	Description
Angular_Velocities	floating point type	Data type of angular velocity objects
Earth_Positions	floating point type	Data type of latitude/longitude values
Sin_Cos_Ratio	floating point type	Data type of values return by sin/cos functions

Data objects:

The following objects are required by this part and maintained in the package body of Longitude_Integration:

Name	Type	Description
Previous_Integration_Variable	Angular_Velocities	Previous value of the integration variable
Previous_Longitude	Earth_Positions	Previous longitude value

Subprograms and task entries:

The following table summarizes the subroutines and task entries required by this part and defined as generic formal parameters to the Longitude_Integration package:

Name	Type	Description
"/"	function	Division operator defining the operation: Angular_Velocities / Sin_Cos_Ratio := Angular_Velocities
Cos	function	Cosine function

3.3.2.2.9.5.10.1.8 LIMITATIONS

None.

3.3.2.2.9.5.10.2 INTEGRATE UNIT DESIGN

This unit, which is a function, calculates the new longitude using trapezoidal integration.

3.3.2.2.9.5.10.2.1 REQUIREMENTS ALLOCATION

This unit partially meets CAMP requirement R031.

3.3.2.2.9.5.10.2.2 LOCAL ENTITIES DESIGN

None.

3.3.2.2.9.5.10.2.3 INPUT/OUTPUT

FORMAL PARAMETERS:

The following table describes this part's formal parameters:

Name	Type	Mode	Description
North_Rho	Angular_Velocities	In	Current value of the North component of the earth-relative rotation rate of the navigation coordinate system
Latitude	Earth_Positions	In	Current latitude
Delta_Time	Intervals	In	Time interval over which the integration is to take place

3.3.2.2.9.5.10.2.4 LOCAL DATA

Data objects:

The following table describes the local data maintained by this part:

Name	Type	Value	Description
Longitude	Earth_Positions	N/A	Longitude being calculated
Temp_PIV	Angular_Velocities	N/A	Temporary previous integration variable

3.3.2.2.9.5.10.2.5 PROCESS CONTROL

Not applicable.

3.3.2.2.9.5.10.2.6 PROCESSING

The following describes the processing performed by this part:

```
      function Integrate (North_Rho  : Angular_Velocities;
                          Latitude   : Earth_Positions;
                          Delta_Time : Intervals := Default_Delta_Time)
                         return Earth_Positions is

--       --------------------------
--       --declaration of variables-
--       --------------------------
```

```
            Longitude : Earth_Positions;
            Temp_PIV  : Angular_Velocities;

--      -------------------------
--      --begin function Integrate
--      -------------------------

      begin

         Temp_PIV  := North_Rho / Cos(Latitude);

         Longitude := Previous_Longitude +
                      ((Temp_PIV - Previous_Integration_Variable) *
                       (0.5 * Delta_Time));

         Previous_Integration_Variable := Temp_PIV;
         Previous_Longitude            := Longitude;

         return Longitude;

      end Integrate;
```

3.3.2.2.9.5.10.2.7 UTILIZATION OF OTHER ELEMENTS

UTILIZATION OF ANCESTRAL ELEMENTS:

The following tables describe the elements used by this part but defined in one or more ancestral units:

Data types:

The following table summarizes the types required by this part and defined as generic formal parameters to the Longitude_Integration package:

Name	Type	Description
Angular_ Velocities	floating point type	Data type of angular velocity objects
Earth_Positions	floating point type	Data type of latitude/longitude values
Intervals	floating point type	Data type of time values
Sin_Cos_Ratio	floating point type	Data type of values return by sin/cos functions

Data objects:

The following objects are required by this part and maintained in the package body of Longitude_Integration:

Name	Type	Description
Previous_Integration_Variable	Angular_Velocities	Previous value of the integration variable
Previous_Longitude	Earth_Positions	Previous longitude value

Subprograms and task entries:

The following table summarizes the subroutines and task entries required by this part and defined as generic formal parameters to the Longitude_Integration package:

Name	Type	Description
"/"	function	Division operator defining the operation: Angular_Velocities / Sin_Cos_Ratio := Angular_Velocities
"*"	function	Multiplication operator defining the operation: Angular_Velocities * Intervals := Earth_Positions
Cos	function	Cosine function

3.3.2.2.9.5.10.2.8 LIMITATIONS

None.

3.3.2.2.10 UNIT DESIGN

3.3.2.2.10.1 COMPUTE CORIOLIS ACCELERATION UNIT DESIGN (CATALOG #P258-0)

This function computes the components of the coriolis acceleration vector when a local level, north pointing coordinate system is being used for navigation.

The calculations performed are as follows:

CA := (2 * Omega_Vector + Rho_Vector) x Velocity_Vector

3.3.2.2.10.1.1 REQUIREMENTS ALLOCATION

This part meets CAMP requirement R008.

3.3.2.2.10.1.2 LOCAL ENTITIES DESIGN

None.

3.3.2.2.10.1.3 INPUT/OUTPUT

GENERIC PARAMETERS:

The following generic parameters were previously defined in package specificatic . for North_Pointing_Navigation_Parts.

Data types:

The following table describes the generic formal types required by this part:

Name	Type	Description
Velocity_Vectors	private	Array of velocities
Angular_Velocity_ Vectors	private	Array of angular velocities
Acceleration_Vectors	private	Array of accelerations

Subprograms:

The following table describes the generic formal subroutines required by this part:

Name	Type	Description
Sparse_Right_X_Add	function	Defines the way two angular velocity vectors are added when the first component in the second vector equals 0
Cross_Product	function	Performs a cross product operation between an angular velocity vector and a velocity vector, resulting in an acceleration vector

FORMAL PARAMETERS:

The following table describes this part's formal parameters:

Name	Type	Mode	Description
Velocity_In	Velocity_Vectors	In	Contains the E, N, and U components of the missile's velocity
Rho_In	Angular_Velocity_ Vectors	In	Contains the E, N, and U components of the missile's velocity
Omega_In	Angular_Velocity_ Vectors	In	Contains the N and U components of the current earth's rotation rate

3.3.2.2.10.1.4 LOCAL DATA

Data objects:

The following table describes the data objects maintained by this part:

Name	Type	Description
Coriolis_Acceleration	Acceleration_Vectors	This is the value being calculated and returned
Temp_AVV	Angular_Velocity_Vectors	Temporary angular velocity vector used for intermediate calculations

3.3.2.2.10.1.5 PROCESS CONTROL

Not applicable.

3.3.2.2.10.1.6 PROCESSING

The following describes the processing performed by this part:

```
   function Compute_Coriolis_Acceleration
              (Velocity_In : Velocity_Vectors;
               Rho_In      : Angular_Velocity_Vectors;
               Omega_In    : Angular_Velocity_Vectors)
              return Acceleration_Vectors is

--    ---------------------------
--    --declaration of variables-
--    ---------------------------

      Coriolis_Acceleration : Acceleration_Vectors;
      Temp_AVV              : Angular_Velocity_Vectors;

-- ------------------------
-- --beginning of function-
-- ------------------------

   begin

      Temp_AVV              := Sparse_Right_X_Add(Omega_In, Omega_In);
      Temp_AVV              := Sparse_Right_X_Add(Rho_In, Temp_AVV);
      Coriolis_Acceleration := Cross_Product(Temp_AVV, Velocity_In);

      return Coriolis_Acceleration;

   end Compute_Coriolis_Acceleration;
```

3.3.2.2.10.1.7 UTILIZATION OF OTHER ELEMENTS

None.

3.3.2.2.10.1.8 LIMITATIONS

None.

3.3.2.2.10.2 TOTAL_PLATFORM_ROTATION_RATES UNIT DESIGN (CATALOG #P260-0)

This function computes the rotation rate of the navigation coordinate system with respect to inertial space.

The calculations performed are as follows:

```
TPR := Rho_Vector + Omega_Vector
```

3.3.2.2.10.2.1 REQUIREMENTS ALLOCATION

See top header.

3.3.2.2.10.2.2 LOCAL ENTITIES DESIGN

None.

3.3.2.2.10.2.3 INPUT/OUTPUT

GENERIC PARAMETERS:

The following generic parameters were previously defined when this part was specified.

Data types:

The following table describes the generic formal types required by this part:

Name	Type	Description
Angular_Velocity_Vectors	private	Array of angular velocities

Subprograms:

The following table describes the generic formal subroutines required by this part:

Name	Type	Description
Sparse_Right_X_Add	function	Adds two angular velocity vectors assuming the x-component of the right vector equals 0.0

FORMAL PARAMETERS:

The following table describes this part's formal parameters:

Name	Type	Mode	Description
Rho_Vector	Angular_Velocity_Vectors	In	Rotation rate vector of the navigation coordinate system with respect to the earth
Omega_Vector	Angular_Velocity_Vectors	In	Earth's rotation rate vector

3.3.2.2.10.2.4 LOCAL DATA

None.

3.3.2.2.10.2.5 PROCESS CONTROL

Not applicable.

3.3.2.2.10.2.6 PROCESSING

The following describes the processing performed by this part:

```
function Total_Platform_Rotation_Rates
            (Rho_Vector   : Angular_Velocity_Vectors;
             Omega_Vector : Angular_Velocity_Vectors)
            return Angular_Velocity_Vectors is
begin
   return Sparse_Right_X_Add (Left  => Rho_Vector,
                              Right => Omega_Vector);
end Total_Platform_Rotation_Rates;
```

3.3.2.2.10.2.7 UTILIZATION OF OTHER ELEMENTS

None.

3.3.2.2.10.2.8 LIMITATIONS

None.

(This page left intentionally blank.)

```
package body North_Pointing_Navigation_Parts is

pragma PAGE;
   function Compute_Coriolis_Acceleration
              (Velocity_In : Velocity_Vectors;
               Rho_In      : Angular_Velocity_Vectors;
               Omega_In    : Angular_Velocity_Vectors)
              return Acceleration_Vectors is

--     ---------------------------
--     --declaration of variables-
--     ---------------------------

      Coriolis_Acceleration : Acceleration_Vectors;
      Temp_Avv              : Angular_Velocity_Vectors;

-- ------------------------
-- --beginning of function-
-- ------------------------

   begin

      Temp_Avv              := Sparse_Right_X_Add(Omega_In, Omega_In);
      Temp_Avv              := Sparse_Right_X_Add(Rho_In, Temp_Avv);
      Coriolis_Acceleration := Cross_Product(Temp_Avv, Velocity_In);

      return Coriolis_Acceleration;

   end Compute_Coriolis_Acceleration;

pragma PAGE;
   package body Radius_Of_Curvature is

--     --------------------
--     --local declarations
--     --------------------

      E                     : constant Indices := Indices'FIRST;
      N                     : constant Indices := Indices'SUCC(E);
      V                     : constant Indices := Indices'LAST;

pragma PAGE;
--     -----------
--     --unit body
--     -----------

      function Compute (Latitude : Earth_Positions;
                        Altitude : Distances)
                       return Distance_Vectors is

--        ---------------------------
--        --declaration of variables-
--        ---------------------------

          Cos_Of_Lat          : Sin_Cos_Ratio;
          Flat_X_Sin_Squared  : Real;
          Long_Denominator    : Real;
```

```
         Radius_Of_Curvature : Distance_Vectors;
         Short_Denominator   : Real;
         Sin_Of_Lat          : Sin_Cos_Ratio;

--    -------------------------
--    --begin function Compute
--    -------------------------

      begin

         Sin_Cos(Latitude, Sin_Of_Lat, Cos_Of_Lat);

         Flat_X_Sin_Squared := Earth_Flattening_Coefficient *
                               (Sin_Of_Lat * Sin_Of_Lat);

         Short_Denominator := 1.0 -
                              Altitude * One_Over_Earth_Radius -
                              Flat_X_Sin_Squared;

         Long_Denominator  := Short_Denominator +
                              2.0 * Earth_Flattening_Coefficient *
                                (Cos_Of_Lat*Cos_Of_Lat);

         Radius_Of_Curvature(E) := Earth_Radius / Short_Denominator;
         Radius_Of_Curvature(N) := Earth_Radius / Long_Denominator;
         Radius_Of_Curvature(V) := Earth_Radius * (1.0 - Flat_X_Sin_Squared) +
                                   Altitude;

         return Radius_Of_Curvature;

      end Compute;

   end Radius_Of_Curvature;

pragma PAGE;
   function Total_Platform_Rotation_Rates
               (Rho_Vector   : Angular_Velocity_Vectors;
                Omega_Vector : Angular_Velocity_Vectors)
               return Angular_Velocity_Vectors is
   begin
      return Sparse_Right_X_Add (Left  => Rho_Vector,
                                 Right => Omega_Vector);
   end Total_Platform_Rotation_Rates;

pragma PAGE;
   package body Earth_Rotation_Rate is

--    ---------------------------
--    --declaration of variables-
--    ---------------------------

      E : constant Indices := Indices'FIRST;
      N : constant Indices := Indices'SUCC(E);
      V : constant Indices := Indices'LAST;

pragma PAGE;
--    -----------
```

```
--     --unit body
--     -----------

       function Compute (Latitude : Earth_Positions)
                         return Angular_Velocity_Vectors is

--        ---------------------------
--        --declaration of variables-
--        ---------------------------

          Cos_Of_Lat    : Sin_Cos_Ratio;
          Rotation_Rate : Angular_Velocity_Vectors;
          Sin_Of_Lat    : Sin_Cos_Ratio;

--     ------------------------
--     --beginning of function-
--     ------------------------

       begin

          Sin_Cos(Latitude, Sin_Of_Lat, Cos_Of_Lat);

          Rotation_Rate(E) := 0.0;
          Rotation_Rate(N) := Earth_Rate * Cos_Of_Lat;
          Rotation_Rate(V) := Earth_Rate * Sin_Of_Lat;

          return Rotation_Rate;

       end Compute;

   end Earth_Rotation_Rate;

pragma PAGE;
   package body Earth_Relative_Navigation_Rotation_Rate is

--     ---------------------------
--     --declaration of variables-
--     ---------------------------

       E   : constant Indices := Indices'FIRST;
       N   : constant Indices := Indices'SUCC(E);
       V   : constant Indices := Indices'LAST;

pragma PAGE;
--     -----------
--     --unit body
--     -----------

       function Compute (Missile_Velocity    : Velocity_Vectors;
                         Radius_Of_Curvature : Distance_Vectors;
                         Latitude            : Earth_Positions)
                        return Angular_Velocity_Vectors is

--        ---------------------------
--        --declaration of variables-
--        ---------------------------
```

```
         Rho : Angular_Velocity_Vectors;

--       ------------------------
--       --beginning of function-
--       ------------------------

      begin

         Rho(E) := - Missile_Velocity(N) / Radius_Of_Curvature(N);
         Rho(N) := Missile_Velocity(E) / Radius_Of_Curvature(E);
         Rho(V) := Rho(N) * Tan(Latitude);

         return Rho;

      end Compute;

   end Earth_Relative_Navigation_Rotation_Rate;

pragma PAGE;
   package body Latitude_Integration is

--       --------------------
--       --local declarations
--       --------------------

      Previous_East_Rho : Angular_Velocities := Initial_East_Rho;
      Previous_Latitude : Earth_Positions    := Initial_Latitude;

pragma PAGE;
      procedure Reinitialize (East_Rho : in Angular_Velocities;
                              Latitude : in Earth_Positions) is

      begin

         Previous_East_Rho := East_Rho;
         Previous_Latitude := Latitude;

      end Reinitialize;

pragma PAGE;
      function Integrate (East_Rho   : Angular_Velocities;
                          Delta_Time : Intervals := Default_Delta_Time)
                         return Earth_Positions is

         Latitude : Earth_Positions;

      begin

         Latitude := Previous_Latitude +
                     ((East_Rho - Previous_East_Rho) * (0.5 * Delta_Time) );

         Previous_Latitude := Latitude;
         Previous_East_Rho := East_Rho;

         return Latitude;

      end Integrate;
```

```
   end Latitude_Integration;

pragma PAGE;
   package body Longitude_Integration is

--    ------------------
--    --local variables-
--    ------------------

      Previous_Integration_Variable : Angular_Velocities
                                    := Initial_North_Rho /
                                       Cos(Initial_Latitude);

      Previous_Longitude            : Earth_Positions := Initial_Longitude;

pragma PAGE;
      procedure Reinitialize (North_Rho : in Angular_Velocities;
                              Latitude  : in Earth_Positions;
                              Longitude : in Earth_Positions) is

      begin

         Previous_Integration_Variable := North_Rho / Cos(Latitude);
         Previous_Longitude            := Longitude;

      end Reinitialize;

pragma PAGE;
      function Integrate (North_Rho  : Angular_Velocities;
                          Latitude   : Earth_Positions;
                          Delta_Time : Intervals := Default_Delta_Time)
                         return Earth_Positions is

--       --------------------------
--       --declaration of variables-
--       --------------------------

           Longitude : Earth_Positions;
           Temp_Piv  : Angular_Velocities;

--    --------------------------
--    --begin function Integrate
--    --------------------------

      begin

         Temp_Piv  := North_Rho / Cos(Latitude);

         Longitude := Previous_Longitude +
                      ((Temp_Piv - Previous_Integration_Variable) *
                       (0.5 * Delta_Time));

         Previous_Integration_Variable := Temp_Piv;
         Previous_Longitude            := Longitude;

         return Longitude;
```

```
      end Integrate;
   end Longitude_Integration;
end North_Pointing_Navigation_Parts;
```

3.3.2.3 WANDER_AZIMUTH_NAVIGATION_PARTS (PACKAGE BODY) TLCSC P002 (CATALOG #P234-0)

This package contairs the specifications for all parts which can be used in a Wander Azimuth navigation coordinate system environment.

The decomposition for this part is the same as that shown in the Top-Level Design Document.

3.3.2.3.1 REQUIREMENTS ALLOCATION

The following chart summarizes the allocation of CAMP requirements to this part:

Name	Requirements Allocation
Compute_East_Velocity	R185
Compute_North_Velocity	R186
Compute_Earth_Relative_Horizontal_Velocities	R001
Compute_Total_Angular_Velocity	R004
Compute_Coriolis_Acceleration	R007
Coriolis_Acceleration_From_Total_Rates	R187
Radius_of_Curvature	R035
Total_Platform_Rotation_Rate	R011
Earth_Rotation_Rate	R013
Earth_Relative_Rotation_Rate	R025
Latitude	R029
Compute_Latitude_using_Arctangent	R030
Compute_Longitude	R033
Compute_Wander_Azimuth_Angle	R028

3.3.2.3.2 LOCAL ENTITIES DESIGN

None.

3.3.2.3.3 INPUT/OUTPUT

None.

3.3.2.3.4 LOCAL DATA

None.

3.3.2.3.5 PROCESS CONTROL

Not applicable.

3.3.2.3.6 PROCESSING

The following describes the processing performed by this part:

```
package body Wander_Azimuth_Navigation_Parts is

   function Compute_East_Velocity
            (Nominal_East_Velocity  : Velocities;
             Nominal_North_Velocity : Velocities;
             Current_Wander_Angle   : Wander_Angles)
            return Velocities is separate;

   function Compute_East_Velocity_with_Sin_Cos_In
            (Nominal_East_Velocity  : Velocities;
             Nominal_North_Velocity : Velocities;
             Sine_of_Wander_Angle   : Sin_Cos_Ratio;
             Cosine_of_Wander_Angle : Sin_Cos_Ratio)
            return Velocities is separate;

   function Compute_North_Velocity
            (Nominal_East_Velocity  : Velocities;
             Nominal_North_Velocity : Velocities;
             Current_Wander_Angle   : Wander_Angles)
            return Velocities is separate;

   function Compute_North_Velocity_with_Sin_Cos_In
            (Nominal_East_Velocity  : Velocities;
             Nominal_North_Velocity : Velocities;
             Sine_of_Wander_Angle   : Sin_Cos_Ratio;
             Cosine_of_Wander_Angle : Sin_Cos_Ratio)
            return Velocities is separate;

   procedure Compute_Earth_Relative_Horizontal_Velocities
             (Nominal_East_Velocity  : in  Velocities;
              Nominal_North_Velocity : in  Velocities;
              Current_Wander_Angle   : in  Wander_Angles;
              East_Velocity          : out Velocities;
              North_Velocity         : out Velocities)
             is separate;

   procedure Compute_Earth_Relative_Horizontal_Velocities_with_Sin_Cos_In
             (Nominal_East_Velocity  : in     Velocities;
              Nominal_North_Velocity : in     Velocities;
              Sine_of_Wander_Angle   : in     Sin_Cos_Ratio;
              Cosine_of_Wander_Angle : in     Sin_Cos_Ratio;
              East_Velocity          :    out Velocities;
              North_Velocity         :    out Velocities)
             is separate;

   function Compute_Total_Angular_Velocity
            (Rho_East  : Angular_Velocities;
             Rho_North : Angular_Velocities)
            return Angular_Velocities is separate;

   function Compute_Coriolis_Acceleration
            (Velocity_in : Velocity_Vectors;
             Rho_in      : Angular_Velocity_Vectors;
```

```
                Omega_in    : Angular_Velocity_Vectors)
              return Acceleration_Vectors is separate;

    package body Coriolis_Acceleration_From_Total_Rates is separate;

    function Compute_Curvatures
              (DC_P_V          : Direction_Cosine_Matrix_Elements;
               DC_P_NE         : Direction_Cosine_Matrix_Elements;
               DC_P_NN         : Direction_Cosine_Matrix_Elements;
               Current_Altitude : Distances)
              return Inverse_Distance_Vectors is separate;

    function Total_Platform_Rotation_Rate
              (Rho_Vector   : Angular_Velocity_Vectors;
               Omega_Vector : Angular_Velocity_Vectors)
              return Angular_Velocity_Vectors is separate;

    package body Earth_Rotation_Rate is separate;

    function Compute_Earth_Relative_Navigation_Rotation_Rate
              (Missile_Velocity    : Velocity_Vectors;
               Curvatures          : Inverse_Distance_Vectors)
              return Angular_Velocity_Vectors is separate;

    function Compute_Latitude (Sin_Latitude : Sin_Cos_Ratio)
                              return Earth_Positions is separate;

    function Compute_Latitude_using_Arctangent
              (DC_P_V  : Direction_Cosine_Matrix_Elements;
               DC_P_NE : Direction_Cosine_Matrix_Elements;
               DC_P_NN : Direction_Cosine_Matrix_Elements)
              return Earth_Positions is separate;

    function Compute_Latitude_using_Two_Value_Arctangent
              (DC_P_V  : Direction_Cosine_Matrix_Elements;
               DC_P_NE : Direction_Cosine_Matrix_Elements;
               DC_P_NN : Direction_Cosine_Matrix_Elements)
              return Earth_Positions is separate;

    function Compute_Longitude
              (DC_R_V : Direction_Cosine_Matrix_Elements;
               DC_G_V : Direction_Cosine_Matrix_Elements)
              return Earth_Positions is separate;

    function Compute_Longitude_using_Two_Value_Arctangent
              (DC_R_V : Direction_Cosine_Matrix_Elements;
               DC_G_V : Direction_Cosine_Matrix_Elements)
              return Earth_Positions is separate;

    function Compute_Wander_Azimuth_Angle
              (DC_P_NE : Direction_Cosine_Matrix_Elements;
               DC_P_NN : Direction_Cosine_Matrix_Elements)
              return Wander_Angles is separate;

    function Compute_Wander_Azimuth_Angle_using_Two_Value_Arctangent
              (DC_P_NE : Direction_Cosine_Matrix_Elements;
               DC_P_NN : Direction_Cosine_Matrix_Elements)
```

```
            return Wander_Angles is separate;

end Wander_Azimuth_Navigation_Parts;
```

3.3.2.3.7 UTILIZATION OF OTHER ELEMENTS

None.

3.3.2.3.8 LIMITATIONS

None.

3.3.2.3.9 LLCSC DESIGN

3.3.2.3.9.1 CORIOLIS_ACCELERATION_FROM_TOTAL_RATES PACKAGE DESIGN (CATALOG #P240-0)

This package contains a function which computes the Coriolis acceleration given the velocity vector, Rho vector, and Total Rates vector.

The computations are performed as follows:

(2 * Omega Vector + Rho Vector) x Velocity Vector

The decomposition for this part is the same as that shown in the Top-Level Design Document.

3.3.2.3.9.1.1 REQUIREMENTS ALLOCATION

This part meets CAMP requirement R187.

3.3.2.3.9.1.2 LOCAL ENTITIES DESIGN

None.

3.3.2.3.9.1.3 INPUT/OUTPUT

GENERIC PARAMETERS:

The following generic parameters for this part were previously defined in the package specification of Wander_Azimuth_Navigation_Parts:

Data types:

The following table describes the generic formal types required by this part:

Name	Type	Description
Acceleration_Vectors	private	Array of acceleration measurements
Angular_Velocities	floating point type	Data type defining angular velocity measurements
Indices	discrete type	Used to dimension Angular_Velocity_Vectors
Angular_Velocity_Vectors	array	Array of angular velocities
Velocity_Vectors	private	Array of velocity measurements

Subprograms:

The following table describes the generic formal subroutines required by this part:

Name	Type	Description
Cross_Product	function	Cross product function performing a cross product operation on Angular_Velocity_Vectors and Velocity_Vectors to obtain Acceleration_Vectors

FORMAL PARAMETERS:

The following table describes the formal parameters to the unit contained in this part:

Name	Type	Mode	Description
Velocity_in	Velocity_Vectors	In	Contains the nominal East, North and vertical components of the missile's velocity
Omega	Angular_Velocity_Vectors	In	Contains the nominal East, North and vertical components of the missile's velocity
Total_Rates	Angular_Velocity_Vectors	In	Contains the nominal East, North and vertical components of the total platform rate

3.3.2.3.9.1.4 LOCAL DATA

Data objects:

The following table describes the data objects maintained by this part:

Name	Type	Value	Description
E	Indices	Indices'FIRST	Index into first element of array
N	Indices	Indices'SUCC(E)	Index into second element of array
U	Indices	Indices'LAST	Index into last element of array

The following table describes the data objects maintained by the unit contained in this part:

Name	Type	Value	Description
Temp_A_V_V	Angular_ Velocity_ Vectors	N/A	Temporary angular velocity vector

3.3.2.3.9.1.5 PROCESS CONTROL

Not applicable.

3.3.2.3.9.1.6 PROCESSING

The following describes the processing performed by this part:

```
separate (Wander_Azimuth_Navigation_Parts)
package body Coriolis_Acceleration_from_Total_Rates is

-- ---------------------------
-- --declaration of variables-
-- ---------------------------

   E               : constant Indices := Indices'FIRST;
   N               : constant Indices := Indices'SUCC(E);
   U               : constant Indices := Indices'LAST;


-- ------------
-- --unit body-
-- ------------

   function Compute (Velocity_In : Velocity_Vectors;
                     Omega       : Angular_Velocity_Vectors;
                     Total_Rates : Angular_Velocity_Vectors)
                    return Acceleration_Vectors is

--    ---------------------------
--    --declaration of variables-
--    ---------------------------

      Temp_A_V_V      : Angular_Velocity_Vectors;

-- ---------------------------
```

```
-- --begin function Compute
-- ------------------------

   begin

      Temp_A_V_V(E) := Omega(E) + Total_Rates(E);
      Temp_A_V_V(N) := Omega(N) + Total_Rates(N);
      Temp_A_V_V(U) := Omega(U) + Omega(U);

      return Cross_Product(Temp_A_V_V, Velocity_In);

   end Compute;

end Coriolis_Acceleration_From_Total_Rates;
```

3.3.2.3.9.1.7 UTILIZATION OF OTHER ELEMENTS

None.

3.3.2.3.9.1.8 LIMITATIONS

None.

3.3.2.3.9.1.9 LLCSC DESIGN

None.

3.3.2.3.9.1.10 UNIT DESIGN

None.

3.3.2.3.9.2 EARTH_ROTATION_RATE PACKAGE DESIGN (CATALOG #P243-0)

This package contains a function which computes the rotation rate of the Earth given some elements of the Direction Cosine matrix.

The computations are performed as follows:

Omega(i) := Direction Cosine Matrix(P,i) * Earth's rotation rate

The decomposition for this part is the same as that shown in the Top-Level Design Document.

3.3.2.3.9.2.1 REQUIREMENTS ALLOCATION

This part meets CAMP requirement R013.

3.3.2.3.9.2.2 LOCAL ENTITIES DESIGN

None.

3.3.2.3.9.2.3 INPUT/OUTPUT

GENERIC PARAMETERS:

The following generic parameters were previously defined in the package specification of Wander_Azimuth_Navigation_Parts:

Data types:

The following table describes the generic formal types required by this part:

Name	Type	Description
Angular_Velocities	floating point type	Data type defining angular velocity measurements
Indices	discrete type	Used to dimension angular_velocity_vector
Angular_Velocity_Vectors	array	Array of angular velocities dimensioned by Indices type
Direction_Cosine_Matrix_Elements	floating point type	Used to define elements of the direction cosine matrix

Data objects:

The following table describes the generic formal objects required by this part:

Name	Type	Value	Description
Earth_Rate	Angular_Velocities	N/A	Rotation rate of the Earth

Subprograms:

The following table describes the generic formal subroutines required by this part:

Name	Type	Description
"*"	function	Multiplication operator defining the operation: Direction_Cosine_Matrix_Elements * Angular_Velocities => Angular_Velocities

FORMAL PARAMETERS:

The following table describes the formal parameters to the unit in this part:

Name	Type	Mode	Description
DC_P_V	Direction_Cosine_Matrix_Elements	In	Sine of the current latitude of the missile
DC_P_NE	Direction_Cosine_Matrix_Elements	In	Product of the sine of the wander angle and the cosine of the latitude
DC_P_NN	Direction_Cosine_Matrix_Elements	In	Product of the cosine of the wander angle and the cosine of the latitude

3.3.2.3.9.2.4 LOCAL DATA

Data objects:

The following table describes the data objects maintained by this part:

Name	Type	Value	Description
NE	Indices	Indices'FIRST	Index into first element of array
NN	Indices	Indices 'SUCC(NE)	Index into second element of array
V	Indices	Indices'LAST	Index into third element of array

The following table describes the data objects maintain by the unit in this part:

Name	Type	Value	Description
Rotation_Rate	Angular_Velocity_Vectors	N/A	Vector being calculated and returned

3.3.2.3.9.2.5 PROCESS CONTROL

Not applicable.

3.3.2.3.9.2.6 PROCESSING

The following describes the processing performed by this part:

```
separate (Wander_Azimuth_Navigation_Parts)
package body Earth_Rotation_Rate is

-- --------------------
-- --local declarations
```

```
-- --------------------

   NE             : constant Indices := Indices'FIRST;
   NN             : constant Indices := Indices'SUCC(NE);
   V              : constant Indices := Indices'LAST;


-- ------------
-- --unit body-
-- ------------

   function Compute (DC_P_V  : Direction_Cosine_Matrix_Elements;
                     DC_P_NE : Direction_Cosine_Matrix_Elements;
                     DC_P_NN : Direction_Cosine_Matrix_Elements)
                    return Angular_Velocity_Vectors is

--    ---------------------------
--    --declaration of variables-
--    ---------------------------

      Rotation_Rate : Angular_Velocity_Vectors;

-- ------------------------
-- --beginning of function-
-- ------------------------

   begin

      Rotation_Rate(NE) := DC_P_NE * Earth_Rate;
      Rotation_Rate(NN) := DC_P_NN * Earth_Rate;
      Rotation_Rate(V)  := DC_P_V * Earth_Rate;

      return Rotation_Rate;

   end Compute;

end Earth_Rotation_Rate;
```

3.3.2.3.9.2.7 UTILIZATION OF OTHER ELEMENTS

None.

3.3.2.3.9.2.8 LIMITATIONS

None.

3.3.2.3.9.2.9 LLCSC DESIGN

None.

3.3.2.3.9.2.10 UNIT DESIGN

None.

3.3.2.3.10 UNIT DESIGN

3.3.2.3.10.1 COMPUTE_EAST_VELOCITY UNIT DESIGN (CATALOG #P235-0)

This function computes the east velocity given the nominal east and north velocities and the wander angle.

The computations are performed as follows:

```
VelE := VelNE * Cos(WA) - VelNN * Sin(WA)

where VelNE = nominal east velocity
      VelNN = nominal north velocity
      WA    = wander angle
```

3.3.2.3.10.1.1 REQUIREMENTS ALLOCATION

This part meets CAMP requirement R158.

3.3.2.3.10.1.2 LOCAL ENTITIES DESIGN

None.

3.3.2.3.10.1.3 INPUT/OUTPUT

GENERIC PARAMETERS:

The following generic parameters were previously defined for this part's specification in the Wander_Azimuth_Navigation_Parts package specification:

Data types:

The following table describes the generic formal types required by this part:

Name	Type	Description
Sin_Cos_Ratio	floating point type	Data type of results from a sine or cosine operation
Velocities	floating point type	Data type of velocity measurements
Wander_Angles	floating point type	Data type of wander angle measurements

Subprograms:

The following table describes the generic formal subroutines required by this part:

Name	Type	Description
"*"	function	Multiplication operator defining the operation: Velocities * Sin_Cos_Ratio => Velocities
Sin_Cos	procedure	Returns the sine and cosine of a wander angle

FORMAL PARAMETERS:

The following table describes this part's formal parameters:

Name	Type	Mode	Description
Nominal_East_Velocity	Velocities	In	Velocity which would be east if the wander angle was 0
Nominal_North_Velocity	Velocities	In	Velocity which would be north if the wander angle was 0
Current_Wander_Angle	Wander_Angles	In	Current wander azimuth angle

3.3.2.3.10.1.4 LOCAL DATA

Data objects:

The following table describes the data objects maintained by this part:

Name	Type	Value	Description
Cos_of_Angle	Sin_Cos_Ratio	N/A	Cosine of current_wander_angle
Sin_of_Angle	Sin_Cos_Ratio	N/A	Sine of current_wander_angle

3.3.2.3.10.1.5 PROCESS CONTROL

Not applicable.

3.3.2.3.10.1.6 PROCESSING

The following describes the processing performed by this part:

```
separate (Wander_Azimuth_Navigation_Parts)
function Compute_East_Velocity
            (Nominal_East_Velocity  : Velocities;
             Nominal_North_Velocity : Velocities;
             Current_Wander_Angle   : Wander_Angles) return Velocities is

-- --------------------------
```

```
-- --declaration of variables-
-- ---------------------------

   Cos_of_Angle  : Sin_Cos_Ratio;
   Sin_of_Angle  : Sin_Cos_Ratio;

-------------------------------------
--begin function Compute_East_Velocity
-------------------------------------

begin

   Sin_Cos(Current_Wander_Angle, Sin_of_Angle, Cos_of_Angle);

   return ( Nominal_East_Velocity * Cos_of_Angle -
            Nominal_North_Velocity * Sin_of_Angle);

end Compute_East_Velocity;
```

3.3.2.3.10.1.7 UTILIZATION OF OTHER ELEMENTS

None.

3.3.2.3.10.1.8 LIMITATIONS

None.

3.3.2.3.10.2 COMPUTE_NORTH_VELOCITY UNIT DESIGN (CATALOG #P236-0)

This function calculates the North velocity component of a missile given the nominal north and east velocities of the missile.

The computations are performed as follows:

```
    VelN := VelNE * Sin(WA) + VelNN * Cos(WA)

    where VelNE = nominal east velocity
          VelNN = nominal north velocity
          WA    = wander angle
```

3.3.2.3.10.2.1 REQUIREMENTS ALLOCATION

This part meets CAMP requirement R186.

3.3.2.3.10.2.2 LOCAL ENTITIES DESIGN

None.

3.3.2.3.10.2.3 INPUT/OUTPUT

GENERIC PARAMETERS:

The following generic parameters were previously defined for this part's specification in the Wander_Azimuth_Navigation_Parts package specification:

Data types:

The following table describes the generic formal types required by this part:

Name	Type	Description
Sin_Cos_Ratio	floating point type	Data type defining results of a sine or cosine operation
Velocities	floating point type	Data type defining velocity measurements
Wander_Angle	floating point type	Data type defining wander angle measurements

Subprograms:

The following table describes the generic formal subroutines required by this part:

Name	Type	Description
"*"	function	Multiplication operator defining the operation: Velocities * Sin_Cos_Ratio => Velocities
Sin_Cos	procedure	Procedure calculating sine and cosine values of a wander angle

FORMAL PARAMETERS:

The following table describes this part's formal parameters:

Name	Type	Mode	Description
Nominal_East_Velocity	Velocities	In	Velocity in the direction which would be East if the wander angle was 0
Nominal_North_Velocity	Velocities	In	Velocity in the direction which would be North if the wander angle was 0
Current_Wander_Angle	Wander_Angles	In	Value of the current wander angle

3.3.2.3.10.2.4 LOCAL DATA

Data objects:

The following table describes the data objects maintained by this part:

Name	Type	Value	Description
Cos_of_Angle	Sin_Cos_Ratio	N/A	Cosine of current_wander_angle
Sin_of_Angle	Sin_Cos_Ratio	N/A	Sine of current_wander_angle

3.3.2.3.10.2.5 PROCESS CONTROL

Not applicable.

3.3.2.3.10.2.6 PROCESSING

The following describes the processing performed by this part:

```
separate (Wander_Azimuth_Navigation_Parts)
function Compute_North_Velocity
            (Nominal_East_Velocity  : Velocities;
             Nominal_North_Velocity : Velocities;
             Current_Wander_Angle   : Wander_Angles) return Velocities is

-- ---------------------------
-- --declaration of variables-
-- ---------------------------

   Cos_of_Angle   : Sin_Cos_Ratio;
   Sin_of_Angle   : Sin_Cos_Ratio;

---------------------------------------
--begin function Compute_North_Velocity
---------------------------------------

begin

   Sin_Cos(Current_Wander_Angle, Sin_of_Angle, Cos_of_Angle);

   return ( Nominal_East_Velocity * Sin_of_Angle +
            Nominal_North_Velocity * Cos_of_Angle);

end Compute_North_Velocity;
```

3.3.2.3.10.2.7 UTILIZATION OF OTHER ELEMENTS

None.

3.3.2.3.10.2.8 LIMITATIONS

None.

3.3.2.3.10.3 COMPUTE_EARTH_RELATIVE_HORIZONTAL_VELOCITIES UNIT DESIGN (CATALOG #P237-0)

This procedure computes the north and east velocity given the nominal east and north velocities and the wander angle.

The computations are performed as follows:

```
VelE := VelNE * Cos(WA) - VelNN * Sin(WA)
VelN := VelNE * Sin(WA) + VelNN * Cos(WA)

where VelNE = nominal east velocity
      VelNN = nominal north velocity
      WA    = wander angle
```

3.3.2.3.10.3.1 REQUIREMENTS ALLOCATION

This part meets CAMP requirement R001.

3.3.2.3.10.3.2 LOCAL ENTITIES DESIGN

None.

3.3.2.3.10.3.3 INPUT/OUTPUT

GENERIC PARAMETERS:

The following generic parameters were previously defined for this part's specification in the Wander_Azimuth_Navigation_Parts package specification:

Data types:

The following table describes the generic formal types required by this part:

Name	Type	Description
Sin_Cos_Ratio	floating point type	Data type defining results of a sine or cosine operation
Velocities	floating point type	Data type defining velocity measurements
Wander_Angles	floating point type	Data type defining wander angle measurements

Subprograms:

The following table describes the generic formal subroutines required by this part:

Name	Type	Description
"*"	function	Multiplication operator defining the operation: Velocities * Sin_Cos_Ratio => Velocities
Sin_Cos	procedure	Returns the sine and cosine of an input angle

FORMAL PARAMETERS:

The following table describes this part's formal parameters:

Name	Type	Mode	Description
Nominal_East_Velocity	Velocities	In	Velocity in the direction that would be East if the wander angle was 0
Nominal_North_Velocity	Velocities	In	Velocity in the direct that would be North if the wander angle was 0
Current_Wander_Angle	Wander_Angles	In	Value of the current wander angle
East_Velocity	Velocities	Out	Current east velocity of the missile
North_Velocity	Velocities	Out	Current north velocity of the missile

3.3.2.3.10.3.4 LOCAL DATA

Data objects:

The following table describes the data objects maintained by this part:

Name	Type	Value	Description
Cos_of_Angle	Sin_Cos_Ratio	N/A	Cosine of current_wander_angle
Sin_of_Angle	Sin_Cos_Ratio	N/A	Sine of current_wander_angle

3.3.2.3.10.3.5 PROCESS CONTROL

Not applicable.

3.3.2.3.10.3.6 PROCESSING

The following describes the processing performed by this part:

separate (Wander_Azimuth_Navigation_Parts)

```
procedure Compute_Earth_Relative_Horizontal_Velocities
            (Nominal_East_Velocity  : in  Velocities;
             Nominal_North_Velocity : in  Velocities;
             Current_Wander_Angle   : in  Wander_Angles;
             East_Velocity          : out Velocities;
             North_Velocity         : out Velocities) is

-- ---------------------------
-- --declaration of variables-
-- ---------------------------

   Cos_of_Angle : Sin_Cos_Ratio;
   Sin_of_Angle : Sin_Cos_Ratio;

------------------------------------------------------------------
--begin procedure Compute_Earth_Relative_Horizontal_Velocities
------------------------------------------------------------------

begin

   Sin_Cos(Current_Wander_Angle, Sin_of_Angle, Cos_of_Angle);

   East_Velocity := Nominal_East_Velocity * Cos_of_Angle -
                    Nominal_North_Velocity * Sin_of_Angle;
   North_Velocity := Nominal_East_Velocity * Sin_of_Angle +
                     Nominal_North_Velocity * Cos_of_Angle;

end Compute_Earth_Relative_Horizontal_Velocities;
```

3.3.2.3.10.3.7 UTILIZATION OF OTHER ELEMENTS

None.

3.3.2.3.10.3.8 LIMITATIONS

None.

3.3.2.3.10.4 COMPUTE_TOTAL_ANGULAR_VELOCITY UNIT DESIGN (CATALOG #P238-0)

This function computes the total angular velocity (which represents the ground speed) given the nominal north and east nominal velocities.

The computations are performed as follows:

```
    TAVel := Sqrt( RhoNN**2 + RhoNE**2)
```

where RhoNN = nominal north component of the current rotation rate of the missile's navigation frame with respect to the earth RhoNE = nominal east component of the current rotation rate of the missile's navigation frame with respect to the earth

3.3.2.3.10.4.1 REQUIREMENTS ALLOCATION

This part meets CAMP requirement R004.

3.3.2.3.10.4.2 LOCAL ENTITIES DESIGN

None.

3.3.2.3.10.4.3 INPUT/OUTPUT

GENERIC PARAMETERS:

The following generic parameters were previously defined for this part's specification in the Wander_Azimuth_Navigation_Parts package specification:

Data types:

The following table describes the generic formal types required by this part:

Name	Type	Description
Angular_Velocities	floating point type	Data type used for angular velocity measurements
Angular_Velocity_Squared	floating point type	Data type of object resulting from the multiplication of two objects of type Angular_Velocity

Subprograms:

The following table describes the generic formal subroutines required by this part:

Name	Type	Description
"*"	function	Multiplication operator defining the operation: Angular_Velocities * Angular_Velocities => Angular_Velocity_Squared
Sqrt	function	Square root function

FORMAL PARAMETERS:

The following table describes this part's formal parameters:

Name	Type	Mode	Description
Rho_East	Angular_ Velocities	In	Nominal east component of the current rotation rate of the missile's navigation frame with repsect to the Earth
Rho_North	Angular_ Velocities	In	Nominal north component of the current rotation rate of the missile's navigation frame with repsect to the Earth

3.3.2.3.10.4.4 LOCAL DATA

None.

3.3.2.3.10.4.5 PROCESS CONTROL

Not applicable.

3.3.2.3.10.4.6 PROCESSING

The following describes the processing performed by this part:

```
separate (Wander_Azimuth_Navigation_Parts)
function Compute_Total_Angular_Velocity
            (Rho_East  : Angular_Velocities;
             Rho_North : Angular_Velocities) return Angular_Velocities is

begin

   return Sqrt(Rho_East * Rho_East + Rho_North * Rho_North);

end Compute_Total_Angular_Velocity;
```

3.3.2.3.10.4.7 UTILIZATION OF OTHER ELEMENTS

None.

3.3.2.3.10.4.8 LIMITATIONS

None.

3.3.2.3.10.5 COMPUTE_CORIOLIS_ACCELERATION UNIT DESIGN (CATALOG #P239-0)

This function computes the Coriolis acceleration given the velocity vector, Rho vector, and Omega vector.

The computations are performed as follows:

CorAccel := (2 * Omega Vector + Rho Vector) x Velocity Vector

3.3.2.3.10.5.1 REQUIREMENTS ALLOCATION

This part meets CAMP requirement R007.

3.3.2.3.10.5.2 LOCAL ENTITIES DESIGN

None.

3.3.2.3.10.5.3 INPUT/OUTPUT

GENERIC PARAMETERS:

The following generic parameters were previously defined for this part's specification in the Wander_Azimuth_Navigation_Parts package specification:

Data types:

The following table describes the generic formal types required by this part:

Name	Type	Description
Acceleration_Vector	private	Contains East, North, and vertical components of the Coriolis acceleration
Angular_Velocity_Vector	private	Contains East, North, and vertical components of the Earth's rotation rate
Velocity_Vector	private	Contains East, North, and vertical components of the missile's velocity

Subprograms:

The following table describes the generic formal subroutines required by this part:

Name	Type	Description
Sparse_Right_X_Add	function	Adds two Angular_Velocity_Vector's assuming the x-component of the second vector equals 0
Sparse_Right_Z_Add	function	Adds two Angular_Velocity_Vector's assuming the z-component of the second vector equals 0
Cross_Product	function	Cross product function crossing an Angular_Velocity_Vector with a Velocity_Vector to obtain an Acceleration_Vector

FORMAL PARAMETERS:

The following table describes this part's formal parameters:

Name	Type	Mode	Description
Velocity_In	Velocity_Vector	In	Contains the nominal East, North, and vertical components of the missile's current velocity
Rho_in	Angular_ Velocity_Vector	In	Contains the nominal East, North, and vertical components of the current earth-relative navigation frame rotation rate (vertical component assumed to equal 0 and is therefore ignored)
Omega_in	Angular_ Velocity_Vector	In	Contains the nominal East, North, and vertical components of the current Earth's rotation rate

3.3.2.3.10.5.4 LOCAL DATA

Data objects:.

The following table describes the data objects maintained by this part:

Name	Type	Value	Description
Temp_A_V_V	Angular_ Velocity_ Vector	N/A	Temporary angular velocity vector

3.3.2.3.10.5.5 PROCESS CONTROL

Not applicable.

3.3.2.3.10.5.6 PROCESSING

The following describes the processing performed by this part:

```
separate (Wander_Azimuth_Navigation_Parts)
function Compute_Coriolis_Acceleration
            (Velocity_In : Velocity_Vectors;
             Rho_In      : Angular_Velocity_Vectors;
             Omega_In    : Angular_Velocity_Vectors)
            return Acceleration_Vectors is

-- ----------------------------
-- --declaration of variables-
```

```
-- -------------------------

   Temp_A_V_V      : Angular_Velocity_Vectors;

-------------------------------------------
--begin function Compute_Coriolis_Acceleration
-------------------------------------------

begin

   Temp_A_V_V        := Omega_In + Omega_In;
   Temp_A_V_V        := Sparse_Right_Z_Add(Temp_A_V_V, Rho_In);

   return Cross_Product(Temp_A_V_V, Velocity_In);

end Compute_Coriolis_Acceleration;
```

3.3.2.3.10.5.7 UTILIZATION OF OTHER ELEMENTS

None.

3.3.2.3.10.5.8 LIMITATIONS

None.

3.3.2.3.10.6 COMPUTE_CURVATURES UNIT DESIGN (CATALOG #P241-0)

This generic function computes the inverse radius of curvature vector given three elements of the direction cosine matrix and the current altitude.

3.3.2.3.10.6.1 REQUIREMENTS ALLOCATION

This part meets CAMP requirement R035.

3.3.2.3.10.6.2 LOCAL ENTITIES DESIGN

None.

3.3.2.3.10.6.3 INPUT/OUTPUT

GENERIC PARAMETERS:

The following generic parameters were previously defined in the package specification of Wander_Azimuth_Navigation_Parts:

Data types:

The following table summarizes the generic formal types required by this part:

Name	Type	Description
Indices	discrete type	Used to dimension Distance_Vectors
Direction_Cosine_Matrix_Elements	floating point type	Used to define elements of a direction cosine matrix
Distances	floating point type	Data type defining distance measurements
Inverse_Distances	floating point type	Data type of one over distance measurements
Real	floating point type	Data type used to define Earth_Flattening_Coefficient
Inverse_Distance_Vectors	array	Array of "Inverse_Distances" dimensioned by "Indices"

Data objects:

The following table summarizes the generic formal objects required by this part:

Name	Type	Description
Earth_Flattening_Coefficient	Real	Coefficient of earth flattening
One_Over_Earth_Radius	Inverse_Distances	Value of 1/Earth radius
NE	Indices	Indices'FIRST
NN	Indices	Indices'SUCC(NE)
V	Indices	Indices'LAST
Two_Flat	Real	2.0 * Earth_Flattening_Coefficient

Subprograms:

The following table summarizes the generic formal subroutines required by this part:

Name	Type	Description
"*"	function	Multiplication operator defining the operation: Distances * Inverse_Distances => Real
"*"	function	Multiplication operator defining the operation: Real * Direction_Cosine_Matrix_Elements => Real
"*"	function	Division operator defining the operation: Inverse_Distances * Real => Inverse_Distances

FORMAL PARAMETERS:

The following table describes this part's formal parameters:

Name	Type	Mode	Description
DC_P_V	Direction_Cosine_Matrix_Elements	In	Sine of the current latitude of the missile
DC_P_NE	Direction_Cosine_Matrix_Elements	In	Product of the sine of the current wander angle and the cosine of the latitude
DC_P_NN	Direction_Cosine_Matrix_Elements	In	Product of the cosine of the wander angle and the cosine of the latitude
Current_Altitude	Distances	In	Current altitude of the missile

3.3.2.3.10.6.4 LOCAL DATA

Data objects:

The following table describes the data objects maintained by this part:

Name	Type	Description
Denominator	Real	Intermediate calculation
Distance_Ratio	Real	Intermediate calculation
Partial_Denominator	Real	Intermediate calculation
Curvatures	Inverse_Distance_Vectors	Vector being calculated and returned
Two_Flat_x_DC_P_NN	Real	Intermediate calculation

3.3.2.3.10.6.5 PROCESS CONTROL

Not applicable.

3.3.2.3.10.6.6 PROCESSING

The following describes the processing performed by this part:

```
separate (Wander_Azimuth_Navigation_Parts)
function Compute_Curvatures
            (DC_P_V           : Direction_Cosine_Matrix_Elements;
             DC_P_NE          : Direction_Cosine_Matrix_Elements;
             DC_P_NN          : Direction_Cosine_Matrix_Elements;
             Current_Altitude : Distances)
            return Inverse_Distance_Vectors is

-- ----------------------------
-- --declaration of variables-
-- ----------------------------

   Denominator         : Real;
   Distance_Ratio      : Real;
```

```
   Partial_Denominator : Real;
   Curvatures          : Inverse_Distance_Vectors;
   Two_Flat_x_DC_P_NN  : Real;

------------------------
--beginning of function-
------------------------

begin

   Distance_Ratio      := Current_Altitude * One_Over_Earth_Radius;
   Partial_Denominator := 1.0 -
                          Distance_Ratio -
                          Earth_Flattening_Coefficient *
                            (DC_P_V * DC_P_V);

   Denominator     := Partial_Denominator +
                        Two_Flat * (DC_P_NE * DC_P_NE);
   Curvatures(NE) := One_Over_Earth_Radius * Denominator;

   Two_Flat_x_DC_P_NN := Two_Flat * DC_P_NN;
   Denominator        := Partial_Denominator +
                           Two_Flat_x_DC_P_NN * DC_P_NN;
   Curvatures(NN)     := One_Over_Earth_Radius * Denominator;

   Denominator    := Two_Flat_x_DC_P_NN * DC_P_NE;
   Curvatures(V) := One_Over_Earth_Radius * Denominator;

   return Curvatures;

end Compute_Curvatures;
```

3.3.2.3.10.6.7 UTILIZATION OF OTHER ELEMENTS

None.

3.3.2.3.10.6.8 LIMITATIONS

None.

3.3.2.3.10.7 TOTAL_PLATFORM_ROTATION_RATE (FUNCTION BODY) UNIT DESIGN (CATALOG #P242-0)

This function computes the rotation rate of the navigation coordinate system with respect to inertial space given the Rho and Omega vectors.

The computations performed are as follows:

TPR := Rho Vector + Omega Vector

3.3.2.3.10.7.1 REQUIREMENTS ALLOCATION

See main header for enclosing package.

3.3.2.3.10.7.2 LOCAL ENTITIES DESIGN

None.

3.3.2.3.10.7.3 INPUT/OUTPUT

GENERIC PARAMETERS:

The following generic parameters were previously defined when this part was specified.

Data types:

The following table describes the generic formal types required by this part:

Name	Type	Description
Angular_Velocity_Vectors	private	Array of angular velocity measurements

Subprograms:

The following table describes the generic formal subroutines required by this part:

Name	Type	Description
Sparse_Right_Z_ Add	function	Adds two angular velocity vectors assuming the third component of the right array equals 0

FORMAL PARAMETERS:

The following table describes this part's formal parameters:

Name	Type	Mode	Description
Rho_Vector	Angular_Velocity_Vectors	In	Contains the nominal East, North, and vertical components of the rotation rate of the navigation coordinate system with respect to the earth (vertical component assumed to equal 0 and is therefore ignored)
Omega_Vector	Angular_Velocity_Vectors	In	Contains the nominal East, North, and vertical components of the earth's rotation rate

3.3.2.3.10.7.4 LOCAL DATA

None.

3.3.2.3.10.7.5 PROCESS CONTROL

Not applicable.

3.3.2.3.10.7.6 PROCESSING

The following describes the processing performed by this part:

```
separate (Wander_Azimuth_Navigation_Parts)
function Total_Platform_Rotation_Rate
           (Rho_Vector   : Angular_Velocity_Vectors;
            Omega_Vector : Angular_Velocity_Vectors)
           return Angular_Velocity_Vectors is
begin
   return Sparse_Right_Z_Add(Left  => Omega_Vector,
                             Right => Rho_Vector);
end Total_Platform_Rotation_Rate;
```

3.3.2.3.10.7.7 UTILIZATION OF OTHER ELEMENTS

None.

3.3.2.3.10.7.8 LIMITATIONS

None.

3.3.2.3.10.8 COMPUTE_EARTH_RELATIVE_NAVIGATION_ROTATION_RATE UNIT DESIGN (CATALOG #P244-0)

This function computes the rotation rate of the navigation coordinate system with respect to the Earth given the Velocity and Radius of Curvature vectors.

The computations are performed as follows:

```
RhoNE := - (VelNE * RCV)  - (VelNN * RCNN)
RhoNN :=   (VelNE * RCNE) + (VelNN * RCV )

where VelNx = nominal x-component of the missile's current velocity
      RCx   = nominal x-component of the radius of curvature
```

3.3.2.3.10.8.1 REQUIREMENTS ALLOCATION

This part meets CAMP requirement RO25.

3.3.2.3.10.8.2 LOCAL ENTITIES DESIGN

None.

3.3.2.3.10.8.3 INPUT/OUTPUT

GENERIC PARAMETERS:

The following generic parameters were previously defined in the package specification of Wander_Azimuth_Navigation_Parts:

Data types:

The following table summarizes the generic formal types required by this part:

Name	Type	Description
Velocities	floating point type	Data type defining velocity measurements
Inverse_Distances	floating point type	Data type defining inverse distance measurements
Angular_Velocities	floating point type	Data type defining angular velocity measurements
Indices	discrete type	Data type used to dimension vector types
Angular_Velocity_Vectors	array	Array of angular velocities dimensioned by "Indices"
Velocity_Vectors	array	Array of velocities dimensioned by "Indices"
Inverse_Distance_Vectors	array	Array of inverse distances dimensioned by "Indices"

Subprograms:

The following table summarizes the generic formal subroutines required by this part:

Name	Type	Description
"/"	function	Division operator defining the operation: Velocities * Inverse_Distances => Angular_Velocities

FORMAL PARAMETERS:

The following table describes the formal parameters to the unit in this part:

Name	Type	Mode	Description
Missile_ Velocity	Velocity_Vectors	In	Contains nominal East and North components of the missile's current velocity
Curvatures	Inverse_ Distance_ Vectors	In	Contains nominal East, North, and vertical components of the inverse radius of curvature

3.3.2.3.10.8.4 LOCAL DATA

Data objects:

The following table describes the data objects maintained by this part:

Name	Type	Value	Description
Rho	Angular_ Velocity_ Vectors	N/A	Vector being calculated and returned

3.3.2.3.10.8.5 PROCESS CONTROL

Not applicable.

3.3.2.3.10.8.6 PROCESSING

The following describes the processing performed by this part:

```
separate (Wander_Azimuth_Navigation_Parts)
function Compute_Earth_Relative_Navigation_Rotation_Rate
           (Missile_Velocity    : Velocity_Vectors;
            Curvatures          : Inverse_Distance_Vectors)
           return Angular_Velocity_Vectors is

-- ---------------------------
-- --declaration of variables-
-- ---------------------------
```

```
   Rho      : Angular_Velocity_Vectors;

------------------------
--beginning of function-
------------------------

begin

   Rho(NE)  := - Missile_Velocity(NE) * Curvatures(V) -
                 Missile_Velocity(NN) * Curvatures(NN);
   Rho(NN)  := Missile_Velocity(NE) * Curvatures(NE) +
               Missile_Velocity(NN) * Curvatures(V);
   Rho(V)   := 0.0;

   return Rho;

end Compute_Earth_Relative_Navigation_Rotation_Rate;
```

3.3.2.3.10.8.7 UTILIZATION OF OTHER ELEMENTS

None.

3.3.2.3.10.8.8 LIMITATIONS

None.

3.3.2.3.10.9 COMPUTE_LATITUDE UNIT DESIGN (CATALOG #P245-0)

This function computes the current latitude given the sine of the latitude.

3.3.2.3.10.9.1 REQUIREMENTS ALLOCATION

See main header for enclosing package.

3.3.2.3.10.9.2 LOCAL ENTITIES DESIGN

None.

3.3.2.3.10.9.3 INPUT/OUTPUT

GENERIC PARAMETERS:

The following generic parameters were previously defined when this part was specified:

Data types:

The following table describes the generic formal types required by this part:

Name	Type	Description
Earth_Positions	floating point type	Data type used to define longitude and latitude measurements
Sin_Cos_Ratio	floating point type	Data type used to define results from a sine or cosine operation

Subprograms:

The following table describes the generic formal subroutines required by this part:

Name	Type	Description
Arcsin	function	Arcsine function

FORMAL PARAMETERS:

The following table describes the formal parameters for this part:

Name	Type	Mode	Description
Sin_Latitude	Sin_Cos_Ratio	In	Sine of the missile's current latitude

3.3.2.3.10.9.4 LOCAL DATA

None.

3.3.2.3.10.9.5 PROCESS CONTROL

Not applicable.

3.3.2.3.10.9.6 PROCESSING

The following describes the processing performed by this part:

```
separate (Wander_Azimuth_Navigation_Parts)
function Compute_Latitude (Sin_Latitude : Sin_Cos_Ratio)
                          return Earth_Positions is
begin
   return Arcsin(Sin_Latitude);
end Compute_Latitude;
```

3.3.2.3.10.9.7 UTILIZATION OF OTHER ELEMENTS

None.

3.3.2.3.10.9.8 LIMITATIONS

None.

3.3.2.3.10.10 COMPUTE_LATITUDE_USING_ARCTANGENT UNIT DESIGN (CATALOG #P246-0)

This function computes the current Latitude given some elements of the Direction Cosine matrix.

The computations are performed as follows:

Lat := Arctan(DCPV / SqRT(DCPNE**2 + DCPNN**2)

(see formal parameters section for explanation of abbreviations)

3.3.2.3.10.10.1 REQUIREMENTS ALLOCATION

This part meets CAMP requirement R030

3.3.2.3.10.10.2 LOCAL ENTITIES DESIGN

None.

3.3.2.3.10.10.3 INPUT/OUTPUT

GENERIC PARAMETERS:

The following generic parameters were previously defined for this part's specification in the Wander_Azimuth_Navigation_Parts package specification:

Data types:

The following table describes the generic formal types required by this part:

Name	Type	Description
Earth_Position	floating point type	Data type used to define latitude and longitude measurements
Tan_Ratio	floating point type	Data type used to define results of a tangent operation
Direction_Cosine_Matrix_Elements	floating point type	Data type used to define elements of a direction cosine matrix

Subprograms:

The following table describes the generic formal subroutines required by this part:

Name	Type	Description
"/"	function	Division operator defining the operation: Direction_Cosine_Matrix_Elements / Direction_Cosine_Matrix_Elements => Tan_Ratio
Arctan	function	Arctangent function
Sqrt	function	Square root function

FORMAL PARAMETERS:

The following table describes this part's formal parameters:

Name	Type	Mode	Description
DC_P_V	Direction_Cosine_Matrix_Elements	In	Sine of the current latitude of the missile
DC_P_NE	Direction_Cosine_Matrix_Elements	In	Product of the sine of the wander angle and the cosine of the latitude
DC_P_NN	Direction_Cosine_Matrix_Elements	In	Product of the cosine of the wander angle and the cosine of the latitude

3.3.2.3.10.10.4 LOCAL DATA

Data objects:

The following table describes the data objects maintained by this part:

Name	Type	Value	Description
Denominator	Direction_Cosine_Matrix_Elements	N/A	Temporary variable used to hold results of intermediate calculations

3.3.2.3.10.10.5 PROCESS CONTROL

Not applicable.

3.3.2.3.10.10.6 PROCESSING

The following describes the processing performed by this part:

```
separate (Wander_Azimuth_Navigation_Parts)
function Compute_Latitude_Using_Arctangent
```

```
            (DC_P_V  : Direction_Cosine_Matrix_Elements;
             DC_P_NE : Direction_Cosine_Matrix_Elements;
             DC_P_NN : Direction_Cosine_Matrix_Elements)
            return Earth_Positions is

-- ---------------------------
-- --declaration of variables-
-- ---------------------------

   Denominator : Direction_Cosine_Matrix_Elements;

------------------------
--beginning of function-
------------------------

begin

   Denominator := Sqrt( DC_P_NE * DC_P_NE +
                        DC_P_NN * DC_P_NN );

   return Arctan(DC_P_V / Denominator);

end Compute_Latitude_Using_Arctangent;
```

3.3.2.3.10.10.7 UTILIZATION OF OTHER ELEMENTS

None.

3.3.2.3.10.10.8 LIMITATIONS

None.

3.3.2.3.10.11 COMPUTE_LONGITUDE UNIT DESIGN (CATALOG #P247-0)

This function computes the current longitude given some elements of the direction cosine matrix.

3.3.2.3.10.11.1 REQUIREMENTS ALLOCATION

This part meets CAMP requirement R033

3.3.2.3.10.11.2 LOCAL ENTITIES DESIGN

None.

3.3.2.3.10.11.3 INPUT/OUTPUT

GENERIC PARAMETERS:

The following generic parameters were previously defined for this part's specification in the Wander_Azimuth_Navigation_Parts package specification:

Data types:

The following table describes the generic formal types required by this part:

Name	Type	Description
Earth_Position	floating point type	Data type used to define latitude and longitude measurements
Tan_Ratio	floating point type	Data type used to define results of a tangent operation
Direction_Cosine_Matrix_Elements	floating point type	Data type used to define elements of a direction cosine matrix

Subprograms:

The following table describes the generic formal subroutines required by this part:

Name	Type	Description
"/"	function	Division operator defining the operation: Direction_Cosine_Matrix_Elements / Direction_Cosine_Matrix_Elements => Tan_Ratio
Arctan	function	Arctangent function

FORMAL PARAMETERS:

The following table describes this part's formal parameters:

Name	Type	Mode	Description
DC_R_V	Direction_Cosine_Matrix_Elements	In	Product of the cosine of the current latitude and the sine of the current longitude
DC_G_V	Direction_Cosine_Matrix_Elements	In	Product of the cosine of the current latitude and the cosine of the current longitude

3.3.2.3.10.11.4 LOCAL DATA

None.

3.3.2.3.10.11.5 PROCESS CONTROL

Not applicable.

3.3.2.3.10.11.6 PROCESSING

The following describes the processing performed by this part:

```
separate (Wander_Azimuth_Navigation_Parts)
function Compute_Longitude (DC_R_V : Direction_Cosine_Matrix_Elements;
                            DC_G_V : Direction_Cosine_Matrix_Elements)
                           return Earth_Positions is

begin

   return Arctan(DC_R_V / DC_G_V);

end Compute_Longitude;
```

3.3.2.3.10.11.7 UTILIZATION OF OTHER ELEMENTS

None.

3.3.2.3.10.11.8 LIMITATIONS

None.

3.3.2.3.10.12 COMPUTE_WANDER_AZIMUTH_ANGLE UNIT DESIGN (CATALOG #P248-0)

This function computes the current wander azimuth angle given some elements of the Direction Cosine matrix.

The computations are performed as follows:

```
    WA := Arctan(DCPNE / DCPNN)
```

(see formal parameters section for explanation of abbreviations)

3.3.2.3.10.12.1 REQUIREMENTS ALLOCATION

This part meets CAMP requirement R028

3.3.2.3.10.12.2 LOCAL ENTITIES DESIGN

None.

3.3.2.3.10.12.3 INPUT/OUTPUT

GENERIC PARAMETERS:

The following generic parameters were previously defined for this part's specification in the Wander_Azimuth_Navigation_Parts package specification:

Data types:

The following table describes the generic formal types required by this part:

Name	Type	Description
Tan_Ratio	floating point type	Data type defining results of a tangent function
Direction_Cosine_Matrix_Elements	floating point type	Data type defining elements in a direction cosine matrix
Wander_Angles	floating point type	Data type defining wander angle measurements

Subprograms:

The following table describes the generic formal subroutines required by this part:

Name	Type	Description
"/"	function	Devision operator defining the operation: Direction_Cosine_Matrix_Elements / Direction_Cosine_Matrix_Elements => Tan_Ratio
Arctan	function	Arctangent function

FORMAL PARAMETERS:

The following table describes this part's formal parameters:

Name	Type	Mode	Description
DC_P_NE	Direction_Cosine_Matrix_Elements	In	Product of the sine of the wander angle and the cosine of the latitude
DC_P_NN	Direction_Cosine_Matrix_Elements	In	Product of the cosine of the wander angle and the cosine of the latitude

3.3.2.3.10.12.4 LOCAL DATA

None.

3.3.2.3.10.12.5 PROCESS CONTROL

Not applicable.

3.3.2.3.10.12.6 PROCESSING

The following describes the processing performed by this part:

```
separate (Wander_Azimuth_Navigation_Parts)
function Compute_Wander_Azimuth_Angle
            (DC_P_NE : Direction_Cosine_Matrix_Elements;
             DC_P_NN : Direction_Cosine_Matrix_Elements)
            return Wander_Angles is

begin

   return Arctan(DC_P_NE / DC_P_NN);

end Compute_Wander_Azimuth_Angle;
```

3.3.2.3.10.12.7 UTILIZATION OF OTHER ELEMENTS

None.

3.3.2.3.10.12.8 LIMITATIONS

None.

3.3.2.3.10.13 COMPUTE_EAST_VELOCITY_WITH_SIN_COS_IN UNIT DESIGN (CATALOG #P1099-0)

This function computes the east velocity given the nominal east and north velocities and the wander angle.

The computations are performed as follows:

```
VelE := VelNE * Cos(WA) - VelNN * Sin(WA)

where VelNE = nominal east velocity
      VelNN = nominal north velocity
      WA    = wander angle
```

3.3.2.3.10.13.1 REQUIREMENTS ALLOCATION

N/A

3.3.2.3.10.13.2 LOCAL ENTITIES DESIGN

None.

3.3.2.3.10.13.3 INPUT/OUTPUT

GENERIC PARAMETERS:

The following generic parameters were previously defined for this part's specification in the Wander_Azimuth_Navigation_Parts package specification:

Data types:

The following table describes the generic formal types required by this part:

Name	Type	Description
Sin_Cos_Ratio	floating point type	Data type of results from a sine or cosine operation
Velocities	floating point type	Data type of velocity measurements

Subprograms:

The following table describes the generic formal subroutines required by this part:

Name	Type	Description
Sin_Cos	procedure	Returns the sine and cosine of a wander angle

FORMAL PARAMETERS:

The following table describes this part's formal parameters:

Name	Type	Mode	Description
Nominal_East_Velocity	Velocities	In	Velocity which would be east if the wander angle was 0
Nominal_North_Velocity	Velocities	In	Velocity which would be north if the wander angle was 0
Sine_of_Wander_Angle	Sin_Cos_Ratio	In	Sine of the current wander azimuth angle
Cosine_of_Wander_Angle	Sin_Cos_Ratio	In	Cosine of the current wander azimuth angle

3.3.2.3.10.13.4 LOCAL DATA

Data objects:

The following table describes the data objects maintained by this part:

Name	Type	Description
Answer	Velocities	East velocity being calculated

3.3.2.3.10.13.5 PROCESS CONTROL

Not applicable.

3.3.2.3.10.13.6 PROCESSING

The following describes the processing performed by this part:

```
separate (Wander_Azimuth_Navigation_Parts)
function Compute_East_Velocity_with_Sin_Cos_In
            (Nominal_East_Velocity  : Velocities;
             Nominal_North_Velocity : Velocities;
             Sine_of_Wander_Angle   : Sin_Cos_Ratio;
             Cosine_of_Wander_Angle : Sin_Cos_Ratio) return Velocities is

-- ---------------------------
-- --declaration of variables-
-- ---------------------------

   Answer : Velocities;

--------------------------------------
--begin function Compute_East_Velocity
--------------------------------------

begin

   Answer := Nominal_East_Velocity  * Cosine_of_Wander_Angle -
             Nominal_North_Velocity * Sine_of_Wander_Angle;

   return Answer;

end Compute_East_Velocity_with_Sin_Cos_In;
```

3.3.2.3.10.13.7 UTILIZATION OF OTHER ELEMENTS

None.

3.3.2.3.10.13.8 LIMITATIONS

None.

3.3.2.3.10.14 COMPUTE_NORTH_VELOCITY_WITH_SIN_COS_IN UNIT DESIGN (CATALOG #P1101-0)

This function calculates the North velocity component of a missile given the nominal north and east velocities of the missile.

The computations are performed as follows:

```
VelN := VelNE * Sin(WA) + VelNN * Cos(WA)

where VelNE = nominal east velocity
      VelNN = nominal north velocity
      WA    = wander angle
```

3.3.2.3.10.14.1 REQUIREMENTS ALLOCATION

N/A

3.3.2.3.10.14.2 LOCAL ENTITIES DESIGN

None.

3.3.2.3.10.14.3 INPUT/OUTPUT

GENERIC PARAMETERS:

The following generic parameters were previously defined for this part's specification in the Wander_Azimuth_Navigation_Parts package specification:

Data types:

The following table describes the generic formal types required by this part:

Name	Type	Description
Sin_Cos_Ratio	floating point type	Data type defining results of a sine or cosine operation
Velocities	floating point type	Data type defining velocity measurements

Subprograms:

The following table describes the generic formal subroutines required by this part:

Name	Type	Description
"*"	function	Multiplication operator defining the operation: Velocities * Sin_Cos_Ratio => Velocities

FORMAL PARAMETERS:

The following table describes this part's formal parameters:

Name	Type	Mode	Description
Nominal_East_Velocity	Velocities	In	Velocity in the direction which would be East if the wander angle was 0
Nominal_North_Velocity	Velocities	In	Velocity in the direction which would be North if the wander angle was 0
Sine_of_Wander_Angle	Sin_Cos_Ratio	In	Sine of the current wander azimuth angle
Cosine_of_Wander_Angle	Sin_Cos_Ratio	In	Cosine of the current wander azimuth angle

3.3.2.3.10.14.4 LOCAL DATA

Data objects:

The following table describes the data objects maintained by this part:

Name	Type	Description
Answer	Velocities	North velocity being calculated

3.3.2.3.10.14.5 PROCESS CONTROL

Not applicable.

3.3.2.3.10.14.6 PROCESSING

The following describes the processing performed by this part:

```
separate (Wander_Azimuth_Navigation_Parts)
function Compute_North_Velocity_with_Sin_Cos_In
            (Nominal_East_Velocity  : Velocities;
             Nominal_North_Velocity : Velocities;
             Sine_of_Wander_Angle   : Sin_Cos_Ratio;
             Cosine_of_Wander_Angle : Sin_Cos_Ratio) return Velocities is

-- ---------------------------
-- --declaration of variables-
-- ---------------------------

   Answer : Velocities;

----------------------------------------
--begin function Compute_North_Velocity
```

```
---------------------------------------

begin

   Answer := Nominal_East_Velocity  * Sine_of_Wander_Angle +
             Nominal_North_Velocity * Cosine_of_Wander_Angle;

   return Answer;

end Compute_North_Velocity_with_Sin_Cos_In;
```

3.3.2.3.10.14.7 UTILIZATION OF OTHER ELEMENTS

None.

3.3.2.3.10.14.8 LIMITATIONS

None.

3.3.2.3.10.15 COMPUTE_EARTH_RELATIVE_HORIZONTAL_VELOCITIES_WITH_SIN_COS_IN UNIT DESIGN (CATALOG #P1107-0)

This procedure computes the north and east velocity given the nominal east and north velocities and the wander angle.

The computations are performed as follows:

```
    VelE := VelNE * Cos(WA) - VelNN * Sin(WA)
    VelN := VelNE * Sin(WA) + VelNN * Cos(WA)

    where VelNE = nominal east velocity
          VelNN = nominal north velocity
          WA    = wander angle
```

3.3.2.3.10.15.1 REQUIREMENTS ALLOCATION

N/A

3.3.2.3.10.15.2 LOCAL ENTITIES DESIGN

None.

3.3.2.3.10.15.3 INPUT/OUTPUT

GENERIC PARAMETERS:

The following generic parameters were previously defined for this part's specification in the Wander_Azimuth_Navigation_Parts package specification:

Data types:

The following table describes the generic formal types required by this part:

Name	Type	Description
Sin_Cos_Ratio	floating point type	Data type defining results of a sine or cosine operation
Velocities	floating point type	Data type defining velocity measurements

Subprograms:

The following table describes the generic formal subroutines required by this part:

Name	Type	Description
"*"	function	Multiplication operator defining the operation: Velocities * Sin_Cos_Ratio => Velocities

FORMAL PARAMETERS:

The following table describes this part's formal parameters:

Name	Type	Mode	Description
Nominal_East_Velocity	Velocities	In	Velocity in the direction that would be East if the wander angle was 0
Nominal_North_Velocity	Velocities	In	Velocity in the direct that would be North if the wander angle was 0
Current_Wander_Angle	Wander_Angles	In	Value of the current wander angle
Sine_of_Wander_Angle	Sin_Cos_Ratio	In	Sine of the current wander azimuth angle
Cosine_of_Wander_Angle	Sin_Cos_Ratio	In	Cosine of the current wander azimuth angle
East_Velocity	Velocities	Out	Current east velocity of the missile
North_Velocity	Velocities	Out	Current north velocity of the missile

3.3.2.3.10.15.4 LOCAL DATA

None.

3.3.2.3.10.15.5 PROCESS CONTROL

Not applicable.

3.3.2.3.10.15.6 PROCESSING

The following describes the processing performed by this part:

```
separate (Wander_Azimuth_Navigation_Parts)
procedure Compute_Earth_Relative_Horizontal_Velocities_with_Sin_Cos_In
             (Nominal_East_Velocity  : in     Velocities;
              Nominal_North_Velocity : in     Velocities;
              Sine_of_Wander_Angle   : in     Sin_Cos_Ratio;
              Cosine_of_Wander_Angle : in     Sin_Cos_Ratio;
              East_Velocity          :    out Velocities;
              North_Velocity         :    out Velocities) is

begin

   East_Velocity := Nominal_East_Velocity * Cosine_of_Wander_Angle -
                    Nominal_North_Velocity * Sine_of_Wander_Angle;
   North_Velocity := Nominal_East_Velocity * Sine_Of_Wander_Angle +
                     Nominal_North_Velocity * Cosine_of_Wander_Angle;

end Compute_Earth_Relative_Horizontal_Velocities_with_Sin_Cos_In;
```

3.3.2.3.10.15.7 UTILIZATION OF OTHER ELEMENTS

None.

3.3.2.3.10.15.8 LIMITATIONS

None.

3.3.2.3.10.16 COMPUTE_LATITUDE_USING_TWO_VALUE_ARCTANGENT UNIT DESIGN (CATALOG #P1104-0)

This function computes the current Latitude given some elements of the Direction Cosine matrix.

The computations are performed as follows:

```
    Lat := Arctan(DCPV / SqRT(DCPNE**2 + DCPNN**2)
```

(see formal parameters section for explanation of abbreviations)

3.3.2.3.10.16.1 REQUIREMENTS ALLOCATION

N/A

3.3.2.3.10.16.2 LOCAL ENTITIES DESIGN

None.

3.3.2.3.10.16.3 INPUT/OUTPUT

GENERIC PARAMETERS:

The following generic parameters were previously defined for this part's specification in the Wander_Azimuth_Navigation_Parts package specification:

Data types:

The following table describes the generic formal types required by this part:

Name	Type	Description
Earth_Position	floating point type	Data type used to define latitude and longitude measurements
Direction_Cosine_Matrix_Elements	floating point type	Data type used to define elements of a direction cosine matrix

Subprograms:

The following table describes the generic formal subroutines required by this part:

Name	Type	Description
Arctan2	function	4-quadrant arctangent function
Sqrt	function	Square root function

FORMAL PARAMETERS:

The following table describes this part's formal parameters:

Name	Type	Mode	Description
DC_P_V	Direction_Cosine_Matrix_Elements	In	Sine of the current latitude of the missile
DC_P_NE	Direction_Cosine_Matrix_Elements	In	Product of the sine of the wander angle and the cosine of the latitude
DC_P_NN	Direction_Cosine_Matrix_Elements	In	Product of the cosine of the wander angle and the cosine of the latitude

3.3.2.3.10.16.4 LOCAL DATA

Data objects:

The following table describes the data objects maintained by this part:

Name	Type	Description
Answer	Earth_Positions	Latitude being calculated
Denominator	Direction_Cosine_Matrix_Elements	Temporary variable used to hold results of intermediate calculations

3.3.2.3.10.16.5 PROCESS CONTROL

Not applicable.

3.3.2.3.10.16.6 PROCESSING

The following describes the processing performed by this part:

```
separate (Wander_Azimuth_Navigation_Parts)
function Compute_Latitude_Using_Two_Value_Arctangent
           (DC_P_V  : Direction_Cosine_Matrix_Elements;
            DC_P_NE : Direction_Cosine_Matrix_Elements;
            DC_P_NN : Direction_Cosine_Matrix_Elements)
           return Earth_Positions is

-- ---------------------------
-- --declaration of variables-
-- ---------------------------

   Answer      : Earth_Positions;
   Denominator : Direction_Cosine_Matrix_Elements;

------------------------
--beginning of function-
------------------------

begin

   Denominator := Sqrt( DC_P_NE * DC_P_NE +
                        DC_P_NN * DC_P_NN );

   Answer := Arctan2 (Y => DC_P_V,
                      X => Denominator);

   return Answer;

end Compute_Latitude_Using_Two_Value_Arctangent;
```

3.3.2.3.10.16.7 UTILIZATION OF OTHER ELEMENTS

None.

3.3.2.3.10.16.8 LIMITATIONS

None.

3.3.2.3.10.17 COMPUTE_LONGITUDE_USING_TWO_VALUE_ARCTANGENT UNIT DESIGN (CATALOG #P1106-0)

This function computes the current longitude given some elements of the direction cosine matrix.

3.3.2.3.10.17.1 REQUIREMENTS ALLOCATION

N/A

3.3.2.3.10.17.2 LOCAL ENTITIES DESIGN

None.

3.3.2.3.10.17.3 INPUT/OUTPUT

GENERIC PARAMETERS:

The following generic parameters were previously defined for this part's specification in the Wander_Azimuth_Navigation_Parts package specification:

Data types:

The following table describes the generic formal types required by this part:

Name	Type	Description
Earth_Position	floating point type	Data type used to define latitude and longitude measurements
Direction_Cosine_Matrix_Elements	floating point type	Data type used to define elements of a direction cosine matrix

Subprograms:

The following table describes the generic formal subroutines required by this part:

Name	Type	Description
Arctan2	function	4-quadrant arctangent function

FORMAL PARAMETERS:

The following table describes this part's formal parameters:

Name	Type	Mode	Description
DC_R_V	Direction_Cosine_Matrix_Elements	In	Product of the cosine of the current latitude and the sine of the current longitude
DC_G_V	Direction_Cosine_Matrix_Elements	In	Product of the cosine of the current latitude and the cosine of the current longitude

3.3.2.3.10.17.4 LOCAL DATA

None.

3.3.2.3.10.17.5 PROCESS CONTROL

Not applicable.

3.3.2.3.10.17.6 PROCESSING

The following describes the processing performed by this part:

```
separate (Wander_Azimuth_Navigation_Parts)
function Compute_Longitude_using_Two_Value_Arctangent
            (DC_R_V : Direction_Cosine_Matrix_Elements;
             DC_G_V : Direction_Cosine_Matrix_Elements)
            return Earth_Positions is

-- ---------------------
-- --declaration section
-- ---------------------

   Answer : Earth_Positions;

-----------------
--begin function
----------------

begin

   return Arctan2 (Y => DC_R_V,
                   X => DC_G_V);

end Compute_Longitude_using_Two_Value_Arctangent;
```

3.3.2.3.10.17.7 UTILIZATION OF OTHER ELEMENTS

None.

3.3.2.3.10.17.8 LIMITATIONS

None.

3.3.2.3.10.18 COMPUTE_WANDER_AZIMUTH_ANGLE_USING_TWO_VALUE_ARCTANGENT UNIT DESIGN (CATALOG #P1109-0)

This function computes the current wander azimuth angle given some elements of the Direction Cosine matrix.

The computations are performed as follows:

WA := Arctan(DCPNE / DCPNN)

(see formal parameters section for explanation of abbreviations)

3.3.2.3.10.18.1 REQUIREMENTS ALLOCATION

This part meets CAMP requirement R028

3.3.2.3.10.18.2 LOCAL ENTITIES DESIGN

None.

3.3.2.3.10.18.3 INPUT/OUTPUT

GENERIC PARAMETERS:

The following generic parameters were previously defined for this part's specification in the Wander_Azimuth_Navigation_Parts package specification:

Data types:

The following table describes the generic formal types required by this part:

Name	Type	Description
Direction_Cosine_Matrix_Elements	floating point type	Data type defining elements in a direction cosine matrix
Wander_Angles	floating point type	Data type defining wander angle measurements

Subprograms:

The following table describes the generic formal subroutines required by this part:

Name	Type	Description
Arctan2	function	4-quadrant Arctangent function

FORMAL PARAMETERS:

The following table describes this part's formal parameters:

Name	Type	Mode	Description
DC_P_NE	Direction_Cosine_Matrix_Elements	In	Product of the sine of the wander angle and the cosine of the latitude
DC_P_NN	Direction_Cosine_Matrix_Elements	In	Product of the cosine of the wander angle and the cosine of the latitude

3.3.2.3.10.18.4 LOCAL DATA

Data objects:

The following table describes the data objects maintained by this part:

Name	Type	Description
Answer	Wander_Angles	Wander angle being calculated

3.3.2.3.10.18.5 PROCESS CONTROL

Not applicable.

3.3.2.3.10.18.6 PROCESSING

The following describes the processing performed by this part:

```
separate (Wander_Azimuth_Navigation_Parts)
function Compute_Wander_Azimuth_Angle_using_Two_Value_Arctangent
            (DC_P_NE : Direction_Cosine_Matrix_Elements;
             DC_P_NN : Direction_Cosine_Matrix_Elements)
            return Wander_Angles is

-- ---------------------
-- --declaration section
-- ---------------------
```

```
   Answer : Wander_Angles;

----------------
--begin function
----------------

begin

   Answer := Arctan2 (Y => DC_P_NE,
                      X => DC_P_NN);

   return Answer;

end Compute_Wander_Azimuth_Angle_using_Two_Value_Arctangent;
```

3.3.2.3.10.18.7 UTILIZATION OF OTHER ELEMENTS

None.

3.3.2.3.10.18.8 LIMITATIONS

None.

(This page left intentionally blank.)

```
package body Wander_Azimuth_Navigation_Parts is

   function Compute_East_Velocity
            (Nominal_East_Velocity  : Velocities;
             Nominal_North_Velocity : Velocities;
             Current_Wander_Angle   : Wander_Angles)
            return Velocities is separate;

   function Compute_East_Velocity_With_Sin_Cos_In
            (Nominal_East_Velocity  : Velocities;
             Nominal_North_Velocity : Velocities;
             Sine_Of_Wander_Angle   : Sin_Cos_Ratio;
             Cosine_Of_Wander_Angle : Sin_Cos_Ratio)
            return Velocities is separate;

   function Compute_North_Velocity
            (Nominal_East_Velocity  : Velocities;
             Nominal_North_Velocity : Velocities;
             Current_Wander_Angle   : Wander_Angles)
            return Velocities is separate;

   function Compute_North_Velocity_With_Sin_Cos_In
            (Nominal_East_Velocity  : Velocities;
             Nominal_North_Velocity : Velocities;
             Sine_Of_Wander_Angle   : Sin_Cos_Ratio;
             Cosine_Of_Wander_Angle : Sin_Cos_Ratio)
            return Velocities is separate;

   procedure Compute_Earth_Relative_Horizontal_Velocities
             (Nominal_East_Velocity  : in  Velocities;
              Nominal_North_Velocity : in  Velocities;
              Current_Wander_Angle   : in  Wander_Angles;
              East_Velocity          : out Velocities;
              North_Velocity         : out Velocities)
             is separate;

   procedure Compute_Earth_Relative_Horizontal_Velocities_With_Sin_Cos_In
             (Nominal_East_Velocity  : in     Velocities;
              Nominal_North_Velocity : in     Velocities;
              Sine_Of_Wander_Angle   : in     Sin_Cos_Ratio;
              Cosine_Of_Wander_Angle : in     Sin_Cos_Ratio;
              East_Velocity          :    out Velocities;
              North_Velocity         :    out Velocities)
             is separate;

   function Compute_Total_Angular_Velocity
            (Rho_East  : Angular_Velocities;
             Rho_North : Angular_Velocities)
            return Angular_Velocities is separate;

   function Compute_Coriolis_Acceleration
            (Velocity_In : Velocity_Vectors;
             Rho_In      : Angular_Velocity_Vectors;
             Omega_In    : Angular_Velocity_Vectors)
            return Acceleration_Vectors is separate;

   package body Coriolis_Acceleration_From_Total_Rates is separate;
```

```
   function Compute_Curvatures
               (Dc_P_V           : Direction_Cosine_Matrix_Elements;
                Dc_P_Ne          : Direction_Cosine_Matrix_Elements;
                Dc_P_Nn          : Direction_Cosine_Matrix_Elements;
                Current_Altitude : Distances)
               return Inverse_Distance_Vectors is separate;

   function Total_Platform_Rotation_Rate
               (Rho_Vector   : Angular_Velocity_Vectors;
                Omega_Vector : Angular_Velocity_Vectors)
               return Angular_Velocity_Vectors is separate;

   package body Earth_Rotation_Rate is separate;

   function Compute_Earth_Relative_Navigation_Rotation_Rate
               (Missile_Velocity    : Velocity_Vectors;
                Curvatures          : Inverse_Distance_Vectors)
               return Angular_Velocity_Vectors is separate;

   function Compute_Latitude (Sin_Latitude : Sin_Cos_Ratio)
                             return Earth_Positions is separate;

   function Compute_Latitude_Using_Arctangent
               (Dc_P_V  : Direction_Cosine_Matrix_Elements;
                Dc_P_Ne : Direction_Cosine_Matrix_Elements;
                Dc_P_Nn : Direction_Cosine_Matrix_Elements)
               return Earth_Positions is separate;

   function Compute_Latitude_Using_Two_Value_Arctangent
               (Dc_P_V  : Direction_Cosine_Matrix_Elements;
                Dc_P_Ne : Direction_Cosine_Matrix_Elements;
                Dc_P_Nn : Direction_Cosine_Matrix_Elements)
               return Earth_Positions is separate;

   function Compute_Longitude
               (Dc_R_V : Direction_Cosine_Matrix_Elements;
                Dc_G_V : Direction_Cosine_Matrix_Elements)
               return Earth_Positions is separate;

   function Compute_Longitude_Using_Two_Value_Arctangent
               (Dc_R_V : Direction_Cosine_Matrix_Elements;
                Dc_G_V : Direction_Cosine_Matrix_Elements)
               return Earth_Positions is separate;

   function Compute_Wander_Azimuth_Angle
               (Dc_P_Ne : Direction_Cosine_Matrix_Elements;
                Dc_P_Nn : Direction_Cosine_Matrix_Elements)
               return Wander_Angles is separate;

   function Compute_Wander_Azimuth_Angle_Using_Two_Value_Arctangent
               (Dc_P_Ne : Direction_Cosine_Matrix_Elements;
                Dc_P_Nn : Direction_Cosine_Matrix_Elements)
               return Wander_Angles is separate;

end Wander_Azimuth_Navigation_Parts;
```

```
separate (Wander_Azimuth_Navigation_Parts)
function Compute_East_Velocity
            (Nominal_East_Velocity  : Velocities;
             Nominal_North_Velocity : Velocities;
             Current_Wander_Angle   : Wander_Angles) return Velocities is

-- ---------------------------
-- --declaration of variables-
-- ---------------------------

   Cos_Of_Angle  : Sin_Cos_Ratio;
   Sin_Of_Angle  : Sin_Cos_Ratio;

---------------------------------------
--begin function Compute_East_Velocity
---------------------------------------

begin

   Sin_Cos(Current_Wander_Angle, Sin_Of_Angle, Cos_Of_Angle);

   return ( Nominal_East_Velocity * Cos_Of_Angle -
            Nominal_North_Velocity * Sin_Of_Angle);

end Compute_East_Velocity;
```

```
separate (Wander_Azimuth_Navigation_Parts)
function Compute_North_Velocity
            (Nominal_East_Velocity  : Velocities;
             Nominal_North_Velocity : Velocities;
             Current_Wander_Angle   : Wander_Angles) return Velocities is

-- ---------------------------
-- --declaration of variables-
-- ---------------------------

   Cos_Of_Angle   : Sin_Cos_Ratio;
   Sin_Of_Angle   : Sin_Cos_Ratio;

-------------- ---------------------------
--begin function Compute_North_Velocity
---------------------------------------

begin

   Sin_Cos(Current_Wander_Angle, Sin_Of_Angle, Cos_Of_Angle);

   return ( Nominal_East_Velocity * Sin_Of_Angle +
            Nominal_North_Velocity * Cos_Of_Angle);

end Compute_North_Velocity;
```

```
separate (Wander_Azimuth_Navigation_Parts)
procedure Compute_Earth_Relative_Horizontal_Velocities
            (Nominal_East_Velocity  : in  Velocities;
             Nominal_North_Velocity : in  Velocities;
             Current_Wander_Angle   : in  Wander_Angles;
             East_Velocity          : out Velocities;
             North_Velocity         : out Velocities) is

-- ---------------------------
-- --declaration of variables-
-- ---------------------------

   Cos_Of_Angle : Sin_Cos_Ratio;
   Sin_Of_Angle : Sin_Cos_Ratio;

------------------------------------------------------------------
--begin procedure Compute_Earth_Relative_Horizontal_Velocities
------------------------------------------------------------------

begin

   Sin_Cos(Current_Wander_Angle, Sin_Of_Angle, Cos_Of_Angle);

   East_Velocity := Nominal_East_Velocity * Cos_Of_Angle -
                    Nominal_North_Velocity * Sin_Of_Angle;
   North_Velocity := Nominal_East_Velocity * Sin_Of_Angle +
                     Nominal_North_Velocity * Cos_Of_Angle;

end Compute_Earth_Relative_Horizontal_Velocities;
```

```
separate (Wander_Azimuth_Navigation_Parts)
function Compute_Total_Angular_Velocity
            (Rho_East  : Angular_Velocities;
             Rho_North : Angular_Velocities) return Angular_Velocities is

begin

   return Sqrt(Rho_East * Rho_East + Rho_North * Rho_North);

end Compute_Total_Angular_Velocity;
```

```
separate (Wander_Azimuth_Navigation_Parts)
function Compute_Coriolis_Acceleration
            (Velocity_In : Velocity_Vectors;
             Rho_In      : Angular_Velocity_Vectors;
             Omega_In    : Angular_Velocity_Vectors)
            return Acceleration_Vectors is

-- ---------------------------
-- --declaration of variables-
-- ---------------------------

   Temp_A_V_V      : Angular_Velocity_Vectors;

-----------------------------------------------
--begin function Compute_Coriolis_Acceleration
-----------------------------------------------

begin

   Temp_A_V_V         := Omega_In + Omega_In;
   Temp_A_V_V         := Sparse_Right_Z_Add(Temp_A_V_V, Rho_In);

   return Cross_Product(Temp_A_V_V, Velocity_In);

end Compute_Coriolis_Acceleration;
```

```
separate (Wander_Azimuth_Navigation_Parts)
package body Coriolis_Acceleration_From_Total_Rates is

-- ---------------------------
-- --declaration of variables-
-- ---------------------------

   E               : constant Indice  := Indices'FIRST;
   N               : constant Indices := Indices'SUCC(E);
   U               : constant Indices := Indices'LAST;

pragma PAGE;
-- ------------
-- --unit body-
-- ------------

   function Compute (Velocity_In : Velocity_Vectors;
                     Omega       : Angular_Velocity_Vectors;
                     Total_Rates : Angular_Velocity_Vectors)
                    return Acceleration_Vectors is

--    ---------------------------
--    --declaration of variables-
--    ---------------------------

      Temp_A_V_V      : Angular_Velocity_Vectors;

-- -------------------------
-- --begin function Compute
-- -------------------------

   begin

      Temp_A_V_V(E) := Omega(E) + Total_Rates(E);
      Temp_A_V_V(N) := Omega(N) + Total_Rates(N);
      Temp_A_V_V(U) := Omega(U) + Omega(U);

      return Cross_Product(Temp_A_V_V, Velocity_In);

   end Compute;

end Coriolis_Acceleration_From_Total_Rates;
```

```
separate (Wander_Azimuth_Navigation_Parts)
function Compute_Curvatures
            (Dc_P_V           : Direction_Cosine_Matrix_Elements;
             Dc_P_Ne          : Direction_Cosine_Matrix_Elements;
             Dc_P_Nn          : Direction_Cosine_Matrix_Elements;
             Current_Altitude : Distances)
            return Inverse_Distance_Vectors is

-- ---------------------------
-- --declaration of variables-
-- ---------------------------

   Denominator         : Real,
   Distance_Ratio      : Real;
   Partial_Denominator : Real;
   Curvatures          : Inverse_Distance_Vectors;
   Two_Flat_X_Dc_P_Nn  : Real;

-------------------------
--beginning of function-
-------------------------

begin

   Distance_Ratio      := Current_Altitude * One_Over_Earth_Radius;
   Partial_Denominator := 1.0 -
                          Distance_Ratio -
                          Earth_Flattening_Coefficient *
                            (Dc_P_V * Dc_P_V);

   Denominator    := Partial_Denominator +
                         Two_Flat * (Dc_P_Ne * Dc_P_Ne);
   Curvatures(Ne) := One_Over_Earth_Radius * Denominator;

   Two_Flat_X_Dc_P_Nn := Two_Flat * Dc_P_Nn;
   Denominator        := Partial_Denominator +
                           Two_Flat_X_Dc_P_Nn * Dc_P_Nn;
   Curvatures(Nn)     := One_Over_Earth_Radius * Denominator;

   Denominator   := Two_Flat_X_Dc_P_Nn * Dc_P_Ne;
   Curvatures(V) := One_Over_Earth_Radius * Denominator;

   return Curvatures;

end Compute_Curvatures;
```

```
separate (Wander_Azimuth_Navigation_Parts)
function Total_Platform_Rotation_Rate
            (Rho_Vector   : Angular_Velocity_Vectors;
             Omega_Vector : Angular_Velocity_Vectors)
            return Angular_Velocity_Vectors is
begin
   return Sparse_Right_Z_Add(Left  => Omega_Vector,
                             Right => Rho_Vector);
end Total_Platform_Rotation_Rate;
```

```
separate (Wander_Azimuth_Navigation_Parts)
package body Earth_Rotation_Rate is

-- --------------------
-- --local declarations
-- --------------------

   Ne             : constant Indices := Indices'FIRST;
   Nn             : constant Indices := Indices'SUCC(Ne);
   V              : constant Indices := Indices'LAST;

pragma PAGE;
-- ------------
-- --unit body-
-- ------------

   function Compute (Dc_P_V  : Direction_Cosine_Matrix_Elements;
                     Dc_P_Ne : Direction_Cosine_Matrix_Elements;
                     Dc_P_Nn : Direction_Cosine_Matrix_Elements)
                    return Angular_Velocity_Vectors is

--    ----------------------------
--    --declaration of variables-
--    ----------------------------

      Rotation_Rate : Angular_Velocity_Vectors;

-- ------------------------
-- --beginning of function-
-- ------------------------

   begin

      Rotation_Rate(Ne) := Dc_P_Ne * Earth_Rate;
      Rotation_Rate(Nn) := Dc_P_Nn * Earth_Rate;
      Rotation_Rate(V)  := Dc_P_V * Earth_Rate;

      return Rotation_Rate;

   end Compute;

end Earth_Rotation_Rate;
```

```
separate (Wander_Azimuth_Navigation_Parts)
function Compute_Earth_Relative_Navigation_Rotation_Rate
           (Missile_Velocity    : Velocity_Vectors;
            Curvatures          : Inverse_Distance_Vectors)
           return Angular_Velocity_Vectors is

-- ---------------------------
-- --declaration of variables-
-- ---------------------------

   Rho      : Angular_Velocity_Vectors;

------------------------
--beginning of function-
------------------------

begin

   Rho(Ne)  := - Missile_Velocity(Ne) * Curvatures(V) -
                 Missile_Velocity(Nn) * Curvatures(Nn);
   Rho(Nn)  := Missile_Velocity(Ne) * Curvatures(Ne) +
               Missile_Velocity(Nn) * Curvatures(V);
   Rho(V)   := 0.0;

   return Rho;

end Compute_Earth_Relative_Navigation_Rotation_Rate;
```

```
separate (Wander_Azimuth_Navigation_Parts)
function Compute_Latitude (Sin_Latitude : Sin_Cos_Ratio)
                          return Earth_Positions is
begin
   return Arcsin(Sin_Latitude);
end Compute_Latitude;
```

```
separate (Wander_Azimuth_Navigation_Parts)
function Compute_Latitude_Using_Arctangent
            (Dc_P_V  : Direction_Cosine_Matrix_Elements;
             Dc_P_Ne : Direction_Cosine_Matrix_Elements;
             Dc_P_Nn : Direction_Cosine_Matrix_Elements)
            return Earth_Positions is

-- ----------------------------
-- --declaration of variables-
-- ----------------------------

   Denominator : Direction_Cosine_Matrix_Elements;

-------------------------
--beginning of function-
-------------------------

begin

   Denominator := Sqrt( Dc_P_Ne * Dc_P_Ne +
                        Dc_P_Nn * Dc_P_Nn );

   return Arctan(Dc_P_V / Denominator);

end Compute_Latitude_Using_Arctangent;
```

```
separate (Wander_Azimuth_Navigation_Parts)
function Compute_Longitude (Dc_R_V : Direction_Cosine_Matrix_Elements;
                            Dc_G_V : Direction_Cosine_Matrix_Elements)
                           return Earth_Positions is

begin

   return Arctan(Dc_R_V / Dc_G_V);

end Compute_Longitude;
```

```
separate (Wander_Azimuth_Navigation_Parts)
function Compute_Wander_Azimuth_Angle
            (Dc_P_Ne : Direction_Cosine_Matrix_Elements;
             Dc_P_Nn : Direction_Cosine_Matrix_Elements)
            return Wander_Angles is

begin

   return Arctan(Dc_P_Ne / Dc_P_Nn);

end Compute_Wander_Azimuth_Angle;
```

```
separate (Wander_Azimuth_Navigation_Parts)
function Compute_East_Velocity_With_Sin_Cos_In
            (Nominal_East_Velocity  : Velocities;
             Nominal_North_Velocity : Velocities;
             Sine_Of_Wander_Angle   : Sin_Cos_Ratio;
             Cosine_Of_Wander_Angle : Sin_Cos_Ratio) return Velocities is

-- ----------------------------
-- --declaration of variables-
-- ----------------------------

   Answer : Velocities;

---------------------------------------
--begin function Compute_East_Velocity
---------------------------------------

begin

   Answer := Nominal_East_Velocity  * Cosine_Of_Wander_Angle -
             Nominal_North_Velocity * Sine_Of_Wander_Angle;

   return Answer;

end Compute_East_Velocity_With_Sin_Cos_In;
```

```
separate (Wander_Azimuth_Navigation_Parts)
function Compute_North_Velocity_With_Sin_Cos_In
            (Nominal_East_Velocity  : Velocities;
             Nominal_North_Velocity : Velocities;
             Sine_Of_Wander_Angle   : Sin_Cos_Ratio;
             Cosine_Of_Wander_Angle : Sin_Cos_Ratio) return Velocities is

-- ---------------------------
-- --declaration of variables-
-- ---------------------------

   Answer : Velocities;

---------------------------------------
--begin function Compute_North_Velocity
---------------------------------------

begin

   Answer := Nominal_East_Velocity  * Sine_Of_Wander_Angle +
             Nominal_North_Velocity * Cosine_Of_Wander_Angle;

   return Answer;

end Compute_North_Velocity_With_Sin_Cos_In;
```

```
separate (Wander_Azimuth_Navigation_Parts)
procedure Compute_Earth_Relative_Horizontal_Velocities_With_Sin_Cos_In
            (Nominal_East_Velocity  : in     Velocities;
             Nominal_North_Velocity : in     Velocities;
             Sine_Of_Wander_Angle   : in     Sin_Cos_Ratio;
             Cosine_Of_Wander_Angle : in     Sin_Cos_Ratio;
             East_Velocity          :    out Velocities;
             North_Velocity         :    out Velocities) is

begin

   East_Velocity := Nominal_East_Velocity * Cosine_Of_Wander_Angle -
                    Nominal_North_Velocity * Sine_Of_Wander_Angle;
   North_Velocity := Nominal_East_Velocity * Sine_Of_Wander_Angle +
                     Nominal_North_Velocity * Cosine_Of_Wander_Angle;

end Compute_Earth_Relative_Horizontal_Velocities_With_Sin_Cos_In;
```

```
separate (Wander_Azimuth_Navigation_Parts)
function Compute_Latitude_Using_Two_Value_Arctangent
            (Dc_P_V  : Direction_Cosine_Matrix_Elements;
             Dc_P_Ne : Direction_Cosine_Matrix_Elements;
             Dc_P_Nn : Direction_Cosine_Matrix_Elements)
            return Earth_Positions is

-- ---------------------------
-- --declaration of variables-
-- ---------------------------

   Answer      : Earth_Positions;
   Denominator : Direction_Cosine_Matrix_Elements;

------------------------
--beginning of function-
------------------------

begin

   Denominator := Sqrt( Dc_P_Ne * Dc_P_Ne +
                        Dc_P_Nn * Dc_P_Nn );

   Answer := Arctan2 (Y => Dc_P_V,
                      X => Denominator);

   return Answer;

end Compute_Latitude_Using_Two_Value_Arctangent;
```

```
separate (Wander_Azimuth_Navigation_Parts)
function Compute_Longitude_Using_Two_Value_Arctangent
            (Dc_R_V : Direction_Cosine_Matrix_Elements;
             Dc_G_V : Direction_Cosine_Matrix_Elements)
            return Earth_Positions is

-- ---------------------
-- --declaration section
-- ---------------------

   Answer : Earth_Positions;

-----------------
--begin function
-----------------

begin

   return Arctan2 (Y => Dc_R_V,
                   X => Dc_G_V);

end Compute_Longitude_Using_Two_Value_Arctangent;
```

```
separate (Wander_Azimuth_Navigation_Parts)
function Compute_Wander_Azimuth_Angle_Using_Two_Value_Arctangent
            (Dc_P_Ne : Direction_Cosine_Matrix_Elements;
             Dc_P_Nn : Direction_Cosine_Matrix_Elements)
            return Wander_Angles is

-- ---------------------
-- --declaration section
-- ---------------------

   Answer : Wander_Angles;

----------------
--begin function
----------------

begin

   Answer := Arctan2 (Y => Dc_P_Ne,
                      X => Dc_P_Nn);

   return Answer;

end Compute_Wander_Azimuth_Angle_Using_Two_Value_Arctangent;
```

3.3.2.4 DIRECTION_COSINE_MATRIX_OPERATIONS (PACKAGE BODY) TLCSC P644 (CATALOG #P286-0)

This part is the body of an Ada package that contains units for all CAMP parts that initialize or update a direction cosine matrix (DCM). A direction cosine matrix is a 3x3 matrix whose elements are cosines of the angles between two coordinate frames - xyz and XYZ. The direction cosines are used to transform data values represented in one coordinate frame to another coordinate frame.

This part consists of two packages. The first package contains all parts which operate on a "general" direction cosine matrix. A "general" DCM is a DCM that does not assume particular coordinate frames, i.e. the Navigation to Earth. However, these parts do assume a particular coordinate frame axes relationship (for an example, see the preceding paragraph). The user should refer to the documentation for each part to determine if the unit can be used for a particular application.

The second package contains all operations that operate on a CNE matrix. The CNE matrix is a direction cosine matrix that is used to transform objects from the navigation frame to the earth frame. The navigation frame assumes "East, North, and Up" (for XYZ), and the earth frame assumes "Greenwich, Right, and Polar" (for xyz) for the axes definitions. These coordinate assumptions are reflected in the generic objects of the package. This package exports a CNE matrix type representing a CNE matrix derived from these coordinate assumptions.

The decomposition for this part is the same as that shown in the Top-Level Design Document.

3.3.2.4.1 REQUIREMENTS ALLOCATION

The following chart summarizes the allocation of CAMP requirements to this part:

Name	Requirements Allocation
DCM_General_Operations	R34
CNE_Operations	R32

3.3.2.4.2 LOCAL ENTITIES DESIGN

None.

3.3.2.4.3 INPUT/OUTPUT

None.

3.3.2.4.4 LOCAL DATA

None.

3.3.2.4.5 PROCESS CONTROL

Not applicable.

3.3.2.4.6 PROCESSING

The following describes the processing performed by this part:

```
package body Direction_Cosine_Matrix_Operations is
   package body DCM_General_Operations is separate;
   package body CNE_Operations is separate;
end Direction_Cosine_Matrix_Operations;
```

3.3.2.4.7 UTILIZATION OF OTHER ELEMENTS

None.

3.3.2.4.8 LIMITATIONS

None.

3.3.2.4.9 LLCSC DESIGN

3.3.2.4.9.1 DCM_GENERAL_OPERATIONS PACKAGE DESIGN (CATALOG #P287-0)

This package contains procedures and functions which operate on a "general" direction cosine matrix. A specific coordinate frame axis ordering has been assumed for the parts; therefore, refer to the documentation for each part before using it.

The decomposition for this part is the same as that shown in the Top-Level Design Document.

3.3.2.4.9.1.1 REQUIREMENTS ALLOCATION

The following table summarizes the allocation of CAMP requirements to this part:

Name	Requirements Allocation
DCM_Initialized_From_Reference	N/A
Perform_Trapezoidal_Integration_of_DCM	R34
Perform_Rectangular_Integration_of_DCM	N/A
Reorthonormalize_DCM	N/A
Frame_Misalignment	N/A
Aligned_DCM_Matrix	N/A
Compute_First_Row_From_Orthonormal	N/A
DCM_From_Quaternion	N/A

3.3.2.4.9.1.2 LOCAL ENTITIES DESIGN

None.

3.3.2.4.9.1.3 INPUT/OUTPUT

None.

3.3.2.4.9.1.4 LOCAL DATA

None.

3.3.2.4.9.1.5 PROCESS CONTROL

Not applicable.

3.3.2.4.9.1.6 PROCESSING

The following describes the processing performed by this part:

```
separate (Direction_Cosine_Matrix_Operations)
package body DCM_General_Operations is
```

```
   function DCM_Initialized_From_Reference
               (Ref_DCM_2_1    : Sin_Cos_Ratio;
                Ref_DCM_2_2    : Sin_Cos_Ratio;
                Ref_DCM_3_1    : Sin_Cos_Ratio;
                Ref_DCM_3_2    : Sin_Cos_Ratio;
                Sign_Of_2_3    : INTEGER;
                Sign_Of_3_3    : INTEGER)
               return Direction_Cosine_Matrix is separate;

   package body DCM_Trapezoidal_Integration is separate;

   procedure Perform_Rectangular_Integration_of_DCM
               (DC_Matrix  : in out Direction_Cosine_Matrix;
                X_Rho      : in     Angular_Velocities;
                Y_Rho      : in     Angular_Velocities;
                Delta_Time : in     Time_Intervals) is separate;

   procedure Reorthonormalize_DCM
               (DCM_Matrix  : in out Direction_Cosine_Matrix) is separate;

   function Frame_Misalignment
               (DCM_Matrix     : Direction_Cosine_Matrix;
                Ref_DC_Matrix  : Direction_Cosine_Matrix)
               return Rotation_Angle_Vec is separate;

   function Aligned_DCM_Matrix
               (DCM_Matrix      : Direction_Cosine_Matrix;
                Rotation_Angle  : Rotation_Angle_Vec)
               return Direction_Cosine_Matrix is separate;

   procedure Compute_First_Row_From_Orthonormal
               (DCM_Matrix      : in out Direction_Cosine_Matrix) is separate;

   function DCM_From_Quaternion
               (Quaternion : Quaternion_Vectors)
               return Direction_Cosine_Matrix is separate;

end DCM_General_Operations;
```

3.3.2.4.9.1.7 UTILIZATION OF OTHER ELEMENTS

None.

3.3.2.4.9.1.8 LIMITATIONS

None.

3.3.2.4.9.1.9 LLCSC DESIGN

3.3.2.4.9.1.9.1 DCM_TRAPEZOIDAL_INTEGRATION PACKAGE DESIGN (CATALOG #P289-0)

This package contains a procedure that integrates a direction cosine matrix using trapezoidal integration. NOTE: Only rows 2 and 3 are integrated by this part. Row 1 is usually updated from rows 2 and 3 - the user must update row 1 manually.

The decomposition for this part is the same as that shown in the Top-Level Design Document.

3.3.2.4.9.1.9.1.1 REQUIREMENTS ALLOCATION

The following table summarizes the allocation of CAMP requirements to this part:

Name	Requirements Allocation
Perform_Trapezoidal_Integration_of_DCM	R34

3.3.2.4.9.1.9.1.2 LOCAL ENTITIES DESIGN

None.

3.3.2.4.9.1.9.1.3 INPUT/OUTPUT

GENERIC PARAMETERS:

Data types:

The following table describes the generic formal types required by this part:

Name	Type	Description
Angular_Velocities	floating point type	This type translates into the type of the angular velocity used to integrate the direction cosine matrix.
Time_Intervals	floating point type	This type translates into the type of the time parameter used to integrate the direction cosine matrix.
Row_Indices	discrete type	Data type which translates into the row indexing of the direction cosine matrix.
Col_Indices	discrete type	Data type which translates into the column indexing of the direction cosine matrix.
Sin_Cos_Ratio	floating point	Type which translates into the type of the elements of the direction cosine matrix.
Direction_Cosine_Matrix	3X3 matrix	Type representing a direction cosine matrix indexed by the column and row types described above with elements of type Sin_Cos_Ratio.

Data objects:

The following table describes the generic formal objects required by this part:

Name	Type	Value	Description
Initial_X_Rho	Angular_Velocities	0 or user-defined	Object used to set the initial angular velocity X_Rho to either 0 (by default) or to a user-specified value. Note that this value is set at compile time. To set the value at run time, use the reinitialize procedure.
Initial_Y_Rho	Angular_Velocities	0 or user-defined	Object used to set the initial angular velocity Y_Rho to either 0 (by default) or to a user-specified value. Note that this value is set at compile time. To set the value at run time, use the reinitialize procedure.
Row1	Row_Indices	'FIRST	Object to be used as an index representing the first row of the direction cosine matrix. This object indexes the direction cosines of the angles between the x-axis and the XYZ axes.
Row2	Row_Indices	SUCC 'FIRST	Object to be used as an index representing the second row of the direction cosine matrix. This object indexes the direction cosines of the angles between the y-axis and the XYZ axes.
Row3	Row_Indices	'LAST	Object to be used as an index representing the third row of the direction cosine matrix. This object indexes the direction cosines of the angles between the z-axis and the XYZ axes.
Col1	Col_Indices	'FIRST	Object to be used as an index representing the first column of the direction cosine matrix. This object indexes the direction cosines of the angles between the X-axis and the xyz axes.
Col2	Col_Indices	SUCC 'FIRST	Object to be used as an index representing the second column of the direction cosine matrix. This object indexes the direction cosines of the angles between the Y-axis and the xyz axes.
Col3	Col_Indices	'LAST	Object to be used as an index representing the third column of the direction cosine matrix. This object indexes the direction cosines of the angles between the Z-axis and the xyz axes.

Subprograms:

The following table describes the generic formal subroutines required by this part:

Name	Type	Description
"*"	function	This function multiplies an object of type Angular_Velocities to an object of type Sin_Cos_Ratio returning an object of type Angular_Velocities.
"*"	function	This function multiplies an object of type Angular_Velocities to an object of type Time_Intervals returning an object of type Sin_Cos_Ratio.

3.3.2.4.9.1.9.1.4 LOCAL DATA

Data objects:

The following table describes the data objects created and used by this part:

Name	Type	Description
Prev_X_Rho	Angular_Velocities	This value is the previous X angular velocity used as the lower bound of the integration.
Prev_Y_Rho	Angular_Velocities	This value is the previous Y angular velocity used as the lower bound of the integration.

Exceptions:

3.3.2.4.9.1.9.1.5 PROCESS CONTROL

Not applicable.

3.3.2.4.9.1.9.1.6 PROCESSING

The following describes the processing performed by this part:

```
separate (Direction_Cosine_Matrix_Operations.DCM_General_Operations)
package body DCM_Trapezoidal_Integration is

   Prev_X_Rho     : Angular_Velocities := Initial_X_Rho;
   Prev_Y_Rho     : Angular_Velocities := Initial_Y_Rho;

   procedure Reinitialize_Angular_Velocities
                  (X_Rho          : in     Angular_Velocities;
                   Y_Rho          : in     Angular_Velocities) is separate;
```

```
   procedure Perform_Trapezoidal_Integration_of_DCM
               (DC_Matrix       : in out Direction_Cosine_Matrix;
                X_Rho           : in     Angular_Velocities;
                Y_Rho           : in     Angular_Velocities;
                Delta_Time      : in     Time_Intervals) is separate;

end DCM_Trapezoidal_Integration;
```

3.3.2.4.9.1.9.1.7 UTILIZATION OF OTHER ELEMENTS

None.

3.3.2.4.9.1.9.1.8 LIMITATIONS

None.

3.3.2.4.9.1.9.1.9 LLCSC DESIGN

None.

3.3.2.4.9.1.9.1.10 UNIT DESIGN

3.3.2.4.9.1.9.1.10.1 REINITIALIZE_ANGULAR_VELOCITIES UNIT DESIGN

This unit reinitalizes the angular velocities which are used as the previous angular velocities in Trapezoidal Integration.

3.3.2.4.9.1.9.1.10.1.1 REQUIREMENTS ALLOCATION

N/A

3.3.2.4.9.1.9.1.10.1.2 LOCAL ENTITIES DESIGN

None.

3.3.2.4.9.1.9.1.10.1.3 INPUT/OUTPUT

GENERIC PARAMETERS:

Data types:

The following table summarizes the generic formal types required by this part:

Name	Type	Description
Angular_Velocities	floating point type	This type translates into the type of the angular velocity used to integrate the direction cosine matrix.

FORMAL PARAMETERS:

The following table describes this part's formal parameters:

Name	Type	Mode	Description
X_Rho	Angular_Velocities	In	This object is the X angular velocity that will be used to reinitialize the previous X angular velocity.
Y_Rho	Angular_Velocities	In	This object is the Y angular velocity that will be used to reinitialize the previous Y angular velocity.

3.3.2.4.9.1.9.1.10.1.4 LOCAL DATA

None.

3.3.2.4.9.1.9.1.10.1.5 PROCESS CONTROL

Not applicable.

3.3.2.4.9.1.9.1.10.1.6 PROCESSING

The following describes the processing performed by this part:

```
separate (Direction_Cosine_Matrix_Operations.DCM_General_Operations.
          DCM_Trapezoidal_Integration)
procedure Reinitialize_Angular_Velocities
             (X_Rho          : in     Angular_Velocities;
              Y_Rho          : in     Angular_Velocities) is

begin

   Prev_X_Rho := X_Rho;
   Prev_Y_Rho := Y_Rho;

end Reinitialize_Angular_Velocities;
```

3.3.2.4.9.1.9.1.10.1.7 UTILIZATION OF OTHER ELEMENTS

UTILIZATION OF EXTERNAL ELEMENTS:

UTILIZATION OF OTHER ELEMENTS IN TOP LEVEL COMPONENT:

The following tables describe the elements used by this part but defined elsewhere in the parent top level component:

Data objects:

The following table summarizes the objects required by this part and defined elsewhere in the parent top level component:

Name	Type	Source	Description
Prev_X_Rho	Angular_Velocities	Direction_Cosine_Matrix_Operations.DCM_General_Operations.DCM_Trapezoidal_Integration	This object is the angular velocity in the X direction that is used as the previous angular velocity for Trapezoidal integration.
Prev_Y_Rho	Angular_Velocities	Direction_Cosine_Matrix_Operations.DCM_General_Operations.DCM_Trapezoidal_Integration	This object is the angular velocity in the Y direction that is used as the previous angular velocity for Trapezoidal integration.

3.3.2.4.9.1.9.1.10.1.8 LIMITATIONS

None.

3.3.2.4.9.1.9.1.10.2 PERFORM_TRAPEZOIDAL_INTEGRATION_OF_DCM UNIT DESIGN

This part integrates a direction cosine matrix using trapezoidal integration. Only the 2nd and 3rd rows are integrated. The first row must be computed by another function.

3.3.2.4.9.1.9.1.10.2.1 REQUIREMENTS ALLOCATION

N/A

3.3.2.4.9.1.9.1.10.2.2 LOCAL ENTITIES DESIGN

None.

3.3.2.4.9.1.9.1.10.2.3 INPUT/OUTPUT

GENERIC PARAMETERS:

Data types:

The following table summarizes the generic formal types required by this part:

Name	Type	Description
Angular_Velocities	floating point type	This type translates into the type of the angular velocity used to integrate the direction cosine matrix.
Time_Intervals	floating point type	This type translates into the type of the time parameter used to integrate the direction cosine matrix.
Row_Indices	discrete type	Data type which translates into the row indexing of the direction cosine matrix.
Col_Indices	discrete type	Data type which translates into the column indexing of the direction cosine matrix.
Sin_Cos_Ratio	floating point	Type which translates into the type of the elements of the direction cosine matrix.
Direction_Cosine_Matrix	3X3 matrix	Type representing a direction cosine matrix indexed by the column and row types described above with elements of type Sin_Cos_Ratio.

Data objects:

The following table summarizes the generic formal objects required by this part:

Name	Type	Value	Description
Initial_X_Rho	Angular_Velocities	0 or user-defined	Object used to set the initial angular velocity X_Rho to either 0 (by default) or to a user-specified value. Note that this value is set at compile time. To set the value at run time, use the reinitialize procedure.
Initial_Y_Rho	Angular_Velocities	0 or user-defined	Object used to set the initial angular velocity Y_Rho to either 0 (by default) or to a user-specified value. Note that this value is set at compile time. To set the value at run time, use the reinitialize procedure.
Row1	Row_Indices	'FIRST	Object to be used as an index representing the first row of the direction cosine matrix. This object indexes the direction cosines of the angles between the x-axis and the XYZ axes.
Row2	Row_Indices	SUCC 'FIRST	Object to be used as an index representing the second row of the direction cosine matrix. This object indexes the direction cosines of the angles between the y-axis and the XYZ axes.
Row3	Row_Indices	'LAST	Object to be used as an index representing the third row of the direction cosine matrix. This object indexes the direction cosines of the angles between the z-axis and the XYZ axes.
Col1	Col_Indices	'FIRST	Object to be used as an index representing the first column of the direction cosine matrix. This object indexes the direction cosines of the angles between the X-axis and the xyz axes.
Col2	Col_Indices	SUCC 'FIRST	Object to be used as an index representing the second column of the direction cosine matrix. This object indexes the direction cosines of the angles between the Y-axis and the xyz axes.
Col3	Col_Indices	'LAST	Object to be used as an index representing the third column of the direction cosine matrix. This object indexes the direction cosines of the angles between the Z-axis and the xyz axes.

Subprograms:

The following table summarizes the generic formal subroutines required by this part:

Name	Type	Description
"*"	function	This function multiplies an object of type Angular_Velocities to an object of type Sin_Cos_Ratio returning an object of type Angular_Velocities.
"*"	function	This function multiplies an object of type Angular_Velocities to an object of type Time_Intervals returning an object of type Sin_Cos_Ratio.

FORMAL PARAMETERS:

The following table describes this part's formal parameters:

Name	Description
DC_Matrix	This object is the direction cosine matrix to be integrated.
X_Rho	This object is the new X angular velocity that is used as the upper bound in the integration.
Y_Rho	This object is the new Y angular velocity that is used as the upper bound in the integration.
Delta_Time	This object is the time increment since the last angular velocity update.

3.3.2.4.9.1.9.1.10.2.4 LOCAL DATA

Data objects:

The following table describes the data objects maintained by this part:

Name	Type	Value	Description
Prev_DCM	Direction_Cosine_Matrix	N.A.	This object holds the incoming DCM matrix for use in the integration.
X_Bound_Sum	Angular_Velocities	N.A.	This object is the sum of the lower and upper bounds (angular velocity in X direction) used in the integration.
Y_Bound_Sum	Angular_Velocities	N.A.	This object is the sum of the lower and upper bounds (angular velocity in Y direction) used in the integration.
Del_T_div_2	Time_Intervals	N.A.	This object is the time interval between the previous and current angular velocities - divided by 2 (for use in the integration).

3.3.2.4.9.1.9.1.10.2.5 PROCESS CONTROL

Not applicable.

3.3.2.4.9.1.9.1.10.2.6 PROCESSING

The following describes the processing performed by this part:

```
separate (Direction_Cosine_Matrix_Operations.DCM_General_Operations.
          DCM_Trapezoidal_Integration)
procedure Perform_Trapezoidal_Integration_of_DCM
             (DC_Matrix       : in out Direction_Cosine_Matrix;
              X_Rho           : in     Angular_Velocities;
              Y_Rho           : in     Angular_Velocities;
              Delta_Time      : in     Time_Intervals) is

-- ------------------------------------------
-- ---<objects for internal calculations>---
-- ------------------------------------------

   Prev_DCM    : Direction_Cosine_Matrix;
   X_Bound_Sum : Angular_Velocities;
   Y_Bound_Sum : Angular_Velocities;
   Del_T_div_2 : Time_Intervals;

------------------
--begin procedure
------------------

begin

-- -----<copy last two rows of DCM into previous DCM before computing>-----

   Prev_DCM(Row2,Col1) := DC_Matrix(Row2,Col1);
```

```
   Prev_DCM(Row2,Col2) := DC_Matrix(Row2,Col2);
   Prev_DCM(Row2,Col3) := DC_Matrix(Row2,Col3);

   Prev_DCM(Row3,Col1) := DC_Matrix(Row3,Col1);
   Prev_DCM(Row3,Col2) := DC_Matrix(Row3,Col2);
   Prev_DCM(Row3,Col3) := DC_Matrix(Row3,Col3);

-- ---<compute sum of X and Y bounds from old and new angular velocity>---

   X_Bound_Sum := (X_Rho + Prev_X_Rho);
   Y_Bound_Sum := (Y_Rho + Prev_Y_Rho);

-- ---<compute delta time divided by 2 - for algorithm simplification>--

   Del_T_div_2 := Delta_Time * Time_Intervals(0.5);

-- ---<compute new DCM>---

   DC_Matrix(row3,col1) := Prev_DCM(row3,col1) -
                           ( Y_Bound_Sum * Prev_DCM(row3,col3)
                             * Del_T_div_2);

   DC_Matrix(row3,col2) := Prev_DCM(row3,col2) +
                           ( X_Bound_Sum * Prev_DCM(row3,col3)
                             * Del_T_div_2);

   DC_Matrix(row3,col3) := Prev_DCM(row3,col3) +
                           ( Y_Bound_Sum * Prev_DCM(row3,col1) -
                             X_Bound_Sum * Prev_DCM(row3,col2)   )
                           * Del_T_div_2;

   DC_Matrix(row2,col1) := Prev_DCM(row2,col1) -
                           ( Y_Bound_Sum * Prev_DCM(row2,col3)
                             * Del_T_div_2);

   DC_Matrix(row2,col2) := Prev_DCM(row2,col2) +
                           ( X_Bound_Sum * Prev_DCM(row2,col3)
                             * Del_T_div_2);

   DC_Matrix(row2,col3) := Prev_DCM(row2,col3) +
                           ( Y_Bound_Sum * Prev_DCM(row2,col1) -
                             X_Bound_Sum * Prev_DCM(row2,col2)   )
                           * Del_T_div_2;

   Prev_X_Rho := X_Rho;
   Prev_Y_Rho := Y_Rho;

end Perform_Trapezoidal_Integration_of_DCM;
```

3.3.2.4.9.1.9.1.10.2.7 UTILIZATION OF OTHER ELEMENTS

UTILIZATION OF OTHER ELEMENTS IN TOP LEVEL COMPONENT:

The following tables describe the elements used by this part but defined elsewhere in the parent top level component:

Data objects:

The following table summarizes the objects required by this part and defined elsewhere in the parent top level component:

Name	Type	Source	Description
Prev_X_Rho	Angular_Velocities	Direction_Cosine_Matrix_Operations.DCM_General_Operations.DCM_Trapezoidal_Integration	This object is the angular velocity in the X direction that is used as the previous angular velocity for Trapezoidal integration.
Prev_Y_Rho	Angular_Velocities	Direction_Cosine_Matrix_Operations.DCM_General_Operations.DCM_Trapezoidal_Integration	This object is the angular velocity in the Y direction that is used as the previous angular velocity for Trapezoidal integration.

3.3.2.4.9.1.9.1.10.2.8 LIMITATIONS

None.

3.3.2.4.9.1.10 UNIT DESIGN

3.3.2.4.9.1.10.1 DCM_INITIALIZED_FROM_REFERENCE UNIT DESIGN (CATALOG #P288-0)

This function initializes a direction cosine matrix to a reference direction cosine matrix. This part uses the orthonormal properties of the DCM to compute all nine elements of the matrix from four elements of the reference DCM plus the signs from two other reference DCM elements.

This implementation assumes that these four elements are (y,X), (y,Y), (z,X), (z,Y), where xyz and XYZ are the two coordinate frames. The signs are computed from elements (y,Z) and (z,Z).

3.3.2.4.9.1.10.1.1 REQUIREMENTS ALLOCATION

N/A

3.3.2.4.9.1.10.1.2 LOCAL ENTITIES DESIGN

None.

3.3.2.4.9.1.10.1.3 INPUT/OUTPUT

GENERIC PARAMETERS:

The following table describes the generic formal types required by this part:

Name	Type	Description
Row_Indices	discrete type	Data type which translates into the row indexing of the direction cosine matrix.
Col_Indices	discrete type	Data type which translates into the column indexing of the direction cosine matrix.
Sin_Cos_Ratio	floating point	Type which translates into the type of the elements of the direction cosine matrix.
Direction_Cosine_Matrix	3X3 matrix	Type representing a direction cosine matrix indexed by the column and row types described above with elements of type Sin_Cos_Ratio.

Data objects:

The following table describes the generic formal objects required by this part:

Name	Type	Value	Description
Row1	Row_Indices	'FIRST	Object to be used as an index representing the first row of the direction cosine matrix. This object indexes the direction cosines of the angles between the x-axis and the XYZ axes.
Row2	Row_Indices	SUCC 'FIRST	Object to be used as an index representing the second row of the direction cosine matrix. This object indexes the direction cosines of the angles between the y-axis and the XYZ axes.
Row3	Row_Indices	'LAST	Object to be used as an index representing the third row of the direction cosine matrix. This object indexes the direction cosines of the angles between the z-axis and the XYZ axes.
Col1	Col_Indices	'FIRST	Object to be used as an index representing the first column of the direction cosine matrix. This object indexes the direction cosines of the angles between the X-axis and the xyz axes.
Col2	Col_Indices	SUCC 'FIRST	Object to be used as an index representing the second column of the direction cosine matrix. This object indexes the direction cosines of the angles between the Y-axis and the xyz axes.
Col3	Col_Indices	'LAST	Object to be used as an index representing the third column of the direction cosine matrix. This object indexes the direction cosines of the angles between the Z-axis and the xyz axes.

Subprograms:

The following table describes the generic formal subroutines required by this part:

Name	Type	Description
SQRT	function	This function computes the square root of a value of the type Sin_Cos_Ratio.

FORMAL PARAMETERS:

The following table describes this part's formal parameters:

Name	Type	Mode	Description
REF_DCM_2_1	Sin_Cos_Ratio	In	This value is the reference direction cosine element located at row 2, column 1. It represents the direction cosine of the angle between the y-axis and the X-axis.
REF_DCM_2_2	Sin_Cos_Ratio	In	This value is the reference direction cosine element located at row 2, column 2. It represents the direction cosine of the angle between the y-axis and the Y-axis.
REF_DCM_3_1	Sin_Cos_Ratio	In	This value is the reference direction cosine element located at row 3, column 1. It represents the direction cosine of the angle between the z-axis and the X-axis.
REF_DCM_3_2	Sin_Cos_Ratio	In	This value is the reference direction cosine element located at row 3, column 2. It represents the direction cosine of the angle between the z-axis and the Y-axis.
Sign_Of_2_3	INTEGER	In	This value represents the sign of the reference direction cosine element located at row 2, column 3. This value is the sign of the direction cosine of the angle between the y-axis and the Z-axis. The value must equal either -1 or 1, which represents minus or plus, respectively.
Sign_Of_3_3	INTEGER	In	This value represents the sign of the reference direction cosine element located at row 3, column 3. This value is the sign of the direction cosine of the angle between the z-axis and the Z-axis. The value must equal either -1 or 1, which represents minus or plus, respectively.

3.3.2.4.9.1.10.1.4 LOCAL DATA

Data objects:

The following table describes the data objects maintained by this part:

Name	Type	Value	Description
DCM_Matrix	Direction_Cosine_Matrix	N.A.	This object is a 3X3 matrix representing the direction consine matrix that is computed from the reference DCM which is passed in.

3.3.2.4.9.1.10.1.5 PROCESS CONTROL

Not applicable.

3.3.2.4.9.1.10.1.6 PROCESSING

The following describes the processing performed by this part:

```
separate (Direction_Cosine_Matrix_Operations.DCM_General_Operations)
function DCM_Initialized_From_Reference
                  (REF_DCM_2_1    : Sin_Cos_Ratio;
                   REF_DCM_2_2    : Sin_Cos_Ratio;
                   REF_DCM_3_1    : Sin_Cos_Ratio;
                   REF_DCM_3_2    : Sin_Cos_Ratio;
                   Sign_Of_2_3    : INTEGER;
                   Sign_Of_3_3    : INTEGER)
                  return Direction_Cosine_Matrix is

-- --<direction cosine matrix to be computed>---

   DCM_Matrix     : Direction_Cosine_Matrix;

begin

-- ---<compute new DCM>---

   if Sign_Of_2_3 = 1 then

      DCM_Matrix(Row2,Col3) := SQRT (1.0 - (   REF_DCM_2_1 *
                                               REF_DCM_2_1
                                           +
                                               REF_DCM_2_2 *
                                               REF_DCM_2_2   )
                                    );
   else

      DCM_Matrix(Row2,Col3) := - SQRT (1.0 - (   REF_DCM_2_1 *
                                                 REF_DCM_2_1
                                             +
                                                 REF_DCM_2_2 *
                                                 REF_DCM_2_2   )
                                       );
   end if;

   if Sign_Of_3_3 = 1 then
```

```
      DCM_Matrix(Row3,Col3) := SQRT (1.0 - (   REF_DCM_3_1 *
                                               REF_DCM_3_1
                                           +
                                               REF_DCM_3_2 *
                                               REF_DCM_3_2   )
                                    );
   else
      DCM_Matrix(Row3,Col3) := - SQRT (1.0 - (   REF_DCM_3_1 *
                                                 REF_DCM_3_1
                                             +
                                                 REF_DCM_3_2 *
                                                 REF_DCM_3_2   )
                                       );
   end if;

   DCM_Matrix(Row1,Col1) :=    REF_DCM_2_2 *
                               DCM_Matrix(Row3,Col3)
                            -
                               DCM_Matrix(Row2,Col3) *
                               REF_DCM_3_2;

   DCM_Matrix(Row1,Col2) :=    DCM_Matrix(Row2,Col3) *
                               REF_DCM_3_1
                            -
                               REF_DCM_2_1 *
                               DCM_Matrix(Row3,Col3);

   DCM_Matrix(Row1,Col3) :=    REF_DCM_2_1 *
                               REF_DCM_3_2
                            -
                               REF_DCM_2_2 *
                               REF_DCM_3_1;

   DCM_Matrix(Row2,Col1) := REF_DCM_2_1;
   DCM_Matrix(Row2,Col2) := REF_DCM_2_2;
   DCM_Matrix(Row3,Col1) := REF_DCM_3_1;
   DCM_Matrix(Row3,Col2) := REF_DCM_3_2;

   return DCM_Matrix;

end DCM_Initialized_From_Reference;
```

3.3.2.4.9.1.10.1.7 UTILIZATION OF OTHER ELEMENTS

None.

3.3.2.4.9.1.10.1.8 LIMITATIONS

None.

3.3.2.4.9.1.10.2 PERFORM_RECTANGULAR_INTEGRATION_OF_DCM UNIT DESIGN (CATALOG #P290-0)

This procedure integrates a direction cosine matrix using rectangular integration. Only rows 2 and 3 are updated. Row 1 must be updated by another function (using the orthonormal property).

3.3.2.4.9.1.10.2.1 REQUIREMENTS ALLOCATION

N/A

3.3.2.4.9.1.10.2.2 LOCAL ENTITIES DESIGN

None.

3.3.2.4.9.1.10.2.3 INPUT/OUTPUT

GENERIC PARAMETERS:

Data types:

The following table describes the generic formal types required by this part:

Name	Type	Description
Angular_Velocities	floating point type	This type translates into the type of the angular velocity used to integrate the direction cosine matrix.
Time_Intervals	floating point type	This type translates into the type of the time parameter used to integrate the direction cosine matrix.
Row_Indices	discrete type	Data type which translates into the row indexing of the direction cosine matrix.
Col_Indices	discrete type	Data type which translates into the column indexing of the direction cosine matrix.
Sin_Cos_Ratio	floating point	Type which translates into the type of the elements of the direction cosine matrix.
Direction_Cosine_Matrix	3X3 matrix	Type representing a direction cosine matrix indexed by the column and row types described above with elements of type Sin_Cos_Ratio.

Data objects:

The following table describes the generic formal objects required by this part:

Name	Type	Value	Description
Row1	Row_Indices	'FIRST	Object to be used as an index representing the first row of the direction cosine matrix. This object indexes the direction cosines of the angles between the x-axis and the XYZ axes.
Row2	Row_Indices	SUCC 'FIRST	Object to be used as an index representing the second row of the direction cosine matrix. This object indexes the direction cosines of the angles between the y-axis and the XYZ axes.
Row3	Row_Indices	'LAST	Object to be used as an index representing the third row of the direction cosine matrix. This object indexes the direction cosines of the angles between the z-axis and the XYZ axes.
Col1	Col_Indices	'FIRST	Object to be used as an index representing the first column of the direction cosine matrix. This object indexes the direction cosines of the angles between the X-axis and the xyz axes.
Col2	Col_Indices	SUCC 'FIRST	Object to be used as an index representing the second column of the direction cosine matrix. This object indexes the direction cosines of the angles between the Y-axis and the xyz axes.
Col3	Col_Indices	'LAST	Object to be used as an index representing the third column of the direction cosine matrix. This object indexes the direction cosines of the angles between the Z-axis and the xyz axes.

Subprograms:

The following table describes the generic formal subroutines required by this part:

Name	Type	Description
"*"	function	This function multiplies an object of type Angular_Velocities to an object of type Time_Intervals returning an object of type Sin_Cos_Ratio.

FORMAL PARAMETERS:

The following table describes this part's formal parameters:

Name	Type	Mode	Description
DC_Matrix	Direction_Cosine_Matrix	in out	This is the direction cosine matrix being integrated.
X_Rho	Angular_Velocities	in	This value represents the angular velocity of the x-axis with respect to the X-axis in the two coordinate frames xyz and XYZ.
Y_Rho	Angular_Velocities	in	This value represents the angular velocity of the y-axis with respect to the Y-axis in the two coordinate frames xyz and XYZ.
Delta_Time	Time_Intervals	in	This value is the time interval over which the integration is to be performed.

3.3.2.4.9.1.10.2.4 LOCAL DATA

Data objects:

The following table describes the data objects maintained by this part:

Name	Type	Value	Description
DCM_Incr	Direction_Cosine_Matrix	N.A.	This object is a direction cosine matrix that is used as temporary storage for computed increments.
Y_Rho_Times_Delta_Time	Sin_Cos_Ratio	Y_Rho * Delta_Time	This object holds the angular velocity Y_Rho times the time increment.
X_Rho_Times_Delta_Time	Sin_Cos_Ratio	X_Rho * Delta_Time	This object holds the angular velocity X_Rho times the time increment.

3.3.2.4.9.1.10.2.5 PROCESS CONTROL

Not applicable.

3.3.2.4.9.1.10.2.6 PROCESSING

The following describes the processing performed by this part:

```
separate (Direction_Cosine_Matrix_Operations.DCM_General_Operations)
procedure Perform_Rectangular_Integration_of_DCM
          (DC_Matrix  : in out Direction_Cosine_Matrix;
           X_Rho      : in     Angular_Velocities;
           Y_Rho      : in     Angular_Velocities;
           Delta_Time : in     Time_Intervals) is

-- ---------------------
-- --declaration section
-- ---------------------

-- ---<DCM for computing increments (temporary) >---

   DCM_Incr                : Direction_Cosine_Matrix;

-- ---< objects for internal calculation >---

   Y_Rho_Times_Delta_Time : Sin_Cos_Ratio := Y_Rho * Delta_Time;
   X_Rho_Times_Delta_Time : Sin_Cos_Ratio := X_Rho * Delta_Time;

-----------------
--begin procedure
-----------------

begin

-- --<compute increments for bottom two rows>--

   DCM_Incr(Row2,Col1) := -DC_Matrix(Row2,Col3) * Y_Rho_Times_Delta_Time;
   DCM_Incr(Row2,Col2) :=  DC_Matrix(Row2,Col3) * X_Rho_Times_Delta_Time;
   DCM_Incr(Row2,Col3) :=  DC_Matrix(Row2,Col1) * Y_Rho_Times_Delta_Time -
                           DC_Matrix(Row2,Col2) * X_Rho_Times_Delta_Time;
   DCM_Incr(Row3,Col1) := -DC_Matrix(Row3,Col3) * Y_Rho_Times_Delta_Time;
   DCM_Incr(Row3,Col2) :=  DC_Matrix(Row3,Col3) * X_Rho_Times_Delta_Time;
   DCM_Incr(Row3,Col3) :=  DC_Matrix(Row3,Col1) * Y_Rho_Times_Delta_Time -
                           DC_Matrix(Row3,Col2) * X_Rho_Times_Delta_Time;

-- --<compute current values of bottom two Rows>--

   DC_Matrix(Row2,Col1) := DC_Matrix(Row2,Col1) + DCM_Incr(Row2,Col1);
   DC_Matrix(Row2,Col2) := DC_Matrix(Row2,Col2) + DCM_Incr(Row2,Col2);
   DC_Matrix(Row2,Col3) := DC_Matrix(Row2,Col3) + DCM_Incr(Row2,Col3);

   DC_Matrix(Row3,Col1) := DC_Matrix(Row3,Col1) + DCM_Incr(Row3,Col1);
   DC_Matrix(Row3,Col2) := DC_Matrix(Row3,Col2) + DCM_Incr(Row3,Col2);
   DC_Matrix(Row3,Col3) := DC_Matrix(Row3,Col3) + DCM_Incr(Row3,Col3);

end Perform_Rectangular_Integration_of_DCM;
```

3.3.2.4.9.1.10.2.7 UTILIZATION OF OTHER ELEMENTS

None.

3.3.2.4.9.1.10.2.8 LIMITATIONS

None.

3.3.2.4.9.1.10.3 REORTHONORMALIZE_DCM UNIT DESIGN (CATALOG #P291-0)

This procedure reorthonormalizes (when executed repeatedly) the bottom two rows of the direction cosine matrix. These two rows represent the y and z axes in a direction cosine matrix that transforms data from coordinate frame xyz to frame XYZ. Note that the first row is not updated.

This procedure will not reorthonormalize the DCM after only one iteration. It must be executed repeatedly at a rate based on the precision of the data, the update frequency of the DCM, and other considerations. For example, in a typical application, this function is executed once per minute.

3.3.2.4.9.1.10.3.1 REQUIREMENTS ALLOCATION

N/A

3.3.2.4.9.1.10.3.2 LOCAL ENTITIES DESIGN

None.

3.3.2.4.9.1.10.3.3 INPUT/OUTPUT

GENERIC PARAMETERS:

Data types:

The following table describes the generic formal types required by this part:

Name	Type	Description
Row_Indices	discrete type	Data type which translates into the row indexing of the direction cosine matrix.
Col_Indices	discrete type	Data type which translates into the column indexing of the direction cosine matrix.
Sin_Cos_Ratio	floating point	Type which translates into the type of the elements of the direction cosine matrix.
Direction_Cosine_Matrix	3X3 matrix	Type representing a direction cosine matrix indexed by the column and row types described above with elements of type Sin_Cos_Ratio.

Data objects:

The following table describes the generic formal objects required by this part:

Name	Type	Value	Description
Row1	Row_Indices	'FIRST	Object to be used as an index representing the first row of the direction cosine matrix. This object indexes the direction cosines of the angles between the x-axis and the XYZ axes.
Row2	Row_Indices	SUCC 'FIRST	Object to be used as an index representing the second row of the direction cosine matrix. This object indexes the direction cosines of the angles between the y-axis and the XYZ axes.
Row3	Row_Indices	'LAST	Object to be used as an index representing the third row of the direction cosine matrix. This object indexes the direction cosines of the angles between the z-axis and the XYZ axes.
Col1	Col_Indices	'FIRST	Object to be used as an index representing the first column of the direction cosine matrix. This object indexes the direction cosines of the angles between the X-axis and the xyz axes.
Col2	Col_Indices	SUCC 'FIRST	Object to be used as an index representing the second column of the direction cosine matrix. This object indexes the direction cosines of the angles between the Y-axis and the xyz axes.
Col3	Col_Indices	'LAST	Object to be used as an index representing the third column of the direction cosine matrix. This object indexes the direction cosines of the angles between the Z-axis and the xyz axes.

FORMAL PARAMETERS:

The following table describes this part's formal parameters:

Name	Type	Mode	Description
DCM_Matrix	Direction_Cosine_Matrix	in	This parameter is the direction cosine matrix to be reorthonormalized.

3.3.2.4.9.1.10.3.4 LOCAL DATA

Data objects:

The following table describes the data objects maintained by this part:

Name	Type	Description
Row2_Dot_Row2	Real	This object is used to store the dot product of row 2 and row 2 of the DCM.
Row3_Dot_Row3	Real	This object is used to store the dot product of row 3 and row 3 of the DCM.
Row2_dot_Row3 Row2_Dot_Row3	Real	This object is used to store the dot product of row 2 and row 3 of the DCM.
Half Row2_Dot_Row3	Sin_Cos_ Ratio	This object is used to store half the dot product of row 2 and row 3 of the DCM.

3.3.2.4.9.1.10.3.5 PROCESS CONTROL

Not applicable.

3.3.2.4.9.1.10.3.6 PROCESSING

The following describes the processing performed by this part:

```
separate (Direction_Cosine_Matrix_Operations.DCM_General_Operations)
procedure Reorthonormalize_DCM
          (DCM_Matrix  : in out Direction_Cosine_Matrix) is

-- ---------------------
-- --declaration section
-- ---------------------

-- ---<objects for storing dot products>---

   Row2_Dot_Row2      : Real;
   Row3_Dot_Row3      : Real;
   Row2_Dot_Row3      : Real;
   Half_Row2_Dot_Row3 : Sin_Cos_Ratio;

-----------------
--begin procedure
-----------------

begin

   Row2_Dot_Row2 := DCM_Matrix(Row2,Col1) * DCM_Matrix(Row2,Col1) +
                    DCM_Matrix(Row2,Col2) * DCM_Matrix(Row2,Col2) +
                    DCM_Matrix(Row2,Col3) * DCM_Matrix(Row2,Col3);

   Row3_Dot_Row3 := DCM_Matrix(Row3,Col1) * DCM_Matrix(Row3,Col1) +
                    DCM_Matrix(Row3,Col2) * DCM_Matrix(Row3,Col2) +
                    DCM_Matrix(Row3,Col3) * DCM_Matrix(Row3,Col3);
```

```
-- -------------------------
-- --calculate column values
-- -------------------------

   DCM_Matrix(Row2,Col1) := DCM_Matrix(Row2,Col1) *
                            (1.5 - Real(0.5) * Row2_Dot_Row2);

   DCM_Matrix(Row3,Col1) := DCM_Matrix(Row3,Col1) *
                            (1.5 - Real(0.5) * Row3_Dot_Row3);

   DCM_Matrix(Row2,Col2) := DCM_Matrix(Row2,Col2) *
                            (1.5 - Real(0.5) * Row2_Dot_Row2);

   DCM_Matrix(Row3,Col2) := DCM_Matrix(Row3,Col2) *
                            (1.5 - Real(0.5) * Row3_Dot_Row3);

   DCM_Matrix(Row2,Col3) := DCM_Matrix(Row2,Col3) *
                            (1.5 - Real(0.5) * Row2_Dot_Row2);

   DCM_Matrix(Row3,Col3) := DCM_Matrix(Row3,Col3) *
                            (1.5 - Real(0.5) * Row3_Dot_Row3);

-- -------------------------
-- --<compute Rows 2 thru 3>
-- -------------------------

   Row2_Dot_Row3 := DCM_Matrix(Row2,Col1) * DCM_Matrix(Row3,Col1) +
                    DCM_Matrix(Row2,Col2) * DCM_Matrix(Row3,Col2) +
                    DCM_Matrix(Row2,Col3) * DCM_Matrix(Row3,Col3);

   Half_Row2_dot_Row3 := 0.5 * Row2_dot_Row3;

   DCM_Matrix(Row2,Col1) := DCM_Matrix(Row2,Col1) -
                            (Half_Row2_Dot_Row3 * DCM_Matrix(Row3,Col1));

   DCM_Matrix(Row3,Col1) := DCM_Matrix(Row3,Col1) -
                            (Half_Row2_Dot_Row3 * DCM_Matrix(Row2,Col1));

   DCM_Matrix(Row2,Col2) := DCM_Matrix(Row2,Col2) -
                            (Half_Row2_Dot_Row3 * DCM_Matrix(Row3,Col2));

   DCM_Matrix(Row3,Col2) := DCM_Matrix(Row3,Col2) -
                            (Half_Row2_Dot_Row3 * DCM_Matrix(Row2,Col2));

   DCM_Matrix(Row2,Col3) := DCM_Matrix(Row2,Col3) -
                            (Half_Row2_Dot_Row3 * DCM_Matrix(Row3,Col3));

   DCM_Matrix(Row3,Col3) := DCM_Matrix(Row3,Col3) -
                            (Half_Row2_Dot_Row3 * DCM_Matrix(Row2,Col3));

end Reorthonormalize_DCM;
```

3.3.2.4.9.1.10.3.7 UTILIZATION OF OTHER ELEMENTS

None.

3.3.2.4.9.1.10.3.8 LIMITATIONS

None.

3.3.2.4.9.1.10.4 FRAME_MISALIGNMENT UNIT DESIGN (CATALOG #P292-0)

This unit computes the rotation of the computed XYZ frame with respect to a reference XYZ frame given the direction cosine matrix of the computed XYZ frame and the direction cosine matrix of the reference XYZ frame. Each rotation angle provided by this part is approximated as the sine of the rotation angle.

3.3.2.4.9.1.10.4.1 REQUIREMENTS ALLOCATION

N/A

3.3.2.4.9.1.10.4.2 LOCAL ENTITIES DESIGN

None.

3.3.2.4.9.1.10.4.3 INPUT/OUTPUT

GENERIC PARAMETERS:

Data types:

The following table describes the generic formal types required by this part:

Name	Type	Description
Row_Indices	discrete type	Data type which translates into the row indexing of the direction cosine matrix.
Col_Indices	discrete type	Data type which translates into the column indexing of the direction cosine matrix.
Sin_Cos_Ratio	floating point	Type which translates into the type of the elements of the direction cosine matrix.
Direction_Cosine_Matrix	3X3 matrix	Type representing a direction cosine matrix indexed by the column and row types described above with elements of type Sin_Cos_Ratio.
Rotation_Indices	discrete type	Data type which translates into the indexing elements of a rotation angle vector.
Rotation_Angles	floating point	Type which translates into the type of the elements of a rotation angle vector.
Rotation_Angle_Vec	1X3 matrix (vector)	Type representing a rotation angle vector.

Data objects:

The following table describes the generic formal objects required by this part:

Name	Type	Value	Description
Row1	Row_Indices	'FIRST	Object to be used as an index representing the first row of the direction cosine matrix. This object indexes the direction cosines of the angles between the x-axis and the XYZ axes.
Row2	Row_Indices	SUCC 'FIRST	Object to be used as an index representing the second row of the direction cosine matrix. This object indexes the direction cosines of the angles between the y-axis and the XYZ axes.
Row3	Row_Indices	'LAST	Object to be used as an index representing the third row of the direction cosine matrix. This object indexes the direction cosines of the angles between the z-axis and the XYZ axes.
Col1	Col_Indices	'FIRST	Object to be used as an index representing the first column of the direction cosine matrix. This object indexes the direction cosines of the angles between the X-axis and the xyz axes.
Col2	Col_Indices	SUCC 'FIRST	Object to be used as an index representing the second column of the direction cosine matrix. This object indexes the direction cosines of the angles between the Y-axis and the xyz axes.
Col3	Col_Indices	'LAST	Object to be used as an index representing the third column of the direction cosine matrix. This object indexes the direction cosines of the angles between the Z-axis and the xyz axes.
X	Rotation _Indices	'FIRST	Object to be used as an index representing the first element of the rotation angle vector.
Y	Rotation _Indices	SUCC 'FIRST	Object to be used as an index representing the second element of the rotation angle vector.
Z	Rotation _Indices	'LAST	Object to be used as an index representing the third element of the rotation angle vector.

Subprograms:

The following table describes the generic formal subroutines required by this part:

Name	Type	Description
"*"	function	This function multiplies an object of type Sin_Cos_Ratio to an object of type Sin_Cos_Ratio returning an object of type Rotation_Angles.

FORMAL PARAMETERS:

The following table describes this part's formal parameters:

Name	Type	Mode	Description
DCM_Matrix	Direction_Cosine_Matrix	In	This parameter is the current active direction cosine matrix.
REF_DC_Matrix	Direction_Cosine_Matrix	In	This parameter is the reference direction cosine matrix. The misalignment angles are computed by determining the angular differences between this DCM and the active DCM.

3.3.2.4.9.1.10.4.4 LOCAL DATA

Data objects:

The following table describes the data objects maintained by this part:

Name	Type	Value	Description
Rot_Angle	Rotation_Angle_Vec	N.A.	This object is the rotation angle vector that is returned by the function.

3.3.2.4.9.1.10.4.5 PROCESS CONTROL

Not applicable.

3.3.2.4.9.1.10.4.6 PROCESSING

The following describes the processing performed by this part:

```
separate (Direction_Cosine_Matrix_Operations.DCM_General_Operations)
function Frame_Misalignment
              (DCM_Matrix     : Direction_Cosine_Matrix;
               REF_DC_Matrix  : Direction_Cosine_Matrix)
              return Rotation_Angle_Vec is
```

```
-- ---------------------
-- --declaration section
-- ---------------------

-- ---<the rotation angle vector that is returned>---

   Rot_Angle    : Rotation_Angle_Vec;

----------------
--begin function
----------------

begin

   Rot_Angle(X) := DCM_Matrix(Row1,Col2) * REF_DC_Matrix(Row1,Col3) +
                   DCM_Matrix(Row2,Col2) * REF_DC_Matrix(Row2,Col3) +
                   DCM_Matrix(Row3,Col2) * REF_DC_Matrix(Row3,Col3);

   Rot_Angle(Y) := DCM_Matrix(Row1,Col3) * REF_DC_Matrix(Row1,Col1) +
                   DCM_Matrix(Row2,Col3) * REF_DC_Matrix(Row2,Col1) +
                   DCM_Matrix(Row3,Col3) * REF_DC_Matrix(Row3,Col1);

   Rot_Angle(Z) := DCM_Matrix(Row1,Col1) * REF_DC_Matrix(Row1,Col2) +
                   DCM_Matrix(Row2,Col1) * REF_DC_Matrix(Row2,Col2) +
                   DCM_Matrix(Row3,Col1) * REF_DC_Matrix(Row3,Col2);

   return Rot_Angle;

end Frame_Misalignment;
```

3.3.2.4.9.1.10.4.7 UTILIZATION OF OTHER ELEMENTS

None.

3.3.2.4.9.1.10.4.8 LIMITATIONS

None.

3.3.2.4.9.1.10.5 ALIGNED_DCM_MATRIX UNIT DESIGN (CATALOG #P293-0)

This part modifies the bottom two rows (y and z) of the direction cosine matrix so that the current coordinate frame is aligned to a true coordinate frame. This part assumes a coordinate frame representation of xyz to XYZ, where xyz is the coordinate frame noted as "true". NOTE: Only rows 2 and 3 are aligned. Row 1 must be computed separately.

3.3.2.4.9.1.10.5.1 REQUIREMENTS ALLOCATION

N/A

3.3.2.4.9.1.10.5.2 LOCAL ENTITIES DESIGN

None.

3.3.2.4.9.1.10.5.3 INPUT/OUTPUT

GENERIC PARAMETERS:

Data types:

The following table describes the generic formal types required by this part:

Name	Type	Description
Row_Indices	discrete type	Data type which translates into the row indexing of the direction cosine matrix.
Col_Indices	discrete type	Data type which translates into the column indexing of the direction cosine matrix.
Sin_Cos_Ratio	floating point	Type which translates into the type of the elements of the direction cosine matrix.
Direction_Cosine_Matrix	3X3 matrix	Type representing a direction cosine matrix indexed by the column and row types described above with elements of type Sin_Cos_Ratio.
Rotation_Indices	discrete type	Data type which translates into the indexing elements of a rotation angle vector.
Rotation_Angles	floating point	Type which translates into the type of the elements of a rotation angle vector.
Rotation_Angle_Vec	1X3 matrix (vector)	Type representing a rotation angle vector.

Data objects:

The following table describes the generic formal objects required by this part:

Name	Type	Value	Description
Row1	Row_Indices	'FIRST	Object to be used as an index representing the first row of the direction cosine matrix. This object indexes the direction cosines of the angles between the x-axis and the XYZ axes.
Row2	Row_Indices	SUCC 'FIRST	Object to be used as an index representing the second row of the direction cosine matrix. This object indexes the direction cosines of the angles between the y-axis and the XYZ axes.
Row3	Row_Indices	'LAST	Object to be used as an index representing the third row of the direction cosine matrix. This object indexes the direction cosines of the angles between the z-axis and the XYZ axes.
Col1	Col_Indices	'FIRST	Object to be used as an index representing the first column of the direction cosine matrix. This object indexes the direction cosines of the angles between the X-axis and the xyz axes.
Col2	Col_Indices	SUCC 'FIRST	Object to be used as an index representing the second column of the direction cosine matrix. This object indexes the direction cosines of the angles between the Y-axis and the xyz axes.
Col3	Col_Indices	'LAST	Object to be used as an index representing the third column of the direction cosine matrix. This object indexes the direction cosines of the angles between the Z-axis and the xyz axes.
X	Rotation_Indices	'FIRST	Object to be used as an index representing the first element of the rotation angle vector.
Y	Rotation_Indices	SUCC 'FIRST	Object to be used as an index representing the second element of the rotation angle vector.
Z	Rotation_Indices	'LAST	Object to be used as an index representing the third element of the rotation angle vector.

Subprograms:

The following table describes the generic formal subroutines required by this part:

Name	Type	Description
"*"	function	This function multiplies an object of type Sin_Cos_Ratio to an object of type Rotation_Angles returning an object of type Sin_Cos_Ratio.

FORMAL PARAMETERS:

The following table describes this part's formal parameters:

Name	Type	Mode	Description
DCM_Matrix	Direction_Cosine_Matrix	In	This parameter is the current active direction cosine matrix.
Rotation_Angle	Rotation_Angle_Vec	In	This parameter represents the rotation angles used to align the DCM matrix.

3.3.2.4.9.1.10.5.4 LOCAL DATA

Data objects:

The following table describes the data objects maintained by this part:

Name	Type	Value	Description
DCM_New	Direction_Cosine_Matrix	N.A.	This object is the direction cosine matrix that is computed and returned.

3.3.2.4.9.1.10.5.5 PROCESS CONTROL

Not applicable.

3.3.2.4.9.1.10.5.6 PROCESSING

The following describes the processing performed by this part:

```
separate (Direction_Cosine_Matrix_Operations.DCM_General_Operations)
function Aligned_DCM_Matrix
         (DCM_Matrix     : Direction_Cosine_Matrix;
          Rotation_Angle : Rotation_Angle_Vec)
         return Direction_Cosine_Matrix is

-- ----------------------
-- --declaration section
```

```
-- ---------------------

-- ---<computed DCM to be returned>---

   DCM_New : Direction_Cosine_Matrix;

-----------------
--begin function
-----------------

begin

   DCM_New(Row2,Col1) := DCM_Matrix(Row2,Col1) + DCM_Matrix(Row2,Col3) *
                         Rotation_Angle(Y);

   DCM_New(Row2,Col2) := DCM_Matrix(Row2,Col2) - DCM_Matrix(Row2,Col3) *
                         Rotation_Angle(X);

   DCM_New(Row2,Col3) := DCM_Matrix(Row2,Col3) - DCM_Matrix(Row2,Col1) *
                         Rotation_Angle(Y) +
                         DCM_Matrix(Row2,Col2) * Rotation_Angle(X);

   DCM_New(Row3,Col1) := DCM_Matrix(Row3,Col1) + DCM_Matrix(Row3,Col3) *
                         Rotation_Angle(Y);

   DCM_New(Row3,Col2) := DCM_Matrix(Row3,Col2) - DCM_Matrix(Row3,Col3) *
                         Rotation_Angle(X);

   DCM_New(Row3,Col3) := DCM_Matrix(Row3,Col3) - DCM_Matrix(Row3,Col1) *
                         Rotation_Angle(Y) +
                         DCM_Matrix(Row3,Col2) * Rotation_Angle(X);

-- ---<set first row equal to original matrix>--
-- ---<NOTE:  The first row is now INVALID data>--

  DCM_New(Row1,Col1) := DCM_Matrix(Row1,Col1);
  DCM_New(Row1,Col2) := DCM_Matrix(Row1,Col2);
  DCM_New(Row1,Col3) := DCM_Matrix(Row1,Col3);

  return DCM_New;

end Aligned_DCM_Matrix;
```

3.3.2.4.9.1.10.5.7 UTILIZATION OF OTHER ELEMENTS

None.

3.3.2.4.9.1.10.5.8 LIMITATIONS

None.

3.3.2.4.9.1.10.6 COMPUTE_FIRST_ROW_FROM_ORTHONORMAL UNIT DESIGN (CATALOG #P294-0)

This part computes the first row of the direction cosine matrix from the second and third rows of the same matrix using the orthonormal property.

In many applications, the second and third rows are updated at a higher rate than the first row. Because of this, this function has been designed as a separate part.

3.3.2.4.9.1.10.6.1 REQUIREMENTS ALLOCATION

N/A

3.3.2.4.9.1.10.6.2 LOCAL ENTITIES DESIGN

None.

3.3.2.4.9.1.10.6.3 INPUT/OUTPUT

GENERIC PARAMETERS:

Data types:

The following table describes the generic formal types required by this part:

Name	Type	Description
Row_Indices	discrete type	Data type which translates into the row indexing of the direction cosine matrix.
Col_Indices	discrete type	Data type which translates into the column indexing of the direction cosine matrix.
Sin_Cos_Ratio	floating point	Type which translates into the type of the elements of the direction cosine matrix.
Direction_Cosine_Matrix	3X3 matrix	Type representing a direction cosine matrix indexed by the column and row types described above with elements of type Sin_Cos_Ratio.

Data objects:

The following table describes the generic formal objects required by this part:

Name	Type	Value	Description
Row1	Row_Indices	'FIRST	Object to be used as an index representing the first row of the direction cosine matrix. This object indexes the direction cosines of the angles between the x-axis and the XYZ axes.
Row2	Row_Indices	SUCC 'FIRST	Object to be used as an index representing the second row of the direction cosine matrix. This object indexes the direction cosines of the angles between the y-axis and the XYZ axes.
Row3	Row_Indices	'LAST	Object to be used as an index representing the third row of the direction cosine matrix. This object indexes the direction cosines of the angles between the z-axis and the XYZ axes.
Col1	Col_Indices	'FIRST	Object to be used as an index representing the first column of the direction cosine matrix. This object indexes the direction cosines of the angles between the X-axis and the xyz axes.
Col2	Col_Indices	SUCC 'FIRST	Object to be used as an index representing the second column of the direction cosine matrix. This object indexes the direction cosines of the angles between the Y-axis and the xyz axes.
Col3	Col_Indices	'LAST	Object to be used as an index representing the third column of the direction cosine matrix. This object indexes the direction cosines of the angles between the Z-axis and the xyz axes.

FORMAL PARAMETERS:

The following table describes this part's formal parameters:

Name	Type	Mode	Description
DCM_Matrix	Direction_Cosine_Matrix	In Out	This parameter is the current active direction cosine matrix.

3.3.2.4.9.1.10.6.4 LOCAL DATA

None.

3.3.2.4.9.1.10.6.5 PROCESS CONTROL

Not applicable.

3.3.2.4.9.1.10.6.6 PROCESSING

The following describes the processing performed by this part:

```
separate (Direction_Cosine_Matrix_Operations.DCM_General_Operations)
procedure Compute_First_Row_From_Orthonormal
            (DCM_Matrix      : in out Direction_Cosine_Matrix) is

begin

   DCM_Matrix(Row1,Col1) :=   DCM_Matrix(Row2,Col2) * DCM_Matrix(Row3,Col3)
                            - DCM_Matrix(Row2,Col3) * DCM_Matrix(Row3,Col2);

   DCM_Matrix(Row1,Col2) :=   DCM_Matrix(Row2,Col3) * DCM_Matrix(Row3,Col1)
                            - DCM_Matrix(Row2,Col1) * DCM_Matrix(Row3,Col3);

   DCM_Matrix(Row1,Col3) :=   DCM_Matrix(Row2,Col1) * DCM_Matrix(Row3,Col2)
                            - DCM_Matrix(Row2,Col2) * DCM_Matrix(Row3,Col1);

end Compute_First_Row_From_Orthonormal;
```

3.3.2.4.9.1.10.6.7 UTILIZATION OF OTHER ELEMENTS

None.

3.3.2.4.9.1.10.6.8 LIMITATIONS

None.

3.3.2.4.9.1.10.7 DCM_FROM_QUATERNION UNIT DESIGN (CATALOG #P295-0)

This part computes the direction cosine matrix that transforms data from coordinate frame xyz to XYZ given the unit quaternion that rotates the XYZ frame into the xyz frame.

3.3.2.4.9.1.10.7.1 REQUIREMENTS ALLOCATION

N/A

3.3.2.4.9.1.10.7.2 LOCAL ENTITIES DESIGN

None.

3.3.2.4.9.1.10.7.3 INPUT/OUTPUT

GENERIC PARAMETERS:

Data types:

The following table describes the generic formal types required by this part:

Name	Type	Description
Row_Indices	discrete type	Data type which translates into the row indexing of the direction cosine matrix.
Col_Indices	discrete type	Data type which translates into the column indexing of the direction cosine matrix.
Sin_Cos_Ratio	floating point	Type which translates into the type of the elements of the direction cosine matrix.
Direction_Cosine_Matrix	3X3 matrix	Type representing a direction cosine matrix indexed by the column and row types described above with elements of type Sin_Cos_Ratio.
Quaternion_Indices	discrete type	Data type which translates into the indexing elements which index the quaternion.
Quaternion_Vector	1X4 matrix	Type representing the unit quaternion which rotates the XYZ frame into the xyz frame.

Data objects:

The following table describes the generic formal objects required by this part:

Name	Type	Value	Description
Row1	Row_Indices	'FIRST	Object to be used as an index representing the first row of the direction cosine matrix. This object indexes the direction cosines of the angles between the x-axis and the XYZ axes.
Row2	Row_Indices	SUCC 'FIRST	Object to be used as an index representing the second row of the direction cosine matrix. This object indexes the direction cosines of the angles between the y-axis and the XYZ axes.
Row3	Row_Indices	'LAST	Object to be used as an index representing the third row of the direction cosine matrix. This object indexes the direction cosines of the angles between the z-axis and the XYZ axes.
Col1	Col_Indices	'FIRST	Object to be used as an index representing the first column of the direction cosine matrix. This object indexes the direction cosines of the angles between the X-axis and the xyz axes.
Col2	Col_Indices	SUCC 'FIRST	Object to be used as an index representing the second column of the direction cosine matrix. This object indexes the direction cosines of the angles between the Y-axis and the xyz axes.
Col3	Col_Indices	'LAST	Object to be used as an index representing the third column of the direction cosine matrix. This object indexes the direction cosines of the angles between the Z-axis and the xyz axes.
Q0	Quaternion _Indices	'FIRST	Object to be used as an index representing the first element of the quaternion.
Q1	Quaternion _Indices	SUCC 'FIRST	Object to be used as an index representing the second element of the quaternion.
Q2	Quaternion _Indices	PRED 'LAST	Object to be used as an index representing the third element of the quaternion.
Q3	Quaternion _Indices	'LAST	Object to be used as an index representing the fourth element of the quaternion.

FORMAL PARAMETERS:

The following table describes this part's formal parameters:

Name	Type	Mode	Description
Quaternion	Quaternion_Vectors	In	This object represents the unit quaternion which rotates XYZ into xyz.

3.3.2.4.9.1.10.7.4 LOCAL DATA

Data objects:

The following table describes the data objects maintained by this part:

Name	Type	Value	Description
Q0_Sqr	Sin_Cos_Ratio	N.A.	This object temporarily hold the square of quaternion element q0.
Q1_Sqr	Sin_Cos_Ratio	N.A.	This object temporarily hold the square of quaternion element q1.
Q2_Sqr	Sin_Cos_Ratio	N.A.	This object temporarily hold the square of quaternion element q2.
Q3_Sqr	Sin_Cos_Ratio	N.A.	This object temporarily hold the square of quaternion element q3.
DCM	Direction_Cosine_Matrix	N.A.	This object is the direction cosine matrix that is computed and returned.

3.3.2.4.9.1.10.7.5 PROCESS CONTROL

Not applicable.

3.3.2.4.9.1.10.7.6 PROCESSING

The following describes the processing performed by this part:

```
separate (Direction_Cosine_Matrix_Operations.DCM_General_Operations)
function DCM_From_Quaternion
            (Quaternion : Quaternion_Vectors)
            return Direction_Cosine_Matrix is

-- ---------------------
-- --declaration section
-- ---------------------

-- ---<objects to hold the squares of quaternions>---

   Q0_Sqr  : Sin_Cos_Ratio;
   Q1_Sqr  : Sin_Cos_Ratio;
```

```
   Q2_Sqr  : Sin_Cos_Ratio;
   Q3_Sqr  : Sin_Cos_Ratio;

-- ---<the DCM to be returned>---

   DCM     : Direction_Cosine_Matrix;

----------------
--begin function
----------------

begin

-- ----- <compute squares of quaternions> ------

   Q0_Sqr := Quaternion(q0) * Quaternion(q0);
   Q1_Sqr := Quaternion(q1) * Quaternion(q1);
   Q2_Sqr := Quaternion(q2) * Quaternion(q2);
   Q3_Sqr := Quaternion(q3) * Quaternion(q3);


-- ----- <compute direction cosines> ---------

   DCM(Row1,Col1) := Q0_Sqr + Q1_Sqr - Q2_Sqr - Q3_Sqr;
   DCM(Row2,Col2) := Q0_Sqr - Q1_Sqr + Q2_Sqr - Q3_Sqr;
   DCM(Row3,Col3) := Q0_Sqr - Q1_Sqr - Q2_Sqr + Q3_Sqr;

   DCM(Row2,Col1) := ( (Quaternion(q0) * Quaternion(q3))
                      +(Quaternion(q1) * Quaternion(q2)) )
                     * 2.0;

   DCM(Row3,Col1) := ( (Quaternion(q1) * Quaternion(q3))
                      -(Quaternion(q0) * Quaternion(q2)) )
                     * 2.0;

   DCM(Row1,Col2) := ( (Quaternion(q1) * Quaternion(q2))
                      -(Quaternion(q0) * Quaternion(q3)) )
                     * 2.0;

   DCM(Row3,Col2) := ( (Quaternion(q0) * Quaternion(q1))
                      +(Quaternion(q2) * Quaternion(q3)) )
                     * 2.0;

   DCM(Row1,Col3) := ( (Quaternion(q0) * Quaternion(q2))
                      +(Quaternion(q1) * Quaternion(q3)) )
                     * 2.0;

   DCM(Row2,Col3) := ( (Quaternion(q2) * Quaternion(q3))
                      -(Quaternion(q0) * Quaternion(q1)) )
                     * 2.0;

   return DCM;

end DCM_From_Quaternion;
```

3.3.2.4.9.1.10.7.7 UTILIZATION OF OTHER ELEMENTS

None.

3.3.2.4.9.1.10.7.8 LIMITATIONS

None.

3.3.2.4.9.2 CNE_OPERATIONS PACKAGE DESIGN (CATALOG #P296-0)

This package contains all functions and procedures that specifically apply to a CNE direction cosine matrix. A CNE DCM is a 3X3 matrix that is used to transform data between the Navigation coordinate frame and the Earth coordinate frame. The Navigation frame assumes the convention "East, North, Up", and the Earth frame assumes the convention "Greenwich, Right, Polar". The elements of this matrix are direction cosines of angles between specific axes of the two coordinate frames.

The decomposition for this part is the same as that shown in the Top-Level Design Document.

3.3.2.4.9.2.1 REQUIREMENTS ALLOCATION

The following table summarizes the allocation of CAMP requirements to this part:

Name	Requirements Allocation
CNE_Initialized_From_Reference	N/A
Reorthonormalize_CNE	N/A
Compute_First_Row_of_CNE_From_Orthonormal	N/A
CNE_Initialized_From_Earth_Position	R32
Perform_Trapezoidal_Integration_of_CNE	N/A
Perform_Rectangular_Integration_of_CNE	N/A
Frame_Misalignment_of_CNE	N/A
Aligned_CNE_Matrix	N/A
CNE_From_Quaternion	N/A

3.3.2.4.9.2.2 LOCAL ENTITIES DESIGN

Subprograms:

The following table describes the subprograms in this part. The first two subprograms are instantiations of parts from DCM_General_Operations.

Name	Type	Description
CNE_Initialized_From_Reference	function	This function initializes a CNE matrix to a reference CNE matrix.
Reorthonormalize_CNE	procedure	This procedure reorthonormalizes a CNE matrix.
Compute_First_Row_of_CNE_From_Orthonormal	procedure	This procedure computes the first row of a CNE matix using the orthonormal property.
CNE_Initialized_From_Earth_Position	generic function	This function initializes the CNE matrix from the latitude, longitude, and wander angle of the missile location.

3.3.2.4.9.2.3 INPUT/OUTPUT

GENERIC PARAMETERS:

Data types:

The following table describes the generic formal types required by this part:

Name	Type	Description
Earth_Axes	discrete type	Data type which translates into the row indexing of the direction cosine matrix. This type should be equivalent (conceptually) to "Greenw, Right, Polar".
Navigation_Axes	discrete type	Data type which translates into the column indexing of the direction cosine matrix. This type should be equivalent (conceptually) to "East, North, Up".
Sin_Cos_Ratio	floating point	Type which translates into the type of the elements of the direction cosine matrix.
CNE_Matrix	3X3 matrix	Type representing a direction cosine matrix indexed by the column and row types described above with elements of type Sin_Cos_Ratio.

Data objects:

The following table describes the generic formal objects required by this part:

Name	Type	Value	Description
Greenw	Earth_Axes	'FIRST	Object to be used as an index representing the first row of the direction cosine matrix. This object indexes the direction cosines of the angles between the x-axis and the XYZ axes.
Right	Earth_Axes	SUCC 'FIRST	Object to be used as an index representing the second row of the direction cosine matrix. This object indexes the direction cosines of the angles between the y-axis and the XYZ axes.
Polar	Earth_Axes	'LAST	Object to be used as an index representing the third row of the direction cosine matrix. This object indexes the direction cosines of the angles between the z-axis and the XYZ axes.
East	Navigation _Axes	'FIRST	Object to be used as an index representing the first column of the direction cosine matrix. This object indexes the direction cosines of the angles between the X-axis and the xyz axes.
North	Navigation _Axes	SUCC 'FIRST	Object to be used as an index representing the second column of the direction cosine matrix. This object indexes the direction cosines of the angles between the Y-axis and the xyz axes.
Up	Navigation _Axes	'LAST	Object to be used as an index representing the third column of the direction cosine matrix. This object indexes the direction cosines of the angles between the Z-axis and the xyz axes.

FORMAL PARAMETERS:

The following table describes subprogram CNE_Initialized_From_Reference's formal parameters:

Name	Type	Mode	Description
REF_CNE_2_1	Sin_Cos_Ratio	In	This value is the reference direction cosine element located at RIGHT, EAST. It represents the direction cosine of the angle between the y-axis and the X-axis.
REF_CNE_2_2	Sin_Cos_Ratio	In	This value is the reference direction cosine element located at RIGHT, NORTH. It represents the direction cosine of the angle between the y-axis and the Y-axis.
REF_CNE_3_1	Sin_Cos_Ratio	In	This value is the reference direction cosine element located at POLAR, EAST. It represents the direction cosine of the angle between the z-axis and the X-axis.
REF_CNE_3_2	Sin_Cos_Ratio	In	This value is the reference direction cosine element located at POLAR, NORTH. It represents the direction cosine of the angle between the z-axis and the Y-axis.
Sign_Of_2_3	INTEGER	In	This value represents the sign of the reference direction cosine element located at RIGHT, column 3. This value is the sign of the direction cosine of the angle between the y-axis and the Z-axis. The value must equal either -1 or 1, which represents minus or plus, respectively.
Sign_Of_3_3	INTEGER	In	This value represents the sign of the reference direction cosine element located at POLAR, column 3. This value is the sign of the direction cosine of the angle between the z-axis and the Z-axis. The value must equal either -1 or 1, which represents minus or plus, respectively.

The following table describes subprogram Reorthonormalize_CNE's formal parameters:

Name	Type	Mode	Description
CNE	CNE_Matrices	in	This parameter is the direction cosine matrix to be reorthonormalized.

3.3.2.4.9.2.4 LOCAL DATA

Data objects:

The following table describes the data objects maintained by this part:

Name	Type	Value	Description
CNE_Matrices	3X3	N.A.	This type is a type representing a DCM derived from the coordinate frames of the Earth and Navigation frames. The earth frame axes are assumed to be "Greenwich, Right, Polar", and the navigation frame axes are assumed to be "North, East, Up".

3.3.2.4.9.2.5 PROCESS CONTROL

Not applicable.

3.3.2.4.9.2.6 PROCESSING

The following describes the processing performed by this part:

```
separate (Direction_Cosine_Matrix_Operations)
package body CNE_Operations is

   package DCM_GEN_PKG renames
                  Direction_Cosine_Matrix_Operations.DCM_General_Operations;

-- ------------------------------------------------------------------
-- --<instantiation of a units from "DCM_General_Operations" with CNE
-- -- specific parameters applied to the generics>
-- ------------------------------------------------------------------

   function Initialize_From_Ref is new
               DCM_GEN_PKG.DCM_Initialized_From_Reference
                  (Row_Indices            => Earth_Axes,
                   Col_Indices            => Navigation_Axes,
                   Sin_Cos_Ratio          => Sin_Cos_Ratio,
                   Direction_Cosine_Matrix => CNE_Matrices);

   procedure Reorthonormalize is new
                DCM_GEN_PKG.Reorthonormalize_DCM
                   (Row_Indices            => Earth_Axes,
                    Col_Indices            => Navigation_Axes,
                    Sin_Cos_Ratio          => Sin_Cos_Ratio,
                    Direction_Cosine_Matrix => CNE_Matrices,
                    Real                   => Real);

   procedure Compute_First_Row is new
                DCM_GEN_PKG.Compute_First_Row_From_Orthonormal
                   (Row_Indices            => Earth_Axes,
```

```
                    Col_Indices             => Navigation_Axes,
                    Sin_Cos_Ratio           => Sin_Cos_Ratio,
                    Direction_Cosine_Matrix => CNE_Matrices);

-- ----------------
-- --separate parts
-- ----------------

   function CNE_Initialized_From_Earth_Position
                           (Wander_Angle : Angles;
                            Latitude     : Earth_Position;
                            Longitude    : Earth_Position)
                            return CNE_Matrices is separate;

   package body CNE_Integration is separate;

   package body Alignment_Parts is separate;

   package body CNE_From_Quaternion is separate;

-- ------------------------------------------------------
-- --routines acting as interfaces to instantiated routines
-- ------------------------------------------------------

   function CNE_Initialized_From_Reference
               (Ref_CNE_2_1    : Sin_Cos_Ratio;
                Ref_CNE_2_2    : Sin_Cos_Ratio;
                Ref_CNE_3_1    : Sin_Cos_Ratio;
                Ref_CNE_3_2    : Sin_Cos_Ratio;
                Sign_Of_2_3    : INTEGER;
                Sign_Of_3_3    : INTEGER)
               return CNE_Matrices is
   begin
      return Initialize_From_Ref
                (Ref_DCM_2_1 => Ref_CNE_2_1,
                 Ref_DCM_2_2 => Ref_CNE_2_2,
                 Ref_DCM_3_1 => Ref_CNE_3_1,
                 Ref_DCM_3_2 => Ref_CNE_3_2,
                 Sign_Of_2_3 => Sign_Of_2_3,
                 Sign_Of_3_3 => Sign_Of_3_3);
   end CNE_Initialized_From_Reference;

   procedure Reorthonormalize_CNE (CNE : in out CNE_Matrices) is
   begin
      Reorthonormalize (CNE);
   end Reorthonormalize_CNE;

   procedure Compute_First_Row_of_CNE_From_Orthonormal
               (CNE : in out CNE_Matrices) is
   begin
      Compute_First_Row (CNE);
   end Compute_First_Row_of_CNE_From_Orthonormal;

end CNE_Operations;
```

3.3.2.4.9.2.7 UTILIZATION OF OTHER ELEMENTS

None.

3.3.2.4.9.2.8 LIMITATIONS

None.

3.3.2.4.9.2.9 LLCSC DESIGN

3.3.2.4.9.2.9.1 CNE_INTEGRATION PACKAGE DESIGN (CATALOG #P298-0)

This generic package contains procedures that integrate a CNE direction cosine matrix. The first two procedures initialize and integrate a CNE using trapezoidal integration. The last procedure integrates a CNE using rectangular integration. Note that the first row is not computed in the integration. It must be computed by another function.

The decomposition for this part is the same as that shown in the Top-Level Design Document.

3.3.2.4.9.2.9.1.1 REQUIREMENTS ALLOCATION

The following table summarizes the allocation of CAMP requirements to this part:

Name	Requirements Allocation
Perform_Trapezoidal_Integration_of_CNE	N/A
Perform_Rectangular_Integration_of_CNE	N/A

3.3.2.4.9.2.9.1.2 LOCAL ENTITIES DESIGN

Packages:

For "Perform_Trapezoidal_Integration_of_CNE", the package "Integrate_Trap" is created and maintained.

Subprograms:

For "Perform_Rectangular_Integration_of_CNE", the procedure "Integrate_Rect" is created and maintained.

3.3.2.4.9.2.9.1.3 INPUT/OUTPUT

GENERIC PARAMETERS:

Data types:

The following table describes the generic formal types required by this part:

Name	Type	Description
Angular_Velocities	floating point	Data type that translates into the type of a component of the angular velocity in a given reference frame.
Time_Intervals	floating point	Data type that translates into the type of an object that represents a time interval (delta time).

Data objects:

The following table describes the generic formal objects required by this part:

Name	Type	Description
Initial_East_Rho	Angular_Velocities	This object can be used to initialize the East component of angular velocity (for trapezoidal integration).
Initial_North_Rho	Angular_Velocities	This object can be used to initialize the North component of angular velocity (for trapezoidal integration).

Subprograms:

The following table describes the generic formal subroutines required by this part:

Name	Type	Description
"*"	function	This function computes the product of an object of type Angular_Velocity and an object of type Time_Intervals producing an object of type Sin_Cos_Ratio.
"*"	function	This function computes the product of an object of type Angular_Velocity and an object of type Sin_Cos_Ratio producing an object of type Angular_Velocities.

FORMAL PARAMETERS:

The following table describes procedure "Perform_Trapezoidal_Integration_of_CNE"'s formal parameters:

Name	Type	Mode	Description
CNE	CNE_Matrices	In Out	This parameter is the direction cosine matrix that is integrated.
East_Rho	Angular_Velocities	In	This value is the East component of angular velocity.
North_Rho	Angular_Velocities	In	This value is the North component of angular velocity.
Delta_Time	Time_Intervals	In	This value is the time interval which equals the time since the last integration.

The following table describes procedure "Reinit_Ang_Vel_For_Trapez_Integ_of_-CNE"'s formal parameters:

Name	Type	Mode	Description
East_Rho	Angular_Velocities	In	This value is the East component of angular velocity to be used for initialization.
North_Rho	Angular_Velocities	In	This value is the North component of angular velocity to be used for initialization.

The following table describes procedure "Perform_Rectangular_Integration_of_-CNE"'s formal parameters:

Name	Type	Mode	Description
CNE	CNE_Matrices	In Out	This parameter is the direction cosine matrix that is integrated.
East_Rho	Angular_Velocities	In	This value is the East component of angular velocity.
North_Rho	Angular_Velocities	In	This value is the North component of angular velocity.
Delta_Time	Time_Intervals	In	This value is the time interval which equals the time since the last integration.

3.3.2.4.9.2.9.1.4 LOCAL DATA

None.

3.3.2.4.9.2.9.1.5 PROCESS CONTROL

Not applicable.

3.3.2.4.9.2.9.1.6 PROCESSING

The following describes the processing performed by this part:

```
separate (Direction_Cosine_Matrix_Operations.CNE_Operations)
package body CNE_Integration is

-- ----------------------
-- --local instantiations
-- ----------------------

-- --<instantiation of a units from "DCM_General_Operations" with CNE
-- -- specific parameters applied to the generics>---

   package Integrate_Trap is new
             DCM_GEN_PKG.DCM_Trapezoidal_Integration
               (Angular_Velocities      => Angular_Velocities,
                Time_Intervals          => Time_Intervals,
                Row_Indices             => Earth_Axes,
                Col_Indices             => Navigation_Axes,
                Sin_Cos_Ratio           => Sin_Cos_Ratio,
                Direction_Cosine_Matrix => CNE_Matrices,
                Initial_X_Rho           => Initial_East_Rho,
                Initial_Y_Rho           => Initial_North_Rho);

   procedure Integrate_Rect is new
             DCM_GEN_PKG.Perform_Rectangular_Integration_of_DCM
               (Angular_Velocities      => Angular_Velocities,
                Time_Intervals          => Time_Intervals,
                Row_Indices             => Earth_Axes,
                Col_Indices             => Navigation_Axes,
                Sin_Cos_Ratio           => Sin_Cos_Ratio,
                Direction_Cosine_Matrix => CNE_Matrices);

-- ------------------------------------------------------
-- --routines acting as interfaces to instantiated routines
-- ------------------ -----------------------------------

-- -------[Trapezoidal Integration]---------

   procedure Perform_Trapezoidal_Integration_of_CNE
               (CNE            : in out CNE_Matrices;
                East_Rho       : in     Angular_Velocities;
                North_Rho      : in     Angular_Velocities;
                Delta_Time     : in     Time_Intervals) is
   begin
      Integrate_Trap.Perform_Trapezoidal_Integration_of_DCM
```

```
                              (CNE, East_Rho, North_Rho, Delta_Time);
   end Perform_Trapezoidal_Integration_of_CNE;


   procedure Reinit_Ang_Vel_for_Trapez_Integ_of_CNE
               (East_Rho     : in     Angular_Velocities;
                North_Rho    : in     Angular_Velocities) is
   begin
      Integrate_Trap.Reinitialize_Angular_Velocities
                              (East_Rho, North_Rho);
   end Reinit_Ang_Vel_for_Trapez_Integ_of_CNE;


-- -------[Rectangular Integration]---------

   procedure Perform_Rectangular_Integration_of_CNE
             (CNE        : in out CNE_Matrices;
              East_Rho   : in     Angular_Velocities;
              North_Rho  : in     Angular_Velocities;
              Delta_Time : in     Time_Intervals) is
   begin
      Integrate_Rect (CNE, East_Rho, North_Rho, Delta_Time);
   end Perform_Rectangular_Integration_of_CNE;


end CNE_Integration;
```

3.3.2.4.9.2.9.1.7 UTILIZATION OF OTHER ELEMENTS

UTILIZATION OF OTHER ELEMENTS IN TOP LEVEL COMPONENT:

The following tables describe the elements used by this part but defined elsewhere in the parent top level component:

Packages:

Name	Type	Source	Description
DCM_General_Operations	package	Direction_Cosine_Matrix_Operations	This package contains general operations for direction cosine matrices. CNE_Integration instantiates functions and procedures from this package.
DCM_Trapezoidal_Integration_of_DCM	package	DCM_General_Operations	This package contains 2 procedures; the first reinitializes the angular velocities for trapezoidal integration, and the second performs the integration.

Data types:

The following table summarizes the types required by this part and defined elsewhere in the parent top level component:

Name	Type	Source	Description
CNE_Matrices	3X3 matrix	CNE_Operations	Type representing a direction cosine matrix indexed by the column and row types representing the navigation and earth frame.

3.3.2.4.9.2.9.1.8 LIMITATIONS

None.

3.3.2.4.9.2.9.1.9 LLCSC DESIGN

None.

3.3.2.4.9.2.9.1.10 UNIT DESIGN

None.

3.3.2.4.9.2.9.2 ALIGNMENT_PARTS PACKAGE DESIGN (CATALOG #P299-0)

This package contains two functions. The first function computes the rotation angles between the CNE and a reference direction cosine matrix. The second function rotates the CNE matrix into alignment by using the rotation angles. The second function only computes the second and third row of the CNE matrix. The first row must be updated separately.

The decomposition for this part is the same as that shown in the Top-Level Design Document.

3.3.2.4.9.2.9.2.1 REQUIREMENTS ALLOCATION

The following table summarizes the allocation of CAMP requirements to this part:

Name	Requirements Allocation
Frame_Misalignment_of_CNE	N/A
Aligned_CNE_Matrix	N/A

3.3.2.4.9.2.9.2.2 LOCAL ENTITIES DESIGN

Packages:

For "Perform_Trapezoidal_Integration_of_CNE", the package "Integrate_Trap" is created and maintained.

Subprograms:

For "Frame_Misalignment_of_CNE", the function "Computed _Frame_Misalignment" is created and maintained. For "Aligned_CNE_Matrix", the function "Align_CNE_-Matrix" is created and maintained.

3.3.2.4.9.2.9.2.3 INPUT/OUTPUT

GENERIC PARAMETERS:

Data types:

The following table summarizes the generic formal types required by this part:

Name	Type	Description
Rotation _Indices	discrete type	Data type which translates into the indexing elements of a rotation angle vector.
Rotation _Angles	floating point	Type which translates into the type of the elements of a rotation angle vector.
Rotation _Angle _Vec	1X3 matrix (vector)	Type representing a rotation angle vector.

Data objects:

The following table describes the generic formal objects required by this part:

Name	Type	Value	Description
X	Rotation _Indices	'FIRST	Object to be used as an index representing the first element of the rotation angle vector.
Y	Rotation _Indices	SUCC 'FIRST	Object to be used as an index representing the second element of the rotation angle vector.
Z	Rotation _Indices	'LAST	Object to be used as an index representing the third element of the rotation angle vector.

Subprograms:

The following table describes the generic formal subroutines required by this part:

Name	Type	Description
"*"	function	This function computes the product of an object of type Sin_Cos_Ratio and an object of type Rotation_Angles producing an object of type Sin_Cos_Ratio.
"*"	function	This function computes the product of an object of type Sin_Cos_Ratio and an object of type Sin_Cos_Ratio producing an object of type Rotation_Angles.

FORMAL PARAMETERS:

The following table describes function "Frame_Misalignment_of_CNE"'s formal parameters:

Name	Type	Mode	Description
CNE	CNE_Matrices	In	This parameter is the active direction cosine matrix (of the missile).
REF_DC_Matrix	CNE_Matrices	In	This parameter is the reference direction cosine matrix.

The following table describes function "Aligned_CNE_Matrix"'s formal parameters:

Name	Type	Mode	Description
CNE	CNE_Matrices	In	This parameter is the active direction cosine matrix (of the missile).
Rotation_Angle	Rotation_Angle_Vec	In	This parameter is the rotation angles to be used to align the DCM.

3.3.2.4.9.2.9.2.4 LOCAL DATA

None.

3.3.2.4.9.2.9.2.5 PROCESS CONTROL

Not applicable.

3.3.2.4.9.2.9.2.6 PROCESSING

The following describes the processing performed by this part:

```
separate (Direction_Cosine_Matrix_Operations.CNE_Operations)
package body Alignment_Parts is

-- ----------------------
-- --local instantiations
-- ----------------------

---<instantiation of a unit from "DCM_General_Operations" with CNE
--  specific parameters applied to the generics>---

   function Computed_Frame_Misalignment is new
            DCM_GEN_PKG.Frame_Misalignment
               (Row_Indices             => Earth_Axes,
                Col_Indices             => Navigation_Axes,
                Sin_Cos_Ratio           => Sin_Cos_Ratio,
                Direction_Cosine_Matrix => CNE_Matrices,
                Rotation_Indices        => Rotation_Indices,
                Rotation_Angles         => Rotation_Angles,
                Rotation_Angle_Vec      => Rotation_Angle_Vec);

   function Align_CNE_Matrix is new
             DCM_GEN_PKG.Aligned_DCM_Matrix
               (Row_Indices             => Earth_Axes,
                Col_Indices             => Navigation_Axes,
                Sin_Cos_Ratio           => Sin_Cos_Ratio,
                Direction_Cosine_Matrix => CNE_Matrices,
                Rotation_Indices        => Rotation_Indices,
                Rotation_Angles         => Rotation_Angles,
                Rotation_Angle_Vec      => Rotation_Angle_Vec);

-- ------------------------------------------------------
-- --routines acting as interfaces to instantiated routines
-- ------------------------------------------------------

-- -------[Frame Misalignment]----------

   function Frame_Misalignment_of_CNE
               (CNE            : CNE_Matrices;
                REF_DC_Matrix  : CNE_Matrices)
               return Rotation_Angle_Vec is
   begin
      return Computed_Frame_Misalignment (CNE, REF_DC_Matrix);
   end Frame_Misalignment_of_CNE;

-- -------[Align CNE Matrix]----------

   function Aligned_CNE_Matrix
               (CNE            : CNE_Matrices;
                Rotation_Angle : Rotation_Angle_Vec)
               return CNE_Matrices is
   begin
      return Align_CNE_Matrix (CNE, Rotation_Angle);
   end Aligned_CNE_Matrix;
```

```
end Alignment_Parts;
```

3.3.2.4.9.2.9.2.7 UTILIZATION OF OTHER ELEMENTS

UTILIZATION OF OTHER ELEMENTS IN TOP LEVEL COMPONENT:

The following tables describe the elements used by this part but defined elsewhere in the parent top level component:

Packages:

The following table summarizes the external packages required by this part:

Name	Type	Source	Description
DCM_General_Operations	package	Direction_Cosine_Matrix_Operations	This package contains general operations for direction cosine matrices. Alignment_Parts instantiates functions and procedures from this package.

Data types:

The following table summarizes the types required by this part and defined elsewhere in the parent top level component:

Name	Type	Source	Description
CNE_Matrices	3X3 matrix	CNE_Operations	Type representing a direction cosine matrix indexed by the column and row types representing the navigation and earth frame.

3.3.2.4.9.2.9.2.8 LIMITATIONS

None.

3.3.2.4.9.2.9.2.9 LLCSC DESIGN

None.

3.3.2.4.9.2.9.2.10 UNIT DESIGN

None.

3.3.2.4.9.2.9.3 CNE_FROM_QUATERNION PACKAGE DESIGN (CATALOG #P300-0)

This package contains a function which initializes the CNE matrix using a quaternion.

The decomposition for this part is the same as that shown in the Top-Level Design Document.

3.3.2.4.9.2.9.3.1 REQUIREMENTS ALLOCATION

The following table summarizes the allocation of CAMP requirements to this part:

Name	Requirements Allocation
CNE_From_Quaternion	N/A

3.3.2.4.9.2.9.3.2 LOCAL ENTITIES DESIGN

Subprograms:

The function "Computed_CNE" is created and maintained by this part.

3.3.2.4.9.2.9.3.3 INPUT/OUTPUT

GENERIC PARAMETERS:

Data types:

The following table summarizes the generic formal types required by this part:

Name	Type	Description
Quaternion_Indices	discrete type	Data type which translates into the indexing elements which index the quaternion.
Quaternion_Vector	1X4 matrix	Type representing the unit quaternion which rotates the XYZ frame into the xyz frame.

Data objects:

The following table summarizes the generic formal objects required by this part:

Name	Type	Value	Description
Q0	Quaternion_Indices	'FIRST	Object to be used as an index representing the first element of the quaternion.
Q1	Quaternion_Indices	SUCC('FIRST)	Object to be used as an index representing the second element of the quaternion.
Q2	Quaternion_Indices	PRED('LAST)	Object to be used as an index representing the third element of the quaternion.
Q3	Quaternion_Indices	'LAST	Object to be used as an index representing the fourth element of the quaternion.

FORMAL PARAMETERS:

The following describes the formal parameters for the function "Compute_CNE":

Name	Type	Mode	Description
Quaternion	Quaternion_Vectors	In	This object represents the unit quaternion which rotates XYZ into xyz.

3.3.2.4.9.2.9.3.4 LOCAL DATA

None.

3.3.2.4.9.2.9.3.5 PROCESS CONTROL

Not applicable.

3.3.2.4.9.2.9.3.6 PROCESSING

The following describes the processing performed by this part:

```
separate (Direction_Cosine_Matrix_Operations.CNE_Operations)
package body CNE_From_Quaternion is

-- ----------------------
-- --local instantiations
-- ----------------------

-- --<instantiation of a unit from "DCM_General_Operations" with CNE
-- -- specific parameters applied to the generics>---

   function Computed_CNE is new
            DCM_GEN_PKG.DCM_From_Quaternion
              (Row_Indices            => Earth_Axes,
               Col_Indices            => Navigation_Axes,
               Sin_Cos_Ratio          => Sin_Cos_Ratio,
```

```
                Direction_Cosine_Matrix => CNE_Matrices,
                Quaternion_Indices      => Quaternion_Indices,
                Quaternion_Vectors      => Quaternion_Vectors);

-- -----------------
-- --function bodies
-- -----------------

-- ---<the function to call the above function>---

   function Compute_CNE
            (quaternion : Quaternion_Vectors)
            return CNE_Matrices is
   begin
       return Computed_CNE (quaternion);
   end Compute_CNE;

end CNE_From_Quaternion;
```

3.3.2.4.9.2.9.3.7 UTILIZATION OF OTHER ELEMENTS

UTILIZATION OF OTHER ELEMENTS IN TOP LEVEL COMPONENT:

The following tables describe the elements used by this part but defined elsewhere in the parent top level component:

Packages:

The following table summarizes the external packages required by this part:

Name	Type	Source	Description
DCM_General_Operations	package	Direction_Cosine_Matrix_Operations	This package contains general operations for direction cosine matrices. CNE_From_Quaternion instantiates functions and procedures from this package.

Data types:

The following table summarizes the types required by this part and defined elsewhere in the parent top level component:

Name	Type	Source	Description
CNE_Matrices	3X3 matrix	CNE_Operations	Type representing a direction cosine matrix indexed by the column and row types representing the navigation and earth frame.

3.3.2.4.9.2.9.3.8 LIMITATIONS

None.

3.3.2.4.9.2.9.3.9 LLCSC DESIGN

None.

3.3.2.4.9.2.9.3.10 UNIT DESIGN

None.

3.3.2.4.9.2.10 UNIT DESIGN

3.3.2.4.9.2.10.1 CNE_INITIALIZED_FROM_EARTH_POSITION UNIT DESIGN (CATALOG #P297-0)

This generic function initializes a CNE matrix from the earth position. Earth position is defined through the wander angle, latitude, and longitude.

3.3.2.4.9.2.10.1.1 REQUIREMENTS ALLOCATION

N/A

3.3.2.4.9.2.10.1.2 LOCAL ENTITIES DESIGN

None.

3.3.2.4.9.2.10.1.3 INPUT/OUTPUT

GENERIC PARAMETERS:

Data types:

The following table describes the generic formal types required by this part:

Name	Type	Description
Earth_Position	floating point	Type which translates into the type of lattitude and longitude.
Angles	floating point	Type which translates into the type of the wander angle.

Subprograms:

The following table describes the generic formal subroutines required by this part:

Name	Type	Description
Sin_Cos	procedure	This procedure computes the sine and cosine of an object of type Angles. The returned values are of type Sin_Cos_Ratio.
Sin_Cos	procedure	This procedure computes the sine and cosine of an object of type Earth_Position. The returned values are of type Sin_Cos_Ratio.

FORMAL PARAMETERS:

The following table describes this part's formal parameters:

Name	Type	Mode	Description
Wander_Angle	Angles	In	This object represents the wander angle which is the angle between the East axis of the navigation frame and the current missile direction of motion.
Latitude	Earth_Position	In	This object represents the current latitude of the missile's earth location.
Longitude	Earth_Position	In	This object represents the current longitude of the missile's earth location.

3.3.2.4.9.2.10.1.4 LOCAL DATA

Data objects:

The following table describes the data objects maintained by this part:

Name	Type	Value	Description
Sin_Lat	Sin_Cos_Ratio	sine of latitude	This object stores the value of the sine of earth latitude.
Cos_Lat	Sin_Cos_Ratio	cosine of latitude	This object stores the value of the cosine of earth latitude.
Sin_Long	Sin_Cos_Ratio	sine of longitude	This object stores the value of the sine of earth longitude.
Cos_Long	Sin_Cos_Ratio	sine of longitude	This object stores the value of the cosine of earth longitude.
Sin_Wa	Sin_Cos_Ratio	sine of wander angle	This object stores the value of the sine of the wander angle.
Cos_Wa	Sin_Cos_Ratio	cosine of wander angle	This object stores the value of the cosine of the wander angle.
CNE	CNE_Matrices	N.A.	This object is a 3X3 direction cosine matrix. This object is computed and returned by this function.

3.3.2.4.9.2.10.1.5 PROCESS CONTROL

Not applicable.

3.3.2.4.9.2.10.1.6 PROCESSING

The following describes the processing performed by this part:

```
separate (Direction_Cosine_Matrix_Operations.CNE_Operations)
function CNE_Initialized_From_Earth_Position
            (Wander_Angle : Angles;
             Latitude     : Earth_Position;
             Longitude    : Earth_Position)
            return CNE_Matrices is

-- ---------------------
-- --declaration section
-- ---------------------

-- ---<objects to hold the sine and cosine of latitude, longitude,
-- --  and wander angle>---

   Sin_Lat    : Sin_Cos_Ratio;
   Cos_Lat    : Sin_Cos_Ratio;
   Sin_Long   : Sin_Cos_Ratio;
   Cos_Long   : Sin_Cos_Ratio;
   Sin_Wa     : Sin_Cos_Ratio;
```

```
   Cos_Wa      : Sin_Cos_Ratio;

-- ---<the CNE to be returned>---

   CNE         : CNE_Matrices;

----------------
--begin function
----------------

begin

-- --<compute sines and cosines>--

   Sin_Cos(Latitude,     Sin_Lat,  Cos_Lat);
   Sin_Cos(Longitude,    Sin_Long, Cos_Long);
   Sin_Cos(Wander_Angle, Sin_Wa,   Cos_Wa);

-- --<compute CNE elements>--

   CNE(Greenw,East)  := -Cos_Wa * Sin_Long - Sin_Wa * Sin_Lat * Cos_Long;
   CNE(Right, East)  :=  Cos_Wa * Cos_Long - Sin_Wa * Sin_Lat * Sin_Long;
   CNE(Polar, East)  :=  Sin_Wa * Cos_Lat;

   CNE(Greenw,North) :=  Sin_Wa * Sin_Long - Cos_Wa * Sin_Lat * Cos_Long;
   CNE(Right, North) := -Sin_Wa * Cos_Long - Cos_Wa * Sin_Lat * Sin_Long;
   CNE(Polar, North) :=  Cos_Wa * Cos_Lat;

   CNE(Greenw,Up)    := Cos_Lat * Cos_Long;
   CNE(Right, Up)    := Cos_Lat * Sin_Long;
   CNE(Polar, Up)    := Sin_Lat;

   return CNE;

end CNE_Initialized_From_Earth_Position;
```

3.3.2.4.9.2.10.1.7 UTILIZATION OF OTHER ELEMENTS

None.

3.3.2.4.9.2.10.1.8 LIMITATIONS

None.

(This page left intentionally blank.)

3.3.2.4.10 UNIT DESIGN

None.

(This page left intentionally blank.)

```
package body Direction_Cosine_Matrix_Operations is

   package body Dcm_General_Operations is separate;

   package body Cne_Operations is separate;

end Direction_Cosine_Matrix_Operations;
```

```
separate (Direction_Cosine_Matrix_Operations)
package body Dcm_General_Operations is

   function Dcm_Initialized_From_Reference
               (Ref_Dcm_2_1    : Sin_Cos_Ratio;
                Ref_Dcm_2_2    : Sin_Cos_Ratio;
                Ref_Dcm_3_1    : Sin_Cos_Ratio;
                Ref_Dcm_3_2    : Sin_Cos_Ratio;
                Sign_Of_2_3    : INTEGER;
                Sign_Of_3_3    : INTEGER)
               return Direction_Cosine_Matrix is separate;

   package body Dcm_Trapezoidal_Integration is separate;

   procedure Perform_Rectangular_Integration_Of_Dcm
               (Dc_Matrix  : in out Direction_Cosine_Matrix;
                X_Rho      : in     Angular_Velocities;
                Y_Rho      : in     Angular_Velocities;
                Delta_Time : in     Time_Intervals) is separate;

   procedure Reorthonormalize_Dcm
               (Dcm_Matrix  : in out Direction_Cosine_Matrix) is separate;

   function Frame_Misalignment
               (Dcm_Matrix     : Direction_Cosine_Matrix;
                Ref_Dc_Matrix  : Direction_Cosine_Matrix)
               return Rotation_Angle_Vec is separate;

   function Aligned_Dcm_Matrix
               (Dcm_Matrix      : Direction_Cosine_Matrix;
                Rotation_Angle  : Rotation_Angle_Vec)
               return Direction_Cosine_Matrix is separate;

   procedure Compute_First_Row_From_Orthonormal
               (Dcm_Matrix      : in out Direction_Cosine_Matrix) is separate;

   function Dcm_From_Quaternion
               (Quaternion : Quaternion_Vectors)
               return Direction_Cosine_Matrix is separate;

end Dcm_General_Operations;
```

```
separate (Direction_Cosine_Matrix_Operations.Dcm_General_Operations)
function Dcm_Initialized_From_Reference
                  (Ref_Dcm_2_1    : Sin_Cos_Ratio;
                   Ref_Dcm_2_2    : Sin_Cos_Ratio;
                   Ref_Dcm_3_1    : Sin_Cos_Ratio;
                   Ref_Dcm_3_2    : Sin_Cos_Ratio;
                   Sign_Of_2_3    : INTEGER;
                   Sign_Of_3_3    : INTEGER)
                  return Direction_Cosine_Matrix is

-- --<direction cosine matrix to be computed>---

   Dcm_Matrix     : Direction_Cosine_Matrix;

begin

-- ---<compute new DCM>---

   if Sign_Of_2_3 = 1 then

      Dcm_Matrix(Row2,Col3) := Sqrt (1.0 - (   Ref_Dcm_2_1 *
                                               Ref_Dcm_2_1
                                           +
                                               Ref_Dcm_2_2 *
                                               Ref_Dcm_2_2   )
                                    );
   else

      Dcm_Matrix(Row2,Col3) := - Sqrt (1.0 - (   Ref_Dcm_2_1 *
                                                 Ref_Dcm_2_1
                                             +
                                                 Ref_Dcm_2_2 *
                                                 Ref_Dcm_2_2   )
                                       );
   end if;


   if Sign_Of_3_3 = 1 then
      Dcm_Matrix(Row3,Col3) := Sqrt (1.0 - (   Ref_Dcm_3_1 *
                                               Ref_Dcm_3_1
                                           +
                                               Ref_Dcm_3_2 *
                                               Ref_Dcm_3_2   )
                                    );
   else
      Dcm_Matrix(Row3,Col3) := - Sqrt (1.0 - (   Ref_Dcm_3_1 *
                                                 Ref_Dcm_3_1
                                             +
                                                 Ref_Dcm_3_2 *
                                                 Ref_Dcm_3_2   )
                                       );
   end if;

   Dcm_Matrix(Row1,Col1) :=    Ref_Dcm_2_2 *
                               Dcm_Matrix(Row3,Col3)
                           -
                               Dcm_Matrix(Row2,Col3) *
```

```
                                Ref_Dcm_3_2;

   Dcm_Matrix(Row1,Col2) :=     Dcm_Matrix(Row2,Col3) *
                                Ref_Dcm_3_1
                             -
                                Ref_Dcm_2_1 *
                                Dcm_Matrix(Row3,Col3);

   Dcm_Matrix(Row1,Col3) :=     Ref_Dcm_2_1 *
                                Ref_Dcm_3_2
                             -
                                Ref_Dcm_2_2 *
                                Ref_Dcm_3_1;

   Dcm_Matrix(Row2,Col1) := Ref_Dcm_2_1;
   Dcm_Matrix(Row2,Col2) := Ref_Dcm_2_2;
   Dcm_Matrix(Row3,Col1) := Ref_Dcm_3_1;
   Dcm_Matrix(Row3,Col2) := Ref_Dcm_3_2;

   return Dcm_Matrix;

end Dcm_Initialized_From_Reference;
```

```
separate (Direction_Cosine_Matrix_Operations.Dcm_General_Operations)
package body Dcm_Trapezoidal_Integration is

   Prev_X_Rho     : Angular_Velocities := Initial_X_Rho;
   Prev_Y_Rho     : Angular_Velocities := Initial_Y_Rho;

   procedure Reinitialize_Angular_Velocities
                  (X_Rho          : in     Angular_Velocities;
                   Y_Rho          : in     Angular_Velocities) is separate;

   procedure Perform_Trapezoidal_Integration_Of_Dcm
               (Dc_Matrix       : in out Direction_Cosine_Matrix;
                X_Rho           : in     Angular_Velocities;
                Y_Rho           : in     Angular_Velocities;
                Delta_Time      : in     Time_Intervals) is separate;

end Dcm_Trapezoidal_Integration;
```

```
separate (Direction_Cosine_Matrix_Operations.Dcm_General_Operations.
          Dcm_Trapezoidal_Integration)
procedure Reinitialize_Angular_Velocities
             (X_Rho           : in     Angular_Velocities;
              Y_Rho           : in     Angular_Velocities) is

begin

   Prev_X_Rho := X_Rho;
   Prev_Y_Rho := Y_Rho;

end Reinitialize_Angular_Velocities;
```

```
separate (Direction_Cosine_Matrix_Operations.Dcm_General_Operations.
          Dcm_Trapezoidal_Integration)
procedure Perform_Trapezoidal_Integration_Of_Dcm
             (Dc_Matrix       : in out Direction_Cosine_Matrix;
              X_Rho           : in     Angular_Velocities;
              Y_Rho           : in     Angular_Velocities;
              Delta_Time      : in     Time_Intervals) is

-- ------------------------------------------
-- ---<objects for internal calculations>---
-- ------------------------------------------

   Prev_Dcm    : Direction_Cosine_Matrix;
   X_Bound_Sum : Angular_Velocities;
   Y_Bound_Sum : Angular_Velocities;
   Del_T_Div_2 : Time_Intervals;

-----------------
--begin procedure
-----------------

begin

-- -----<copy last two rows of DCM into previous DCM before computing>-----

   Prev_Dcm(Row2,Col1) := Dc_Matrix(Row2,Col1);
   Prev_Dcm(Row2,Col2) := Dc_Matrix(Row2,Col2);
   Prev_Dcm(Row2,Col3) := Dc_Matrix(Row2,Col3);

   Prev_Dcm(Row3,Col1) := Dc_Matrix(Row3,Col1);
   Prev_Dcm(Row3,Col2) := Dc_Matrix(Row3,Col2);
   Prev_Dcm(Row3,Col3) := Dc_Matrix(Row3,Col3);

-- ---<compute sum of X and Y bounds from old and new angular velocity>---

   X_Bound_Sum := (X_Rho + Prev_X_Rho);
   Y_Bound_Sum := (Y_Rho + Prev_Y_Rho);

-- ---<compute delta time divided by 2 - for algorithm simplification>--

   Del_T_Div_2 := Delta_Time * Time_Intervals(0.5);

-- ---<compute new DCM>---

   Dc_Matrix(Row3,Col1) := Prev_Dcm(Row3,Col1) -
                           ( Y_Bound_Sum * Prev_Dcm(Row3,Col3)
                             * Del_T_Div_2);

   Dc_Matrix(Row3,Col2) := Prev_Dcm(Row3,Col2) +
                           ( X_Bound_Sum * Prev_Dcm(Row3,Col3)
                             * Del_T_Div_2);

   Dc_Matrix(Row3,Col3) := Prev_Dcm(Row3,Col3) +
                           ( Y_Bound_Sum * Prev_Dcm(Row3,Col1) -
                             X_Bound_Sum * Prev_Dcm(Row3,Col2)   )
                           * Del_T_Div_2;
```

```
   Dc_Matrix(Row2,Col1) := Prev_Dcm(Row2,Col1) -
                           ( Y_Bound_Sum * Prev_Dcm(Row2,Col3)
                             * Del_T_Div_2);

   Dc_Matrix(Row2,Col2) := Prev_Dcm(Row2,Col2) +
                           ( X_Bound_Sum * Prev_Dcm(Row2,Col3)
                             * Del_T_Div_2);

   Dc_Matrix(Row2,Col3) := Prev_Dcm(Row2,Col3) +
                           ( Y_Bound_Sum * Prev_Dcm(Row2,Col1) -
                             X_Bound_Sum * Prev_Dcm(Row2,Col2)   )
                           * Del_T_Div_2;

   Prev_X_Rho := X_Rho;
   Prev_Y_Rho := Y_Rho;

end Perform_Trapezoidal_Integration_Of_Dcm;
```

```
separate (Direction_Cosine_Matrix_Operations.Dcm_General_Operations)
procedure Perform_Rectangular_Integration_Of_Dcm
          (Dc_Matrix  : in out Direction_Cosine_Matrix;
           X_Rho      : in     Angular_Velocities;
           Y_Rho      : in     Angular_Velocities;
           Delta_Time : in     Time_Intervals) is

-- ----------------------
-- --declaration section
-- ----------------------

-- ---<DCM for computing increments (temporary) >---

   Dcm_Incr                 : Direction_Cosine_Matrix;

-- ---< objects for internal calculation >---

   Y_Rho_Times_Delta_Time : Sin_Cos_Ratio := Y_Rho * Delta_Time;
   X_Rho_Times_Delta_Time : Sin_Cos_Ratio := X_Rho * Delta_Time;

------------------
--begin procedure
------------------

begin

-- --<compute increments for bottom two rows>--

   Dcm_Incr(Row2,Col1) := -DC_Matrix(Row2,Col3) * Y_Rho_Times_Delta_Time;
   Dcm_Incr(Row2,Col2) :=  Dc_Matrix(Row2,Col3) * X_Rho_Times_Delta_Time;
   Dcm_Incr(Row2,Col3) :=  Dc_Matrix(Row2,Col1) * Y_Rho_Times_Delta_Time -
                           Dc_Matrix(Row2,Col2) * X_Rho_Times_Delta_Time;
   Dcm_Incr(Row3,Col1) := -DC_Matrix(Row3,Col3) * Y_Rho_Times_Delta_Time;
   Dcm_Incr(Row3,Col2) :=  Dc_Matrix(Row3,Col3) * X_Rho_Times_Delta_Time;
   Dcm_Incr(Row3,Col3) :=  Dc_Matrix(Row3,Col1) * Y_Rho_Times_Delta_Time -
                           Dc_Matrix(Row3,Col2) * X_Rho_Times_Delta_Time;

-- --<compute current values of bottom two Rows>--

   Dc_Matrix(Row2,Col1) := Dc_Matrix(Row2,Col1) + Dcm_Incr(Row2,Col1);
   Dc_Matrix(Row2,Col2) := Dc_Matrix(Row2,Col2) + Dcm_Incr(Row2,Col2);
   Dc_Matrix(Row2,Col3) := Dc_Matrix(Row2,Col3) + Dcm_Incr(Row2,Col3);

   Dc_Matrix(Row3,Col1) := Dc_Matrix(Row3,Col1) + Dcm_Incr(Row3,Col1);
   Dc_Matrix(Row3,Col2) := Dc_Matrix(Row3,Col2) + Dcm_Incr(Row3,Col2);
   Dc_Matrix(Row3,Col3) := Dc_Matrix(Row3,Col3) + Dcm_Incr(Row3,Col3);

end Perform_Rectangular_Integration_Of_Dcm;
```

```
separate (Direction_Cosine_Matrix_Operations.Dcm_General_Operations)
procedure Reorthonormalize_Dcm
          (Dcm_Matrix  : in out Direction_Cosine_Matrix) is

-- ---------------------
-- --declaration section
-- ---------------------

-- ---<objects for storing dot products>---

   Row2_Dot_Row2      : Real;
   Row3_Dot_Row3      : Real;
   Row2_Dot_Row3      : Real;
   Half_Row2_Dot_Row3 : Sin_Cos_Ratio;

-----------------
--begin procedure
-----------------

begin

   Row2_Dot_Row2 := Dcm_Matrix(Row2,Col1) * Dcm_Matrix(Row2,Col1) +
                    Dcm_Matrix(Row2,Col2) * Dcm_Matrix(Row2,Col2) +
                    Dcm_Matrix(Row2,Col3) * Dcm_Matrix(Row2,Col3);

   Row3_Dot_Row3 := Dcm_Matrix(Row3,Col1) * Dcm_Matrix(Row3,Col1) +
                    Dcm_Matrix(Row3,Col2) * Dcm_Matrix(Row3,Col2) +
                    Dcm_Matrix(Row3,Col3) * Dcm_Matrix(Row3,Col3);

-- -------------------------
-- --calculate column values
-- -------------------------

   Dcm_Matrix(Row2,Col1) := Dcm_Matrix(Row2,Col1) *
                            (1.5 - Real(0.5) * Row2_Dot_Row2);

   Dcm_Matrix(Row3,Col1) := Dcm_Matrix(Row3,Col1) *
                            (1.5 - Real(0.5) * Row3_Dot_Row3);

   Dcm_Matrix(Row2,Col2) := Dcm_Matrix(Row2,Col2) *
                            (1.5 - Real(0.5) * Row2_Dot_Row2);

   Dcm_Matrix(Row3,Col2) := Dcm_Matrix(Row3,Col2) *
                            (1.5 - Real(0.5) * Row3_Dot_Row3);

   Dcm_Matrix(Row2,Col3) := Dcm_Matrix(Row2,Col3) *
                            (1.5 - Real(0.5) * Row2_Dot_Row2);

   Dcm_Matrix(Row3,Col3) := Dcm_Matrix(Row3,Col3) *
                            (1.5 - Real(0.5) * Row3_Dot_Row3);

-- -------------------------
-- --<compute Rows 2 thru 3>
-- -------------------------

   Row2_Dot_Row3 := Dcm_Matrix(Row2,Col1) * Dcm_Matrix(Row3,Col1) +
```

```
                    Dcm_Matrix(Row2,Col2) * Dcm_Matrix(Row3,Col2) +
                    Dcm_Matrix(Row2,Col3) * Dcm_Matrix(Row3,Col3);

   Half_Row2_Dot_Row3 := 0.5 * Row2_Dot_Row3;

   Dcm_Matrix(Row2,Col1) := Dcm_Matrix(Row2,Col1) -
                            (Half_Row2_Dot_Row3 * Dcm_Matrix(Row3,Col1));

   Dcm_Matrix(Row3,Col1) := Dcm_Matrix(Row3,Col1) -
                            (Half_Row2_Dot_Row3 * Dcm_Matrix(Row2,Col1));

   Dcm_Matrix(Row2,Col2) := Dcm_Matrix(Row2,Col2) -
                            (Half_Row2_Dot_Row3 * Dcm_Matrix(Row3,Col2));

   Dcm_Matrix(Row3,Col2) := Dcm_Matrix(Row3,Col2) -
                            (Half_Row2_Dot_Row3 * Dcm_Matrix(Row2,Col2));

   Dcm_Matrix(Row2,Col3) := Dcm_Matrix(Row2,Col3) -
                            (Half_Row2_Dot_Row3 * Dcm_Matrix(Row3,Col3));

   Dcm_Matrix(Row3,Col3) := Dcm_Matrix(Row3,Col3) -
                            (Half_Row2_Dot_Row3 * Dcm_Matrix(Row2,Col3));

end Reorthonormalize_Dcm;
```

```
separate (Direction_Cosine_Matrix_Operations.Dcm_General_Operations)
function Frame_Misalignment
              (Dcm_Matrix     : Direction_Cosine_Matrix;
               Ref_Dc_Matrix  : Direction_Cosine_Matrix)
              return Rotation_Angle_Vec is

-- ----------------------
-- --declaration section
-- ----------------------

-- ---<the rotation angle vector that is returned>--

   Rot_Angle    : Rotation_Angle_Vec;

----------------
--begin function
----------------

begin

   Rot_Angle(X) := Dcm_Matrix(Row1,Col2) * Ref_Dc_Matrix(Row1,Col3) +
                   Dcm_Matrix(Row2,Col2) * Ref_Dc_Matrix(Row2,Col3) +
                   Dcm_Matrix(Row3,Col2) * Ref_Dc_Matrix(Row3,Col3);

   Rot_Angle(Y) := Dcm_Matrix(Row1,Col3) * Ref_Dc_Matrix(Row1,Col1) +
                   Dcm_Matrix(Row2,Col3) * Ref_Dc_Matrix(Row2,Col1) +
                   Dcm_Matrix(Row3,Col3) * Ref_Dc_Matrix(Row3,Col1);

   Rot_Angle(Z) := Dcm_Matrix(Row1,Col1) * Ref_Dc_Matrix(Row1,Col2) +
                   Dcm_Matrix(Row2,Col1) * Ref_Dc_Matrix(Row2,Col2) +
                   Dcm_Matrix(Row3,Col1) * Ref_Dc_Matrix(Row3,Col2);

   return Rot_Angle;

end Frame_Misalignment;
```

```
separate (Direction_Cosine_Matrix_Operations.Dcm_General_Operations)
function Aligned_Dcm_Matrix
         (Dcm_Matrix      : Direction_Cosine_Matrix;
          Rotation_Angle  : Rotation_Angle_Vec)
         return Direction_Cosine_Matrix is

-- ---------------------
-- --declaration section
-- ---------------------

-- ---<computed DCM to be returned>---

   Dcm_New : Direction_Cosine_Matrix;

----------------
--begin function
----------------

begin

   Dcm_New(Row2,Col1) := Dcm_Matrix(Row2,Col1) + Dcm_Matrix(Row2,Col3) *
                         Rotation_Angle(Y);

   Dcm_New(Row2,Col2) := Dcm_Matrix(Row2,Col2) - Dcm_Matrix(Row2,Col3) *
                         Rotation_Angle(X);

   Dcm_New(Row2,Col3) := Dcm_Matrix(Row2,Col3) - Dcm_Matrix(Row2,Col1) *
                         Rotation_Angle(Y) +
                         Dcm_Matrix(Row2,Col2) * Rotation_Angle(X);

   Dcm_New(Row3,Col1) := Dcm_Matrix(Row3,Col1) + Dcm_Matrix(Row3,Col3) *
                         Rotation_Angle(Y);

   Dcm_New(Row3,Col2) := Dcm_Matrix(Row3,Col2) - Dcm_Matrix(Row3,Col3) *
                         Rotation_Angle(X);

   Dcm_New(Row3,Col3) := Dcm_Matrix(Row3,Col3) - Dcm_Matrix(Row3,Col1) *
                         Rotation_Angle(Y) +
                         Dcm_Matrix(Row3,Col2) * Rotation_Angle(X);

-- ---<set first row equal to original matrix>--
-- ---<NOTE: The first row is now INVALID data>--

  Dcm_New(Row1,Col1) := Dcm_Matrix(Row1,Col1);
  Dcm_New(Row1,Col2) := Dcm_Matrix(Row1,Col2);
  Dcm_New(Row1,Col3) := Dcm_Matrix(Row1,Col3);

  return Dcm_New;

end Aligned_Dcm_Matrix;
```

```
separate (Direction_Cosine_Matrix_Operations.Dcm_General_Operations)
procedure Compute_First_Row_From_Orthonormal
            (Dcm_Matrix      : in out Direction_Cosine_Matrix) is

begin

   Dcm_Matrix(Row1,Col1) :=   Dcm_Matrix(Row2,Col2) * Dcm_Matrix(Row3,Col3)
                            - Dcm_Matrix(Row2,Col3) * Dcm_Matrix(Row3,Col2);

   Dcm_Matrix(Row1,Col2) :=   Dcm_Matrix(Row2,Col3) * Dcm_Matrix(Row3,Col1)
                            - Dcm_Matrix(Row2,Col1) * Dcm_Matrix(Row3,Col3);

   Dcm_Matrix(Row1,Col3) :=   Dcm_Matrix(Row2,Col1) * Dcm_Matrix(Row3,Col2)
                            - Dcm_Matrix(Row2,Col2) * Dcm_Matrix(Row3,Col1);

end Compute_First_Row_From_Orthonormal;
```

```
separate (Direction_Cosine_Matrix_Operations.Dcm_General_Operations)
function Dcm_From_Quaternion
            (Quaternion : Quaternion_Vectors)
            return Direction_Cosine_Matrix is

-- ---------------------
-- --declaration section
-- ---------------------

-- ---<objects to hold the squares of quaternions>---

   Q0_Sqr  : Sin_Cos_Ratio;
   Q1_Sqr  : Sin_Cos_Ratio;
   Q2_Sqr  : Sin_Cos_Ratio;
   Q3_Sqr  : Sin_Cos_Ratio;

-- ---<the DCM to be returned>---

   Dcm     : Direction_Cosine_Matrix;

-----------------
--begin function
-----------------

begin

-- -----  <compute squares of quaternions> -----

   Q0_Sqr := Quaternion(Q0) * Quaternion(Q0);
   Q1_Sqr := Quaternion(Q1) * Quaternion(Q1);
   Q2_Sqr := Quaternion(Q2) * Quaternion(Q2);
   Q3_Sqr := Quaternion(Q3) * Quaternion(Q3);


-- -----  <compute direction cosines> --------

   Dcm(Row1,Col1) := Q0_Sqr + Q1_Sqr - Q2_Sqr - Q3_Sqr;
   Dcm(Row2,Col2) := Q0_Sqr - Q1_Sqr + Q2_Sqr - Q3_Sqr;
   Dcm(Row3,Col3) := Q0_Sqr - Q1_Sqr - Q2_Sqr + Q3_Sqr;

   Dcm(Row2,Col1) := ( (Quaternion(Q0) * Quaternion(Q3))
                      +(Quaternion(Q1) * Quaternion(Q2)) )
                     * 2.0;

   Dcm(Row3,Col1) := ( (Quaternion(Q1) * Quaternion(Q3))
                      -(Quaternion(Q0) * Quaternion(Q2)) )
                     * 2.0;

   Dcm(Row1,Col2) := ( (Quaternion(Q1) * Quaternion(Q2))
                      -(Quaternion(Q0) * Quaternion(Q3)) )
                     * 2.0;

   Dcm(Row3,Col2) := ( (Quaternion(Q0) * Quaternion(Q1))
                      +(Quaternion(Q2) * Quaternion(Q3)) )
                     * 2.0;

   Dcm(Row1,Col3) := ( (Quaternion(Q0) * Quaternion(Q2))
```

```
                        +(Quaternion(Q1) * Quaternion(Q3)) )
                       * 2.0;

   Dcm(Row2,Col3) := ( (Quaternion(Q2) * Quaternion(Q3))
                        -(Quaternion(Q0) * Quaternion(Q1)) )
                       * 2.0;

   return Dcm;

end Dcm_From_Quaternion;
```

```
separate (Direction_Cosine_Matrix_Operations)
package body Cne_Operations is

   package Dcm_Gen_Pkg renames
                  Direction_Cosine_Matrix_Operations.Dcm_General_Operations;

-- ------------------------------------------------------------------
-- --<instantiation of a units from "DCM_General_Operations" with CNE
-- --  specific parameters applied to the generics>
-- ------------------------------------------------------------------

   function Initialize_From_Ref is new
               Dcm_Gen_Pkg.Dcm_Initialized_From_Reference
                  (Row_Indices            => Earth_Axes,
                   Col_Indices            => Navigation_Axes,
                   Sin_Cos_Ratio          => Sin_Cos_Ratio,
                   Direction_Cosine_Matrix => Cne_Matrices);

   procedure Reorthonormalize is new
                Dcm_Gen_Pkg.Reorthonormalize_Dcm
                   (Row_Indices            => Earth_Axes,
                    Col_Indices            => Navigation_Axes,
                    Sin_Cos_Ratio          => Sin_Cos_Ratio,
                    Direction_Cosine_Matrix => Cne_Matrices,
                    Real                   => Real);

   procedure Compute_First_Row is new
                Dcm_Gen_Pkg.Compute_First_Row_From_Orthonormal
                   (Row_Indices            => Earth_Axes,
                    Col_Indices            => Navigation_Axes,
                    Sin_Cos_Ratio          => Sin_Cos_Ratio,
                    Direction_Cosine_Matrix => Cne_Matrices);

-- ----------------
-- --separate parts
-- ----------------

   function Cne_Initialized_From_Earth_Position
                           (Wander_Angle : Angles;
                            Latitude     : Earth_Position;
                            Longitude    : Earth_Position)
                            return Cne_Matrices is separate;

   package body Cne_Integration is separate;

   package body Alignment_Parts is separate;

   package body Cne_From_Quaternion is separate;

-- ------------------------------------------------------
-- --routines acting as interfaces to instantiated routines
-- ------------------------------------------------------

   function Cne_Initialized_From_Reference
               (Ref_Cne_2_1    : Sin_Cos_Ratio;
                Ref_Cne_2_2    : Sin_Cos_Ratio;
                Ref_Cne_3_1    : Sin_Cos_Ratio;
```

```
                Ref_Cne_3_2    : Sin_Cos_Ratio;
                Sign_Of_2_3    : INTEGER;
                Sign_Of_3_3    : INTEGER)
              return Cne_Matrices is
   begin
      return Initialize_From_Ref
                (Ref_Dcm_2_1 => Ref_Cne_2_1,
                 Ref_Dcm_2_2 => Ref_Cne_2_2,
                 Ref_Dcm_3_1 => Ref_Cne_3_1,
                 Ref_Dcm_3_2 => Ref_Cne_3_2,
                 Sign_Of_2_3 => Sign_Of_2_3,
                 Sign_Of_3_3 => Sign_Of_3_3);
   end Cne_Initialized_From_Reference;

   procedure Reorthonormalize_Cne (Cne : in out Cne_Matrices) is
   begin
      Reorthonormalize (Cne);
   end Reorthonormalize_Cne;

   procedure Compute_First_Row_Of_Cne_From_Orthonormal
                (Cne : in out Cne_Matrices) is
   begin
      Compute_First_Row (Cne);
   end Compute_First_Row_Of_Cne_From_Orthonormal;

end Cne_Operations;
```

```
separate (Direction_Cosine_Matrix_Operations.Cne_Operations)
function Cne_Initialized_From_Earth_Position
            (Wander_Angle : Angles;
             Latitude     : Earth_Position;
             Longitude    : Earth_Position)
            return Cne_Matrices is

-- ---------------------
-- --declaration section
-- ---------------------

-- ---<objects to hold the sine and cosine of latitude, longitude,
-- --   and wander angle>---

   Sin_Lat    : Sin_Cos_Ratio;
   Cos_Lat    : Sin_Cos_Ratio;
   Sin_Long   : Sin_Cos_Ratio;
   Cos_Long   : Sin_Cos_Ratio;
   Sin_Wa     : Sin_Cos_Ratio;
   Cos_Wa     : Sin_Cos_Ratio;

-- ---<the CNE to be returned>---

   Cne        : Cne_Matrices;

----------------
--begin function
----------------

begin

-- --<compute sines and cosines>--

   Sin_Cos(Latitude,     Sin_Lat,  Cos_Lat);
   Sin_Cos(Longitude,    Sin_Long, Cos_Long);
   Sin_Cos(Wander_Angle, Sin_Wa,   Cos_Wa);

-- --<compute CNE elements>--

   Cne(Greenw,East)  := -COs_Wa * Sin_Long - Sin_Wa * Sin_Lat * Cos_Long;
   Cne(Right, East)  :=  Cos_Wa * Cos_Long - Sin_Wa * Sin_Lat * Sin_Long;
   Cne(Polar, East)  :=  Sin_Wa * Cos_Lat;

   Cne(Greenw,North) :=  Sin_Wa * Sin_Long - Cos_Wa * Sin_Lat * Cos_Long;
   Cne(Right, North) := -SIn_Wa * Cos_Long - Cos_Wa * Sin_Lat * Sin_Long;
   Cne(Polar, North) :=  Cos_Wa * Cos_Lat;

   Cne(Greenw,Up)    := Cos_Lat * Cos_Long;
   Cne(Right, Up)    := Cos_Lat * Sin_Long;
   Cne(Polar, Up)    := Sin_Lat;

   return Cne;

end Cne_Initialized_From_Earth_Position;
```

```
separate (Direction_Cosine_Matrix_Operations.Cne_Operations)
package body Cne_Integration is

-- ----------------------
-- --local instantiations
-- ----------------------

-- --<instantiation of a units from "DCM_General_Operations" with CNE
-- --  specific parameters applied to the generics>---

   package Integrate_Trap is new
             Dcm_Gen_Pkg.Dcm_Trapezoidal_Integration
               (Angular_Velocities      => Angular_Velocities,
                Time_Intervals          => Time_Intervals,
                Row_Indices             => Earth_Axes,
                Col_Indices             => Navigation_Axes,
                Sin_Cos_Ratio           => Sin_Cos_Ratio,
                Direction_Cosine_Matrix => Cne_Matrices,
                Initial_X_Rho           => Initial_East_Rho,
                Initial_Y_Rho           => Initial_North_Rho);

   procedure Integrate_Rect is new
             Dcm_Gen_Pkg.Perform_Rectangular_Integration_Of_Dcm
               (Angular_Velocities      => Angular_Velocities,
                Time_Intervals          => Time_Intervals,
                Row_Indices             => Earth_Axes,
                Col_Indices             => Navigation_Axes,
                Sin_Cos_Ratio           => Sin_Cos_Ratio,
                Direction_Cosine_Matrix => Cne_Matrices);

-- ------------------------------------------------------
-- --routines acting as interfaces to instantiated routines
-- ------------------------------------------------------

-- --------[Trapezoidal Integration]---------

   procedure Perform_Trapezoidal_Integration_Of_Cne
               (Cne            : in out Cne_Matrices;
                East_Rho       : in     Angular_Velocities;
                North_Rho      : in     Angular_Velocities;
                Delta_Time     : in     Time_Intervals) is
   begin
      Integrate_Trap.Perform_Trapezoidal_Integration_Of_Dcm
                           (Cne, East_Rho, North_Rho, Delta_Time);
   end Perform_Trapezoidal_Integration_Of_Cne;


   procedure Reinit_Ang_Vel_For_Trapez_Integ_Of_Cne
               (East_Rho     : in     Angular_Velocities;
                North_Rho    : in     Angular_Velocities) is
   begin
      Integrate_Trap.Reinitialize_Angular_Velocities
                           (East_Rho, North_Rho);
   end Reinit_Ang_Vel_For_Trapez_Integ_Of_Cne;


-- --------[Rectangular Integration]---------
```

```
   procedure Perform_Rectangular_Integration_Of_Cne
             (Cne        : in out Cne_Matrices;
              East_Rho   : in     Angular_Velocities;
              North_Rho  : in     Angular_Velocities;
              Delta_Time : in     Time_Intervals) is
   begin
      Integrate_Rect (Cne, East_Rho, North_Rho, Delta_Time);
   end Perform_Rectangular_Integration_Of_Cne;


end Cne_Integration;
```

```
separate (Direction_Cosine_Matrix_Operations.Cne_Operations)
package body Alignment_Parts is

-- ----------------------
-- --local instantiations
-- ----------------------

--- <instantiation of a unit from "DCM_General_Operations" with CNE
--   specific parameters applied to the generics>---

   function Computed_Frame_Misalignment is new
            Dcm_Gen_Pkg.Frame_Misalignment
               (Row_Indices             => Earth_Axes,
                Col_Indices             => Navigation_Axes,
                Sin_Cos_Ratio           => Sin_Cos_Ratio,
                Direction_Cosine_Matrix => Cne_Matrices,
                Rotation_Indices        => Rotation_Indices,
                Rotation_Angles         => Rotation_Angles,
                Rotation_Angle_Vec      => Rotation_Angle_Vec);

   function Align_Cne_Matrix is new
             Dcm_Gen_Pkg.Aligned_Dcm_Matrix
               (Row_Indices             => Earth_Axes,
                Col_Indices             => Navigation_Axes,
                Sin_Cos_Ratio           => Sin_Cos_Ratio,
                Direction_Cosine_Matrix => Cne_Matrices,
                Rotation_Indices        => Rotation_Indices,
                Rotation_Angles         => Rotation_Angles,
                Rotation_Angle_Vec      => Rotation_Angle_Vec);

-- ------------------------------------------------------------
-- --routines acting as interfaces to instantiated routines
-- ------------------------------------------------------------

-- --------[Frame Misalignment]----------

   function Frame_Misalignment_Of_Cne
               (Cne            : Cne_Matrices;
                Ref_Dc_Matrix  : Cne_Matrices)
               return Rotation_Angle_Vec is
   begin
      return Computed_Frame_Misalignment (Cne, Ref_Dc_Matrix);
   end Frame_Misalignment_Of_Cne;

-- --------[Align CNE Matrix]----------

   function Aligned_Cne_Matrix
               (Cne             : Cne_Matrices;
                Rotation_Angle  : Rotation_Angle_Vec)
               return Cne_Matrices is
   begin
      return Align_Cne_Matrix (Cne, Rotation_Angle);
   end Aligned_Cne_Matrix;

end Alignment_Parts;
```

```
separate (Direction_Cosine_Matrix_Operations.Cne_Operations)
package body Cne_From_Quaternion is

-- ----------------------
-- --local instantiations
-- ----------------------

-- --<instantiation of a unit from "DCM_General_Operations" with CNE
-- --  specific parameters applied to the generics>---

   function Computed_Cne is new
            Dcm_Gen_Pkg.Dcm_From_Quaternion
              (Row_Indices             => Earth_Axes,
               Col_Indices             => Navigation_Axes,
               Sin_Cos_Ratio           => Sin_Cos_Ratio,
               Direction_Cosine_Matrix => Cne_Matrices,
               Quaternion_Indices      => Quaternion_Indices,
               Quaternion_Vectors      => Quaternion_Vectors);

-- -----------------
-- --function bodies
-- -----------------

-- ---<the function to call the above function>---

   function Compute_Cne
            (Quaternion : Quaternion_Vectors)
            return Cne_Matrices is
   begin
       return Computed_Cne (Quaternion);
   end Compute_Cne;

end Cne_From_Quaternion;
```

(This page left intentionally blank.)

3.3.3 KALMAN FILTER

(This page intentionally left blank.)

3.3.3.1 KALMAN FILTER COMMON PARTS (BODY) TLCSC 651 (CATALOG #P163-0)

This part, which is designed as an Ada package, contains specifications for all CAMP parts which can be used to implement a Kalman Filter regardless of the type of H matrix used.

The decomposition for this part is the same as that shown in the Top-Level Design Document.

3.3.3.1.1 REQUIREMENTS ALLOCATION

The following chart summarizes the allocation of CAMP requirements to this

Tlcsc:

Name	Requirements Allocation
State_Transition_And_Process_Noise_Matrices_ Manager	R145
Error_Covariance_Matrix_Manager	R146
State_Transition_Matrix_Manger	R148

3.3.3.1.2 LOCAL ENTITIES DESIGN

None.

3.3.3.1.3 INPUT/OUTPUT

None.

3.3.3.1.4 LOCAL DATA

None.

3.3.3.1.5 PROCESS CONTROL

Not applicable.

3.3.3.1.6 PROCESSING

The following describes the processing performed by this part:

```
package body Kalman_Filter_Common_Parts is

   package body State_Transition_And_Process_Noise_Matrices_Manager is separate;

   package body Error_Covariance_Matrix_Manager                     is separate;

   package body State_Transition_Matrix_Manager                     is separate;
```

```
end Kalman_Filter_Common_Parts;
```

3.3.3.1.7 UTILIZATION OF OTHER ELEMENTS

None.

3.3.3.1.8 LIMITATIONS

None.

3.3.3.1.9 LLCSC DESIGN

3.3.3.1.9.1 STATE_TRANSITION_AND_PROCESS_NOISE_MATRICES_MANAGER PACKAGE DESIGN (CATALOG #P164-0)

This LLCSC is a generic package which manages the State Transition (Phi) and Process Noise (Q) matrices. It consists of an Initialize procedure, a Propagation function, and functions which return the stored value of each of the two matrices.

The decomposition for this part is the same as that shown in the Top-Level Design Document.

3.3.3.1.9.1.1 REQUIREMENTS ALLOCATION

This part meets CAMP requirement R145.

3.3.3.1.9.1.2 LOCAL ENTITIES DESIGN

None.

3.3.3.1.9.1.3 INPUT/OUTPUT

GENERIC PARAMETERS:

Data types:

The following table summarizes the generic formal types required by this part:

Name	Base Type	Description
Time_Intervals	floating point	Type for the delta time variable
Phi_Matrices	private	Data type of N x N Phi Matrix
Integrated_F_ Matrices	private	Data type of N x N Matrix for F integration
Integrated_Q_ Matrices	private	Data type of N x N Matrix for Q integration
Q_Matrices	private	Data type of N x N Q Matrix

Subprograms:

The following table summarizes the generic formal subroutines required by this part:

Name	Type	Description
Add_To_Identity	procedure	Adds the identity matrix to a Integrated_F_Matrices
Set_To_Identity_Matrix	function	Sets a Phi_Matrices type matrix to the identity matrix
Set_To_Zero_Matrix	function	Sets a Q_Matrices type matrix to the zer matrix
ABA_Transpose	function	Multiplies a Integrated_F_Matrices type matrix by the transpose of a Q_Matrices type matrix yielding a Q_Matrices type matrix
"*"	function	Multiplies a Integrated F Matrix by a Time_Interval yielding a Integrated Q Matrix
"*"	function	Multiplies a Integrated F Matrix by a Phi Matrix yielding a Phi Matrix
"+"	function	Adds a Integrated Q Matrix to a Q Marix yielding a Q Matrix

3.3.3.1.9.1.4 LOCAL DATA

Data objects:

The following table describes the data objects maintained by this part:

Name	Type	Value	Description
Prop_Phi	Statically_Sparse_Matrices	variable	The propagated state transition matrix
Prop_Q	Symmetric Matrices	variable	The propagated process noise matrix

3.3.3.1.9.1.5 PROCESS CONTROL

Not applicable.

3.3.3.1.9.1.6 PROCESSING

The following describes the processing performed by this part:

```
separate (Kalman_Filter_Common_Parts)
package body State_Transition_And_Process_Noise_Matrices_Manager is

   Prop_Phi : Phi_Matrices;
   Prop_Q   : Q_Matrices;

end State_Transition_And_Process_Noise_Matrices_Manager;
```

3.3.3.1.9.1.7 UTILIZATION OF OTHER ELEMENTS

None.

3.3.3.1.9.1.8 LIMITATIONS

None.

3.3.3.1.9.1.9 LLCSC DESIGN

None.

3.3.3.1.9.1.10 UNIT DESIGN

3.3.3.1.9.1.10.1 INITIALIZE UNIT DESIGN

This unit initializes the Propagated Phi matrix to the identity, and the Propagated Q matrix to the zero matrix.

3.3.3.1.9.1.10.1.1 REQUIREMENTS ALLOCATION

This part meets CAMP requirement R145

3.3.3.1.9.1.10.1.2 LOCAL ENTITIES DESIGN

None.

3.3.3.1.9.1.10.1.3 INPUT/OUTPUT

None.

3.3.3.1.9.1.10.1.4 LOCAL DATA

None.

3.3.3.1.9.1.10.1.5 PROCESS CONTROL

Not applicable.

3.3.3.1.9.1.10.1.6 PROCESSING

The following describes the processing performed by this part:

```
procedure Initialize is

begin

   Set_To_Identity_Matrix (Source => Prop_Phi);

   Set_To_Zero_Matrix (Source => Prop_Q);

end Initialize;
```

3.3.3.1.9.1.10.1.7 UTILIZATION OF OTHER ELEMENTS

UTILIZATION OF OTHER ELEMENTS IN TOP LEVEL COMPONENT:

The following tables describe the elements used by this part but defined elsewhere in the parent top level component:

Data types:

The following table summarizes the types required by this part and defined in the parent component:

Name	Base Type	Description
Phi_Matrices	private	Data type of matrix used to represent Phi matrices
Q_Matrices	private	Data type of matrix used to represent Q matrices

Data objects:

The following table summarizes the objects required by this part and defined in the parent component:

Name	Type	Value	Description
Prop_Phi	Phi_Matrices	variable	The propagated state transition matrix
Prop_Q	Q_Matrices	variable	The propagated process noise matrix

3.3.3.1.9.1.10.1.8 LIMITATIONS

None.

3.3.3.1.9.1.10.2 PROPAGATE UNIT DESIGN

This unit propagates new Propagated Phi and Propagated Q matrices given a new Integrated System Description Matrix and Time Interval

3.3.3.1.9.1.10.2.1 REQUIREMENTS ALLOCATION

This part meets CAMP requirement R145.

3.3.3.1.9.1.10.2.2 LOCAL ENTITIES DESIGN

None.

3.3.3.1.9.1.10.2.3 INPUT/OUTPUT

FORMAL PARAMETERS:

The following table describes this unit's formal parameters:

Name	Type	Mode	Description
Integrated_F	Integrated_F_Matrices	in	Current value for the System Description matrix already integrated across time
Q	Integrated_Q_Matrices	in	Current process noise matrix
DT	Time_Intervals	in	Time since the last propagation

3.3.3.1.9.1.10.2.4 LOCAL DATA

Data objects:

The following table describes the data objects maintained by this part:

Name	Type	Value	Description
Phi	Integrated_F_Matrices	variable	Computed process noise matrix matrix

3.3.3.1.9.1.10.2.5 PROCESS CONTROL

Not applicable.

3.3.3.1.9.1.10.2.6 PROCESSING

The following describes the processing performed by this part:

```
   procedure Propagate (Integrated_F : in Integrated_F_Matrices;
                        Q            : in Integrated_Q_Matrices;
                        DT           : in Time_Intervals) is

      Phi      : Integrated_F_Matrices;

   begin

--    -- Propagate Phi

      Phi := Add_To_Identity (Source_Matrix => Integrated_F);

      Prop_Phi := Phi * Prop_Phi;

--    -- Propagate Q

      Prop_Q := Q * DT + ABA_Transpose( Phi, Prop_Q ) ;

   end Propagate;
```

3.3.3.1.9.1.10.2.7 UTILIZATION OF OTHER ELEMENTS

UTILIZATION OF OTHER ELEMENTS IN TOP LEVEL COMPONENT:

The following tables describe the elements used by this part but defined elsewhere in the parent top level component:

Data types:

The following table summarizes the generic formal types required by this part:

Name	Base Type	Description
Time_Intervals	floating point type	Type for the delta time variable
Phi_Matrices	private	Data type of N x N Phi Matrix
Integrated_F_Matrices	private	Data type of N x N Matrix for F integration
Integrated_Q_Matrices	private	Data type of N x N Matrix for Q integration
Q_Matrices	private	Data type of N x N Q Matrix

Data objects:

The following table summarizes the objects required by this part and defined in the parent component:

Name	Type	Value	Description
Prop_Phi	Phi_Matrices	variable	The propagated state transition matrix
Prop_Q	Q_Matrices	variable	The propagated process noise matrix

3.3.3.1.9.1.10.2.8 LIMITATIONS

None.

3.3.3.1.9.1.10.3 GET_CURRENT UNIT DESIGN

This unit returns the current value of the Propagated Phi and Propagated Q matrices.

3.3.3.1.9.1.10.3.1 REQUIREMENTS ALLOCATION

This part meets CAMP requirement R145.

3.3.3.1.9.1.10.3.2 LOCAL ENTITIES DESIGN

None.

3.3.3.1.9.1.10.3.3 INPUT/OUTPUT

FORMAL PARAMETERS:

The following table describes this unit's formal parameters:

Name	Type	Mode	Description
Propagated_Phi	Phi_Matrices	out	Stored Propagated State Transition matrix
Propagated_Q	Q_Matrices	out	Stored Propagated Process Noise Matrix

3.3.3.1.9.1.10.3.4 LOCAL DATA

None.

3.3.3.1.9.1.10.3.5 PROCESS CONTROL

Not applicable.

3.3.3.1.9.1.10.3.6 PROCESSING

The following describes the processing performed by this part:

```
procedure Get_Current (Propagated_Phi : out Phi_Matrices;
                       Propagated_Q   : out Q_Matrices ) is

begin

   Propagated_Phi := Prop_Phi;

   Propagated_Q   := Prop_Q;

end Get_Current;
```

3.3.3.1.9.1.10.3.7 UTILIZATION OF OTHER ELEMENTS

UTILIZATION OF OTHER ELEMENTS IN TOP LEVEL COMPONENT:

The following tables describe the elements used by this part but defined elsewhere in the parent top level component:

Data types:

The following table summarizes the types required by this part and defined in the parent component:

Name	Base Type	Description
Phi_Matrices	private	Data type of N x N Phi Matrix
Q_Matrices	private	Data type of N x N Q Matrix

Data objects:

The following table summarizes the objects required by this part and defined in the parent component:

Name	Type	Value	Description
Prop_Phi	Phi_Matrices	variable	The propagated state transition matrix
Prop_Q	Q_Matrices	variable	The propagated process noise matrix

3.3.3.1.9.1.10.3.8 LIMITATIONS

None.

3.3.3.1.9.1.10.4 PROPAGATED_PHI UNIT DESIGN

This unit returns the current value of the Propagated Phi matrix

3.3.3.1.9.1.10.4.1 REQUIREMENTS ALLOCATION

This part meets CAMP requirement R145.

3.3.3.1.9.1.10.4.2 LOCAL ENTITIES DESIGN

None.

3.3.3.1.9.1.10.4.3 INPUT/OUTPUT

FORMAL PARAMETERS:

The following table describes this unit's formal parameters:

Name	Type	Mode	Description
<returned value>	Phi_Matrices	out	Stored Propagated State Transition matrix

3.3.3.1.9.1.10.4.4 LOCAL DATA

None.

3.3.3.1.9.1.10.4.5 PROCESS CONTROL

Not applicable.

3.3.3.1.9.1.10.4.6 PROCESSING

The following describes the processing performed by this part:

```
function Propagated_Phi return Phi_Matrices is

begin

   return Prop_Phi;

end Propagated_Phi;
```

3.3.3.1.9.1.10.4.7 UTILIZATION OF OTHER ELEMENTS

UTILIZATION OF OTHER ELEMENTS IN TOP LEVEL COMPONENT:

The following tables describe the elements used by this part but defined elsewhere in the parent top level component:

Data types:

The following table summarizes the types required by this part and defined in the parent component:

Name	Base Type	Description
Phi_Matrices	private	Data type for N x N phi matrix

Data objects:

The following table summarizes the objects required by this part and defined in the parent component:

Name	Type	Value	Description
Prop_Phi	Phi_Matrices	variable	The propagated state transition matrix

3.3.3.1.9.1.10.4.8 LIMITATIONS

None.

3.3.3.1.9.2 ERROR_COVARIANCE_MATRIX_MANAGER PACKAGE DESIGN (CATALOG #P165-0)

This LLCSC is a generic package which manages the Error Covariance Matrix; it consists of an Initialize procedure, a Propagation procedure, and a P function, which returns the current Error Covariance matrix value

The decomposition for this part is the same as that shown in the Top-Level Design Document.

3.3.3.1.9.2.1 REQUIREMENTS ALLOCATION

This part meets CAMP requirement R146.

3.3.3.1.9.2.2 LOCAL ENTITIES DESIGN

None.

3.3.3.1.9.2.3 INPUT/OUTPUT

GENERIC PARAMETERS:

Data types:

The following table summarizes the generic formal types required by this part:

Name	Base Type	Description
Phi_Matrices	private	Data type of N x N Phi matrix
P_And_Q_Matrices	private	Data type of N x N P and Q matrix

Subprograms:

The following table summarizes the generic formal subroutines required by this part:

Name	Type	Description
ABA_Transpose	function	Multiplies a Phi matrix by the by the transpose of a P and Q matrix yielding a P and Q matrix
"+"	function	Adds two P and Q matrices yielding a P and Q matrix

3.3.3.1.9.2.4 LOCAL DATA

The following table describes the data objects maintained by this part:

Name	Type	Value	Description
Stored_P	P_And_Q_Matrices	variable	The propagated error covariance matrix which is stored by part

3.3.3.1.9.2.5 PROCESS CONTROL

Not applicable.

3.3.3.1.9.2.6 PROCESSING

The following describes the processing performed by this part:

```
separate (Kalman_Filter_Common_Parts)
package body Error_Covariance_Matrix_Manager is
   Stored_P : P_And_Q_Matrices;
end Error_Covariance_Matrix_Manager;
```

3.3.3.1.9.2.7 UTILIZATION OF OTHER ELEMENTS

None.

3.3.3.1.9.2.8 LIMITATIONS

None.

3.3.3.1.9.2.9 LLCSC DESIGN

None.

3.3.3.1.9.2.10 UNIT DESIGN

3.3.3.1.9.2.10.1 INITIALIZE UNIT DESIGN

This unit initializes the Error Covariance matrix to the matrix sent in.

3.3.3.1.9.2.10.1.1 REQUIREMENTS ALLOCATION

This part meets CAMP requirement R146

3.3.3.1.9.2.10.1.2 LOCAL ENTITIES DESIGN

None.

3.3.3.1.9.2.10.1.3 INPUT/OUTPUT

FORMAL PARAMETERS:

The following table describes this unit's formal parameters:

Name	Type	Mode	Description
Initial_P	P_And_Q_ Matrices	in	This is the value to which the Error Covariance Matrix will be initialized

3.3.3.1.9.2.10.1.4 LOCAL DATA

None.

3.3.3.1.9.2.10.1.5 PROCESS CONTROL

Not applicable.

3.3.3.1.9.2.10.1.6 PROCESSING

The following describes the processing performed by this part:

```
procedure Initialize (Initial_P : in P_And_Q_Matrices) is

begin

   Stored_P := Initial_P;

end Initialize;
```

3.3.3.1.9.2.10.1.7 UTILIZATION OF OTHER ELEMENTS

UTILIZATION OF OTHER ELEMENTS IN TOP LEVEL COMPONENT:

The following tables describe the elements used by this part but defined elsewhere in the parent top level component:

Data types:

The following table summarizes the types required by this part and defined in the parent component:

Name	Base Type	Description
P_And_Q_Matrices	private	Data type of N x N P and Q matrix

Data objects:

The following table summarizes the objects required by this part and defined in the parent component:

Name	Type	Value	Description
Stored_P	P_And_Q_ Matrices	variable	The propagated error covariance matrix which is stored by part

3.3.3.1.9.2.10.1.8 LIMITATIONS

None.

3.3.3.1.9.2.10.2 PROPAGATE UNIT DESIGN

This unit propagates the Error Covariance Matrix given the Propagated Phi and Propagated Q matrices

3.3.3.1.9.2.10.2.1 REQUIREMENTS ALLOCATION

This part meets CAMP requirement R146.

3.3.3.1.9.2.10.2.2 LOCAL ENTITIES DESIGN

None.

3.3.3.1.9.2.10.2.3 INPUT/OUTPUT

FORMAL PARAMETERS:

The following table describes this unit's formal parameters:

Name	Type	Mode	Description
Propagated_Phi	Phi_Matrices	in	Current value for the Propagated State Transition Matrix
Propagated_Q	P_And_Q_ Matrices	in	Current value for the Propagated Process Noise Matrix

3.3.3.1.9.2.10.2.4 LOCAL DATA

None.

3.3.3.1.9.2.10.2.5 PROCESS CONTROL

Not applicable.

3.3.3.1.9.2.10.2.6 PROCESSING

The following describes the processing performed by this part:

```
procedure Propagate  (Propagated_Phi : in Phi_Matrices;
                      Propagated_Q   : in P_And_Q_Matrices) is

begin

   Stored_P := ABA_Transpose( Propagated_Phi, Stored_P ) + Propagated_Q;

end Propagate;
```

3.3.3.1.9.2.10.2.7 UTILIZATION OF OTHER ELEMENTS

UTILIZATION OF OTHER ELEMENTS IN TOP LEVEL COMPONENT:

The following tables describe the elements used by this part but defined elsewhere in the parent top level component:

Data types:

The following table summarizes the types required by this part and defined in the parent component:

Name	Base Type	Description
Phi_Matrices	private	Data type of n x n Phi matrix
P_And_Q_Matrices	private	Data type of n x n P and Q matrix

Data objects:

The following table summarizes the objects required by this part and defined in the parent component:

Name	Type	Value	Description
Stored_P	P_And_Q_ Matrices	variable	The propagated error covariance matrix which is stored by part

3.3.3.1.9.2.10.2.8 LIMITATIONS

None.

3.3.3.1.9.2.10.3 P UNIT DESIGN

This unit returns the current value of the P matrix

3.3.3.1.9.2.10.3.1 REQUIREMENTS ALLOCATION

This part meets CAMP requirement R146.

3.3.3.1.9.2.10.3.2 LOCAL ENTITIES DESIGN

None.

3.3.3.1.9.2.10.3.3 INPUT/OUTPUT

FORMAL PARAMETERS:

The following table describes this unit's formal parameters:

Name	Type	Mode	Description
<returned value>	P_And_Q_ Matrices	out	Stored Propagated Error Covariance matrix

3.3.3.1.9.2.10.3.4 LOCAL DATA

None.

3.3.3.1.9.2.10.3.5 PROCESS CONTROL

Not applicable.

3.3.3.1.9.2.10.3.6 PROCESSING

The following describes the processing performed by this part:

```
function P return P_And_Q_Matrices is
begin
   return Stored_P;
end P;
```

3.3.3.1.9.2.10.3.7 UTILIZATION OF OTHER ELEMENTS

UTILIZATION OF OTHER ELEMENTS IN TOP LEVEL COMPONENT:

The following tables describe the elements used by this part but defined elsewhere in the parent top level component:

Data types:

The following table summarizes the types required by this part and defined in the parent component:

Name	Base Type	Description
P_and_Q_Matrices	private	Data type of n x n P and Q matrix

Data objects:

The following table summarizes the objects required by this part and defined in the parent component:

Name	Type	Value	Description
Stored_P	P_and_Q_ Matrices	variable	The current propagated error covariance matrix

3.3.3.1.9.2.10.3.8 LIMITATIONS

None.

3.3.3.1.9.3 STATE_TRANSITION_MATRIX_MANAGER PACKAGE DESIGN (CATALOG #P166-0)

This LLCSC is a generic package which manages the State Transition Matrix, commonly known as the Phi matrix. It consists of an Initialization procedure, a Propagation function, and a function which returns the stored Propagated_Phi value.

The decomposition for this part is the same as that shown in the Top-Level Design Document.

3.3.3.1.9.3.1 REQUIREMENTS ALLOCATION

This part meets CAMP requirement R148.

3.3.3.1.9.3.2 LOCAL ENTITIES DESIGN

None.

3.3.3.1.9.3.3 INPUT/OUTPUT

GENERIC PARAMETERS:

Data types:

The following table summarizes the generic formal types required by this part:

Name	Base Type	Description
Integrated_F_Matrices	private	Data type for n x n F integration matrix
Phi_Matrices	private	Data type for n x n Phi matrix

Subprograms:

The following table summarizes the generic formal subroutines required by this part:

Name	Type	Description
Set_To_Identity_	function	Sets a Phi matrix to the Identity
"*"	function	Multiplies an Integrated F matrix by Phi matrix yielding a Phi matrix

3.3.3.1.9.3.4 LOCAL DATA

Data objects:

The following table describes the data objects maintained by this part:

Name	Type	Value	Description
Prop_Phi	Phi_Matrices	variable	The propagated state transition matrix

3.3.3.1.9.3.5 PROCESS CONTROL

Not applicable.

3.3.3.1.9.3.6 PROCESSING

The following describes the processing performed by this part:

separate (Kalman_Filter_Common_Parts)

```
package body State_Transition_Matrix_Manager is

   Prop_Phi : Phi_Matrices;

end State_Transition_Matrix_Manager;
```

3.3.3.1.9.3.7 UTILIZATION OF OTHER ELEMENTS

None.

3.3.3.1.9.3.8 LIMITATIONS

None.

3.3.3.1.9.3.9 LLCSC DESIGN

None.

3.3.3.1.9.3.10 UNIT DESIGN

3.3.3.1.9.3.10.1 INITIALIZE UNIT DESIGN

This unit initializes the Propagated Phi matrix to the identity.

3.3.3.1.9.3.10.1.1 REQUIREMENTS ALLOCATION

This part meets CAMP requirement R148

3.3.3.1.9.3.10.1.2 LOCAL ENTITIES DESIGN

None.

3.3.3.1.9.3.10.1.3 INPUT/OUTPUT

None.

3.3.3.1.9.3.10.1.4 LOCAL DATA

None.

3.3.3.1.9.3.10.1.5 PROCESS CONTROL

Not applicable.

3.3.3.1.9.3.10.1.6 PROCESSING

The following describes the processing performed by this part:

```
procedure Initialize is

begin

   Set_To_Identity_Matrix (Matrix => Prop_Phi);

end Initialize;
```

3.3.3.1.9.3.10.1.7 UTILIZATION OF OTHER ELEMENTS

UTILIZATION OF OTHER ELEMENTS IN TOP LEVEL COMPONENT:

The following tables describe the elements used by this part but defined elsewhere in the parent top level component:

Data types:

The following table summarizes the data types defined by the parent component:

Name	Base Type	Description
Phi_Matrices	private	Data type for n x n Phi matrix

Data objects:

The following table summarizes the objects required by this part and defined in the parent component:

Name	Type	Value	Description
Prop_Phi	Integrated_F_Matrix	variable	The propagated state transition matrix

3.3.3.1.9.3.10.1.8 LIMITATIONS

None.

3.3.3.1.9.3.10.2 PROPAGATE UNIT DESIGN

This unit propagates a new Propagated Phi matrix.

3.3.3.1.9.3.10.2.1 REQUIREMENTS ALLOCATION

This part meets CAMP requirement R148.

3.3.3.1.9.3.10.2.2 LOCAL ENTITIES DESIGN

None.

3.3.3.1.9.3.10.2.3 INPUT/OUTPUT

FORMAL PARAMETERS:

The following table describes this unit's formal parameters:

Name	Type	Mode	Description
Phi	Integrated_F_Matrices	in	Current value for the State Transition Matrix

3.3.3.1.9.3.10.2.4 LOCAL DATA

None.

3.3.3.1.9.3.10.2.5 PROCESS CONTROL

Not applicable.

3.3.3.1.9.3.10.2.6 PROCESSING

The following describes the processing performed by this part:

```
procedure Propagate (Integrated_F : in Integrated_F_Matrices) is
begin
   Prop_Phi := Integrated_F * Prop_Phi;
end Propagate;
```

3.3.3.1.9.3.10.2.7 UTILIZATION OF OTHER ELEMENTS

UTILIZATION OF OTHER ELEMENTS IN TOP LEVEL COMPONENT:

The following tables describe the elements used by this part but defined elsewhere in the parent top level component:

Data types:

The following table summarizes the types required by this part and defined in the parent component:

Name	Base Type	Description
Integrated_F_ Matrices	private	Data type for N by N matrix for F Integration
Phi_Matrices	private	Data Type for N by N Phi matrix

Data objects:

The following table summarizes the objects required by this part and defined in the parent component:

Name	Type	Value	Description
Prop_Phi	Integrated_F_Matrices	variable	The propagated state transition matrix

3.3.3.1.9.3.10.2.8 LIMITATIONS

None.

3.3.3.1.9.3.10.3 PROPAGATED_PHI UNIT DESIGN

This unit returns the current value of the Propagated Phi matrix

3.3.3.1.9.3.10.3.1 REQUIREMENTS ALLOCATION

This part meets CAMP requirement R148.

3.3.3.1.9.3.10.3.2 LOCAL ENTITIES DESIGN

None.

3.3.3.1.9.3.10.3.3 INPUT/OUTPUT

FORMAL PARAMETERS:

The following table describes this unit's formal parameters:

Name	Type	Mode	Description
<returned value>	Phi_Matrices	out	Stored Propagated State Transition matrix

3.3.3.1.9.3.10.3.4 LOCAL DATA

None.

3.3.3.1.9.3.10.3.5 PROCESS CONTROL

Not applicable.

3.3.3.1.9.3.10.3.6 PROCESSING

The following describes the processing performed by this part:

```
function Propagated_Phi return Phi_Matrices is

begin

   return Prop_Phi;

end Propagated_Phi;
```

3.3.3.1.9.3.10.3.7 UTILIZATION OF OTHER ELEMENTS

UTILIZATION OF OTHER ELEMENTS IN TOP LEVEL COMPONENT:

The following tables describe the elements used by this part but defined elsewhere in the parent top level component:

Data types:

The following table summarizes the types required by this part and defined in the parent component:

Name	Base Type	Description
Phi_Matrices	private	Data type for n x n Phi matrix

Data objects:

The following table summarizes the objects required by this part and defined in the parent component:

Name	Type	Value	Description
Prop_Phi	Phi_Matrices	variable	The propagated state transition matrix

3.3.3.1.9.3.10.3.8 LIMITATIONS

None.

3.3.3.1.10 UNIT DESIGN

None.

(This page left intentionally blank.)

```
package body Kalman_Filter_Common_Parts is

   package body State_Transition_And_Process_Noise_Matrices_Manager is separate;

   package body Error_Covariance_Matrix_Manager                     is separate;

   package body State_Transition_Matrix_Manager                     is separate;

end Kalman_Filter_Common_Parts;
```

```
separate (Kalman_Filter_Common_Parts)
package body State_Transition_And_Process_Noise_Matrices_Manager is

   Prop_Phi : Phi_Matrices;
   Prop_Q   : Q_Matrices;

pragma PAGE;
   procedure Initialize is

   begin

      Set_To_Identity_Matrix (Source => Prop_Phi);

      Set_To_Zero_Matrix (Source => Prop_Q);

   end Initialize;

pragma PAGE;
   procedure Propagate (Integrated_F : in Integrated_F_Matrices;
                        Q            : in Integrated_Q_Matrices;
                        Dt           : in Time_Intervals) is

      Phi      : Integrated_F_Matrices;

   begin


--       -- Propagate Phi

      Phi := Add_To_Identity (Source_Matrix => Integrated_F);

      Prop_Phi := Phi * Prop_Phi;


--       -- Propagate Q

      Prop_Q := Q * Dt + Aba_Transpose( Phi, Prop_Q ) ;

   end Propagate;

pragma PAGE;
   procedure Get_Current (Propagated_Phi : out Phi_Matrices;
                          Propagated_Q   : out Q_Matrices ) is

   begin

      Propagated_Phi := Prop_Phi;

      Propagated_Q   := Prop_Q;

   end Get_Current;

pragma PAGE;
   function Propagated_Phi return Phi_Matrices is

   begin
```

```
        return Prop_Phi;
    end Propagated_Phi;
end State_Transition_And_Process_Noise_Matrices_Manager;
```

```
separate (Kalman_Filter_Common_Parts)

package body Error_Covariance_Matrix_Manager is

   Stored_P : P_And_Q_Matrices;

pragma PAGE;
   procedure Initialize (Initial_P : in P_And_Q_Matrices) is

   begin

      Stored_P := Initial_P;

   end Initialize;

pragma PAGE;
   procedure Propagate  (Propagated_Phi : in Phi_Matrices;
                         Propagated_Q   : in P_And_Q_Matrices) is

   begin

      Stored_P := Aba_Transpose( Propagated_Phi, Stored_P ) + Propagated_Q;

   end Propagate;

pragma PAGE;
   function P return P_And_Q_Matrices is

   begin

      return Stored_P;

   end P;

end Error_Covariance_Matrix_Manager;
```

```
separate (Kalman_Filter_Common_Parts)

package body State_Transition_Matrix_Manager is

   Prop_Phi : Phi_Matrices;

pragma PAGE;
   procedure Initialize is

   begin

      Set_To_Identity_Matrix (Matrix => Prop_Phi);

   end Initialize;

pragma PAGE;
   procedure Propagate (Integrated_F : in Integrated_F_Matrices) is

   begin

      Prop_Phi := Integrated_F * Prop_Phi;

   end Propagate;

pragma PAGE;
   function Propagated_Phi return Phi_Matrices is

   begin

      return Prop_Phi;

   end Propagated_Phi;

end State_Transition_Matrix_Manager;
```

(This page left intentionally blank.)

3.3.3.2 KALMAN FILTER COMPACT H PARTS (BODY) TLCSC (CATALOG #P137-0)

This part, which is designed as an Ada package, contains bodies for all CAMP parts which can be used to implement a Kalman Filter when a compact Measurement Sensitivity Matrix (Compact H Matrix) is used

The decomposition for this part is the same as that shown in the Top-Level Design Document.

3.3.3.2.1 REQUIREMENTS ALLOCATION

The following chart summarizes the allocation of CAMP requirements to this

Tlcsc:

Name	Requirements Allocation
Kalman_Update	R147
Compute_Kalman_Gain	R149
Update_Error_Covariance_Matrix	R150
Update_State_Vector	R151
Sequentially_Update_Covariance_Matrix_and_ State_Vector	R152

3.3.3.2.2 LOCAL ENTITIES DESIGN

None.

3.3.3.2.3 INPUT/OUTPUT

None.

3.3.3.2.4 LOCAL DATA

None.

3.3.3.2.5 PROCESS CONTROL

Not applicable.

3.3.3.2.6 PROCESSING

The following describes the processing performed by this part:

```
package body Kalman_Filter_Compact_H_Parts is

   function Compute_Kalman_Gain
               (P                    : P_Matrices;
                Measurement_Number   : Measurement_Indices;
```

```
                Compact_H           : Compact_H_Matrices;
                Measurement_Variance : Measurement_Variance_Vectors)
            return K_Column_Vectors is separate;

   procedure Update_Error_Covariance_Matrix
               (P                  : in out P_Matrices;
                Measurement_Number : in     Measurement_Indices;
                K                  : in     K_Column_Vectors;
                Compact_H          : in     Compact_H_Matrices)
               is separate;

   procedure Update_State_Vector
               (X                  : in out State_Vectors;
                Z                  : in     Measurement_Vectors;
                K                  : in     K_Column_Vectors;
                Measurement_Number : in     Measurement_Indices;
                Compact_H          : in     Compact_H_Matrices)
               is separate;

   package body Sequentially_Update_Covariance_Matrix_and_State_Vector
                  is separate;

   package body Kalman_Update is separate;

   procedure Update_Error_Covariance_Matrix_General_Form
               (P                  : in out P_Matrices;
                Measurement_Number : in     Measurement_Indices;
                K                  : in     K_Column_Vectors;
                Compact_H          : in     Compact_H_Matrices;
                Measurement_Variance : in   Measurement_Variance_Vectors)
            is separate;

end Kalman_Filter_Compact_H_Parts;
```

3.3.3.2.7 UTILIZATION OF OTHER ELEMENTS

None.

3.3.3.2.8 LIMITATIONS

None.

3.3.3.2.9 LLCSC DESIGN

3.3.3.2.9.1 SEQUENTIALLY UPDATE COVARIANCE MATRIX AND STATE VECTOR PACKAGE DESIGN (CATALOG #P141-0)

This LLCSC is a generic package which contains one procedure, "Update", which updates the Covariance Matrix, P, and state Vector, X.

The decomposition for this part is the same as that shown in the Top-Level Design Document.

3.3.3.2.9.1.1 REQUIREMENTS ALLOCATION

This part meets CAMP requirement R152.

3.3.3.2.9.1.2 LOCAL ENTITIES DESIGN

Packages:

As described above, this package body instantiates Part R149, Compute Kalman Gain, part R150, Update Error Covariance Matrix, and part R151, Update State Vector

3.3.3.2.9.1.3 INPUT/OUTPUT

GENERIC PARAMETERS:

Data types:

The following table summarizes the generic formal data types required by this part:

Name	Base Type	Description
State_Indices	discrete	Index to the arrays which depend on the number of states
Measurement_Indices	discrete	Index to the arrays which depend on the number of measurements
Kalman_Filter_Elements	floating point type	Elements making up the Kalman Filter aggregates
P_Matrices	private	Data type of P matrix
Measurement_Variance_Vectors	vector	Vector indexed by Measurement Indices containing Kalman_Filter_Elements
Measurement_Vectors	vector	Vector indexed by Measurement_Indices containing Kalman_Filter_Elements
P_Row_Vectors	vector	Vector indexed by State_Indices containing Kalman_Filter_Elements
K_Column_Vectors	vector	Vector indexed by State_Indices containing Kalman_Filter_Elements
State_Vectors	vector	Vector indexed by State_Indices containing Kalman_Filter_Elements
Compact_H_Matrices	vector	Data type of Compact H matrix

Subprograms:

The following table summarizes the generic formal subroutines required by this part:

Name	Type	Description
Element	function	Extracts an element of a P Matrix
Row	function	Extracts a row of a P matrix
"*"	function	A K Column Vector is multiplied by the transpose of a P Row vector, yielding a P matrix
"-"	function	Two P matrices are added, yielding a Symmetric matrix
"+"	function	Add a state vector and a K column vector, yielding a state vector
"*"	function	Multiply a K column vector by a Kalman Filter Element, yielding a K column vector

3.3.3.2.9.1.4 LOCAL DATA

None.

3.3.3.2.9.1.5 PROCESS CONTROL

Not applicable.

3.3.3.2.9.1.6 PROCESSING

The following describes the processing performed by this part:

```
separate (Kalman_Filter_Compact_H_Parts)
package body Sequentially_Update_Covariance_Matrix_and_State_Vector is

   K : K_Column_Vectors;


   function Compute_K is new Compute_Kalman_Gain
               (State_Indices           => State_Indices,
                Measurement_Indices     => Measurement_Indices,
                Kalman_Filter_Elements  => Kalman_Filter_Elements,
                P_Matrices              => P_Matrices,
                Measurement_Variance_Vectors
                                        => Measurement_Variance_Vectors,
                K_Column_Vectors        => K_Column_Vectors,
                Compact_H_Matrices      => Compact_H_Matrices);
```

```
   procedure Update_P is new Update_Error_Covariance_Matrix
             (State_Indices           => State_Indices,
              Measurement_Indices     => Measurement_Indices,
              Kalman_Filter_Elements  => Kalman_Filter_Elements,
              P_Matrices              => P_Matrices,
              P_Row_Vectors           => P_Row_Vectors,
              K_Column_Vectors        => K_Column_Vectors,
              Compact_H_Matrices      => Compact_H_Matrices);


   procedure Update_X is new Update_State_Vector
             (State_Indices           => State_Indices,
              Measurement_Indices     => Measurement_Indices,
              Kalman_Filter_Elements  => Kalman_Filter_Elements,
              Measurement_Vectors     => Measurement_Vectors,
              K_Column_Vectors        => K_Column_Vectors,
              State_Vectors           => State_Vectors,
              Compact_H_Matrices      => Compact_H_Matrices);

end Sequentially_Update_Covariance_Matrix_and_State_Vector;
```

3.3.3.2.9.1.7 UTILIZATION OF OTHER ELEMENTS

UTILIZATION OF OTHER ELEMENTS IN TOP LEVEL COMPONENT:

The following tables describe the elements used by this part but defined elsewhere in the parent top level component:

Subprograms and task entries:

The following table summarizes the subroutines and task entries required by this part and defined elsewhere in the parent top level component:

Name	Type	Source	Description
Compute_Kalman_Gain	generic function	Kalman_Filter_Common	Computes the Kalman Gain Vector, K
Update_State_Vector	generic procedure	Kalman_Filter_Common	Updates the State Vector, X
Update_Error_Covariance_Matrix	generic procedure	Kalman_Filter_Common	Updates the Error Covariance Matrix, P

3.3.3.2.9.1.8 LIMITATIONS

None.

3.3.3.2.9.1.9 LLCSC DESIGN

None.

3.3.3.2.9.1.10 UNIT DESIGN

3.3.3.2.9.1.10.1 UPDATE UNIT DESIGN

This unit is a procedure which does the update of the Covariance Matrix, P, and the State_Vector, X.

3.3.3.2.9.1.10.1.1 REQUIREMENTS ALLOCATION

This parts meets CAMP Requirement R152.

3.3.3.2.9.1.10.1.2 LOCAL ENTITIES DESIGN

None.

3.3.3.2.9.1.10.1.3 INPUT/OUTPUT

FORMAL PARAMETERS:

The following table describes this part's formal parameters:

Name	Type	Mode	Description
P	P_Matrices	in/out	Error Covariance Matrix (P) to be updated
X	State_Vectors	in/out	State Vector (X), to be updatd
Z	Measurement_ Vectors	in	Current Measurement Vector(Z)
Compact_H	Compact_H_ Matrices	in	Current Measurement Sensitivity Array (Compact H)
Measurement_ Variance	Measurement_ Variance_Vectors	in	Current Measurement Variance Array

3.3.3.2.9.1.10.1.4 LOCAL DATA

None.

3.3.3.2.9.1.10.1.5 PROCESS CONTROL

Not applicable.

3.3.3.2.9.1.10.1.6 PROCESSING

The following describes the processing performed by this part:

```
procedure Update
                (P                    : in out P_Matrices;
                 X                    : in out State_Vectors;
                 Z                    : in     Measurement_Vectors;
                 Compact_H            : in     Compact_H_Matrices;
                 Measurement_Variance : in     Measurement_Variance_Vectors) is

begin

   for Measurement_Number in Measurement_Indices loop

      K := Compute_K (P                    => P,
                      Measurement_Number   => Measurement_Number,
                      Compact_H            => Compact_H,
                      Measurement_Variance => Measurement_Variance);

      Update_P (P                  => P,
                Measurement_Number => Measurement_Number,
                K                  => K,
                Compact_H          => Compact_H);

      Update_X (X                  => X,
                Z                  => Z,
                K                  => K,
                Measurement_Number => Measurement_Number,
                Compact_H          => Compact_H);

   end loop;

end Update;
```

3.3.3.2.9.1.10.1.7 UTILIZATION OF OTHER ELEMENTS

UTILIZATION OF OTHER ELEMENTS IN TOP LEVEL COMPONENT:

The following tables describe the elements used by this part but defined elsewhere in the parent top level component:

Subprograms and task entries:

The following table summarizes the subroutines and task entries required by this part and defined elsewhere in the parent top level component:

Name	Type	Source	Description
Compute_Kalman_Gain	generic function	parent	Computes the Kalman Gain Vector, K
Update_State_Vector	generic procedure	parent	Updates the State Vector, X
Update_Error_Covariance_Matrix	generic procedure	parent	Updates the Error Covariance Matrix, P

Data types:

The following table summarizes the types required by this part and defined elsewhere in the parent top level component:

Name	Base Type	Description
Measurement_Indices	discrete	Index to the arrays which depend on the number of measurements
P_Matrices	private	Data type of P matrix
Measurement_Variance_Vectors	vector	Vector indexed by Measurement Indices containing Kalman_Filter_Elements
Measurement_Vectors	vector	Vector indexed by Measurement_Indices containing Kalman_Filter_Elements
State_Vectors	vector	Vector indexed by State_Indices containing Kalman_Filter_Elements
Compact_H_Matrices	vector	Data type of Compact H matrix

Data objects:

The following table summarizes the objects required by this part and defined elsewhere in the parent top level component:

Name	Type	Value	Description
K	K_Column_Vectors	variable	Stores the value of the Kalman Gain Vector (K)

3.3.3.2.9.1.10.1.8 LIMITATIONS

None.

3.3.3.2.9.2 KALMAN_UPDATE (BODY) PACKAGE DESIGN (CATALOG #P142-0)

This LLCSC is a generic package body which contains 1 procedure, "Update" which updates the State Vector, X, given the old X vector, the Z vector, the K vector, the Measurement Number, and the Compact H array.

The decomposition for this part is the same as that shown in the Top-Level Design Document.

3.3.3.2.9.2.1 REQUIREMENTS ALLOCATION

This part meets CAMP requirement R147.

3.3.3.2.9.2.2 LOCAL ENTITIES DESIGN

None.

3.3.3.2.9.2.3 INPUT/OUTPUT

GENERIC PARAMETERS:

Data types:

The following table summarizes the generic formal data types required by this part:

Name	Base Type	Description
State_Indices	discrete	Index to the arrays which depend on the number of states
Measurement_Indices	discrete	Index to the arrays which depend on the number of measurements
Kalman_Filter_Elements	floating point type	Elements making up the Kalman Filter aggregates
Phi_Matrices	private	Data type of Phi matrix
P_And_Q_Matrices	private	Data type of a P and Q matrix
Measurement_Variance_Vectors	vector	Vector indexed by Measurement Indices containing Kalman_Filter_Elements
Measurement_Vectors	vector	Vector indexed by Measurement_Indices containing Kalman_Filter_Elements
P_Row_Vectors	vector	Vector indexed by State_Indices containing Kalman_Filter_Elements
K_Column_Vectors	vector	Vector indexed by State_Indices containing Kalman_Filter_Elements
State_Vectors	vector	Vector indexed by State_Indices containing Kalman_Filter_Elements
Compact_H_Matrices	vector	Data type of Compact H matrix

Subprograms:

The following table summarizes the generic formal subroutines required by this part:

Name	Type	Description
Element	function	Extracts an element of a P and Q Matrix
Row	function	Extracts a row of a P and Q matrix
Phi_P_Phi_ Transpose	function	Performs an ABA transpose on a Phi Matrix and a P and Q Matrix
"*"	function	A K Column Vector is multiplied by the transpose of a P Row vector, yielding a P and Q matrix
"-"	function	Two P and Q matrices are added, yielding a P and Q matrix
"+"	function	Add a state vector and a K column vector, yielding a state vector
"*"	function	Multiply a K column vector by a Kalman Filter Element, yielding a K column vector

3.3.3.2.9.2.4 LOCAL DATA

None.

3.3.3.2.9.2.5 PROCESS CONTROL

Not applicable.

3.3.3.2.9.2.6 PROCESSING

The following describes the processing performed by this part:

```
with Kalman_Filter_Common_Parts;
separate (Kalman_Filter_Compact_H_Parts)
package body Kalman_Update is

   package P_Manager is new
             Kalman_Filter_Common_Parts.Error_Covariance_Matrix_Manager
                (Phi_Matrices     => Phi_Matrices,
                 P_And_Q_Matrices => P_And_Q_Matrices);

   package Update_P_And_X is new
             Sequentially_Update_Covariance_Matrix_And_State_Vector
                (State_Indices             => State_Indices,
                 Measurement_Indices       => Measurement_Indices,
                 Kalman_Filter_Elements    => Kalman_Filter_Elements,
                 P_Matrices                => P_And_Q_Matrices,
                 Measurement_Variance_Vectors
```

```
                                        => Measurement_Variance_Vectors,
                Measurement_Vectors     => Measurement_Vectors,
                P_Row_Vectors           => P_Row_Vectors,
                K_Column_Vectors        => K_Column_Vectors,
                State_Vectors           => State_Vectors,
                Compact_H_Matrices      => Compact_H_Matrices);

   Zero_State_Vector : State_Vectors := (others => 0.0);

end Kalman_Update;
```

3.3.3.2.9.2.7 UTILIZATION OF OTHER ELEMENTS

The following library units are with'd by this part:
1. Kalman_Filter_Common_Parts

UTILIZATION OF EXTERNAL ELEMENTS:

Packages:

The following table summarizes the external packages required by this part:

Name	Type	Source	Description
Covariance_Matrix_Manager	generic package	Kalman_Filter_Common	Manages the Covariance Matrix (P)

UTILIZATION OF OTHER ELEMENTS IN TOP LEVEL COMPONENT:

The following tables describe the elements used by this part but defined elsewhere in the parent top level component:

Packages:

The following table summarizes the packages required by this part and defined elsewhere in the parent top level component:

Name	Type	Source	Description
Sequentially_Update_Covariance_Matrix_And_State_Vector	generic package	parent	Updates the P matrix and X Vector

3.3.3.2.9.2.8 LIMITATIONS

None.

3.3.3.2.9.2.9 LLCSC DESIGN

None.

3.3.3.2.9.2.10 UNIT DESIGN

3.3.3.2.9.2.10.1 UPDATE UNIT DESIGN

This unit is a procedure which does the Kalman Update

3.3.3.2.9.2.10.1.1 REQUIREMENTS ALLOCATION

This parts meets CAMP Requirement R147.

3.3.3.2.9.2.10.1.2 LOCAL ENTITIES DESIGN

None.

3.3.3.2.9.2.10.1.3 INPUT/OUTPUT

FORMAL PARAMETERS:

The following table describes this part's formal parameters:

Name	Type	Mode	Description
X	State_Vectors	in/out	State Vector (X) to be updatd
P	P_And_Q_Matrices	in/out	Error Covariance Matrix (P) to be updated
Z	Measurement_ Vectors	in	Current Measurement Vector(Z)
Compact_H	Compact_H_ Matrices	in	Current Measurement Sensitivity Array (Compact H)
Measurement_ Variance	Measurement_ Variance_Vectors	in	Current Measurement Variance Array
Propagated_Phi	Phi_Matrices	in	Current value of the propagated State Transition Matrix
Propagated_Q	P_And_Q_Matrices	in	Current value of the propagated Process Noise Matrix

3.3.3.2.9.2.10.1.4 LOCAL DATA

None.

3.3.3.2.9.2.10.1.5 PROCESS CONTROL

Not applicable.

3.3.3.2.9.2.10.1.6 PROCESSING

The following describes the processing performed by this part:

```
   procedure Update
                    (X                    : in out State_Vectors;
                     P                    : in out P_And_Q_Matrices;
                     Z                    : in     Measurement_Vectors;
                     Compact_H            : in     Compact_H_Matrices;
                     Measurement_Variance : in     Measurement_Variance_Vectors;
                     Propagated_Phi       : in     Phi_Matrices;
                     Propagated_Q         : in     P_And_Q_Matrices) is
   begin

--       -- Propagate the Error Covariance Matrix

         P_Manager.Initialize (Initial_P => P);

         P_Manager.Propagate (Propagated_Phi => Propagated_Phi,
                              Propagated_Q   => Propagated_Q);

         P := P_Manager.P;


--       -- If the state vector is not zero, multiply it by Propagated Phi

         if X /= Zero_State_Vector then

            X := Propagated_Phi * X;

         end if;


--       -- Update Error Covariance Matrix and State Vector

         Update_P_And_X.Update (P                    => P,
                                X                    => X,
                                Z                    => Z,
                                Compact_H            => Compact_H,
                                Measurement_Variance => Measurement_Variance);

   end Update;
```

3.3.3.2.9.2.10.1.7 UTILIZATION OF OTHER ELEMENTS

UTILIZATION OF OTHER ELEMENTS IN TOP LEVEL COMPONENT:

The following tables describe the elements used by this part but defined elsewhere in the parent top level component:

Packages:

The following table summarizes the subroutines and task entries required by this part and defined elsewhere in the parent top level component:

Name	Type	Source	Description
P_Manager	package	parent	Manages the Error Covariance matrix (it is an instantiation of part R146)
Update_P_And_X	package	parent	Updates the Covariance Matrix and X Vector (it is an instantiation of part R152)

Data types:

The following table summarizes the types required by this part and defined elsewhere in the parent top level component:

Name	Base Type	Description
State_Indices	discrete	Index to the arrays which depend on the number of states
Measurement_Indices	discrete	Index to the arrays which depend on the number of measurements
Kalman_Filter_Elements	floating point type	Elements making up the Kalman Filter aggregates
Phi_Matrices	private	Data type of Phi matrix
P_And_Q_Matrices	private	Data type of a P and Q matrix
Measurement_Variance_Vectors	vector	Vector indexed by Measurement Indices containing Kalman_Filter_Elements
Measurement_Vectors	vector	Vector indexed by Measurement_Indices containing Kalman_Filter_Elements
State_Vectors	vector	Vector indexed by State_Indices containing Kalman_Filter_Elements
Compact_H_Matrices	vector	Data type of Compact H matrix

Data objects:

The following table summarizes the objects required by this part and defined elsewhere in the parent top level component:

Name	Type	Value	Description
Zero_State_Vector	State_Vector	vector of 0 values	Used for comparison to zero vector

3.3.3.2.9.2.10.1.8 LIMITATIONS

None.

3.3.3.2.10 UNIT DESIGN

3.3.3.2.10.1 COMPUTE_KALMAN_GAIN UNIT DESIGN (CATALOG #P138-0)

This unit is a generic function body which computes the Kalman gain vector resulting from the processing of a single component of the measurement vector, Z.

3.3.3.2.10.1.1 REQUIREMENTS ALLOCATION

This part meets CAMP requirement R149.

3.3.3.2.10.1.2 LOCAL ENTITIES DESIGN

None.

3.3.3.2.10.1.3 INPUT/OUTPUT

GENERIC PARAMETERS:

Data types:

The following table summarizes the generic formal data types required by this part:

Name	Base Type	Description
State_Indices	discrete	Index to the arrays which depend on the number of states
Measurement_Indices	discrete	Index to the arrays which depend on the number of measurements
Kalman_Filter_Elements	floating point type	Elements contained in the Kalman Filter aggregates
P_Matrices	private	Data type of P matrix
Measurement_Variance_Vectors	vector	Vector indexed by Measurement Indices containing Kalman_Filter_Elements
K_Column_Vectors	vector	Vector indexed by State_Indices containing Kalman_Filter_Elements
Compact_H_Matrices	vector	Data type of Compact H matrix

Subprograms:

The following table summarizes the generic formal subroutines required by this part:

Name	Type	Description
Element	function	Extracts an element of a P Matrix

3.3.3.2.10.1.4 LOCAL DATA

None.

3.3.3.2.10.1.5 PROCESS CONTROL

Not applicable.

3.3.3.2.10.1.6 PROCESSING

The following describes the processing performed by this part:

```
separate (Kalman_Filter_Compact_H_Parts)
function Compute_Kalman_Gain
            (P                    : P_Matrices;
             Measurement_Number   : Measurement_Indices;
             Compact_H            : Compact_H_Matrices;
             Measurement_Variance : Measurement_Variance_Vectors)
          return K_Column_Vectors is

   J : State_Indices;
   K : K_Column_Vectors;

begin

   J := Compact_H (Measurement_Number);

   for I in State_Indices loop

      K (I) := Element (P, I, J) /
               (Element (P, J, J) + Measurement_Variance (Measurement_Number));

   end loop;

   return K;

end Compute_Kalman_Gain;
```

3.3.3.2.10.1.7 UTILIZATION OF OTHER ELEMENTS

None.

3.3.3.2.10.1.8 LIMITATIONS

None.

3.3.3.2.10.2 UPDATE_ERROR_COVARIANCE_MATRIX UNIT DESIGN (CATALOG #P139-0)

This unit is a generic procedure body which computes the updated covariance matrix resulting from the processing of a single component of the measurement vector, Z.

3.3.3.2.10.2.1 REQUIREMENTS ALLOCATION

This part meets CAMP requirement R150.

3.3.3.2.10.2.2 LOCAL ENTITIES DESIGN

None.

3.3.3.2.10.2.3 INPUT/OUTPUT

GENERIC PARAMETERS:

Data types:

The following table summarizes the generic formal data types required by this part:

Name	Base Type	Description
State_Indices	discrete	Index to the arrays which depend on the number of states
Measurement_Indices	discrete	Index to the arrays which depend on the number of measurements
Kalman_Filter_Elements	floating point type	Elements contained in the Kalman Filter aggregates
P_Matrices	private	Data type of P matrix
P_Row_Vectors	vector	Vector indexed by State_Indices containing Kalman_Filter_Elements
K_Column_Vectors	vector	Vector indexed by State_Indices containing Kalman_Filter_Elements
Compact_H_Matrices	vector	Data type of Compact H matrix

Subprograms:

The following table summarizes the generic formal subroutines required by this part:

Name	Type	Description
Row	function	Extracts a row of a P matrix
"*"	function	A K Column Vector is multiplied by the transpose of a P Row vector, yielding a P matrix
"-"	function	Two P matrices are added, yielding a Symmetric matrix

3.3.3.2.10.2.4 LOCAL DATA

None.

3.3.3.2.10.2.5 PROCESS CONTROL

Not applicable.

3.3.3.2.10.2.6 PROCESSING

The following describes the processing performed by this part:

```
separate (Kalman_Filter_Compact_H_Parts)

procedure Update_Error_Covariance_Matrix
                (P                  : in out P_Matrices;
                 Measurement_Number : in     Measurement_Indices;
                 K                  : in     K_Column_Vectors;
                 Compact_H          : in     Compact_H_Matrices) is

   J : State_Indices;

begin

   J := Compact_H (Measurement_Number);

   P := P - K * Row (P, J);

end Update_Error_Covariance_Matrix;
```

3.3.3.2.10.2.7 UTILIZATION OF OTHER ELEMENTS

None.

3.3.3.2.10.2.8 LIMITATIONS

None.

3.3.3.2.10.3 UPDATE_STATE_VECTOR UNIT DESIGN (CATALOG #P140-0)

This unit is a generic procedure body which updates the State Vector, X, given the old X vector, the Z vector, the K vector, the Measurement Number, and the Compact H array.

3.3.3.2.10.3.1 REQUIREMENTS ALLOCATION

This part meets CAMP requirement R152.

3.3.3.2.10.3.2 LOCAL ENTITIES DESIGN

None.

3.3.3.2.10.3.3 INPUT/OUTPUT

GENERIC PARAMETERS:

Data types:

The following table summarizes the generic formal data types required by this part:

Name	Base Type	Description
State_Indices	discrete	Index to the arrays which depend on the number of states
Measurement_Indices	discrete	Index to the arrays which depend on the number of measurements
Kalman_Filter_Elements	floating point type	Elements making up the Kalman Filter aggregates
Measurement_Vectors	vector	Vector indexed by Measurement_Indices containing Kalman_Filter_Elements
K_Column_Vectors	vector	Vector indexed by State_Indices containing Kalman_Filter_Elements
State_Vectors	vector	Vector indexed by State_Indices containing Kalman_Filter_Elements
Compact_H_Matrices Vectors	vector	Vector indexed by Measurement_Indices containing State_Indices

Subprograms:

The following table summarizes the generic formal subroutines required by this part:

Name	Type	Description
"+"	function	Add a state vector and a K column vector, yielding a state vector
"*"	function	Multiply a K column vector by a Kalman Filter Element, yielding a K column vector

3.3.3.2.10.3.4 LOCAL DATA

None.

3.3.3.2.10.3.5 PROCESS CONTROL

Not applicable.

3.3.3.2.10.3.6 PROCESSING

The following describes the processing performed by this part:

```
separate (Kalman_Filter_Compact_H_Parts)
procedure Update_State_Vector
              (X                  : in out State_Vectors;
               Z                  : in     Measurement_Vectors;
               K                  : in     K_Column_Vectors;
               Measurement_Number : in     Measurement_Indices;
               Compact_H          : in     Compact_H_Matrices) is

   J : State_Indices;

begin

   J := Compact_H (Measurement_Number);

   X := K * (Z (Measurement_Number) - X (J)) + X;

end Update_State_Vector;
```

3.3.3.2.10.3.7 UTILIZATION OF OTHER ELEMENTS

None.

3.3.3.2.10.3.8 LIMITATIONS

None.

3.3.3.2.10.4 UPDATE_ERROR_COVARIANCE_MATRIX_GENERAL_FORM UNIT DESIGN (CATALOG #P1121-0)

This unit is a generic procedure body which computes the updated covariance matrix resulting from the processing of a single component of the measurement vector, Z.

3.3.3.2.10.4.1 REQUIREMENTS ALLOCATION

This part meets CAMP requirement R150.

3.3.3.2.10.4.2 LOCAL ENTITIES DESIGN

None.

3.3.3.2.10.4.3 INPUT/OUTPUT

GENERIC PARAMETERS:

Data types:

The following table summarizes the generic formal data types required by this part:

Name	Base Type	Description
State_Indices	discrete	Index to the arrays which depend on the number of states
Measurement_Indices	discrete	Index to the arrays which depend on the number of measurements
Kalman_Filter_Elements	floating point type	Elements contained in the Kalman Filter aggregates
P_Matrices	private	Type of covariance matrix. It represents a symmetric matrix index by (states, states)
K_H_Product_Matrices	private	A matrix of the form I_KH, where K is a K_Column_Vector and H is a row of the H matrix. It represents a matrix indexed by (states, states)
Measurement_Variance_	vector	Type of Measurement Variance Vector (R). It is indexed by (states)
K_Column_Vectors	vector	Type of a Column of the Kalman Gain Matrix (K). It is indexed by (states)
Compact_H_Matrices	vector	A vector, index by Measurement_Indices containing the indices of the measured states. It represents a matrix indexed by (Measurement, states) that is all zeroes except for locations specified by row=I, column= Compact_H_Matrices(I)

Subprograms:

The following table summarizes the generic formal subroutines required by this part:

Name	Type	Description
I_Minus_Column_ Matrix	function	K and the current state measured (as indicated by whicvh element of the current row of H is a "1") => I_KH
A_B_A_Transpose	function	K_H_Product Matrices * Kalman_Filter Elements * transpose(K_H_Product_ Matrices) => P_Matrices
A_B_A_Transpose	function	K_Colunm_Vectors * Kalman_Filter_ Elements * transpose(K_Column_ Vectors) => P_Matrices
"+"	function	P_Matrices + P_Matrices => P_Matrices

3.3.3.2.10.4.4 LOCAL DATA

None.

3.3.3.2.10.4.5 PROCESS CONTROL

Not applicable.

3.3.3.2.10.4.6 PROCESSING

The following describes the processing performed by this part:

```
separate (Kalman_Filter_Compact_H_Parts)

procedure Update_Error_Covariance_Matrix_General_Form
            (P                    : in out P_Matrices;
             Measurement_Number   : in     Measurement_Indices;
             K                    : in     K_Column_Vectors;
             Compact_H            : in     Compact_H_Matrices;
             Measurement_Variance : in     Measurement_Variance_Vectors) is

   I_Minus_K_H_Matrix : K_H_Product_Matrices;
   J                  : State_Indices;

begin

   J := Compact_H (Measurement_Number);

   I_Minus_K_H_Matrix := I_Minus_Column_Matrix (K             => K,
                                                State_Measure => J);

   P := ABA_Transpose (I_Minus_K_H_Matrix, P) +
        ABA_Transpose (K, Measurement_Variance( Measurement_Number ));

end Update_Error_Covariance_Matrix_General_Form;
```

3.3.3.2.10.4.7 UTILIZATION OF OTHER ELEMENTS

None.

3.3.3.2.10.4.8 LIMITATIONS

None.

```
package body Kalman_Filter_Compact_H_Parts is

   function Compute_Kalman_Gain
               (P                    : P_Matrices;
                Measurement_Number   : Measurement_Indices;
                Compact_H            : Compact_H_Matrices;
                Measurement_Variance : Measurement_Variance_Vectors)
            return K_Column_Vectors is separate;

   procedure Update_Error_Covariance_Matrix
               (P                  : in out P_Matrices;
                Measurement_Number : in     Measurement_Indices;
                K                  : in     K_Column_Vectors;
                Compact_H          : in     Compact_H_Matrices)
               is separate;

   procedure Update_State_Vector
               (X                  : in out State_Vectors;
                Z                  : in     Measurement_Vectors;
                K                  : in     K_Column_Vectors;
                Measurement_Number : in     Measurement_Indices;
                Compact_H          : in     Compact_H_Matrices)
               is separate;

   package body Sequentially_Update_Covariance_Matrix_And_State_Vector
                  is separate;

   package body Kalman_Update is separate;

   procedure Update_Error_Covariance_Matrix_General_Form
               (P                    : in out P_Matrices;
                Measurement_Number   : in     Measurement_Indices;
                K                    : in     K_Column_Vectors;
                Compact_H            : in     Compact_H_Matrices;
                Measurement_Variance : in     Measurement_Variance_Vectors)
            is separate;

end Kalman_Filter_Compact_H_Parts;
```

```
separate (Kalman_Filter_Compact_H_Parts)
function Compute_Kalman_Gain
            (P                    : P_Matrices;
             Measurement_Number   : Measurement_Indices;
             Compact_H            : Compact_H_Matrices;
             Measurement_Variance : Measurement_Variance_Vectors)
          return K_Column_Vectors is

   J : State_Indices;
   K : K_Column_Vectors;

begin

   J := Compact_H (Measurement_Number);

   for I in State_Indices loop

      K (I) := Element (P, I, J) /
               (Element (P, J, J) + Measurement_Variance (Measurement_Number));

   end loop;

   return K;

end Compute_Kalman_Gain;
```

```
separate (Kalman_Filter_Compact_H_Parts)

procedure Update_Error_Covariance_Matrix
                (P                  : in out P_Matrices;
                 Measurement_Number : in     Measurement_Indices;
                 K                  : in     K_Column_Vectors;
                 Compact_H          : in     Compact_H_Matrices) is

   J : State_Indices;

begin

   J := Compact_H (Measurement_Number);

   P := P - K * Row (P, J`·

end Update_Error_Covariance_Matrix;
```

```
separate (Kalman_Filter_Compact_H_Parts)
procedure Update_State_Vector
              (X                  : in out State_Vectors;
               Z                  : in     Measurement_Vectors;
               K                  : in     K_Column_Vectors;
               Measurement_Number : in     Measurement_Indices;
               Compact_H          : in     Compact_H_Matrices) is

   J : State_Indices;

begin

   J := Compact_H (Measurement_Number);

   X := K * (Z (Measurement_Number) - X (J)) + X;

end Update_State_Vector;
```

THIS REPORT HAS BEEN DELIMITED
AND CLEARED FOR PUBLIC RELEASE
UNDER DOD DIRECTIVE 5200.20 AND
NO RESTRICTIONS ARE IMPOSED UPON
ITS USE AND DISCLOSURE.

DISTRIBUTION STATEMENT A

APPROVED FOR PUBLIC RELEASE;
DISTRIBUTION UNLIMITED.

```
separate (Kalman_Filter_Compact_H_Parts)
package body Sequentially_Update_Covariance_Matrix_And_State_Vector is

   K : K_Column_Vectors;


   function Compute_K is new Compute_Kalman_Gain
               (State_Indices           => State_Indices,
                Measurement_Indices     => Measurement_Indices,
                Kalman_Filter_Elements  => Kalman_Filter_Elements,
                P_Matrices              => P_Matrices,
                Measurement_Variance_Vectors
                                        => Measurement_Variance_Vectors,
                K_Column_Vectors        => K_Column_Vectors,
                Compact_H_Matrices      => Compact_H_Matrices);

   procedure Update_P is new Update_Error_Covariance_Matrix
               (State_Indices           => State_Indices,
                Measurement_Indices     => Measurement_Indices,
                Kalman_Filter_Elements  => Kalman_Filter_Elements,
                P_Matrices              => P_Matrices,
                P_Row_Vectors           => P_Row_Vectors,
                K_Column_Vectors        => K_Column_Vectors,
                Compact_H_Matrices      => Compact_H_Matrices);


   procedure Update_X is new Update_State_Vector
               (State_Indices           => State_Indices,
                Measurement_Indices     => Measurement_Indices,
                Kalman_Filter_Elements  => Kalman_Filter_Elements,
                Measurement_Vectors     => Measurement_Vectors,
                K_Column_Vectors        => K_Column_Vectors,
                State_Vectors           => State_Vectors,
                Compact_H_Matrices      => Compact_H_Matrices);

                                                          pragma PAGE;
   procedure Update
                (P                    : in out P_Matrices;
                 X                    : in out State_Vectors;
                 Z                    : in     Measurement_Vectors;
                 Compact_H            : in     Compact_H_Matrices;
                 Measurement_Variance : in     Measurement_Variance_Vectors) is

   begin

      for Measurement_Number in Measurement_Indices loop

         K := Compute_K (P                    => P,
                         Measurement_Number   => Measurement_Number,
                         Compact_H            => Compact_H,
                         Measurement_Variance => Measurement_Variance);

         Update_P (P                  => P,
                   Measurement_Number => Measurement_Number,
                   K                  => K,
                   Compact_H          => Compact_H);
```

```
         Update_X (X                  => X,
                   Z                  => Z,
                   K                  => K,
                   Measurement_Number => Measurement_Number,
                   Compact_H          => Compact_H);

      end loop;

   end Update;

end Sequentially_Update_Covariance_Matrix_And_State_Vector;
```

```
with Kalman_Filter_Common_Parts;
separate (Kalman_Filter_Compact_H_Parts)
package body Kalman_Update is

   package P_Manager is new
             Kalman_Filter_Common_Parts.Error_Covariance_Matrix_Manager
                (Phi_Matrices     => Phi_Matrices,
                 P_And_Q_Matrices => P_And_Q_Matrices);

   package Update_P_And_X is new
             Sequentially_Update_Covariance_Matrix_And_State_Vector
                (State_Indices             => State_Indices,
                 Measurement_Indices       => Measurement_Indices,
                 Kalman_Filter_Elements    => Kalman_Filter_Elements,
                 P_Matrices                => P_And_Q_Matrices,
                 Measurement_Variance_Vectors
                                           => Measurement_Variance_Vectors,
                 Measurement_Vectors       => Measurement_Vectors,
                 P_Row_Vectors             => P_Row_Vectors,
                 K_Column_Vectors          => K_Column_Vectors,
                 State_Vectors             => State_Vectors,
                 Compact_H_Matrices        => Compact_H_Matrices);

   Zero_State_Vector : State_Vectors := (others => 0.0);

                                                          pragma PAGE;
   procedure Update
                  (X                    : in out State_Vectors;
                   P                    : in out P_And_Q_Matrices;
                   Z                    : in     Measurement_Vectors;
                   Compact_H            : in     Compact_H_Matrices;
                   Measurement_Variance : in     Measurement_Variance_Vectors;
                   Propagated_Phi       : in     Phi_Matrices;
                   Propagated_Q         : in     P_And_Q_Matrices) is
   begin

--    -- Propagate the Error Covariance Matrix

      P_Manager.Initialize (Initial_P => P);

      P_Manager.Propagate (Propagated_Phi => Propagated_Phi,
                           Propagated_Q   => Propagated_Q);

      P := P_Manager.P;


--    -- If the state vector is not zero, multiply it by Propagated Phi

      if X /= Zero_State_Vector then

         X := Propagated_Phi * X;

      end if;


--    -- Update Error Covariance Matrix and State Vector
```

```
      Update_P_And_X.Update (P                  => P,
                             X                  => X,
                             Z                  => Z,
                             Compact_H          => Compact_H,
                             Measurement_Variance => Measurement_Variance);

   end Update;

end Kalman_Update;
```

```
separate (Kalman_Filter_Compact_H_Parts)

procedure Update_Error_Covariance_Matrix_General_Form
            (P                    : in out P_Matrices;
             Measurement_Number   : in     Measurement_Indices;
             K                    : in     K_Column_Vectors;
             Compact_H            : in     Compact_H_Matrices;
             Measurement_Variance : in     Measurement_Variance_Vectors) is

   I_Minus_K_H_Matrix : K_H_Product_Matrices;
   J                  : State_Indices;

begin

   J := Compact_H (Measurement_Number);

   I_Minus_K_H_Matrix := I_Minus_Column_Matrix (K             => K,
                                                State_Measure => J);

   P := Aba_Transpose (I_Minus_K_H_Matrix, P) +
        Aba_Transpose (K, Measurement_Variance( Measurement_Number ));

end Update_Error_Covariance_Matrix_General_Form;
```

(This page left intentionally blank.)

3.3.3.3 KALMAN FILTER COMPLICATED H PARTS (BODY) TLCSC (CATALOG #P149-0)

This part, which is designed as an Ada package, contains bodies for all CAMP parts which can be used to implement a Kalman Filter when a complicated Measurement Sensitivity Matrix (Complicated H Matrix) is used

The decomposition for this part is the same as that shown in the Top-Level Design Document.

3.3.3.3.1 REQUIREMENTS ALLOCATION

The following chart summarizes the allocation of CAMP requirements to this

Tlcsc:

Name	Requirements Allocation
Compute_Kalman_Gain	R182
Update_Error_Covariance_Matrix	R183
Update_State_Vector	R184
Sequentially_Update_Covariance_Matrix_and_ State_Vector	R201
Kalman_Update	R181
Update_Error_Covariance_Matrix_General_Form	R183

3.3.3.3.2 LOCAL ENTITIES DESIGN

None.

3.3.3.3.3 INPUT/OUTPUT

None.

3.3.3.3.4 LOCAL DATA

None.

3.3.3.3.5 PROCESS CONTROL

Not applicable.

3.3.3.3.6 PROCESSING

The following describes the processing performed by this part:

```
package body Kalman_Filter_Complicated_H_Parts is

   function Compute_Kalman_Gain
               ( P                    : P_Matrices;
```

```
                Measurement_Number   : Measurement_Indices;
                Complicated_H        : H_Matrices;
                Measurement_Variance : Measurement_Variance_Vectors)
             return K_Column_Vectors is separate;

   procedure Update_Error_Covariance_Matrix
             (P                 : in out P_Matrices;
              Measurement_Number : in     Measurement_Indices;
              K                 : in     K_Column_Vectors;
              Complicated_H     : in     H_Matrices) is separate;

   procedure Update_State_Vector
             (X                 : in out State_Vectors;
              Z                 : in     Measurement_Vectors;
              K                 : in     K_Column_Vectors;
              Measurement_Number : in     Measurement_Indices;
              Complicated_H     : in     H_Matrices) is separate;

   package body Sequentially_Update_Covariance_Matrix_and_State_Vector is separate;

   package body Kalman_Update is separate;

   procedure Update_Error_Covariance_Matrix_General_Form
             ( P                  : in out P_Matrices;
               Measurement_Number  : in     Measurement_Indices;
               K                  : in     K_Column_Vectors;
               Complicated_H      : in     H_Matrices;
               Measurement_Variance : in     Measurement_Variance_Vectors )
          is separate;

end Kalman_Filter_Complicated_H_Parts;
```

3.3.3.3.7 UTILIZATION OF OTHER ELEMENTS

None.

3.3.3.3.8 LIMITATIONS

None.

3.3.3.3.9 LLCSC DESIGN

3.3.3.3.9.1 SEQUENTIALLY UPDATE COVARIANCE MATRIX AND STATE VECTOR PACKAGE DESIGN (CATALOG #P153-0)

This LLCSC is a generic package which contains 1 procedure, "Update", which updates the Covariance Matrix, P, and state Vector, X.

The decomposition for this part is the same as that shown in the Top-Level Design Document.

3.3.3.3.9.1.1 REQUIREMENTS ALLOCATION

This part meets CAMP requirement R152.

3.3.3.3.9.1.2 LOCAL ENTITIES DESIGN

Packages:

As described above, this package body instantiates Compute Kalman Gain, Update Error Covariance Matrix, and Update State Vector

3.3.3.3.9.1.3 INPUT/OUTPUT

GENERIC PARAMETERS:

Data types:

The following table describes the generic formal types required by this part:

Name	Base Type	Description
States_Indices	discrete	Index to the arrays which depend on the number of states
Measurement_Indices	discrete	Index to the arrays which depend on the number of measurements
Kalman_Filter_Elements	floating point type	Elements contained in the Kalman Filter aggregates
P_Matrices	private	Data type for n x n private P matrix
H_Matrices	private	Data type for private m x n H matrix
K_H_Product_Matrices	private	Data type for private n x n K and H matrix
H_Row_Vectors	private	Vector representing a row of a P matrix
Measurement_Vectors	vector	Vector indexed by Measurement_Indices containing Kalman_Filter_Elements
Measurement_Variance_Vectors	vector	Vector indexed by Measurement_Indices containing Kalman_Filter_Elements
K_Column_Vectors	vector	Vector indexed by State_Indices containing Kalman_Filter_Elements
State_Vectors	vector	Vector indexed by State_Indices containing Kalman_Filter_Elements

Subprograms:

The following table summarizes the generic formal subroutines required by this part:

Name	Type	Description
Active_H_Vector	function	Returns the row of an (MxN) H matrix that is referenced
Subtract_From_ Identity	procedure	Subtracts a K and H Product matrix from the Identity Matrix(i.e., I - S)
Times_Transpose	function	Multiplies a P matrix by the transpose of a H Row Vector, yielding a K Column Vector
"*"	function	Multiplies a H Row Vector by a K Column vector, yielding a Kalman Filter Element
"*"	function	Multiplies a H Row Vector by a State vector, yielding a Kalman Filter Element
"*"	function	Multiplies a K Column Vector by a H Row Vector, yielding a K and H Product Matrix
"*"	function	Multiplies a K and H Product Matrix by a P Matrix, yielding a P Matrix
"*"	function	Multiplies a K Column Vector by a Kalman Filter Element, yielding a K Column Vector
"/"	function	Divides a K Column Vector by a Kalman Filter Element, yielding a K Column Vector
"+"	function	Adds a State Vector and a K Column Vector, yielding a State Vector

3.3.3.3.9.1.4 LOCAL DATA

None.

3.3.3.3.9.1.5 PROCESS CONTROL

Not applicable.

3.3.3.3.9.1.6 PROCESSING

The following describes the processing performed by this part:

separate (Kalman_Filter_Complicated_H_Parts)

```
package body Sequentially_Update_Covariance_Matrix_and_State_Vector is

   K : K_Column_Vectors;


   function Compute_K is new Compute_Kalman_Gain
               (State_Indices              => State_Indices,
                Measurement_Indices        => Measurement_Indices,
                Kalman_Filter_Elements     => Kalman_Filter_Elements,
                P_Matrices                 => P_Matrices,
                H_Matrices                 => H_Matrices,
                H_Row_Vectors              => H_Row_Vectors,
                Measurement_Variance_Vectors
                                           => Measurement_Variance_Vectors,
                K_Column_Vectors           => K_Column_Vectors);


   procedure Update_P is new Update_Error_Covariance_Matrix
               (State_Indices              => State_Indices,
                Measurement_Indices        => Measurement_Indices,
                Kalman_Filter_Elements     => Kalman_Filter_Elements,
                P_Matrices                 => P_Matrices,
                H_Matrices                 => H_Matrices,
                H_Row_Vectors              => H_Row_Vectors,
                K_H_Product_Matrices       => K_H_Product_Matrices,
                K_Column_Vectors           => K_Column_Vectors);


   procedure Update_X is new Update_State_Vector
               (State_Indices          => State_Indices,
                Measurement_Indices    => Measurement_Indices,
                Kalman_Filter_Elements => Kalman_Filter_Elements,
                H_Matrices             => H_Matrices,
                H_Row_Vectors          => H_Row_Vectors,
                Measurement_Vectors    => Measurement_Vectors,
                K_Column_Vectors       => K_Column_Vectors,
                State_Vectors          => State_Vectors);

end Sequentially_Update_Covariance_Matrix_and_State_Vector;
```

3.3.3.3.9.1.7 UTILIZATION OF OTHER ELEMENTS

UTILIZATION OF OTHER ELEMENTS IN TOP LEVEL COMPONENT:

The following tables describe the elements used by this part but defined elsewhere in the parent top level component:

Subprograms and task entries:

The following table summarizes the subroutines and task entries required by this part and defined elsewhere in the parent top level component:

Name	Type	Source	Description
Compute_Kalman_Gain	generic function	parent	Computes the Kalman Gain Vector, K
Update_State_Vector	generic procedure	parent	Updates the State Vector, X
Update_Error_Covariance_Matrix	generic procedure	parent	Updates the Error Covariance Matrix, P

3.3.3.3.9.1.8 LIMITATIONS

None.

3.3.3.3.9.1.9 LLCSC DESIGN

None.

3.3.3.3.9.1.10 UNIT DESIGN

3.3.3.3.9.1.10.1 UPDATE UNIT DESIGN

This unit is a procedure which does the update of the Covariance Matrix, P, and the State_Vector, X.

3.3.3.3.9.1.10.1.1 REQUIREMENTS ALLOCATION

This parts meets CAMP Requirement R201.

3.3.3.3.9.1.10.1.2 LOCAL ENTITIES DESIGN

None.

3.3.3.3.9.1.10.1.3 INPUT/OUTPUT

FORMAL PARAMETERS:

The following table describes this part's formal parameters:

Name	Type	Mode	Description
P	P_Matrices	in/out	Error Covariance Matrix (P) to be updated
X	State_Vectors	in/out	State Vector (X) to be updatd
Z	Measurement_ Vectors	in	Current Measurement Vector(Z)
Complicated_H	H_Matrices	in	Current Measurement Sensitivity Matrix (Complicated H)
Measurement_ Variance	Measurement_ Variance_	in	Current Measurement Variance Array

3.3.3.3.9.1.10.1.4 LOCAL DATA

None.

3.3.3.3.9.1.10.1.5 PROCESS CONTROL

Not applicable.

3.3.3.3.9.1.10.1.6 PROCESSING

The following describes the processing performed by this part:

```
   procedure Update
               (P                    : in out P_Matrices;
                X                    : in out State_Vectors;
                Z                    : in     Measurement_Vectors;
                Complicated_H        : in     H_Matrices;
                Measurement_Variance : in     Measurement_Variance_Vectors) is
   begin

      for Measurement_Number in Measurement_Indices loop

         K := Compute_K (P                    => P,
                         Measurement_Number   => Measurement_Number,
                         Complicated_H        => Complicated_H,
                         Measurement_Variance => Measurement_Variance);

         Update_P (P                  => P,
                   Measurement_Number => Measurement_Number,
                   K                  => K,
                   Complicated_H      => Complicated_H);

         Update_X (X                  => X,
                   Z                  => Z,
                   K                  => K,
                   Measurement_Number => Measurement_Number,
```

```
                    Complicated_H      => Complicated_H);

      end loop;

   end Update;
```

3.3.3.3.9.1.10.1.7 UTILIZATION OF OTHER ELEMENTS

UTILIZATION OF OTHER ELEMENTS IN TOP LEVEL COMPONENT:

The following tables describe the elements used by this part but defined elsewhere in the parent top level component:

Subprograms and task entries:

The following table summarizes the subroutines and task entries required by this part and defined elsewhere in the parent top level component:

Name	Type	Source	Description
Compute_Kalman_Gain	generic function	parent	Computes the Kalman Gain Vector, K
Update_State_Vector	generic procedure	parent	Updates the State Vector, X
Update_Error_Covariance_Matrix	generic procedure	parent	Updates the Error Covariance Matrix, P

Data types:

The following table summarizes the types required by this part and defined elsewhere in the parent top level component:

Name	Base Type	Description
States_Indices	discrete	Index to the arrays which depend on the number of states
Measurement_Indices	discrete	Index to the arrays which depend on the number of measurements
Kalman_Filter_Elements	floating point type	Elements contained in the Kalman Filter aggregates
P_Matrices	private	Data type for n x n private P matrix
H_Matrices	private	Data type for private m x n H matrix
Measurement_Vectors	vector	Vector indexed by Measurement_Indices containing Kalman_Filter_Elements
Measurement_Variance_Vectors	vector	Vector indexed by Measurement_Indices containing Kalman_Filter_Elements
K_Column_Vectors	vector	Vector indexed by State_Indices containing Kalman_Filter_Elements
State_Vectors	vector	Vector indexed by State_Indices containing Kalman_Filter_Elements

Data objects:

The following table summarizes the objects required by this part and defined elsewhere in the parent top level component:

Name	Type	Value	Description
K	K_Column_Vectors	variable	Stores the value of the Kalman Gain Vector (K)

3.3.3.3.9.1.10.1.8 LIMITATIONS

None.

3.3.3.3.9.2 KALMAN_UPDATE PACKAGE DESIGN (CATALOG #P155-0)

This LLCSC is a generic package which updates the State Vector, X, given the old X vector, the Z vector, the K vector, the Measurement Number, and the Complicated H array.

The decomposition for this part is the same as that shown in the Top-Level Design Document.

3.3.3.3.9.2.1 REQUIREMENTS ALLOCATION

This part meets CAMP requirement R181.

3.3.3.3.9.2.2 LOCAL ENTITIES DESIGN

Packages:

As mentioned above, this package instantiates Sequentially Update Covariance Matrix and State Vector, and Error Covariance Matrix Manager

3.3.3.3.9.2.3 INPUT/OUTPUT

GENERIC PARAMETERS:

Data types:

The following table describes the generic formal types required by this part:

Name	Base Type	Description
States_Indices	discrete	Index to the arrays which depend on the number of states
Measurement_Indices	discrete	Index to the arrays which depend on the number of measurements
Kalman_Filter_Elements	floating point type	Elements contained in the Kalman Filter aggregates
Phi_Matrices	private	Data type for private n x n Phi matrix
P_And_Q_Matrices	private	Data type for n x n private P matrix
H_Matrices	private	Data type for private m x n H matrix
K_H_Product_Matrices	private	Data type for private n x n K and H matrix
H_Row_Vectors	private	Vector representing a row of a P matrix
Measurement_Vectors	vector	Vector indexed by Measurement_Indices containing Kalman_Filter_Elements
Measurement_Variance_Vectors	vector	Vector indexed by Measurement_Indices containing Kalman_Filter_Elements
K_Column_Vectors	vector	Vector indexed by State_Indices containing Kalman_Filter_Elements
State_Vectors	vector	Vector indexed by State_Indices containing Kalman_Filter_Elements

Subprograms:

The following table summarizes the generic formal subroutines required by this part:

Name	Type	Description
Active_H_Vector	function	Returns the row of an (MxN) H matrix that is referenced
Subtract_From_ Identity	procedure	Subtracts a K and H Product matrix from the Identity Matrix(i.e., I - S)
Times_Transpose	function	Multiplies a P matrix by the transpose of a H Row Vector, yielding a K Column Vector
Phi_P_Phi_	function	Does an ABA Transpose opertion on a Phi matrix and a P and Q matrix, yielding a P and Q matrix
"*"	function	Multiplies a H Row Vector by a K Column vector, yielding a Kalman Filter Element
"*"	function	Multiplies a H Row Vector by a State vector, yielding a Kalman Filter Element
"*"	function	Multiplies a K Column Vector by a H Row Vector, yielding a K and H Product Matrix
"*"	function	Multiplies a K and H Product Matrix by a P Matrix, yielding a P Matrix
"*"	function	Multiplies a K Column Vector by a Kalman Filter Element, yielding a K Column Vector
"*"	function	Multiplies a Phi Matrix by a State Vector yielding a State Vector
"/"	function	Divides a K Column Vector by a Kalman Filter Element, yielding a K Column Vector
"+"	function	Adds a State Vector and a K Column Vector, yielding a State Vector

3.3.3.3.9.2.4 LOCAL DATA

Data objects:

The following table describes the data objects maintained by this part:

Name	Type	Value	Description
Zero_State_Vector	State_Vector	vector of 0 values	Used for comparison to zero vector

3.3.3.3.9.2.5 PROCESS CONTROL

Not applicable.

3.3.3.3.9.2.6 PROCESSING

The following describes the processing performed by this part:

```
with Kalman_Filter_Common_Parts;
separate (Kalman_Filter_Complicated_H_Parts)
package body Kalman_Update is

   package P_Manager is new
             Kalman_Filter_Common_Parts.Error_Covariance_Matrix_Manager
                ( Phi_Matrices     => Phi_Matrices,
                  P_And_Q_Matrices => P_And_Q_Matrices );

   package Update_P_And_X is new
             Sequentially_Update_Covariance_Matrix_And_State_Vector
                (State_Indices                => State_Indices,
                 Measurement_Indices          => Measurement_Indices,
                 Kalman_Filter_Elements       => Kalman_Filter_Elements,
                 P_Matrices                   => P_And_Q_Matrices,
                 H_Matrices                   => H_Matrices,
                 K_H_Product_Matrices         => K_H_Product_Matrices,
                 H_Row_Vectors                => H_Row_Vectors,
                 Measurement_Variance_Vectors => Measurement_Variance_Vectors,
                 Measurement_Vectors          => Measurement_Vectors,
                 K_Column_Vectors             => K_Column_Vectors,
                 State_Vectors                => State_Vectors );

   Zero_State_Vector : State_Vectors := (others => 0.0);

end Kalman_Update;
```

3.3.3.3.9.2.7 UTILIZATION OF OTHER ELEMENTS

The following library units are with'd by this part:
1. Kalman_Filter_Common_Parts

UTILIZATION OF EXTERNAL ELEMENTS:

Packages:

The following table summarizes the external packages required by this part:

Name	Type	Source	Description
Error_Covariance_Matrix_Manager	generic package	Kalman_Filter_Common_Parts	Manages the Covariance Matrix (P)

UTILIZATION OF OTHER ELEMENTS IN TOP LEVEL COMPONENT:

The following tables describe the elements used by this part but defined elsewhere in the parent top level component:

Packages:

The following table summarizes the packages required by this part and defined elsewhere in the parent top level component:

Name	Type	Source	Description
Sequentially_Update_Covariance_Matrix_And_State_Vector	generic package	parent	Updates the P matrix and X Vector

3.3.3.3.9.2.8 LIMITATIONS

None.

3.3.3.3.9.2.9 LLCSC DESIGN

None.

3.3.3.3.9.2.10 UNIT DESIGN

3.3.3.3.9.2.10.1 UPDATE UNIT DESIGN

This unit is a procedure which does the Kalman Update

3.3.3.3.9.2.10.1.1 REQUIREMENTS ALLOCATION

This parts meets CAMP Requirement R181.

3.3.3.3.9.2.10.1.2 LOCAL ENTITIES DESIGN

None.

3.3.3.3.9.2.10.1.3 INPUT/OUTPUT

FORMAL PARAMETERS:

The following table describes this part's formal parameters:

Name	Type	Mode	Description
P	P_And_Q_Matrices	in/out	Error Covariance Matrix (P) to be updated
X	State_Vectors	in/out	State Vector (X) to be updatd
Z	Measurement_Vectors	in	Current Measurement Vector(Z)
Complicated_H	H_Matrices	in	Current Measurement Sensitivity Matrix (Complicated H)
Measurement_Variance	Measurement_Variance_Vectors	in	Current Measurement Variance Array
Propagated_Phi	Phi_Matrices	in	Current value of the propagated State Transition Matrix
Propagated_Q	P_And_Q_Matrices	in	Current value of the propagated Process Noise Matrix

3.3.3.3.9.2.10.1.4 LOCAL DATA

None.

3.3.3.3.9.2.10.1.5 PROCESS CONTROL

Not applicable.

3.3.3.3.9.2.10.1.6 PROCESSING

The following describes the processing performed by this part:

```
   procedure Update (X                    : in out State_Vectors;
                     P                    : in out P_And_Q_Matrices;
                     Z                    : in     Measurement_Vectors;
                     Complicated_H        : in     H_Matrices;
                     Measurement_Variance : in     Measurement_Variance_Vectors;
                     Propagated_Phi       : in     Phi_Matrices;
                     Propagated_Q         : in     P_And_Q_Matrices) is

   begin

--     -- Propagate the Error Covariance Matrix
```

```
      P_Manager.Initialize (Initial_P => P);

      P_Manager.Propagate (Propagated_Phi => Propagated_Phi,
                           Propagated_Q   => Propagated_Q);

      P := P_Manager.P;

--    -- If the state vector is not zero, multiply it by Propagated Phi

      if X /= Zero_State_Vector then

         X := Propagated_Phi * X;

      end if;

--    -- Update Error Covariance Matrix and State Vector

      Update_P_And_X.Update (P                    => P,
                             X                    => X,
                             Z                    => Z,
                             Complicated_H        => Complicated_H,
                             Measurement_Variance => Measurement_Variance);

   end Update;
```

3.3.3.3.9.2.10.1.7 UTILIZATION OF OTHER ELEMENTS

UTILIZATION OF OTHER ELEMENTS IN TOP LEVEL COMPONENT:

The following tables describe the elements used by this part but defined elsewhere in the parent top level component:

Packages:

The following table summarizes the subroutines and task entries required by this part and defined elsewhere in the parent top level component:

Name	Type	Source	Description
P_Manager	package	Kalman_ Filter_ Common_ Parts	Manages the Error Covariance matrix (it is an instantiation of part R146)
Update_P_And_X	package	parent	Updates the Covariance Matrix and X Vector (it is an instantiation of part R201)

Data types:

The following table summarizes the types required by this part and defined elsewhere in the parent top level component:

Name	Base Type	Description
States_Indices	discrete	Index to the arrays which depend on the number of states
Measurement_Indices	discrete	Index to the arrays which depend on the number of measurements
Kalman_Filter_Elements	floating point type	Elements contained in the Kalman Filter aggregates
Phi_Matrices	private	Data type for private n x n Phi matrix
P_And_Q_Matrices	private	Data type for n x n private P matrix
H_Matrices	private	Data type for private m x n H matrix
K_H_Product_Matrices	private	Data type for private n x n K and H matrix
H_Row_Vectors	private	Vector representing a row of a P matrix
Measurement_Vectors	vector	Vector indexed by Measurement_Indices containing Kalman_Filter_Elements
Measurement_Variance_Vectors	vector	Vector indexed by Measurement_Indices containing Kalman_Filter_Elements
K_Column_Vectors	vector	Vector indexed by State_Indices containing Kalman_Filter_Elements
State_Vectors	vector	Vector indexed by State_Indices containing Kalman_Filter_Elements

Data objects:

The following table summarizes the objects required by this part and defined elsewhere in the parent top level component:

Name	Type	Value	Description
Zero_State_Vector	State_Vectors	vector of 0 values	Used for comparison to zero vector

3.3.3.3.9.2.10.1.8 LIMITATIONS

None.

3.3.3.3.10 UNIT DESIGN

3.3.3.3.10.1 COMPUTE_KALMAN_GAIN (BODY) UNIT DESIGN (CATALOG #P150-0)

This LLCSC is a generic function body which computes the Kalman gain vector resulting from the processing of a single component of the measurement vector, Z.

3.3.3.3.10.1.1 REQUIREMENTS ALLOCATION

This part meets CAMP requirement R182.

3.3.3.3.10.1.2 LOCAL ENTITIES DESIGN

None.

3.3.3.3.10.1.3 INPUT/OUTPUT

GENERIC PARAMETERS:

Data types:

The following table describes the generic formal types required by this part:

Name	Base Type	Description
State_Indices	discrete	Index to the arrays which depend on the number of states
Measurement_Indices	discrete	Index to the arrays which depend on the number of measurements
Kalman_Filter_Elements	floating point type	Elements contained in the Kalman Filter aggregates
P_Matrices	private	Data type for n x n P matrixx
H_Matrices	private	Data type for n x n H matrix
H_Row_Vectors	private	Data type of vector representing an H vector
Measurement_Variance_Vectors	array	Vector indexed by Measurement_Indices containing Kalman_Filter_Elements
K_Column_Vectors	array	Vector indexed by State_Indices containing Kalman_Filter_Elements

Subprograms:

The following table summarizes the generic formal subroutines required by this part:

Name	Type	Description
Active_H_Vector	function	Returns the row of an (MxN) H matrix that is referenced
Times_Transpose	function	Multiplies a P matrix by the transpose of a n x 1 H Row Vector, yielding a K Column Vector
Dot_Product	function	Multiplies a (Nx1) H Row Vector by a K Column Vector, yielding a Kalman Filter Element
"/"	function	Divides a (Nx1) K Column Vector by a Kalman Filter Element, yielding a K Column Vector

3.3.3.3.10.1.4 LOCAL DATA

Data objects:

The following table describes the data objects maintained by this part:

Name	Type	Value	Description
Denominator	Kalman_Filter_Elements	variable	One element of the Error Covariance matrix
K	K_Column_Vectors	variable	Vector which is calculated and returned
V	H_Row_Vectors	variable	Column slice of the H matrix representing the current measurement

3.3.3.3.10.1.5 PROCESS CONTROL

Not applicable.

3.3.3.3.10.1.6 PROCESSING

The following describes the processing performed by this part:

```
separate (Kalman_Filter_Complicated_H_Parts)

function Compute_Kalman_Gain
            (P                    : P_Matrices;
             Measurement_Number   : Measurement_Indices;
             Complicated_H        : H_Matrices;
             Measurement_Variance : Measurement_Variance_Vectors)
            return K_Column_Vectors is


   Denominator : Kalman_Filter_Elements;
   K           : K_Column_Vectors;
   V           : H_Row_Vectors;

begin

   V := Active_H_Vector (Source => Complicated_H,
                         Row    => Measurement_Number);

   K := Times_Transpose (Left  => P,
                         Right => V);

   Denominator := Dot_Product(Left  => V,
                              Right => K) +
                  Measurement_Variance( Measurement_Number );

   return K / Denominator;

end Compute_Kalman_Gain;
```

3.3.3.3.10.1.7 UTILIZATION OF OTHER ELEMENTS

None.

3.3.3.3.10.1.8 LIMITATIONS

None.

3.3.3.3.10.2 UPDATE_ERROR_COVARIANCE_MATRIX (BODY) UNIT DESIGN (CATALOG #P151-0)

This unit is a generic procedure body which computes the updated covariance matrix resulting from the processing of a single component of the measurement vector, Z.

3.3.3.3.10.2.1 REQUIREMENTS ALLOCATION

This part meets CAMP requirement R183.

3.3.3.3.10.2.2 LOCAL ENTITIES DESIGN

None.

3.3.3.3.10.2.3 INPUT/OUTPUT

GENERIC PARAMETERS:

Data types:

The following table describes the generic formal types required by this part:

Name	Base Type	Description
State_Indices	discrete	Index to the arrays which depend on the number of states
Measurement_Indices	discrete	Index to the arrays which depend on the number of measurements
Kalman_Filter_Elements	floating point type	Elements contained in the Kalman Filter aggregates
P_Matrices	private	Data type for private n x n P matrix
H_Matrices	private	Data type for private H matrix
H_Row_Vectors	private	Private vector representing a row of the H matrix
K_H_Product_Matrices	array	Private matrix representing product of K and H matrices
K_Column_Vectors	vector	Vector indexed by State_Indices containing Kalman_Filter_Elements

Subprograms:

The following table summarizes the generic formal subroutines required by this part:

Name	Type	Description
Active_H_Vector	function	Returns the row of a (MxN) H Matrix that is referenced
Subtract_From_Identity	procedure	Subtracts a K_H_Product matrix from the Identity Matrix (i.e., I - S)
"*"	function	Multiplies a K Column Vector by a H Row Vector, yielding a K H Product Matrix
"*"	function	Multiplies a K H Product Matrix by a P Matrix, yielding a P Matrix

3.3.3.3.10.2.4 LOCAL DATA

None.

3.3.3.3.10.2.5 PROCESS CONTROL

Not applicable.

3.3.3.3.10.2.6 PROCESSING

The following describes the processing performed by this part:

```
separate (Kalman_Filter_Complicated_H_Parts)

procedure Update_Error_Covariance_Matrix
            (P                  : in out P_Matrices;
             Measurement_Number : in     Measurement_Indices;
             K                  : in     K_Column_Vectors;
             Complicated_H      : in     H_Matrices) is

   V : H_Row_Vectors;

begin

   V := Active_H_Vector (Source => Complicated_H,
                         Row    => Measurement_Number);

   P := Subtract_From_Identity (K * V) * P;

end Update_Error_Covariance_Matrix;
```

3.3.3.3.10.2.7 UTILIZATION OF OTHER ELEMENTS

None.

3.3.3.3.10.2.8 LIMITATIONS

None.

3.3.3.3.10.3 UPDATE_STATE_VECTOR UNIT DESIGN (CATALOG #P152-0)

This unit is a generic procedure which updates the State Vector, X, given the old X vector, the Z vector, the K vector, the Measurement Number, and the Complicated H array.

3.3.3.3.10.3.1 REQUIREMENTS ALLOCATION

This part meets CAMP requirement R184.

3.3.3.3.10.3.2 LOCAL ENTITIES DESIGN

None.

3.3.3.3.10.3.3 INPUT/OUTPUT

GENERIC PARAMETERS:

Data types:

The following table describes the generic formal types required by this part:

Name	Base Type	Description
State_Indices	discrete	Index to the arrays which depend on the number of states
Measurement_Indices	discrete	Index to the arrays which depend on the number of measurements
Kalman_Filter_Elements	floating point type	Elements contained in the Kalman Filter aggregates
H_Matrices	private	Data type of m x n H matrix
H_Row_Vectors	private	Vector representing a row of an H matrix
Measurement_Vectors	vector	Vector indexed by Measurement Indices
K_Column_Vectors	vector	Vector indexed by State_Indices containing Kalman_Filter_Elements
State_Vectors	vector	Vector indexed by State_Indices containing Kalman_Filter_Elements

Subprograms:

The following table summarizes the generic formal subroutines required by this part:

Name	Type	Description
Active_H_Vector	function	Returns the row of a (MxN) H matrix that is referenced
"*"	function	Computes the product of an H Row vector and a State vector, yeilding a Kalman Filter Element
Dot_Product	function	Multiplies a (Nx1) K Column Vector by a Kalman Filter Element, yielding a K Column Vector
"+"	function	Adds a state vector and a K column vector (both n x 1) yielding a state vector

3.3.3.3.10.3.4 LOCAL DATA

Data objects:

The following table describes the data objects maintained by this part:

Name	Type	Value	Description
V	H_Row_Vectors	variable	Column slice of the H matrix representing the current measurement

3.3.3.3.10.3.5 PROCESS CONTROL

Not applicable.

3.3.3.3.10.3.6 PROCESSING

The following describes the processing performed by this part:

```
separate (Kalman_Filter_Complicated_H_Parts)
procedure Update_State_Vector
             (X                  : in out State_Vectors;
              Z                  : in     Measurement_Vectors;
              K                  : in     K_Column_Vectors;
              Measurement_Number : in     Measurement_Indices;
              Complicated_H      : in     H_Matrices) is

   V : H_Row_Vectors;

begin

   V := Active_H_Vector (Source => Complicated_H,
```

```
                        Row    => Measurement_Number);

   X := K * (Z (Measurement_Number) -
             Dot_Product(Left  => V,
                         Right => X)) + X;

end Update_State_Vector;
```

3.3.3.3.10.3.7 UTILIZATION OF OTHER ELEMENTS

None.

3.3.3.3.10.3.8 LIMITATIONS

None.

3.3.3.3.10.4 UPDATE_ERROR_COVARIANCE_MATRIX_GENERAL_FORM UNIT DESIGN (CATALOG #P1119-0)

This unit is a generic procedure body which computes the updated covariance matrix resulting from the processing of a single component of the measurement vector, Z. This solution uses the general form of the equation.

3.3.3.3.10.4.1 REQUIREMENTS ALLOCATION

This part meets CAMP requirement R183.

3.3.3.3.10.4.2 LOCAL ENTITIES DESIGN

None.

3.3.3.3.10.4.3 INPUT/OUTPUT

GENERIC PARAMETERS:

Data types:

The following table describes the generic formal types required by this part:

Name	Base Type	Description
State_Indices	discrete	Index to the arrays which depend on the number of states
Measurement_Indices	discrete	Index to the arrays which depend on the number of measurements
Kalman_Filter_Elements	floating point type	Elements contained in the Kalman Filter aggregates
P_Matrices	private	Data type for private n x n P matrix
H_Matrices	private	Data type for private H matrix
H_Row_Vectors	private	Private vector representing a row of the H matrix
K_H_Product_Matrices	array	Private matrix representing product of K and H matrices
Measurement_Variance_Vectors	array	Vector indixed by Measurement Indices containing KF Elements
K_Column_Vectors	vector	Vector indexed by State_Indices containing Kalman_Filter_Elements

Subprograms:

The following table summarizes the generic formal subroutines required by this part:

Name	Type	Description
Active_H_Vector	function	Returns the row of a (MxN) H Matrix that is referenced
Subtract_From_ Identity	procedure	Subtracts a K_H_Product matrix from the Identity Matrix (i.e., I - S)
ABA_Transpose	function	Does an ABA transpose on a K H Matrix and a P Matrix, yielding a P Matrix
ABA_Transpose	function	Does an ABA transpose on a K Column Vector and a Kalman Filter Element, yielding a P Matrix
"*"	function	Multiplies a K Column Vector by a H Row Vector, yielding a K H Product Matrix
"+"	function	Adds two P Matrices, yielding a P Matrix

3.3.3.3.10.4.4 LOCAL DATA

Data objects:

The following table describes the data objects maintained by this part:

Name	Type	Value	Description
A	K_H_Product_Matrices	variable	Product of K and H Matrices
V	H_Row_Vectors	variable	Column slice of the H matrix representing the current measurement

3.3.3.3.10.4.5 PROCESS CONTROL

Not applicable.

3.3.3.3.10.4.6 PROCESSING

The following describes the processing performed by this part:

```
separate (Kalman_Filter_Complicated_H_Parts)

procedure Update_Error_Covariance_Matrix_General_Form
            (P                    : in out P_Matrices;
```

```
            Measurement_Number   : in     Measurement_Indices;
            K                    : in     K_Column_Vectors;
            Complicated_H        : in     H_Matrices;
            Measurement_Variance : in     Measurement_Variance_Vectors ) is

   A : K_H_Product_Matrices;
   V : H_Row_Ve_tors;

begin

   V := Active_H_Vector (Source => Complicated_H,
                         Row    => Measurement_Number);

   A := Subtract_From_Identity (K * V);

   P := ABA_Transpose( A, P ) +
        ABA_Transpose( K, Measurement_Variance( Measurement_Number ) );

end Update_Error_Covariance_Matrix_General_Form;
```

3.3.3.3.10.4.7 UTILIZATION OF OTHER ELEMENTS

None.

3.3.3.3.10.4.8 LIMITATIONS

None.

```
package body Kalman_Filter_Complicated_H_Parts is

   function Compute_Kalman_Gain
               ( P                    : P_Matrices;
                 Measurement_Number   : Measurement_Indices;
                 Complicated_H        : H_Matrices;
                 Measurement_Variance : Measurement_Variance_Vectors)
              return K_Column_Vectors is separate;

   procedure Update_Error_Covariance_Matrix
               (P                  : in out P_Matrices;
                Measurement_Number : in     Measurement_Indices;
                K                  : in     K_Column_Vectors;
                Complicated_H      : in     H_Matrices) is separate;

   procedure Update_State_Vector
               (X                  : in out State_Vectors;
                Z                  : in     Measurement_Vectors;
                K                  : in     K_Column_Vectors;
                Measurement_Number : in     Measurement_Indices;
                Complicated_H      : in     H_Matrices) is separate;

   package body Sequentially_Update_Covariance_Matrix_And_State_Vector is separate;

   package body Kalman_Update is separate;

   procedure Update_Error_Covariance_Matrix_General_Form
               ( P                    : in out P_Matrices;
                 Measurement_Number   : in     Measurement_Indices;
                 K                    : in     K_Column_Vectors;
                 Complicated_H        : in     H_Matrices;
                 Measurement_Variance : in     Measurement_Variance_Vectors )
              is separate;

end Kalman_Filter_Complicated_H_Parts;
```

```
separate (Kalman_Filter_Complicated_H_Parts)

function Compute_Kalman_Gain
            (P                    : P_Matrices;
             Measurement_Number   : Measurement_Indices;
             Complicated_H        : H_Matrices;
             Measurement_Variance : Measurement_Variance_Vectors)
            return K_Column_Vectors is


   Denominator : Kalman_Filter_Elements;
   K           : K_Column_Vectors;
   V           : H_Row_Vectors;

begin

   V := Active_H_Vector (Source => Complicated_H,
                         Row    => Measurement_Number);

   K := Times_Transpose (Left  => P,
                         Right => V);

   Denominator := Dot_Product(Left  => V,
                              Right => K) +
                  Measurement_Variance( Measurement_Number );

   return K / Denominator;

end Compute_Kalman_Gain;
```

```
separate (Kalman_Filter_Complicated_H_Parts)

procedure Update_Error_Covariance_Matrix
            (P                  : in out P_Matrices;
             Measurement_Number : in     Measurement_Indices;
             K                  : in     K_Column_Vectors;
             Complicated_H      : in     H_Matrices) is

   V : H_Row_Vectors;

begin

   V := Active_H_Vector (Source => Complicated_H,
                         Row    => Measurement_Number);

   P := Subtract_From_Identity (K * V) * P;

end Update_Error_Covariance_Ma  ix;
```

```
separate (Kalman_Filter_Complicated_H_Parts)
procedure Update_State_Vector
            (X                  : in out State_Vectors;
             Z                  : in     Measurement_Vectors;
             K                  : in     K_Column_Vectors;
             Measurement_Number : in     Measurement_Indices;
             Complicated_H      : in     H_Matrices) is

   V : H_Row_Vectors;

begin

   V := Active_H_Vector (Source => Complicated_H,
                         Row    => Measurement_Number);

   X := K * (Z (Measurement_Number) -
             Dot_Product(Left  => V,
                         Right => X)) + X;

end Update_State_Vector;
```

```
separate (Kalman_Filter_Complicated_H_Parts)

package body Sequentially_Update_Covariance_Matrix_And_State_Vector is

   K : K_Column_Vectors;


   function Compute_K is new Compute_Kalman_Gain
               (State_Indices              => State_Indices,
                Measurement_Indices        => Measurement_Indices,
                Kalman_Filter_Elements     => Kalman_Filter_Elements,
                P_Matrices                 => P_Matrices,
                H_Matrices                 => H_Matrices,
                H_Row_Vectors              => H_Row_Vectors,
                Measurement_Variance_Vectors
                                           => Measurement_Variance_Vectors,
                K_Column_Vectors           => K_Column_Vectors);


   procedure Update_P is new Update_Error_Covariance_Matrix
               (State_Indices              => State_Indices,
                Measurement_Indices        => Measurement_Indices,
                Kalman_Filter_Elements     => Kalman_Filter_Elements,
                P_Matrices                 => P_Matrices,
                H_Matrices                 => H_Matrices,
                H_Row_Vectors              => H_Row_Vectors,
                K_H_Product_Matrices       => K_H_Product_Matrices,
                K_Column_Vectors           => K_Column_Vectors);


   procedure Update_X is new Update_State_Vector
               (State_Indices          => State_Indices,
                Measurement_Indices    => Measurement_Indices,
                Kalman_Filter_Elements => Kalman_Filter_Elements,
                H_Matrices             => H_Matrices,
                H_Row_Vectors          => H_Row_Vectors,
                Measurement_Vectors    => Measurement_Vectors,
                K_Column_Vectors       => K_Column_Vectors,
                State_Vectors          => State_Vectors);
                                                          pragma PAGE;
   procedure Update
               (P                    : in out P_Matrices;
                X                    : in out State_Vectors;
                Z                    : in     Measurement_Vectors;
                Complicated_H        : in     H_Matrices;
                Measurement_Variance : in     Measurement_Variance_Vectors) is
   begin

      for Measurement_Number in Measurement_Indices loop

         K := Compute_K (P                    => P,
                         Measurement_Number   => Measurement_Number,
                         Complicated_H        => Complicated_H,
                         Measurement_Variance => Measurement_Variance);

         Update_P (P                  => P,
                   Measurement_Number => Measurement_Number,
```

```
                K                   => K,
                Complicated_H       => Complicated_H);

        Update_X (X                 => X,
                Z                   => Z,
                K                   => K,
                Measurement_Number  => Measurement_Number,
                Complicated_H       => Complicated_H);

     end loop;

   end Update;

end Sequentially_Update_Covariance_Matrix_And_State_Vector;
```

```
with Kalman_Filter_Common_Parts;
separate (Kalman_Filter_Complicated_H_Parts)
package body Kalman_Update is

   package P_Manager is new
             Kalman_Filter_Common_Parts.Error_Covariance_Matrix_Manager
                ( Phi_Matrices     => Phi_Matrices,
                  P_And_Q_Matrices => P_And_Q_Matrices );

   package Update_P_And_X is new
             Sequentially_Update_Covariance_Matrix_And_State_Vector
                (State_Indices                => State_Indices,
                 Measurement_Indices          => Measurement_Indices,
                 Kalman_Filter_Elements       => Kalman_Filter_Elements,
                 P_Matrices                   => P_And_Q_Matrices,
                 H_Matrices                   => H_Matrices,
                 K_H_Product_Matrices         => K_H_Product_Matrices,
                 H_Row_Vectors                => H_Row_Vectors,
                 Measurement_Variance_Vectors => Measurement_Variance_Vectors,
                 Measurement_Vectors          => Measurement_Vectors,
                 K_Column_Vectors             => K_Column_Vectors,
                 State_Vectors                => State_Vectors );

   Zero_State_Vector : State_Vectors := (others => 0.0);

pragma PAGE;
   procedure Update (X                    : in out State_Vectors;
                     P                    : in out P_And_Q_Matrices;
                     Z                    : in     Measurement_Vectors;
                     Complicated_H        : in     H_Matrices;
                     Measurement_Variance : in     Measurement_Variance_Vectors;
                     Propagated_Phi       : in     Phi_Matrices;
                     Propagated_Q         : in     P_And_Q_Matrices) is

   begin

--       -- Propagate the Error Covariance Matrix

      P_Manager.Initialize (Initial_P => P);

      P_Manager.Propagate (Propagated_Phi => Propagated_Phi,
                           Propagated_Q   => Propagated_Q);

      P := P_Manager.P;

--       -- If the state vector is not zero, multiply it by Propagated Phi

      if X /= Zero_State_Vector then

         X := Propagated_Phi * X;

      end if;

--       -- Update Error Covariance Matrix and State Vector

      Update_P_And_X.Update (P                    => P,
                             X                    => X,
```

```
                                Z                    => Z,
                                Complicated_H        => Complicated_H,
                                Measurement_Variance => Measurement_Variance);

   end Update;

end Kalman_Update;
```

```
separate (Kalman_Filter_Complicated_H_Parts)

procedure Update_Error_Covariance_Matrix_General_Form
            (P                  : in out P_Matrices;
             Measurement_Number : in     Measurement_Indices;
             K                  : in     K_Column_Vectors;
             Complicated_H      : in     H_Matrices;
             Measurement_Variance : in   Measurement_Variance_Vectors ) is

   A : K_H_Product_Matrices;
   V : H_Row_Vectors;

begin

   V := Active_H_Vector (Source => Complicated_H,
                         Row    => Measurement_Number);

   A := Subtract_From_Identity (K * V);

   P := Aba_Transpose( A, P ) +
        Aba_Transpose( K, Measurement_Variance( Measurement_Number ) );

end Update_Error_Covariance_Matrix_General_Form;
```

SUPPLEMENTARY

INFORMATION

DEPARTMENT OF THE AIR FORCE
WRIGHT LABORATORY (AFSC)
EGLIN AIR FORCE BASE, FLORIDA, 32542-5434

ERRATA
AD-B120254

REPLY TO ATTN OF MNOI

13 Feb 92

SUBJECT Removal of Distribution Statement and Export-Control Warning Notices

TO Defense Technical Information Center
ATTN: DTIC/HAR (Mr William Bush)
Bldg 5, Cameron Station
Alexandria, VA 22304-6145

1. The following technical reports have been approved for public release by the local Public Affairs Office (copy attached).

Technical Report Number	AD Number
1. 88-18-Vol-4	ADB 120 251
2. 88-18-Vol-5	ADB 120 252
3. 88-18-Vol-6	ADB 120 253
4. 88-25-Vol-1	ADB 120 309
5. 88-25-Vol-2	ADB 120 310
6. 88-62-Vol-1	ADB 129 568
7. 88-62-Vol-2	ADB 129 569
8. 88-62-Vol-3	ADB 129-570
9. 85-93-Vol-1	ADB 102-654
10. 85-93-Vol-2	ADB 102-655
11. 85-93-Vol-3	ADB 102-656
12. 88-18-Vol-1	ADB 120 248
13. 88-18-Vol-2	ADB 120 249
14. 88-18-Vol-7	ADB 120 254
15. 88-18-Vol-8	ADB 120 255
16. 88-18-Vol-9	ADB 120 256
17. 88-18-Vol-10	ADB 120 257
18. 88-18-Vol-11	ADB 120 258
19. 88-18-Vol-12	ADB 120 259

2. If you have any questions regarding this request call me at DSN 872-4620.

LYNN S. WARGO
Chief, Scientific and Technical
Information Branch

1 Atch
AFDTIC/PA Ltr, dtd 30 Jan 92

ERRATA

DEPARTMENT OF THE AIR FORCE
HEADQUARTERS AIR FORCE DEVELOPMENT TEST CENTER (AFSC)
EGLIN AIR FORCE BASE, FLORIDA 32542-5000

REPLY TO
ATTN OF: PA (Jim Swinson, 882-3931) 30 January 1992

SUBJECT: Clearance for Public Release

TO: WL/MNA

The following technical reports have been reviewed and are approved for public release: AFATL-TR-88-18 (Volumes 1 & 2), AFATL-TR-88-18 (Volumes 4 thru 12), AFATL-TR-88-25 (Volumes 1 & 2), AFATL-TR-88-62 (Volumes 1 thru 3) and AFATL-TR-85-93 (Volumes 1 thru 3).

Virginia N. Pribyla

VIRGINIA N. PRIBYLA, Lt Col, USAF
Chief of Public Affairs

AFDTC/PA 92-039