Calhoun: The NPS Institutional Archive

DSpace Repository

Theses and Dissertations | 1. Thesis and Dissertation Collection, all items

1990-12

A software prototype for a Command, Control, Communications and Intelligence (C_x001B_p3_x001B_sI) workstation

Coskun, Vedat; Kesoglu, Cengiz

Monterey, California: Naval Postgraduate School

http://hdl.handle.net/10945/27566

Downloaded from NPS Archive: Calhoun

http://www.nps.edu/library

Calhoun is a project of the Dudley Knox Library at NPS, furthering the precepts and goals of open government and government transparency. All information contained herein has been approved for release by the NPS Public Affairs Officer.

Dudley Knox Library / Naval Postgraduate School
411 Dyer Road / 1 University Circle
Monterey, California USA 93943

AD-A245 390

NAVAL POSTGRADUATE SCHOOL
Monterey, California

DTIC
ELECTE
FEB 0 4 1992
S D
D

THESIS

A SOFTWARE PROTOTYPE FOR A COMMAND, CONTROL, COMMUNICATIONS AND INTELLIGENCE (C^3I) WORKSTATION

by

Vedat Coskun and Cengiz Kesoglu

December, 1990

Thesis Advisor: Luqi
Co-Advisor: Gary Hughes

Approved for public release; distribution is unlimited.

REPRODUCED BY
U.S. DEPARTMENT OF COMMERCE
NATIONAL TECHNICAL
INFORMATION SERVICE
SPRINGFIELD, VA 22161

92-02459

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION: UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT: Approved for public release; distribution is unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION: Computer Science Dept. Naval Postgraduate School; 6b. OFFICE SYMBOL (if applicable): CS	7a. NAME OF MONITORING ORGANIZATION: Office of Chief of Naval Research, Code D1124
6c. ADDRESS (City, State, and ZIP Code): Monterey, CA 93943-5000	7b. ADDRESS (City, State, and ZIP Code): Arlington, VA 22217-5000
8a. NAME OF FUNDING/SPONSORING ORGANIZATION: Office of Chief of Naval Operations; 8b. OFFICE SYMBOL (if applicable): OP-942F4	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code): Washington D.C. 20350 - 2000	10. SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)
A SOFTWARE PROTOTYPE FOR A COMMAND, CONTROL, COMMUNICATIONS AND INTELLIGENCE (C^3I) WORKSTATION

12. PERSONAL AUTHOR(S)
Coskun, Vedat and Kesoglu, Cengiz

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Master's Thesis	FROM ____ TO ____	December, 1990	328

16. SUPPLEMENTARY NOTATION
The views expressed in this thesis are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government

17. COSATI CODES		
FIELD	GROUP	SUB-GROUP

18. SUBJECT TERMS (Continue or reverse if necessary and identify by block number)
Computer Aided Prototyping, C^3I Systems, Hard-Real-Time Systems, User Interface, Ada, Command and Control, Prototyping Language

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Developing large hard-real-time systems in a traditional way usually creates inconsistencies among the user's needs, the requirements and the implementation.

Rapid Prototyping by using Prototype System Description Language (PSDL) and Computer Aided Prototyping System (CAPS) minimizes the time and resource costs, and maximizes reliability. In this technique, designer builds the prototype with the initial requirements, and the user evaluates the actual behavior of the prototype against its expected behavior. If prototype fails to execute properly, the user and the designer work together to change the requirements and the prototype, until the prototype captures the critical aspects of the software system.

This thesis uses the rapid prototyping approach to produce an Ada software prototype of C^3I workstation, which provides commonality and connectivity between naval platforms and land bases by providing the ability to process tactical data from many interfaces in real-time. The major emphasis of the prototype is to support

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT: ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION: UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL: Prof. Luqi	22b. TELEPHONE (Include Area Code): (408) 646-2735; 22c. OFFICE SYMBOL: CS / LQ

C^3I information management functions, message generation and information display.

This thesis has successfully produced an executable Ada prototype, and further demonstrates the feasibility of the software research and development policies proposed by the Navy's Next Generation Computer Resources program.

Approved for public release; distribution is unlimited.

A SOFTWARE PROTOTYPE FOR A COMMAND, CONTROL, COMMUNICATIONS AND INTELLIGENCE (C^3I) WORKSTATION

by

Vedat Coskun
Ltjg., Turkish Navy
B.S., Turkish Naval Academy, 1984

and

Cengiz Kesoglu
Ltjg., Turkish Navy
B.S., Turkish Naval Academy, 1984

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
December 1990

Authors: [redacted]
Vedat Coskun

[redacted]
Cengiz Kesoglu

Approved By: [redacted]
CDR Hughes, Gary, Thesis Co-Advisor

[redacted]
Luqi, Thesis Advisor

[redacted]
Robert B. McGhee, Chairman,
Department of Computer Science

Accesion For
NTIS CRA&I
DTIC TAB
Unannounced
Justification
By
Distribution /
Availability Codes
Dist | Avail and / or Special
A-1

DTIC QUALITY INSPECTED 4

ABSTRACT

Developing large hard-real-time systems in a traditional way usually creates inconsistencies among the user's needs, the requirements and the implementation.

Rapid Prototyping by using Prototype System Description Language (PSDL) and Computer Aided Prototyping System (CAPS) minimizes the time and resource costs, and maximizes reliability. In this technique, designer builds the prototype with the initial requirements, and the user evaluates the actual behavior of the prototype against its expected behavior. If prototype fails to execute properly, the user and the designer work together to change the requirements and the prototype, until the prototype captures the critical aspects of the software system.

This thesis uses the rapid prototyping approach to produce an Ada software prototype of C^3I workstation, which provides commonality and connectivity between naval platforms and land bases by providing the ability to process tactical data from many interfaces in real-time. The major emphasis of the prototype is to support C^3I information management functions, message generation and information display.

This thesis has successfully produced an executable Ada prototype, and demonstrates the feasibility of the software research and development policies proposed by the Navy's Next Generation Computer Resources program.

THESIS DISCLAIMER

Ada is a registered trademark of the United States Government, Ada Joint Program Office.

TAE Plus is a registered trademark of the National Aeronautics and Space Administration (NASA).

X Window System is a registered trademark of the Massachusetts Institute of Technology (MIT).

SUN is a registered trademark of Sun Microsystems.

UNIX is a registered trademark of AT&T.

TABLE OF CONTENTS

I. INTRODUCTION 1

A. SOFTWARE DEVELOPMENT 1

B. RAPID PROTOTYPING 3

II. BACKGROUND 5

A. THE COMPUTER AIDED PROTOTYPING SYSTEM (CAPS) 5

1. The User Interface 5

2. The Software Database System 7

3. The Execution Support System 8

B. THE PROTOTYPING SYSTEM DESCRIPTION LANGUAGE (PSDL) 9

1. PSDL Computational Model 11

III. REQUIREMENTS FOR THE PROTOTYPE OF A GENERIC C^3I WORKSTATION 17

A. BACKGROUND 17

B. THE ESSENTIAL MODEL 17

1. The Environmental Model 17

i. The Statement of Purpose 17

ii. The Context Diagram 18

iii. The Event List 19

2. The Behavioral Model 19

C. IMPLEMENTATION CONSTRAINTS 20

D. REQUIREMENTS AND CONSTRAINTS FOR PROTOTYPING EFFORT 20

1. Unclassified Environment 20
2. Prototyping Hardware 21
3. Prototyping Software 21
4. Implementation Configuration 21
5. Timing Constraints 21

IV. PSDL DESCRIPTIONS OF THE PROTOTYPE OF GENERIC C3I WORKSTATION 25

A. GENERIC C3I WORKSTATION 26

B. COMMUNICATIONS INTERFACE 29

1. Operator resolve_incoming_messages 34

i. Operator parse_input_file 36
ii. Operator decide_for_relaying 36
iii. Operator decide_for_archiving 37
iv. Operator extract_tracks 38

2. Operator resolve_outgoing_messages 38

i. Operator make_routing 40
ii. Operator forward_for_translation 41
iii. Operator forward_for_transmission 42
iv. Operator convert_to_text_file 43

3. Operator prepare_periodic_report 43
4. Operator translate_message 44

C. SENSOR INTERFACE 44

1. Operator analyze_sensor_data 47
2. Operator normalize_sensor_information 47
3. Operator prepare_sensor_track 48

D. TRACK DATABASE MANAGER 49

1. Operator update_tracks 52
i. Operator filter_comms_tracks 54
ii. Operator filter_sensor_tracks 54
iii. Operator add_comms_track 55
iv. Operator add_sensor_track 55
v. Operator monitor_ownship_position 56
vi. Operator update_the_track 56
vii. Operator delete_the_track 57
viii Operator add_user_track 57
2. Operator monitor_database 58
i. Operator check_for_timeout_and_range 59
ii. Operator identify_similarities 60
E. USER INTERFACE 60
1. Operator manage_user_interface 63
2. Operator get_user_inputs 65
3. Operator resolution_notice_panel 65
4. Operator emergency_status_screen 66
5. Operator message_arrival_panel 66
6. Operator intelligence_report_panel 67
7. Operator get_modification_data 67
8. Operator status_screen 67
9. Operator message_editor 68
10. Operator display_tracks 68
11. Operator display_graphic_tracks 69
F. WEAPONS INTERFACE 69
G. EXTERNAL SYSTEMS 70

1. Operator comms_links 71
2. Operator sensors 71
3. Operator navigation_system 72
4. Operator weapons_systems 72
G. DATA TYPES 73
V. IMPLEMENTATION OF THE C^3I WORKSTATION PROTOTYPE THROUGH CAPS 79
A. PREPARING PSDL DESCRIPTION FILE, PSDL.TXT 80
B. PRODUCING DRIVER PACKAGE, TL.A 80
C. BUILDING THE STATIC SCHEDULE, SS.A 81
D. BUILDING THE DYNAMIC SCHEDULE, DS.A 81
E. BUILDING THE SOFTWARE BASE, SB.A 81
VI. CONCLUSIONS AND RECOMMENDATIONS 83
A. CONCLUSIONS 83
B. RECOMMENDATIONS 84
APPENDIX A Data Flow Diagrams 85
APPENDIX B PSDL Description of the Prototype, psdl.txt 93
APPENDIX C Driver Package, tl.a 105
APPENDIX D Static Schedule, ss.a 135
APPENDIX E Dynamic Schedule, ds.a 145
APPENDIX F Software Base, sb.a 147
APPENDIX G An Algorithm to Calculate Minimum Calling Periods and Maximum Response Times 265
APPENDIX H Producing Executable User Interface Code in Ada 275
APPENDIX I Ada Code for Excluded Functions 281

LIST OF REFERENCES 301

BIBLIOGRAPHY 303

INITIAL DISTRIBUTION LIST 305

ABBREVIATIONS

ABBREVIATION	MEANING
CAPS	Computer Aided Prototyping System
COMMS	Communications
CWC	Composite Warfare Commander
C3I	Command, Control, Communications and Intelligence
ECM	Electronic Counter Measures
EMAIL	Electronic Mail
EMCON	Emissions Control
EMREP	Emergency Report
ESM	Electronic Support Measures
FIFO	First In First Out
INTREP	Intelligence Report
MCP	Minimum Calling Period
MET	Maximum Execution Time
MRT	Maximum Response Time
NGCR	Next Generation Computer Resources
OTH-T	Over The Horizon Targeting
PSDL	Prototype System Description Language
STATREP	Status Report
TAE	Transportable Applications Environment
TCD	Tactical Command Display
TD	Track Database
TDD	Track Data Display
TDM	Track Database Manager

To my father, whom I love very much, but sadly, I don't have any more chance to share moments;

and to my mother, whom I love very much, and happily, I still have more chance to do so.

Vedat Coskun

I. INTRODUCTION

This thesis describes the development of a software prototype of a workstation for a Command, Control, Communications and Intelligence (C^3I) system.

This chapter presents a brief description of the software engineering problem, current methodologies being used, and the major differences between them. The Prototype System Description Language (PSDL), the Computer Aided Prototyping System (CAPS) and C^3I system are also described briefly to provide a foundation for this thesis. The requirements for C^3I system, and the detailed stages of prototyping are given in successive chapters. These chapters are followed with our conclusions and recommendations for further research.

The design of the prototype with PSDL, and prototyping steps through CAPS are performed by Cengiz Kesoglu. Vedat Coskun wrote the Ada code for the modules, and designed the user interface for the prototype.

A. SOFTWARE DEVELOPMENT

Computers are useful only if they perform the functions needed correctly. Most computer faults are due to software design errors rather than hardware malfunctions. Software engineering is important because software has a large and increasing effect on people's lives, and has a large and increasing cost. Developing reliable, useful, and flexible software systems is one of the great challenges facing software engineers today [Ref. 1].

There exist many methodologies to support software engineering. The two major approaches are the *waterfall model* and the *prototyping method* [Ref. 2].

The waterfall model describes a sequential approach to software development by using a bottom-up implementation, as shown in the Figure 1-1. The use of bottom-up implementation is the major weakness in this life cycle. With this approach, nothing seems to be produced until the whole system is finished. There are some difficulties in the testing phase, because the most trivial bugs are found at the beginning of testing period, and the most serious ones are encountered at the last. Unfortunately, debugging is extremely hard during this stage since there are thousands of modules to search for the bug. Another weakness of the waterfall model is the sequential progression. It insists that a phase must be finished completely before starting the next one. The requirement analysis stage can be considered as the most critical

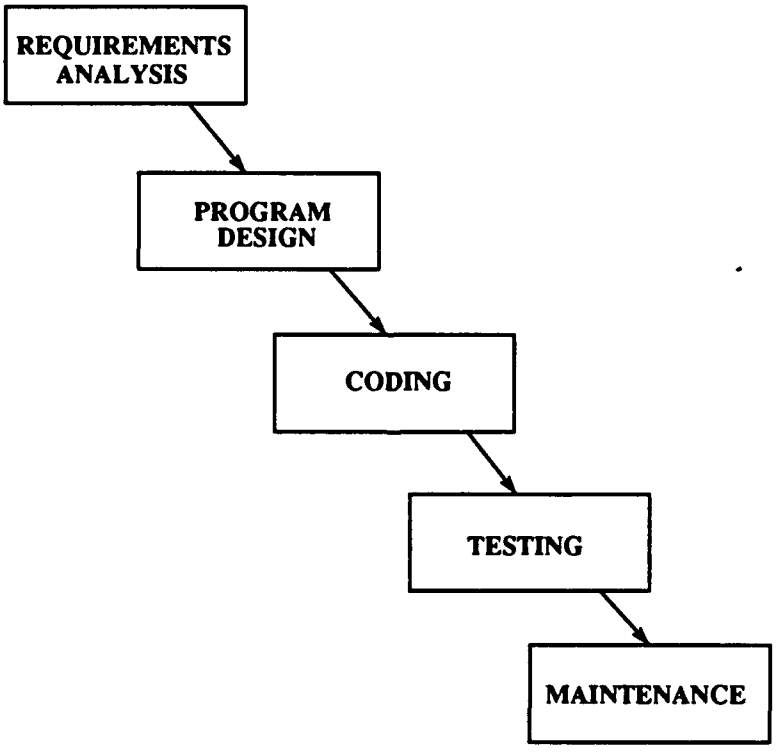


Figure 1-1. The Waterfall Model

phase, since the errors made during this phase effect the rest of the project very seriously. Because the user begins to see the important results only after coding

phase ends, if there are some decisions done at the design phase that the user does not approve, which is a usual case, lots of effort will be lost, since much work is done based on that wrong design decisions. Improvements in the user's understanding of what the system will be like when it is completed would be very helpful if they can be achieved before much effort is spent.

B. RAPID PROTOTYPING

The demand for large, high-quality systems has increased to the point where a jump in software technology is needed. Rapid prototyping is one of the most promising solutions to this problem. Rapid prototyping is particularly effective for ensuring that the requirements accurately reflect the user's real needs, increasing reliability and reducing costly requirement changes [Ref. 3].

Figure 1-2 illustrates the prototyping process. In prototyping cycle, the system designer and the user work together at the beginning to determine the critical parts of the proposed system. Then the designer prepares a prototype of the system based on these critical requirements by using a prototype description language. The resulting system is presented to the user for validation. During these demonstrations, the user evaluates if the prototype behaves as it supposed to do. If errors are found at this point, the user and the designer work together again on the specified requirements and correct them. This process continues until the user determines that the prototype successfully captures the critical aspects of the proposed system. Then designer uses the prototype as a basis for designing the production software [Ref. 4].

The rapid iterative construction of prototypes within a computer aided environment automates the prototyping method of software development and is called *rapid prototyping*[Ref. 5]. The potential benefits of prototyping depend critically on the ability to modify the behavior of the prototype with less effort than required to modify the production software. Computer aided and object-based rapid prototyping provides a solution to this problem.

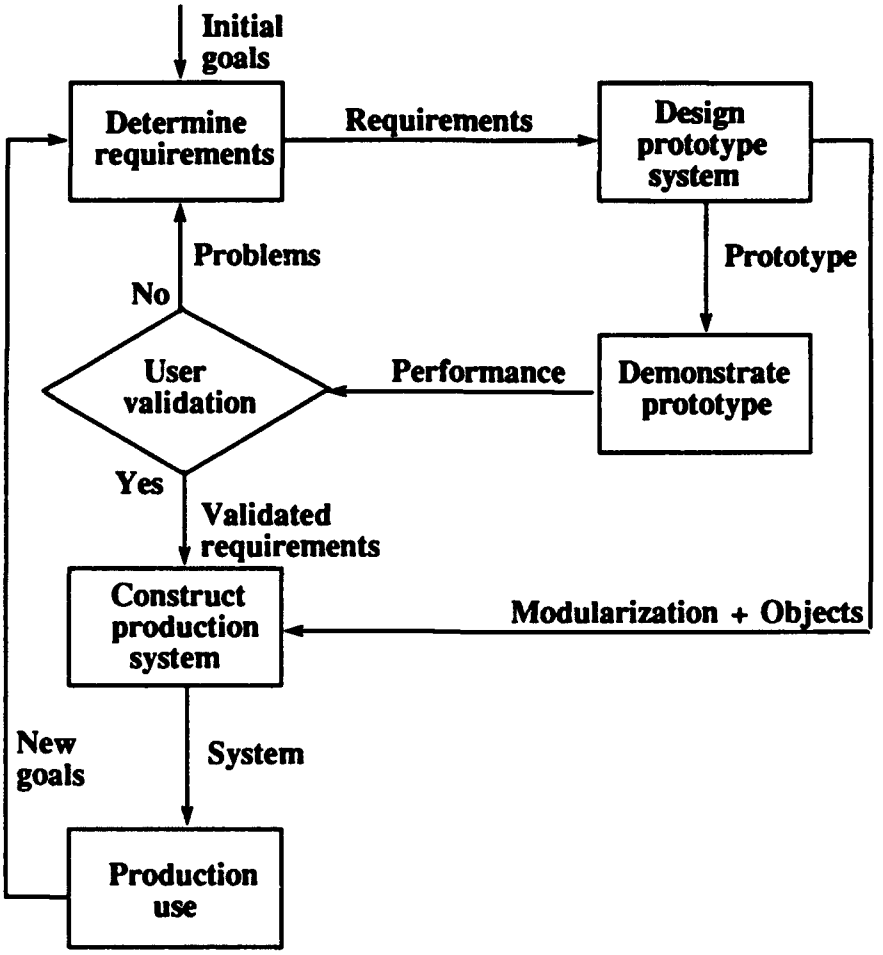


Figure 1-2. The prototyping cycle

II. BACKGROUND

A. THE COMPUTER AIDED PROTOTYPING SYSTEM (CAPS)

An integrated set of computer aided software tools, the Computer Aided Prototyping System, has been designed to support prototyping of complex software systems, such as control systems with hard-real-time constraints.

If carried out manually, the prototyping process has limited benefits because of the time and effort involved. CAPS can increase the leverage of the prototyping strategy reducing and adapting a prototype to perceived user needs.

The main components of CAPS are *The User Interface*, *The Software Database System*, and *The Execution Support System*, as shown in the Figure 2-1. Since the CAPS system is a new approach, there is ongoing research on some aspects of it, and the system is not finished yet.

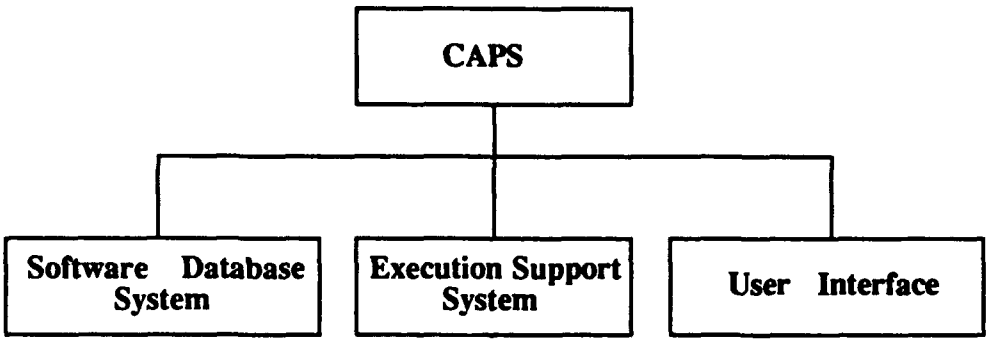


Figure 2-1. CAPS Components

1. The User Interface

The user interface aids evolution by providing facilities for entering information about the requirements and design, presenting the results of prototype

execution to the customer, guiding the choice of which aspects of the prototype to demonstrate, and helping the designer propagate the effects of a change. The user interface consists of a *Syntax-directed Editor* with graphic capabilities, an *Expert System* for communicating with end users, and a *Browser*, as shown in Figure 2-2. PSDL is the main component of the user interface, and will be explained in the next part.

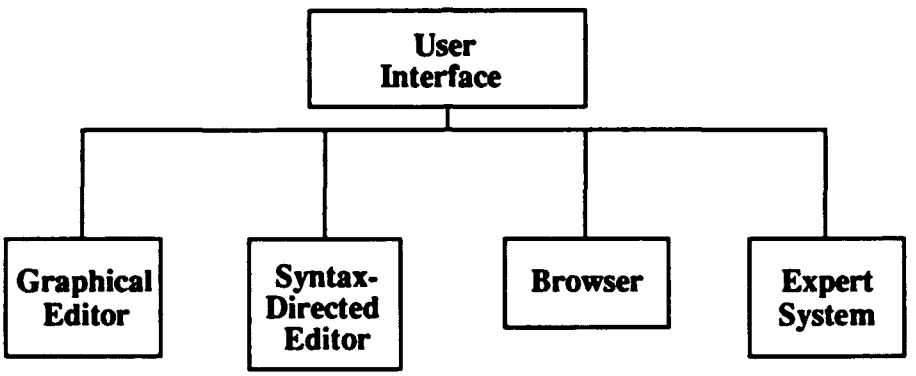


Figure 2-2. User Interface Components

The Graphic Editor is a tool which permits the designer to specify the portions of PSDL prototype using graphical objects to represent the system. Graphical objects include *Operators*, *Inputs*, *Outputs*, *Data Flows*, and *Self Loops* on operators [Ref. 6].

The Syntax Directed Editor is used by the designer to enter the textual portions of the prototype design not represented by the graphic editor in order to completely represent the system [Ref. 4].

The Browser provides the ability for the designer to view the reusable components in the software base [Ref. 4].

The Expert System provides the capability that generates English text descriptions of PSDL specifications. This tool permits the understanding of PSDL components to the users who are not familiar with them.

2. The Software Database System

The Software Database System provides reusable software components for realizing given functional (PSDL) specifications, and it consists of *Design Database, Software Base, Software Design Management System,* and *Rewrite System* as shown in the Figure 2-3.

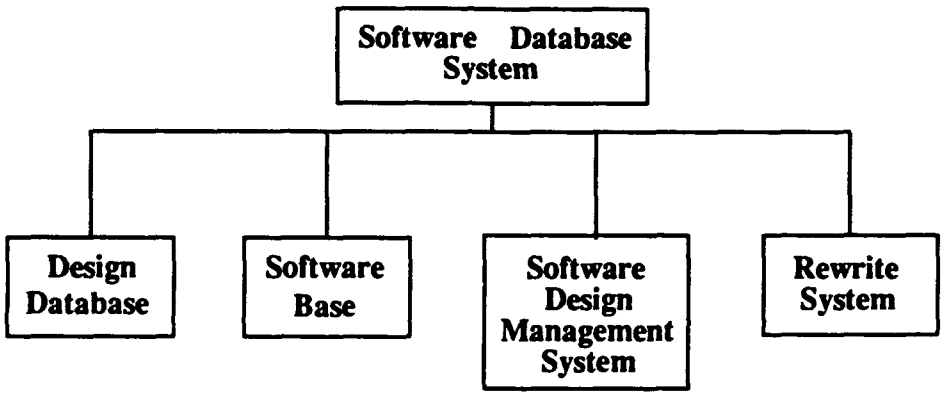


Figure 2-3. Software Database System Components

The Design Database contains PSDL prototype descriptions for all software projects developed using CAPS [Ref. 4].

The Software Base contains PSDL descriptions and implementations for all reusable software components developed using CAPS [Ref 4].

The Software Design Managements System manages and retrieves the versions, refinements and alternatives of the prototypes in the design database and the reusable components in the software base [Ref 4].

The Software Base speeds up evolution by providing many different versions of commonly used components, making it easier to try out alternative designs. In the PSDL prototyping method, modules are realized by three main mechanisms:

1. Retrieval of suitable components from the Software Base. The Software Base contains generic modules with parameters determined as part of the retrieval process. It also contains rules for matching a specification by means of a composite operator realized by a network of operators, at least one of which must be an available reusable component. The retrieval mechanism can therefore perform some routine aspects of bottom-up design, freeing the designer from the need to be familiar with all of the reusable components in the software base.

2. Decomposition of the component into a network of simpler components. The designer does this if the component can not be retrieved directly from the Software Base, and if the component is sufficiently complex to benefit from decomposition into simpler parts.

3. Direct implementation in a programming language. This is essential if the Software Base does not contain a component that performs the required function with the required speed.

3. The Execution Support System

The Execution Support System contains of a *Translator*, a *Static Scheduler*, a *Dynamic Scheduler*, and a *Debugger* as shown in the Figure 2-4.

The Translator generates code that binds together the reusable components extracted from the software base. Its main functions are to implement data streams, control constraints, and timers.

The Static Scheduler allocates time slots for operators with real time constraints before execution begins. If the allocator succeeds, all operators are guaranteed to meet their deadlines even with the worst case execution times. If the static scheduler fails to find a valid schedule, it provides diagnostic information useful

for determining the cause of the difficulty and whether or not the difficulty can be solved by adding more processors.

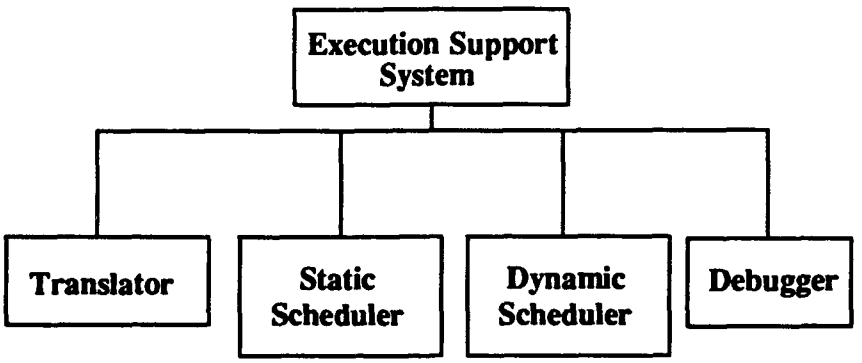


Figure 2-4. Execution Support System Components

As execution proceeds, the dynamic scheduler invokes operators without real-time constraints in the time slots not used by operators with real-time constraints.

The Debugger allows the designer to interact with the execution support system. The Debugger has facilities for initiating the execution of a prototype, displaying execution results or tracing information of the execution, and gathering statistics about a prototype's behavior and performance.

B. THE PROTOTYPING SYSTEM DESIGN LANGUAGE(PSDL)

CAPS tools communicate by means of PSDL, which integrates the tools and provides the prototype designer with a uniform conceptual framework and a high level description of the system. PSDL supports frequent design modifications by meeting the subgoals *Modularity*, *Simplicity*, *Reuse*, *Adaptability*, *Abstraction*, and *Requirements Tracing*.

Modularity is important for easy modification, and is supported by PSDL via *operators* and *data streams*. Good modularity means that the prototype should be

realized by a set of independent modules with narrow and explicitly specified interfaces.

PSDL is simple and easy to use because it contains a small number of powerful constructs. Designs are described in PSDL as networks of operators connected by data streams. These networks can be represented as dataflow diagrams augmented with control and timing constraints. The operators in the network can either be functions or state machines. The data streams can carry exception conditions or values of arbitrary abstract data types.

PSDL supports reusable components with uniform specifications suitable for retrieving modules from a software base. The specification part of a PSDL component contains several attributes that describe the interface and behavior of the component.

PSDL supports adaptability of modules by making small modifications to them when necessary, by means of control constraints. Control constraints can be used to impose preconditions on the execution of a module, to add filters to the output of a module, to suppress or raise exceptions in specified conditions, and to control timers. These facilities allow small modifications to the behavior of a module to be expressed independently of its implementation.

A set of abstractions suitable for describing complex software systems with real-time constrains is important, and hence PSDL provides abstractions suitable for describing large systems and real-time constraints. These include the nonprocedural control constraints such as *timing constraints*, and *timers*. Examples of timing constraints include the maximum execution time, the maximum response time, and the minimum calling period. Timing constraints implicitly determine when operators with hard-real-time constraints will execute. This simplifies evolution by removing explicit scheduling decisions from the design, thus allowing a software tool rather than the designer to handle rescheduling caused by design changes.

Requirements tracing is important because the prototype must adapt the changing perceptions of the requirements resulting from demonstrations of prototype behavior.

PSDL supports requirements by means of a construct for declaring the requirements associated with each part of the prototype.

1. PSDL Computational Model

PSDL is based on a computational model. Formally it is an **augmented graph**. The form of the computational model is:

Graph = (Vertex, Edge, Time(Vertex), Constraint(Vertex))

This implies that each graph is represented by its set of vertices, set of edges, set of maximum execution times for each vertices, and set of constraints for each vertices.

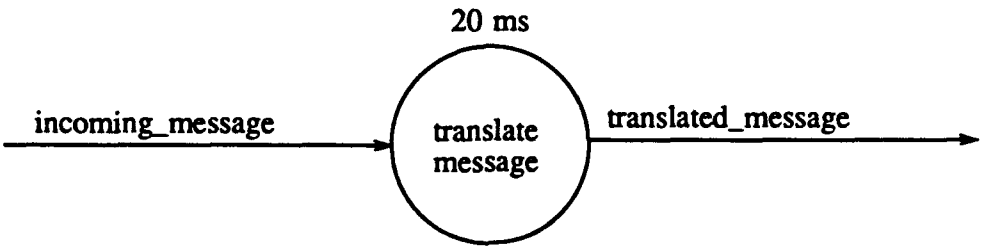


Figure 2-5. A PSDL Graph Example

An example of a PSDL graph is shown in the Figure 2-5. In this graph, operator **translate message** takes the input **incoming message**, outputs into **translated message**, and whole process should take a maximum of 20 milliseconds, as the value of time constraint.

Operators are *state machines*, and the internal state is modeled by a set of state variables. Operators with empty set of state variables behave like *functions*. Operators may be either *atomic* or *composite*. Atomic operators represent single operations and not need to be decomposed into other operators.

Data streams are data which move between operators. They are either *data-flow* streams or *sampled* data streams. Data-flow streams in PSDL act as FIFO buffers of capacity one, and are used for synchronizing data-driven computations.

Data-flow streams guarantee that each of the data values written into the stream is read exactly once. Sampled streams in PSDL model are streams for which only the most recent information is meaningful. One distinction between sampled streams and data flows streams is that reading a data value from stream does not remove it from the stream.

Operators may be triggered by data streams and/or periodic timing constraints. Operators triggered by data streams are called *sporadic*, and operators triggered by periodic timing constraints are called *periodic*.

Every time critical sporadic operator has a ***maximum response time (mrt)*** and a ***minimum calling period (mcp)*** in addition to a ***maximum execution time (met)***.

The met is the upper bound on the length of the time between the instant when a module begins execution and the instant when it completes execution.

The mrt defines an upper bound on the time that may elapse between the point in time at which an operator is activated to read from its input streams and the time when its write event occurs.

A mcp defines a lower bound on the time between two successive activations of the read transitions of operator f (Figure 2-6). The mcp can be considered as the window of opportunity for the operator to use, and the mrt as the used portion of it.

Periodic operators are triggered by temporal events and must occur at regular time intervals. For each operator f, these time intervals are determined by the specified period (OPERATOR f PERIOD t) and deadline (OPERATOR f FINISH WITHIN t).

The period is the time interval between two successive activation times for the read transition of a periodic operator.

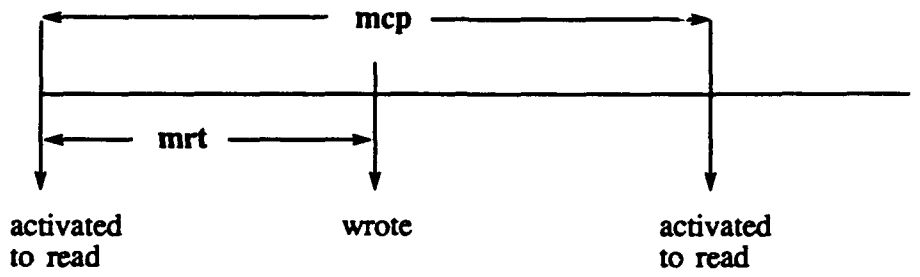


Figure 2-6. MCP and MRT of an Operator

The deadline defines an upper bound on the occurrence time of the write transition of a periodic operator relative to the activation of its read transition. By default, the deadline is equal to the maximum execution time, and a static feasibility constraint requires that deadline must be greater than or equal to maximum execution time (Figure 2-7).

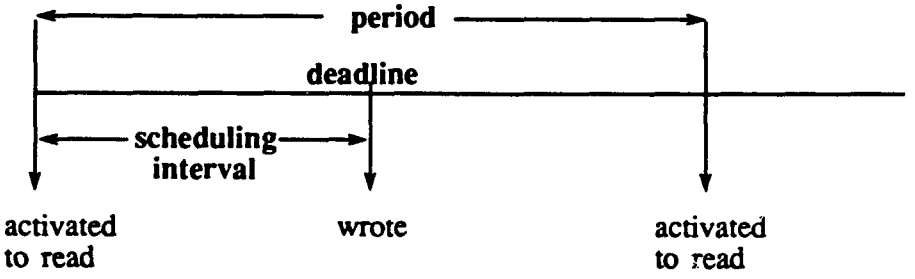


Figure 2-7. Period and Deadline of an Operator

The difference between the activation time of a read transition and the deadline for the corresponding write transition is called the scheduling interval. The scheduling intervals of a periodic operator can be viewed as sliding windows, whose position on time axis relative to each other is fixed by the period, and whose absolute position on

the time axis is fixed by the occurrence time t_0 of the first read transition. This time may vary within the interval 0 to period of the operator (Figure 2-8).

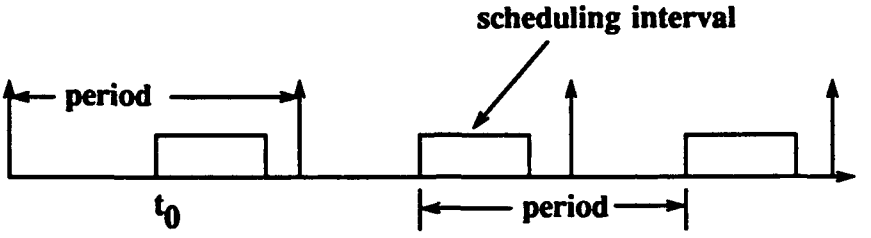


Figure 2-8. Scheduling Interval

The control constraints and the timing constraints determine both the conditions under which the operators are triggered and the buffering disciplines for the data streams. Control constraints can express conditional execution and output, and can control exceptions and timers.

Triggering conditions and guarded outputs are expressed by predicates. If an input stream is guarded by a triggering condition, input data which do not satisfy the condition are read from the stream but do not fire the operator. Similarly, guarded output streams of an operator prevent the specified output data from being written into the guarded streams if the output guard conditions are not satisfied.

Synchronization between different operators in PSDL is achieved by *precedence constraints*. These constraints are introduced by data streams as follows.

Data-flow streams ensure that values are not read until they are written, and that a value is not overwritten before it has been read. This property ensures that transactions are not lost or repeated, and can be used to correlate data from different sources, such as preprocessor operators operating in parallel.

Sampled streams cannot guarantee that values will never be overwritten before they are read. The purpose of a sampled stream is to provide the most recent available version of data.

The precedence constraints associated with sporadic operators are implicit. Periodic operators are triggered by temporal events rather than by arrival of data values, and in certain conditions the precedence constraints can affect these timing constraints.

III. REQUIREMENTS FOR THE

PROTOTYPE OF A GENERIC C^3I WORKSTATION

A. BACKGROUND

A *Command, Control, Communications and Intelligence (C^3I) System* assists the commander in understanding a tactical situation within his geographical area of responsibility. The generic C^3I workstation would be designed to be implemented on a wide variety platforms, in support of a Composite Warfare Commander (CWC) command and control architecture. The workstation would provide the CWC and his subordinate commanders and coordinators with a system that supports them in monitoring air, surface, subsurface, and power-projection (strike) tactical environments, and aid in tactical decision making in those areas. The architecture provides for connectivity between naval platforms, shore-bases, and external forces and information sources, and enables the processing of tactical data from internal and external sources [Ref. 7].

Requirements for the generic C^3I workstation is given in great detail in the thesis written by Steven E. Anderson [Ref. 7]. A summary of his study will be provided in this chapter as a basis for the prototyping of the C^3I workstation.

B. THE ESSENTIAL MODEL

1. The Environmental Model

i. The Statement of Purpose

The purpose of the generic C^3I workstation is to provide commonality and connectivity between naval platforms and land bases by providing the ability to

process, in real-time, tactical data from many interfaces. This includes the ability of the C^3I workstation to receive and transmit command and control data via communications links, to receive track information from organic platform sensors, to provide a tactical display interface to the user, to provide a modern text editor for generating and forwarding communications messages, and to provide a means for verifying track data integrity.

ii. The Context Diagram

In the Yourdon Environmental Model, the context diagram shows the external interfaces with which the designed system communicates. For C^3I workstation, these external interfaces are the users, weapon systems, platform sensors, navigation system, and communications links. The context diagram is given in Figure 3-1.

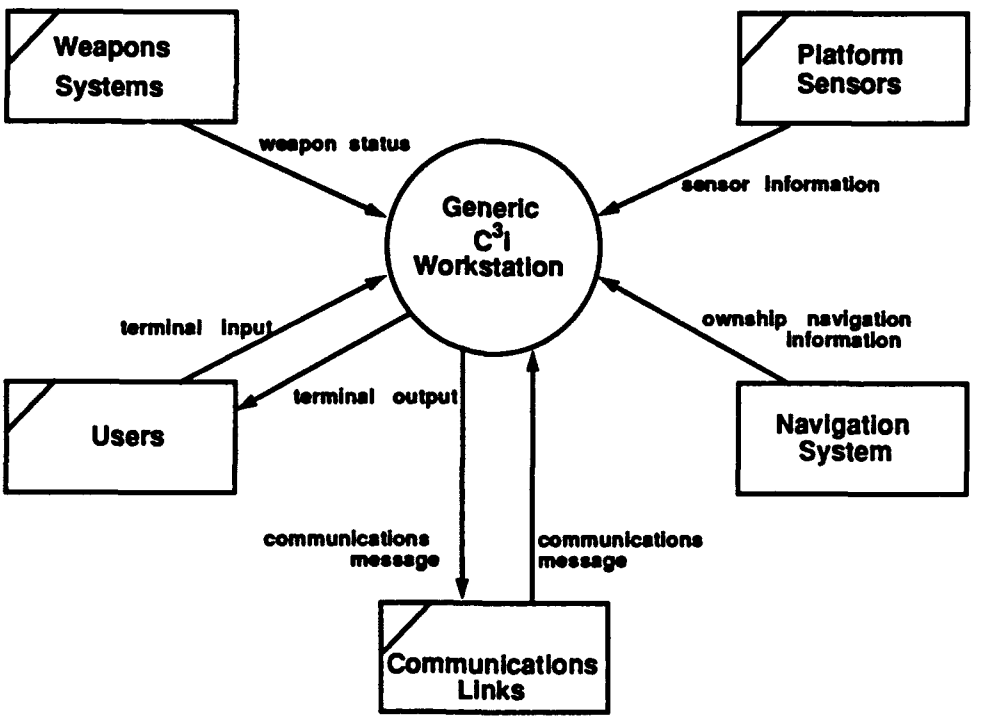


Figure 3-1. The Context Diagram

iii. The Event List

The following is an informal list of events that occur outside of the generic C^3I workstation and invoke a response from the station. This list represents a list of generic stimuli that could apply to any specific C^3I workstation implementation.

- Network communications message received (via communications links).
- Sensor system data update received.
- Weapons system change of status received.
- Navigation system updates own-ship navigation information.
- User chooses to view track tuple information (textually).
- User chooses to manually add new track to database.
- User chooses to manually modify existing track data.
- User chooses to manually delete track from database.
- User chooses to view own-ship weapon status.
- User chooses to view track data (graphically).
- User chooses to generate a message.
- User enters message text.
- User chooses to read a message.
- User chooses to set system parameters: initiate transmission sequence.
- User chooses to set system parameters: set monitor constraints.
- User chooses to set system parameters: archive set-up.
- User chooses to set system parameters: set track filter.
- User chooses to set system parameters: reporting set-up.
- User chooses to set system parameters: network set-up.
- User chooses to set system parameters: set monitor constraints.

2. The Behavioral Model

Yourdon Behavioral Model provides an informal means for describing the internal behavior of the proposed software system. It consists of data-flow diagrams, process specifications for the atomic modules, and the data dictionary. Level 0 diagram and the data-flow diagrams for the first level decomposition are given in

Appendix A. The detailed decomposition for PSDL is based on these first level decompositions, which will be discussed in further chapters.

C. IMPLEMENTATION CONSTRAINTS

The following constraints are specified in the functional specifications document for a generic C^3I workstation [Ref. 7]:

- Software must adhere to Department of Defense Military Standard 2167-A *Defense System Software Development*, 29 February 1988 (or latest revision) and Department of Defence Military Standard 2168 *Defense System Software Quality Program*, 29 April 1988 (or latest revision).
- The performance constraints for the generic C^3I workstation include hard-real-time information processing and display. Since the current systems are so antiquated, it is somewhat unknown exactly how well such a system can perform. Hence, a policy of "best possible performance" will be adopted while looking at the system performance constraints.
- When interacting with information from multiple sources, the most recent information should be displayed, unless specified otherwise by the user.
- The implementation constraints are somewhat sketchy at present. With today's technology, it is possible to implement such a system on a 100 MIPS class machine (such as a militarized SUN workstation). However, the software developed must be modular and anticipate the portability of this system, as much as possible, onto other machines in the future. Careful attention must be paid to distributed architectures in the design and implementation of this software.
- The system hardware must also provide for a "modern" computer interface including color graphics, windows/menus, mouse/trackball mechanism, keypad, etc. in keeping with the Next Generation Computer Resources (NGCR) effort.

D. REQUIREMENTS AND CONSTRAINTS FOR PROTOTYPING EFFORT

1. Unclassified Environment

Since most C^3I systems and procedures are classified, many generalizing assumptions will be made for the prototype system. Arbitrary representative data values have been provided. The OTH-T Gold Reporting Format has been chosen to

serve as the basis for the prototyping examples since it is both character oriented and unclassified.

2. Prototyping Hardware

The generic C^3I workstation will be implemented on a commercially available Sun Microsystems workstation operated by the Naval Postgraduate School Computer Science Department, with the goal of producing software directly transferable to the Genisco Workstation.

3. Prototyping Software

The Sun Microsystems workstation operating system is derived from UC Berkeley Version 4.2BSD and Bell Laboratories' UNIX System Version 32V.

TAE Plus, a windowing software package developed by NASA Goddard, will be used to generate the user interface to the generic C^3I workstation.

In accordance with Department of Defence policy, Ada shall be used as the implementation language for the prototype.

4. Implementation Configuration

Initial prototyping efforts will focus on a single user system with multiple weapons, sensors, and external communications (Figure 3-2).

5. Timing Constraints

- The Inertial Navigation System updates the ownship velocity every 40.96 ms and transmits this nominally every 983.04 ms. The latitude and longitude are updated every 1.3 seconds.
- Provisionally the Generic C^3I Workstation should be capable of retrieving data from up to (potentially) 1000 tracks in less than 1 second.
- The time difference between receiving a track data message by the system and entering its contents into the track database should be less than 2 seconds.

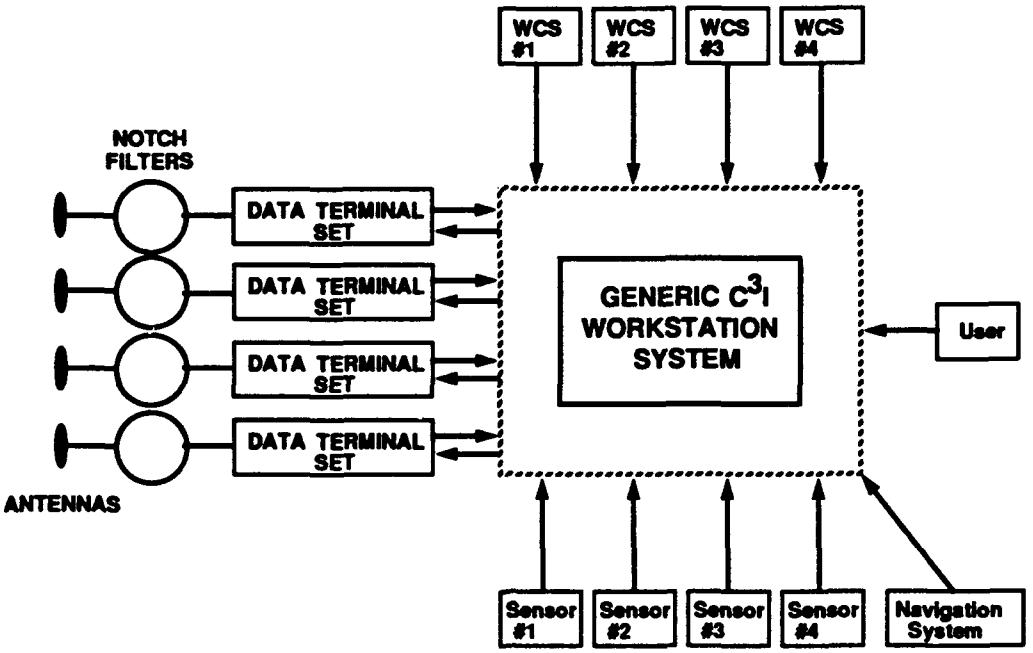


Figure 3-2. Single User Generic C^3I Workstation

- Provisionally for prototyping purposes, the maximum number of tracks per sensor will be assumed to be 100 tracks per second.
- System response times for man-machine interface are as given in Table 3-1:

TABLE 3-1. Dialogue Type Versus System Response

Dialogue Type	System Response Time
Question and Answer	0.5 to less than 2 seconds
Menu Selection	less than 0.2 seconds
Form Filling	greater than 2 seconds
Function Keys	less than 0.2 seconds
Command Language	0.5 to greater than 2 seconds
Natural/Query Language	0.2 to less than 0.5 seconds
Graphic Interaction	less than 0.2 seconds

- System response times for ingressing and egressing communications messages are provided in Table 3-2.

TABLE 3-2. Message Priorities and System Response

Message Precedence	Time Between Message Completion and Transmission	Time Between Message Reception and Display
FLASH	Very Fast (less than 1 second)	Very Fast (less than 1 second)
IMMEDIATE	Fast (less than 2 seconds	Fast (less than 2 seconds
PRIORITY	Moderate (less than 3 seconds	Moderate (less than 3 seconds
ROUTINE	Slow (less than 4 seconds)	Slow (less than 4 seconds)

- Provisionally a nominal weapon status update rate would be once every second.

IV. PSDL DESCRIPTIONS OF THE PROTOTYPE OF GENERIC C^3I WORKSTATION

Based on the system requirements given in the previous chapter, it is possible to construct a module decomposition diagram, consisting of modules that are responsible for implementing these requirements. The first level module decomposition is given in Figure 4-1. Modules in this diagram will be decomposed into lower levels to show the implementation of the higher level module requirements. A detailed description of

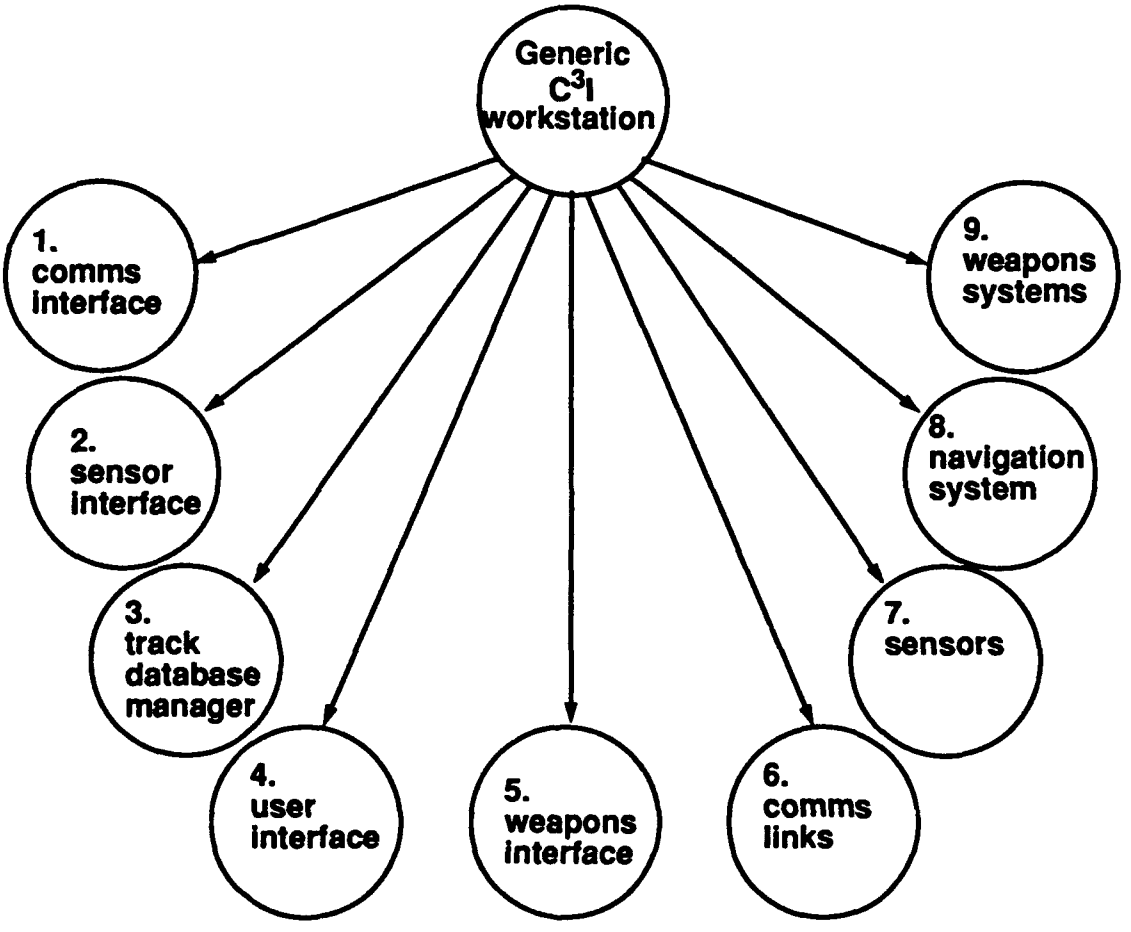


Figure 4-1. First Level Module Decomposition

each module with its corresponding PSDL prototype description will be presented in following sections.

Comms Links, Sensors, Navigation System and Weapons Systems modules are actually external to the prototype, and should not be included in the modular decomposition. However, the prototype workstation will need their data, so these modules are included in the prototype to simulate the related external systems.

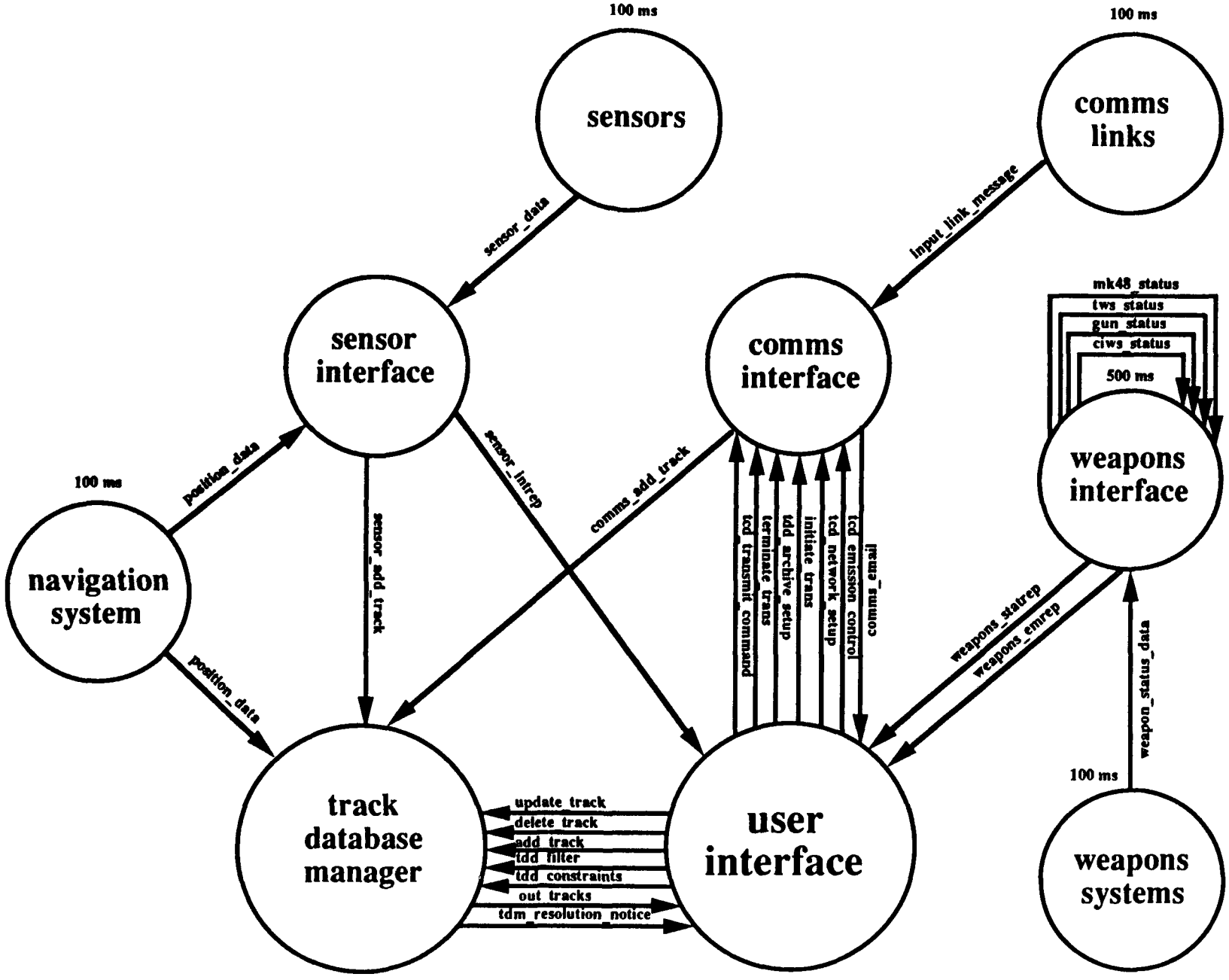
A. GENERIC C^{3}I WORKSTATION

This module is responsible for the complete implementation of the system described. This is the highest level module, and includes the *communications interface*, the *sensor interface*, the *track database manager*, the *user interface*, and the *weapons interface* along with the modules simulating the external interfaces.

The periods of the external interface simulators are determined according to the total maximum execution times of the atomic operators in the system. Other timing constraints given at this level are calculated based upon the requirements, and will be explained with the related operator explanation. The PSDL description follows:

```
OPERATOR c3i_system
SPECIFICATION

   DESCRIPTION { This module is responsible for the imple-
                 mentation of the complete generic C3I
                 workstation }
END
IMPLEMENTATION
  GRAPH
```


100 ms
sensors
sensor_data
100 ms
comms links
input_link_message
sensor interface
comms interface
mk48_status
tws_status
gun_status
ciws_status
500 ms
weapons interface
100 ms
position_data
navigation system
sensor_add_track
sensor_intrep
comms_add_track
tcd_transmit_command
terminate_trans
tdd_archive_setup
initiate_trans
tcd_network_setup
tcd_emission_control
comms_email
weapons_statrep
weapons_emrep
weapon_status_data
position_data
track database manager
update_track
delete_track
add_track
tdd_filter
tdd_constraints
out_tracks
tdm_resolution_notice
user interface
100 ms
weapons systems

```
DATA STREAMS
      sensor_data             : sensor_record,
      position_data           : ownship_navigation_info,
      input_link_message      : filename,
      weapon_status_data      : weapon_status,
      comms_email             : filename,
      sensor_intrep           : intelligence_report,
      tdd_archive_setup       : archive_setup,
      tdm_resolution_notice   : resolution_notice,
      initiate_trans          : initiate_transmission_sequence,
      terminate_trans         : boolean,
      tdd_filter              : set_track_filter,
      tdd_constraints         : set_monitor_constraints,
      tcd_transmit_command    : transmit_command,
      tcd_emission_control    : emissions_control_command,
      tcd_network_setup       : network_setup,
      weapons_emrep           : weapon_status_report,
      weapons_statrep         : weapon_status_report,
      comms_add_track         : add_track_tuple,
      sensor_add_track        : add_track_tuple,
      update_track            : update_track_tuple,
      delete_track            : delete_track_tuple,
      add_track               : add_track_tuple,
      out_tracks              : track_tuple

  CONTROL CONSTRAINTS
     OPERATOR comms_interface
     OPERATOR sensor_interface
       MINIMUM CALLING PERIOD 2.5 ms
       MAXIMUM RESPONSE TIME 500 ms
     OPERATOR track_database_manager
     OPERATOR user_interface
     OPERATOR weapons_interface
       TRIGGERED BY SOME weapon_status_data
       OUTPUT weapons_emrep IF
           weapon_status_data.status = damaged OR
           weapon_status_data.status = service_required OR
           weapon_status_data.status = out_of_ammunition
       MINIMUM CALLING PERIOD 250 ms
       MAXIMUM RESPONSE TIME 500 ms
     OPERATOR comms_links
       PERIOD 10 sec
     OPERATOR sensors
```

```
      PERIOD 10 sec
    OPERATOR navigation_system
      PERIOD 10 sec
  OPERATOR WEAPONS_SYSTEMS
      PERIOD 10 sec
END
```

B. COMMUNICATIONS INTERFACE

The Communications Interface performs those functions directly related to the reception and transmission of the communications messages. Because of the differences between the stations utilizing the communications equipment, this module will vary greatly from one instantiation of a generic C^3I workstation to another. As it was recommended in the requirements analysis [Ref. 7], implementation of this module is highly modularized.

Communications messages which arrive at the interface will be analyzed to determine if they contain track information, if they need to be relayed to other participants in the network, or if they need to be archived for future reference.

This interface will need to monitor, relay and transmit messages on the various link networks. It performs filtering functions on incoming messages, address routing, track precedence sorting and message format translation.

For prototyping purposes, as proposed in the requirements analysis, the primary communications systems of interested will be JTIDS, LINK-11, LINK-16, and OTCIXS.

The complete modular decomposition of the communications interface is given in Figure 4-2.

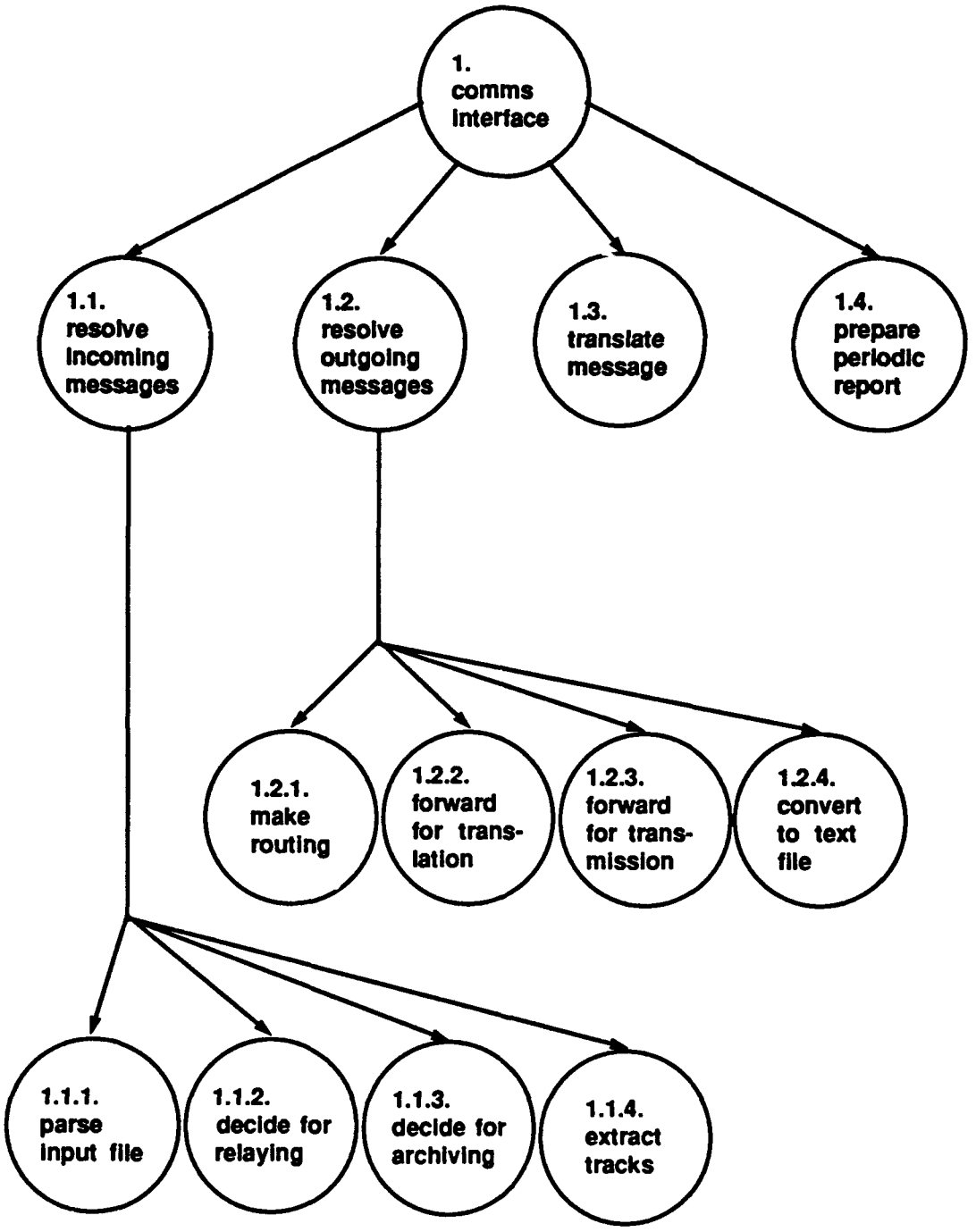


Figure 4-2. Decomposition of Communications Interface Module

The timing constraints related to this interface follow :

- For incoming messages, time between message reception and display varies between less than 1 second and less than 4 seconds, depending upon the precedence of the incoming message. The prototype is designed to be fast enough to handle the highest precedence message. Thus the system shall display any incoming message within 1 second after arriving at the interface. On the data flow path for a message from the communications interface to the CRT there are many operators, and these are limited with 500 ms to process on the message.
- Similarly, requirements state that each outgoing message should be transmitted between 1 second and 2 seconds depending on the precedence. Again the prototype is designed to be limited with the lower boundary. Thus each outgoing message shall be processed and forwarded within 1 second after arriving at the interface.
- There is no time constraint for periodic track reports in the requirements. The prototype is designed to produce periodic track report every 2 seconds when this feature is activated.

These timing constraints are reflected on the submodules as minimum calling periods and maximum response times.

Based on the speeds of the links connected to the workstation (1300-5000 bits/sec), and the assumption that the shortest navy message is about 100 characters (800 bits, if start/stop bits and redundant data are neglected), the system will receive a maximum of 25 messages from 4 communications links in 1 second. Therefore, the operator *resolve_incoming_messages* has 500 ms to handle 12.5 messages, and its minimum calling period is 500 / 12.5 = 40 ms. Maximum response time is 500 ms as indicated above.

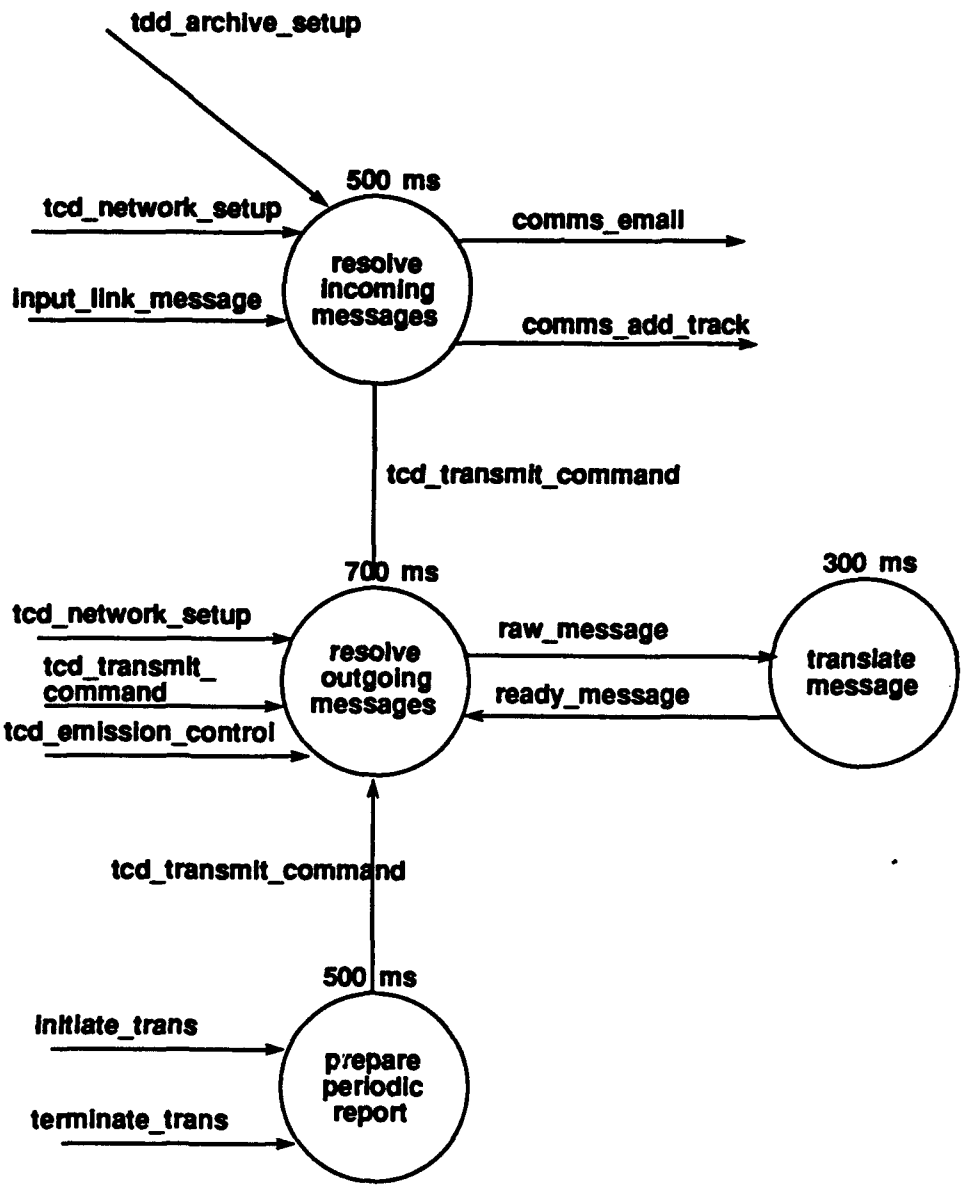
The maximum number of messages arriving at the operator *resolve_outgoing_messages* in 1 second is calculated as follows:

- 1 message/2 sec from periodic operator prepare_track_report.
- 12.5 messages/sec to relay (an assumption that half of the messages that the system received should be relayed).
- 1 message/ sec from the user (an assumption).

So totally there are 14.5 messages to process within 1 second. Thus the minimum calling period for this operator is 1000 / 14.5 = 68.9 ms. Maximum response time is 1000 ms as given by the constraint.

Maximum execution times for operators are determined such that the addition of maximum execution times for a certain process does not exceed the maximum response time of that process. The PSDL description follows:

```
OPERATOR comms_interface
SPECIFICATION
   INPUT  input_link_message    : filename,
          tcd_transmit_command  : transmit_command,
          tcd_emission_control  : emissions_control_command,
          tcd_network_setup     : network_setup,
          tdd_archive_setup     : archive_setup,
          initiate_trans        : initiate_transmission_
                                  sequence,
          terminate_trans       : boolean
   OUTPUT comms_email           : filename,
          comms_add_track       : add_track_tuple
   DESCRIPTION  { This operator is responsible for process-
                  ing incoming and outgoing messages as
                  well as producing periodic track reports
                  and format translating }
END
IMPLEMENTATION
  GRAPH
```




```
DATA STREAMS
  raw_message    : translation_command,
  ready_message  : transmission_command
CONTROL CONSTRAINTS
  OPERATOR resolve_incoming_messages
```

```
      MINIMUM CALLING PERIOD 40 ms
      MAXIMUM RESPONSE TIME 500 ms
   OPERATOR resolve_outgoing_messages
      MINIMUM CALLING PERIOD 68.9 ms
      MAXIMUM RESPONSE TIME 700 ms
   OPERATOR translate_message
      TRIGGERED BY SOME raw_message
   OPERATOR prepare_periodic_report
      PERIOD 2 sec
      TRIGGERED IF not terminate_trans
END
```

Detailed explanations of the decomposed composite operators are provided below with their PSDL descriptions.

1. Operator resolve_incoming_messages

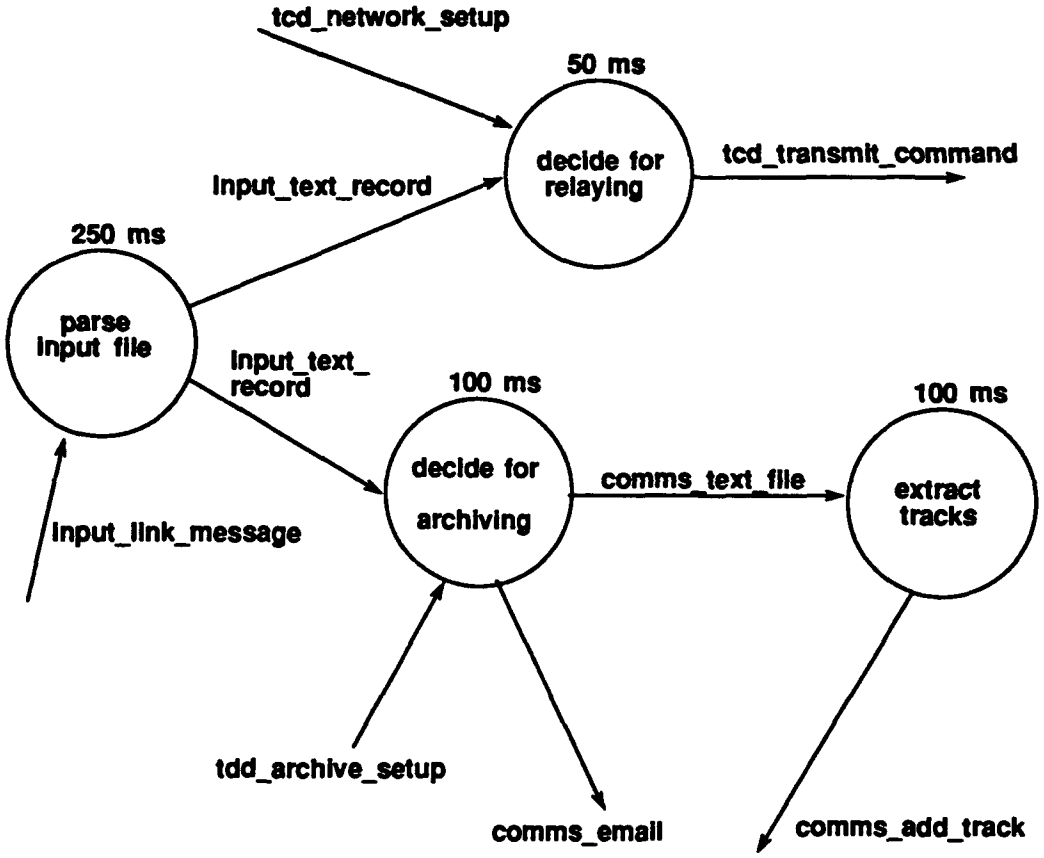
The primary function of this operator is to process communications messages received from the connected links. The operator is still complicated for implementation and it is decomposed into four atomic operators as shown in the graph in the PSDL description.

Timing constraints for the operators are given arbitrarily, but the total execution time for the operators is less than or equal to the maximum response time of the operator resolve_incoming_messages.

Some input and output guards, and triggering conditions are specified in the implementation part of the PSDL description, and they will be explained under the definitions of the atomic operators. The PSDL description follows:

```
OPERATOR resolve_incoming_messages
SPECIFICATION
    INPUT  input_link_message    : filename,
           tcd_network_setup     : network_setup,
           tdd_archive_setup     : archive_setup
```

```
    OUTPUT comms_email           : filename,
           comms_add_track       : add_track_tuple,
           tcd_transmit_command  : transmit_command
    DESCRIPTION { This operator processes the incoming
                  communications messages }
END
IMPLEMENTATION
  GRAPH
```




```
  DATA STREAMS
    input_text_record : text_record,
    comms_text_file   : text_record

  CONTROL CONSTRAINTS
    OPERATOR parse_input_file
      TRIGGERED BY SOME input_link_message
```

```
    OPERATOR decide_for_relaying
      TRIGGERED BY SOME input_text_record
      OUTPUT tcd_transmit_command IF
             input_text_record.relayed
    OPERATOR decide_for_archiving
      TRIGGERED BY SOME input_text_record
      OUTPUT comms_text_file IF comms_text_file.archive,
      OUTPUT comms_email IF NOT input_text_record.is_track
    OPERATOR extract_tracks
      TRIGGERED IF comms_text_file.is_track
END
```

The atomic operators resulting from the decomposition are explained below.

i. Operator parse_input_file

This is an atomic operator responsible for parsing the incoming communications message and writing the data in the message in an Ada record type for further processing. It is a sporadic operator, and triggered by input_link_message. The PSDL description follows:

```
OPERATOR parse_input_file
SPECIFICATION
   INPUT  input_link_message    : filename
   OUTPUT input_text_record     : text_record
   MAXIMUM EXECUTION TIME 250 ms
   DESCRIPTION { Parses the input string and writes the
                 data in a record type }
END
IMPLEMENTATION Ada parse_input_file
END
```

ii. Operator decide_for_relaying

This is a sporadic operator triggered by input_text_record as specified one level above. Depending on the network conditions described in variable

tcd_network_setup, and the address fields in input_text_record, this operator outputs tcd_transmit_command. As it was determined by the control constraints of this operator, output is guarded, and this guarding flag (tcd_transmit_command.text.relay) is set by the operator depending on the conditions explained above. The PSDL description follows:

```
OPERATOR decide_for_relaying
SPECIFICATION
   INPUT  tcd_network_setup     : network_setup,
          input_text_record     : text_record
   OUTPUT tcd_transmit_command : transmit_command
   MAXIMUM EXECUTION TIME 50 ms
   DESCRIPTION { Checks the input message record if it is
                 needed to be relayed }
END
IMPLEMENTATION Ada decide_for_relaying
END
```

iii. Operator decide_for_archiving

There are two main functions performed by this atomic operator. First, it is triggered by the input_text_record and depending on the tdd_archive_setup conditions, the output guard flag for comms_text_file (comms_text_file.archive) is set. Second, the input_text_record is checked if it contains track information. This will set the output guard flag for comms_email (input_text_record.is_track). Depending on the value of this last flag either comms_email will be placed on the output stream, or the next operator extract_tracks will be triggered as determined by its input guard. The PSDL description follows:

```
OPERATOR decide_for_archiving
SPECIFICATION
   INPUT  input_text_record     : text_record,
          tdd_archive_setup     : archive_setup
```

```
   OUTPUT comms_text_file       : text_record,
          comms_email           : filename
   MAXIMUM EXECUTION TIME 100 ms
   DESCRIPTION { Checks if the input file is needed to be
                 archived, and sets the is_track flag
                 depending on the input data }
END
IMPLEMENTATION Ada decide_for_archiving
END
```

iv. Operator extract_tracks

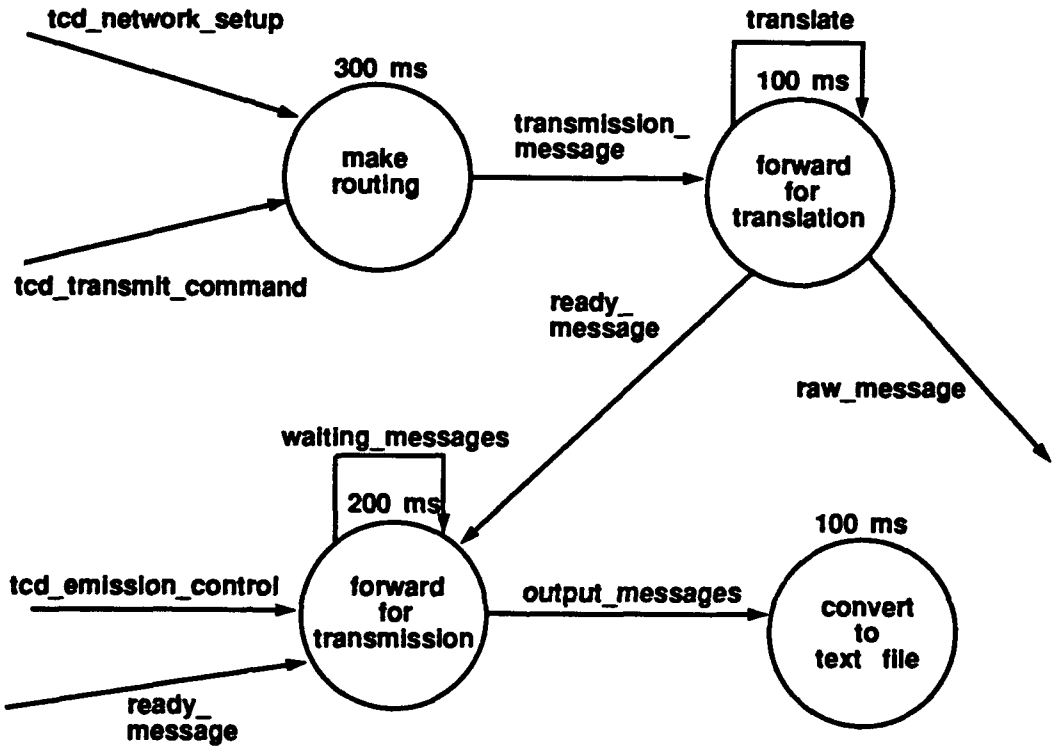
This sporadic operator is triggered if the comms_text_file.is_track flag is set to true by the previous operator, and extracts the tracks from the input variable. Then the operator puts them into the output stream. The PSDL description follows:

```
OPERATOR extract_tracks
SPECIFICATION
   INPUT  comms_text_file       : text_record
   OUTPUT comms_add_track       : add_track_tuple
   MAXIMUM EXECUTION TIME 100 ms
   DESCRIPTION { Extracts the tracks from comms_text_file
                 and writes them into add_track }
END
IMPLEMENTATION Ada extract_tracks
END
```

2. Operator resolve_outgoing_messages

This operator performs the functions for processing all messages for transmission. It is decomposed into four atomic operators. Maximum execution times for the operators are again given arbitrarily, and the triggering and guarding conditions will be explained later with the atomic operators. The PSDL description follows:

```
OPERATOR resolve_outgoing_messages
SPECIFICATION
   INPUT  tcd_transmit_command : transmit_command,
          tcd_emission_control : emissions_control_command,
          tcd_network_setup    : network_setup,
          ready_message        : transmission_command
   OUTPUT raw_message          : translation_command
DESCRIPTION { This operator processes the outgoing
              messages }
END
IMPLEMENTATION
  GRAPH
```




```
DATA STREAMS
  output_messages      : message_list,
  transmission_message : transmission_command
CONTROL CONSTRAINTS
  OPERATOR make_routing
    TRIGGERED BY SOME tcd_transmit_command
```

```
    OPERATOR forward_for_translation
      TRIGGERED BY SOME transmission_message
      OUTPUT raw_message IF translate,
             ready_message IF NOT translate
    OPERATOR forward_for_transmission
      TRIGGERED BY SOME tcd_emission_control,
                        ready_message
      OUTPUT output_messages
            IF tcd_emission_control.no_silence
    OPERATOR convert_to_text_file
      TRIGGERED BY SOME output_messages
END
```

The details of the atomic operators with their PSDL descriptions are below.

i. Operator make_routing

This operator prepares the routing field of the transmission_message depending on the tcd_network_setup (it determines which station is in which link), and the addresses of the tcd_transmit_command when triggered by the tcd_transmit_command as explained in triggering conditions. The PSDL description follows :

```
OPERATOR make_routing
SPECIFICATION
   INPUT  tcd_network_setup     : network_setup,
          tcd_transmit_command : transmit_command
   OUTPUT transmission_message : transmission_command
   MAXIMUM EXECUTION TIME 300 ms
   DESCRIPTION { Prepares the routing field of
                 transmission_message }
END
IMPLEMENTATION Ada make_routing
END
```

ii. Operator forward_for_translation

This atomic operator is implemented as a state machine with the state variable *translate*. It decides if the transmission_message requires translation depending on the format of the message and the links on which it is going to be transmitted. If the format of the original message is not compatible with the formats allowed in transmission link, than it is necessary to translate it to the appropriate format for that link. The state variable is set upon this decision, and output guard determined in one level above checks for the value of this flag.

As it is explained in requirements document [Ref. 7], the message formats used in communications links are classified. Therefore, for prototyping purposes a graph similar to one in Figure 4-3 is used.

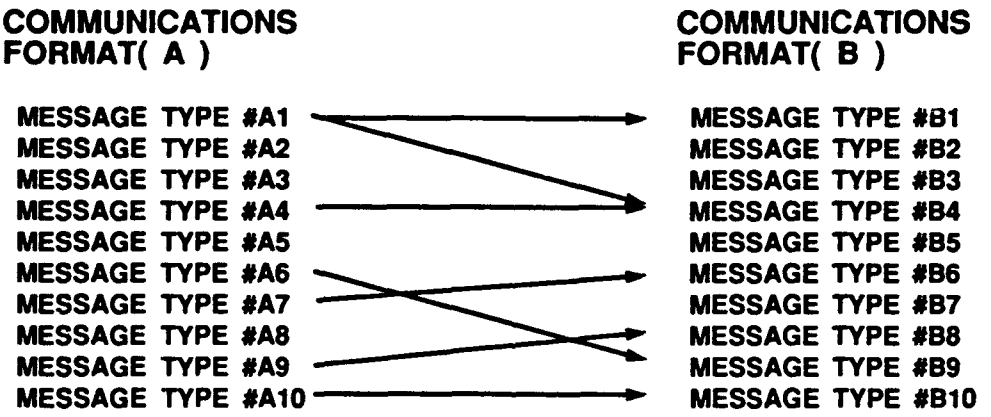


Figure 4-3. Notional Message Translation Mapping[6]

The PSDL description follows:

```
OPERATOR forward_for_translation
SPECIFICATION
   INPUT  transmission_message : transmission_command
   OUTPUT raw_message          : translation_command,
          ready_message        : transmission_command
   STATES translate : boolean INITIALLY false
```

```
   MAXIMUM EXECUTION TIME 100 ms
   DESCRIPTION { checks if message is needed to be
                 translated, and sets the state variable }
END
IMPLEMENTATION Ada forward_for_translation
END
```

iii. Operator forward_for_transmission

As it is determined in the control constraints, this operator is triggered by either ready_message or tcd_emission_control. It is a state machine with state variable *waiting_messages*. As long as the emissions are restricted by tcd_emission_control, the messages for transmission are written into the state variable waiting_messages. When silence is lifted, again by tcd_emission_control, all messages in state variable are written into output stream output_messages. While there is no silence, input ready_message is written into output stream. Output of this operator is guarded by the values of the input stream tcd_emission_control. The PSDL description follows:

```
OPERATOR forward_for_transmission
SPECIFICATION
   INPUT  ready_message         : transmission_command,
          tcd_emission_control  : emissions_control_command
   OUTPUT output_messages       : message_list
   STATES waiting_messages : message_list INITIALLY null
   MAXIMUM EXECUTION TIME 200 ms
   DESCRIPTION { queues the input while there is silence,
                 and dequeues when silence lifted }
END
IMPLEMENTATION Ada forward_for_transmission
END
```

iv. Operator convert_to_text_file

This sporadic operator prepares a text file which includes the message to be transmitted. The PSDL description follows.

```
OPERATOR convert_to_text_file
SPECIFICATION
   INPUT  output_messages       : message_list
   MAXIMUM EXECUTION TIME 100 ms
   DESCRIPTION { writes the message into a text file }
END
IMPLEMENTATION Ada convert_to_text_file
END
```

3. Operator prepare_periodic_report

This operator is responsible for producing periodic track reports. This atomic operator gets the tracks from the database, and periodically prepares a message that contains tracks with matching values determined by initiate_trans command. This process terminates by terminate_trans command. The PSDL description follows:

```
OPERATOR prepare_track_report
SPECIFICATION
   INPUT  initiate_trans        : initiate_transmission_
                                  sequence,
          terminate_trans       : boolean
   OUTPUT tcd_transmit_command : transmit_command
   MAXIMUM EXECUTION TIME 500 ms
   DESCRIPTION { produces periodic track report  }
END
IMPLEMENTATION Ada prepare_track_report
END
```

4. Operator translate_message

This operator is supposed to perform message format translation in the real implementation. Since the message formats are classified, for prototyping purposes, it just makes a type conversion between the input and output data streams. The PSDL description follows :

```
OPERATOR translate_message
SPECIFICATION
   INPUT  raw_message    : translation_command
   OUTPUT ready_message  : transmission_command
   MAXIMUM EXECUTION TIME 300 ms
   DESCRIPTION { performs message format translation }
END
IMPLEMENTATION Ada translate_message
END
```

C. SENSOR INTERFACE

The type of sensors used by the U.S. Navy varies from platform to platform. There are many differences between the sensors such as speed, data accuracy, tracking capability, maximum number of tracks that can be tracked, etc. As it was advised in the requirements document [Ref. 7], the prototype is designed for a shipboard example. The primary sensor systems in the prototype are SPY-1 radar, SAR-8 IRST, SLQ-32 ESM device, and SQS-53C bow mounted sonar.

The Sensor Interface is still thought to be complex and it is decomposed into atomic operators. The functional decomposition of the interface is in Figure 4-4. The functions performed by these atomic modules will be discussed in detail in following sections.

The timing constraints related with this operator were given as follows:

- The inertial navigation system updates the ownship velocity every 40.96 ms and transmits this nominally every 983.04 ms. The latitude and longitude are updated every 1.3 seconds.

- Provisionally for prototyping purposes, the maximum number of tracks per sensor will be assumed to be 100 tracks per second.

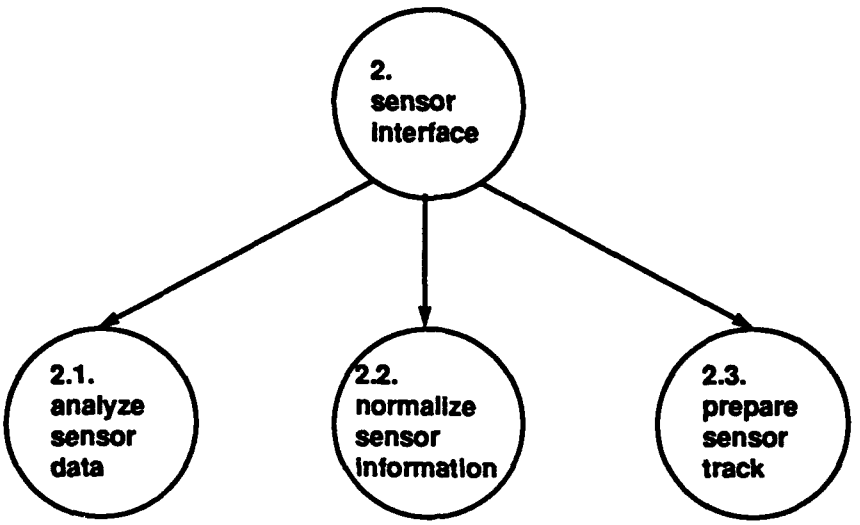


Figure 4-4. Decomposition of Sensor Interface Module

The second constraint is used to determine the minimum calling period of this interface. Since there will be 4 sensors connected to the prototype C^3I workstation, maximum number of tracks coming into the system will be 4 * 100 = 400 per second.

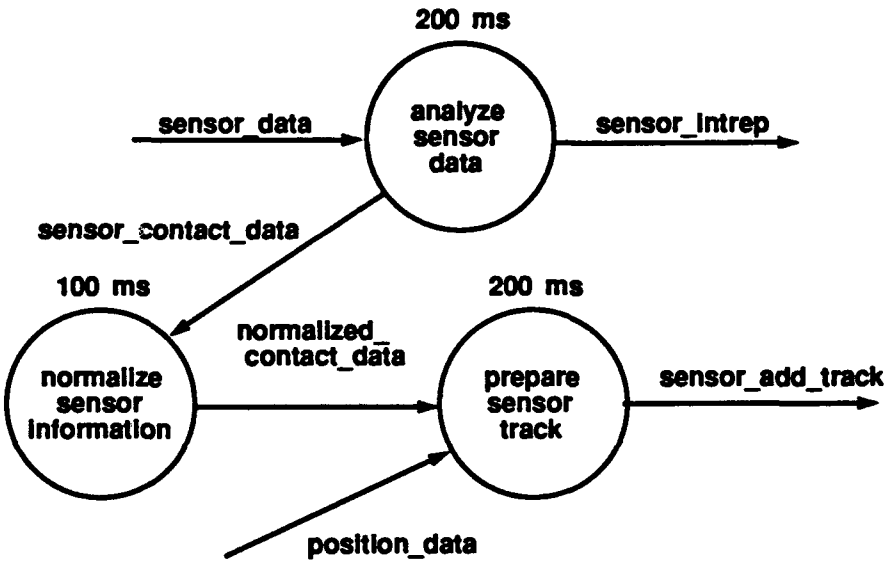
There was not a specific timing constraint for maximum time required to process a sensor track data. An assumption was made that a sensor track data shall be processed by the sensor interface within 500 ms after arriving the interface. Therefore minimum calling period for this interface is 500 / 200 = 2.5 ms. Maximum response time is 500 ms as assumed. These timing constraints were stated in PSDL description of operator c3i_system.

Maximum execution time for the atomic operators are again given arbitrarily. The PSDL description follows:

```
OPERATOR sensor_interface
SPECIFICATION
   INPUT  sensor_data        : sensor_record,
```

```
        position_data      : ownship_navigation_info
  OUTPUT sensor_add_track  : add_track_tuple,
        sensor_intrep      : intelligence_report

  DESCRIPTION { this operator processes the sensor
                track data }
END
IMPLEMENTATION
  GRAPH
```




```
  DATA STREAMS
    sensor_contact_data      : local_track_info,
    normalized_contact_data : local_track_info
  CONTROL CONSTRAINTS
    OPERATOR analyze_sensor_data
      TRIGGERED BY SOME sensor_data
    OPERATOR normalize_sensor_information
      TRIGGERED BY SOME sensor_contact_data
    OPERATOR prepare_sensor_track
      TRIGGERED BY normalized_contact_data, position_data
END
```

2. Operator analyze_sensor_data

In the actual system this operator is an expert system that acts as a track filter, eliminating duplicate or redundant information. It insures the consistency and uniqueness of track identification and labeling. As a second major function, based upon the intelligence data supplied by the sensors, such as ESM and ECM information, it also provides decision support.

For prototyping purposes, this operator is designed to separate contact and intelligence information provided by the sensors. The PSDL description follows:

```
OPERATOR analyze_sensor_data
SPECIFICATION
   INPUT  sensor_data          : sensor_record
   OUTPUT sensor_contact_data : local_track_info,
          sensor_intrep        : intelligence_report
   MAXIMUM EXECUTION TIME 200 ms
   DESCRIPTION { separates the contact and intelligence
                 data portions of input sensor data }
END
IMPLEMENTATION Ada analyze_sensor_data
END
```

3. Operator normalize_sensor_information

Sensors on a platform are located separately anywhere from a few meters apart, to a hundred meters apart, or more. This module modifies the information provided by a sensor depending on its physical location relative to a reference location on the platform. Thus all track information from different sensors will be reported from the same reference point.

For prototyping purposes, since this module will completely depend on the type of the platform and positions of each individual sensors on that platform, no

computations and modifications are made. It has been placed in the decomposition for the completeness. The PSDL description follows:

```
OPERATOR normalize_sensor_information
SPECIFICATION
   INPUT  sensor_contact_data      : local_track_info
   OUTPUT normalized_contact_data : local_track_info
   MAXIMUM EXECUTION TIME 100 ms
   DESCRIPTION { for prototyping purposes, it just
                 assigns the input into output }
END
IMPLEMENTATION Ada normalize_sensor_information
END
```

4. Operator prepare_sensor_track

The position information is provided relative to own platform by the sensors. This operator is triggered by this relative track information, and it calculates the absolute position of the track by using the latest own platform position time information. The PSDL description follows :

```
OPERATOR prepare_sensor_track
SPECIFICATION
   INPUT  normalized_contact_data : local_track_info,
          position_data           : ownship_navigation_
                                    information
   OUTPUT sensor_add_track        : add_track_tuple
   MAXIMUM EXECUTION TIME 200 ms
   DESCRIPTION { converts relative location data to
                 absolute values }
END
IMPLEMENTATION Ada prepare_sensor_track
END
```

D. TRACK DATABASE MANAGER

The track database manager is responsible for integration and synthesis of tracks received from communications systems and organic platform sensors. It is responsible for including all new tracks, delete old tracks and permitting rapid access to information fields for use by the user. A monitor will scan the database periodically to match, merge, correlate or delete unnecessary tracks.

Based on these simplified general functions, the decomposition shown in Figure 4-5 was produced.

The timing constraints specified for this operator are :

- Provisionally the generic C^3I workstation should be capable of retrieving data from track database up to 1000 tracks in less than one second.
- From the time a track data message is received by the system and its contents are entered into the track database should be less than 2 seconds.
- The inertial navigation system updates the ownship velocity every 40.96 ms and transmits this nominally every 983.04 ms. The latitude and longitude are updated every 1.3 seconds.

The second constraint is used to determine the maximum response time of the operator update_tracks. 500 ms of the given 2 seconds was used by the communications interface to process the incoming message. The maximum response time of operator update_tracks is determined to be 500 ms. So, for a track data message, it takes 1000 ms to enter its contents into the track database, and these timing constraints satisfy the given requirement.

The last constraint is used to determine the type of the operator monitor_ownship_position. Since the input data comes periodically, this operator is designed as a periodic operator with a period equal to the input data flow rate. There was no timing constraint specified in the requirements document for the period of monitoring the database. The prototype is designed to perform this function every 2 seconds.

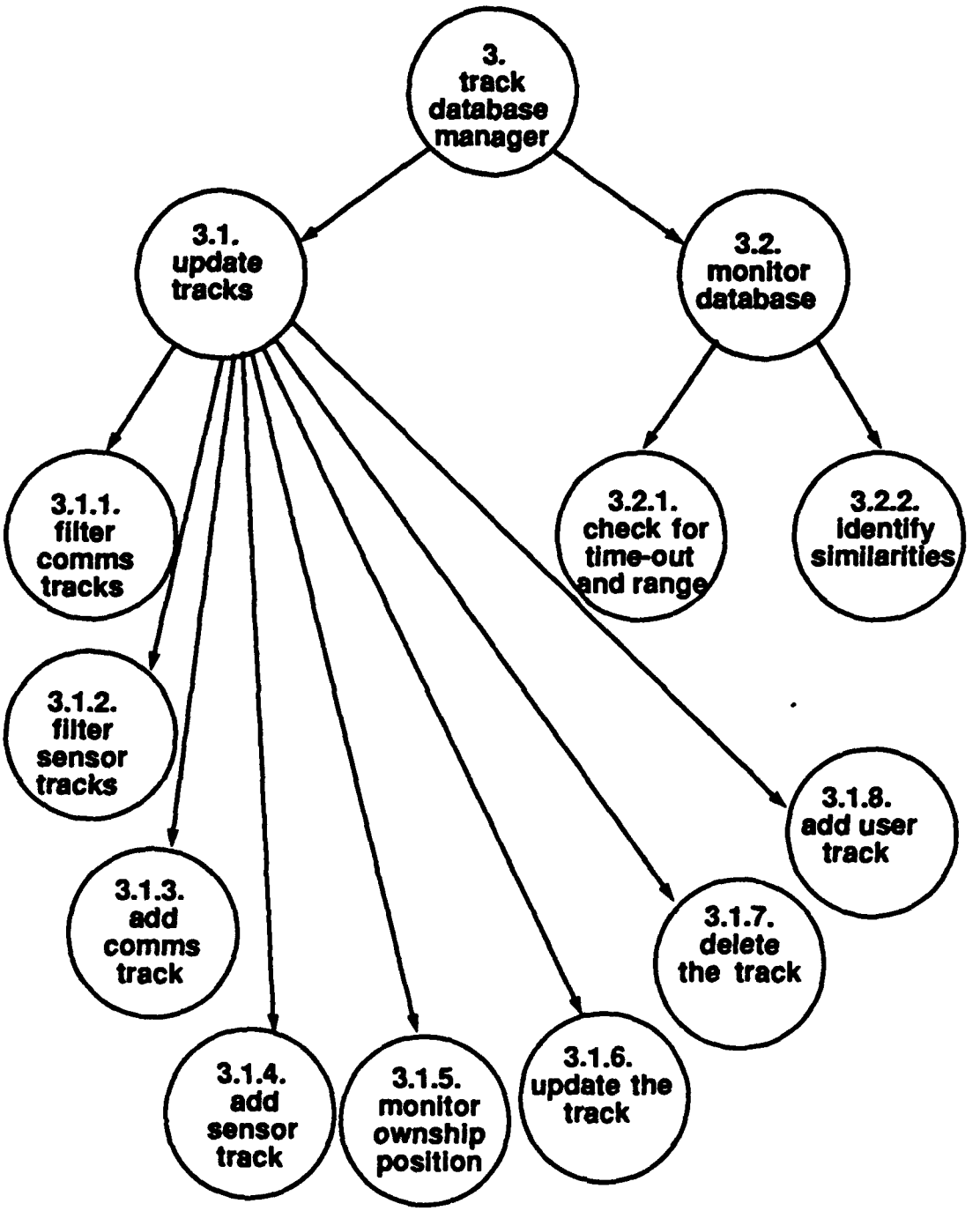
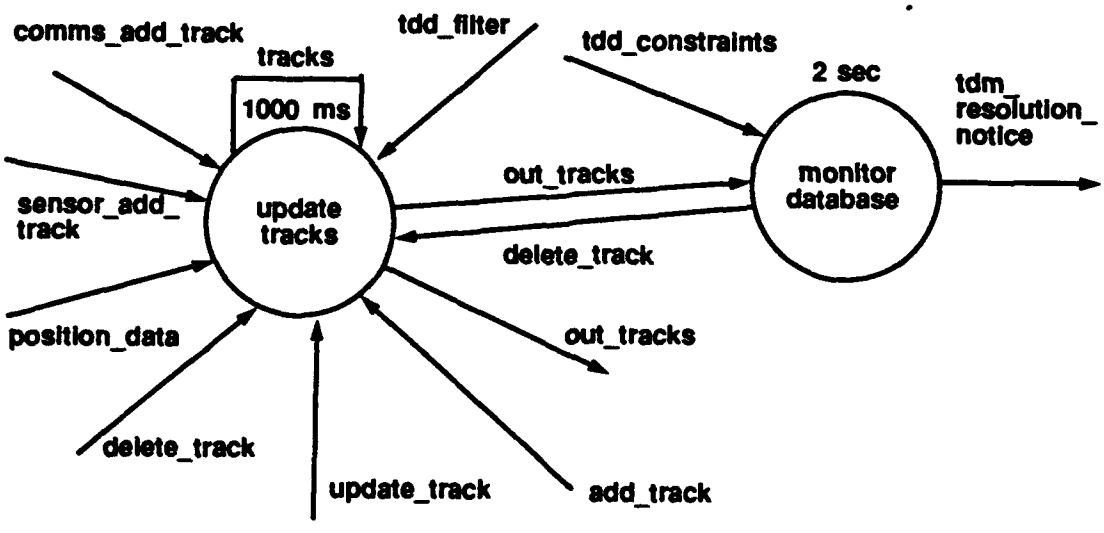


Figure 4-5. Decomposition of Track Database Manager Module

The PSDL description follows:

```
OPERATOR track_database_manager
SPECIFICATION
   INPUT    comms_add_track        : add_track_tuple,
            sensor_add_track       : add_track_tuple,
            add_track              : add_track_tuple,
            update_track           : update_track_tuple,
            delete_track           : delete_track_tuple,
            tdd_filter             : set_track_filter,
            tdd_constraints        : set_monitor_constraints,
            position_data          : ownship_navigation_info
    OUTPUT  out_tracks             : track_tuple,
            tdm_resolution_notice  : resolution_notice,

    DESCRIPTION { manages the track database }
END
IMPLEMENTATION
 GRAPH
```




```
  CONTROL CONSTRAINTS
    OPERATOR update_tracks
    OPERATOR monitor_database
```

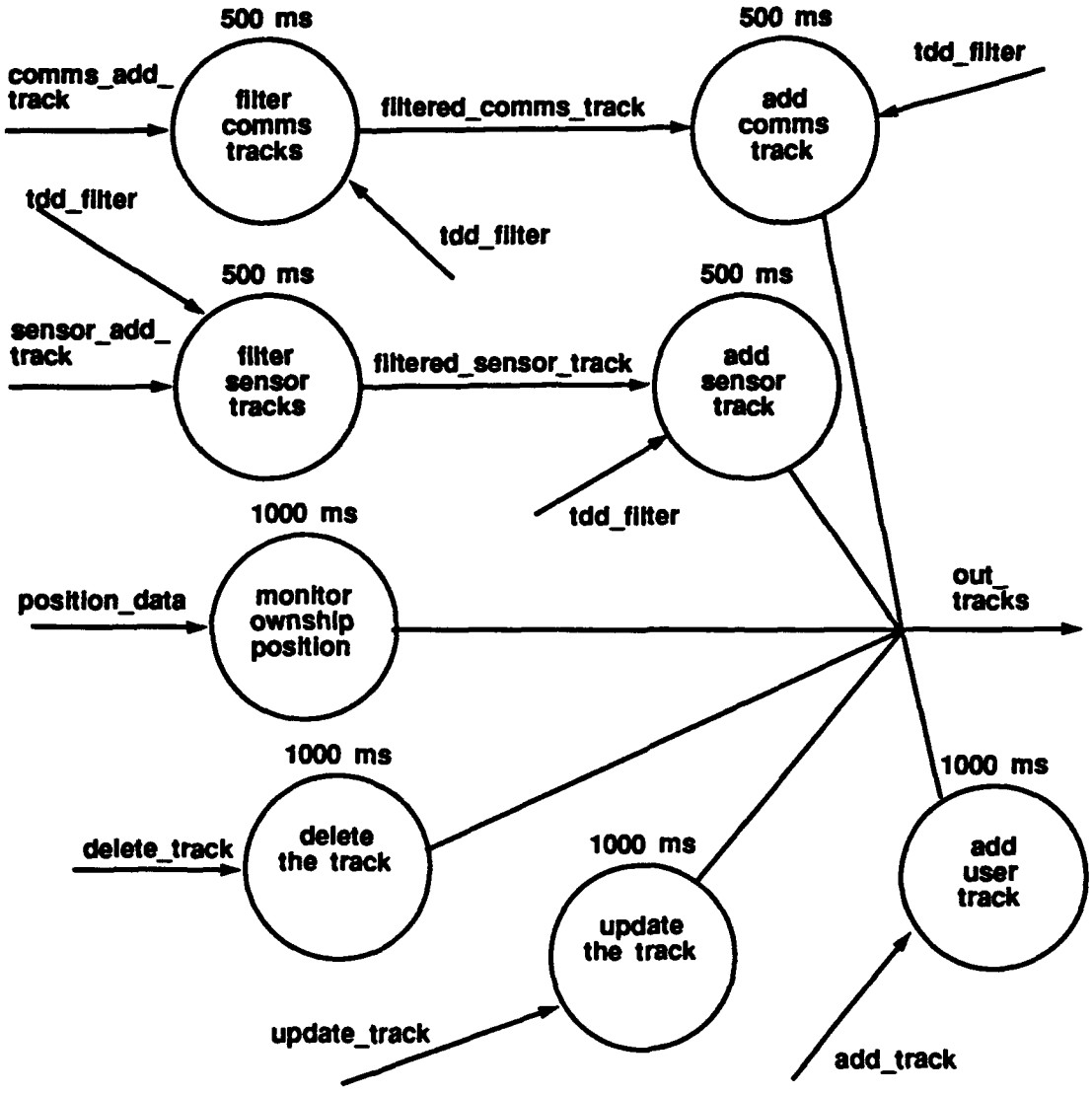
```
        PERIOD 2 sec
END
```

The details about the operators in the decomposition at this level are given below.

1. *Operator update_tracks*

This module performs the functions for adding new tracks into the track database, deleting tracks from the database, and updating tracks in the database. It is composed of several atomic operators and the details of them will be provided in the following sections. The PSDL description follows:

```
OPERATOR update_tracks
 SPECIFICATION
    INPUT    comms_add_track   : add_track_tuple,
             sensor_add_track  : add_track_tuple,
             add_track         : add_track_tuple,
             update_track      : update_track_tuple,
             delete_track      : delete_track_tuple,
             tdd_filter        : set_track_filter,
             position_data     : ownship_navigation_info
    OUTPUT   out_tracks        : track_tuple
    STATES   tracks            : track_tuple
    INITIALLY null
    DESCRIPTION { adds, deletes, updates tracks }
 END
 IMPLEMENTATION
   GRAPH
```




```
DATA STREAMS
  filtered_comms_track  : add_track_tuple,
  filtered_sensor_track : add_track_tuple
CONTROL CONSTRAINTS
  OPERATOR filter_comms_tracks
    TRIGGERED BY SOME comms_add_track
  OPERATOR filter_sensor_tracks
    TRIGGERED BY SOME sensor_add_track
  OPERATOR add_comms_track
```

```
    TRIGGERED BY SOME filtered_comms_track
  OPERATOR add_sensor_track
    TRIGGERED BY SOME filtered_sensor_track
  OPERATOR monitor_ownship_position
    TRIGGERED BY SOME position_data
  OPERATOR delete_the_track
    TRIGGERED BY SOME delete_track
  OPERATOR update_the_track
    TRIGGERED BY SOME update_track
  OPERATOR add_user_track
    TRIGGERED BY SOME add_track
END
```

i. Operator filter_comms_tracks

This operator prevents unwanted tracks from entering the track database. It is triggered by *comms_add_track*, and depending on the filtering conditions set by the user via *tdd_filter*, only the tracks satisfying the filtering conditions are placed into the output stream. The PSDL description follows :

```
OPERATOR filter_comms_tracks
SPECIFICATION
   INPUT   tdd_filter           : set_track_filter,
           comms_add_track      : add_track_tuple
   OUTPUT  filtered_comms_track : add_track_tuple
   MAXIMUM EXECUTION TIME 500 ms
   DESCRIPTION { filters the tracks depending on the
                 values specified by tdd_filter }
END
IMPLEMENTATION Ada filter_comms_tracks
END
```

ii. Operator filter_sensor_tracks

This operator performs a similar filtering operation for the tracks coming from the sensor interface. The PSDL description follows :

```
OPERATOR filter_sensor_tracks
SPECIFICATION
   INPUT   tdd_filter            : set_track_filter,
           sensor_add_track      : add_track_tuple
   OUTPUT  filtered_sensor_track : add_track_tuple
   MAXIMUM EXECUTION TIME 500 ms
   DESCRIPTION { filters the tracks depending on the
                 values specified by tdd_filter }
END
IMPLEMENTATION Ada filter_sensor_tracks
END
```

iii. Operator add_comms_track

This atomic operator adds the filtered comms tracks to the track database. Tracks are added to the database with a prioritizing scheme as defined in the requirements document [Ref. 7] . Air tracks are placed at the top of the database, followed by surface and then subsurface tracks. Within these demarcations, the tracks are ordered on the range from our station. The PSDL description follows:

```
OPERATOR add_comms_track
SPECIFICATION
   INPUT  filtered_comms_track : add_track_tuple,
          tdd_filter           : set_track_filter
   OUTPUT out_tracks           : track_tuple
   MAXIMUM EXECUTION TIME 500 ms
   DESCRIPTION { adds comms tracks to the database }
END
IMPLEMENTATION Ada add_comms_track
END
```

iv. Operator add_sensor_track

This atomic operator adds the filtered sensor tracks to the track database as explained above. The PSDL description follows:

```
OPERATOR add_sensor_track
SPECIFICATION
   INPUT  filtered_sensor_track : add_track_tuple,
          tdd_filter            : set_track_filter
   OUTPUT out_tracks            : track_tuple
   MAXIMUM EXECUTION TIME 500 ms
   DESCRIPTION { adds sensor tracks to the database }
END
IMPLEMENTATION Ada add_sensor_track
END
```

v. Operator monitor_ownship_position

This is an atomic operator that acts as an interface with ownship navigation system. In the actual implementation, this operator would be concerned with the message protocols, and bit patterns necessary to retrieve information from the navigation system. For the prototype system, as an interface, this operator parses through representative navigation data and modifies the own position data in the track database. The PSDL description follows:

```
OPERATOR monitor_ownship_position
SPECIFICATION
   INPUT  position_data : ownship_navigation_info
   OUTPUT out_tracks    : track_tuple
   MAXIMUM EXECUTION TIME 1000 ms
   DESCRIPTION { parses through the input and modifies the
                 own position in database }
END
IMPLEMENTATION Ada monitor_ownship_position
END
```

vi. Operator update_the_track

This atomic operator updates the specified track in the track database. The PSDL description follows:

```
OPERATOR update_the_track
SPECIFICATION
   INPUT  update_track : update_track_tuple
   OUTPUT out_tracks   : track_tuple
   MAXIMUM EXECUTION TIME 1000 ms
   DESCRIPTION { updates the specified track }
END
IMPLEMENTATION Ada update_the_track
END
```

vii. Operator delete_the_track

This atomic operator deletes the specified track from the database. The PSDL description follows:

```
OPERATOR delete_the_track
SPECIFICATION
   INPUT  delete_track : delete_track_tuple
   OUTPUT out_tracks   : track_tuple
   MAXIMUM EXECUTION TIME 1000 ms
   DESCRIPTION { deletes the track from the database }
END
IMPLEMENTATION Ada delete_the_track
END
```

viii. Operator add_user_track

This atomic operator adds the tracks entered by the user. The PSDL description follows:

```
OPERATOR add_user_track
SPECIFICATION
   INPUT  add_track  : add_track_tuple
   OUTPUT out_tracks : track_tuple
   MAXIMUM EXECUTION TIME 1000 ms
   DESCRIPTION { adds user tracks to the database }
```

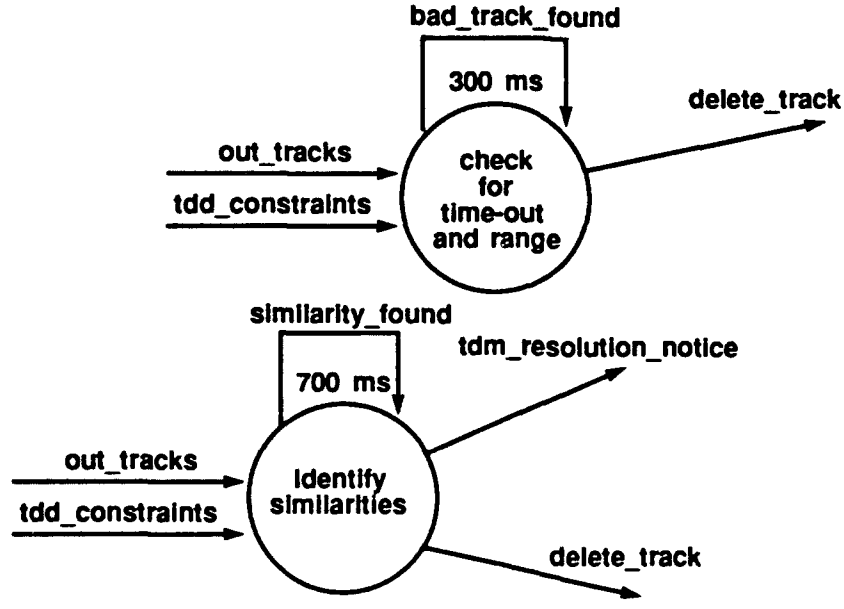
```
END
IMPLEMENTATION Ada add_user_track
END
```

2. Operator monitor_database

The purpose of this module is to scan the track database periodically with the goal of removing old, unwanted, and redundant tracks. This is performed by specified atomic operators. The PSDL description follows:

```
OPERATOR monitor_database
SPECIFICATION
   INPUT   tdd_constraints        : set_monitor_constraints,
           out_tracks             : track_tuple
   OUTPUT  delete_track           : delete_track_tuple,
           tdm_resolution_notice  : resolution_notice,

   DESCRIPTION { scans the database for old, unwanted and
                 redundant tracks }
END
IMPLEMENTATION
  GRAPH
```



```
CONTROL CONSTRAINTS
  OPERATOR check_for_time-out_and_range
    OUTPUT delete_track IF bad_track_found
  OPERATOR identify_similarities
    TRIGGERED IF NOT mode = OFF
    OUTPUT tdm_resolution_notice IF mode = ADVISE AND
           similarity_found,
           delete_track IF mode = AUTOMATIC AND
           similarity_found
END
```

The details about the atomic operators are explained below.

i. Operator check_for_timeout_and_range

This atomic operator scans the track database and forwards the id's of the tracks with older refreshing time and greater range than specified by the constraints. It is implemented as a state machine, and output is placed if the state variable *bad_track_found* is set to true. The state variable is set to true if any out-timed or out-ranged track is found in the track database. The PSDL description follows:

```
OPERATOR check_for_time-out_and_range
SPECIFICATION
   INPUT   out_tracks          : track_tuple,
           tdd_constraints     : set_monitor_constraints
   OUTPUT  delete_track        : delete_track_tuple
   STATES  bad_track_found     : boolean INITIALLY false
   MAXIMUM EXECUTION TIME 300 ms
   DESCRIPTION { scans database for aged and out ranged
                 tracks }
END
IMPLEMENTATION Ada check_for_timeout_and_range
END
```

ii. Operator identify_similarities

This operator scans the database for redundant tracks. If a similarity is found, than the state variable is set to true, and depending on the operation mode specified by tdd_constraints (advise or automatic), either the track id is forwarded to operator update_tracks for deletion, or a notice is sent to the user. The PSDL description follows:

```
OPERATOR identify_similarities
SPECIFICATION
   INPUT    out_tracks             : track_tuple,
            tdd_constraints        : set_monitor_constraints
   OUTPUT   tdm_resolution_notice  : resolution_notice,
            delete_track           : delete_track_tuple
   STATES   similarity_found : boolean INITIALLY false
   MAXIMUM EXECUTION TIME 700 ms
   DESCRIPTION { scans the database for track resolution }
END
IMPLEMENTATION Ada identify_similarities
END
```

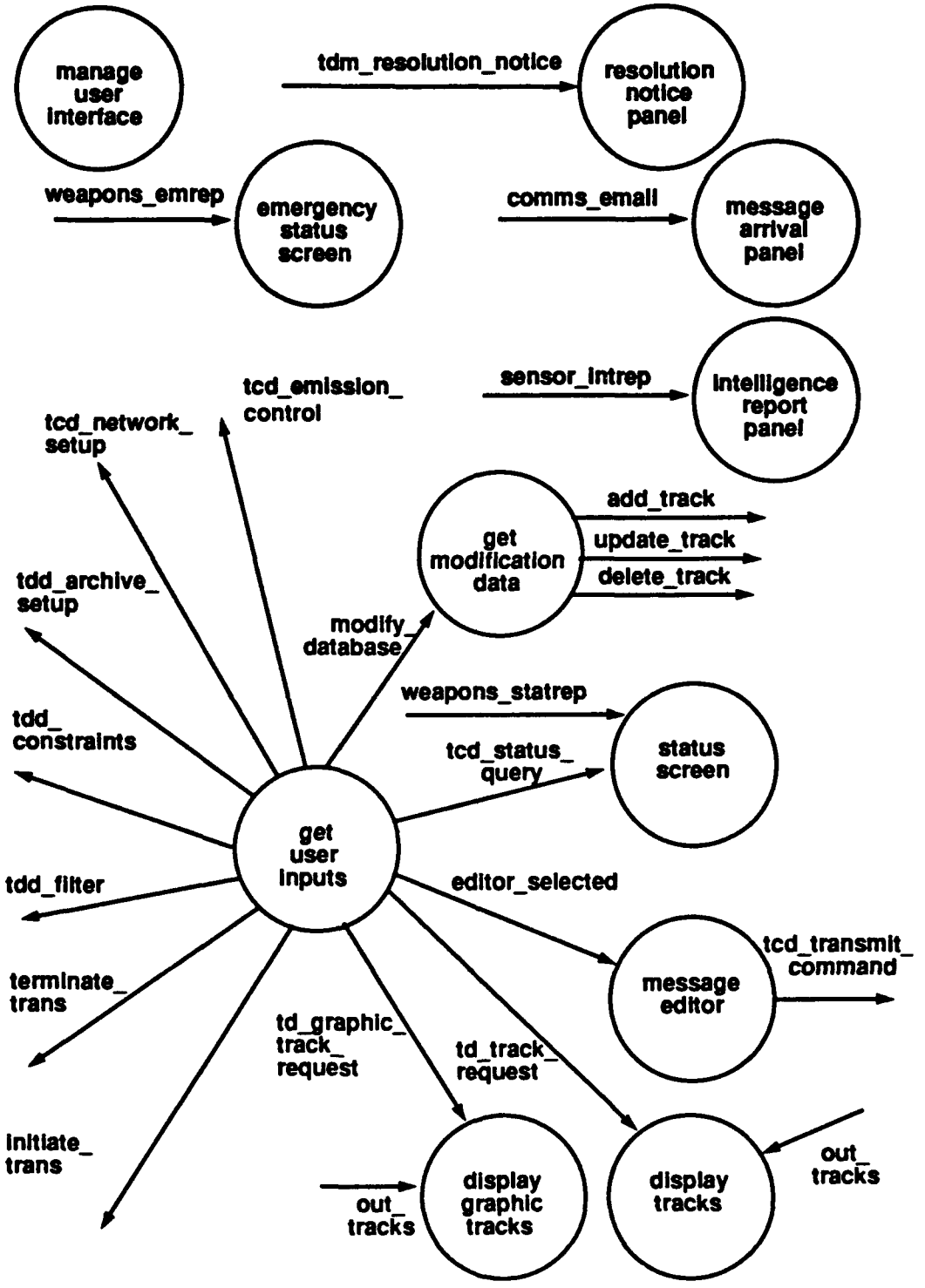
E. USER INTERFACE

The user interface for generic C^3I workstation prototype is implemented with version 4.1 release of Transportable Applications Environment (TAE) Plus. TAE Plus provides an integrated and consistent environment for developing and running both contemporary window-based applications with graphical, point-and-click user interfaces based on the X Window System, and more traditional alpha-numeric-based applications [Ref. 8].

After preparing and running the implemented user interface within TAE Plus workbench, the software provides a skeleton of an executable Ada program for the

user interface. Then, this Ada program is modified to meet the requirements of the whole system. The PSDL description follows:

```
OPERATOR user_interface
SPECIFICATION
   INPUT   comms_email             : filename,
           sensor_intrep           : intelligence_report,
           out_tracks              : track_tuple,
           tdm_resolution_notice   : resolution_notice,
           weapons_statrep         : weapon_status_report,
           weapons_emrep           : weapon_status_report,
   OUTPUT  delete_track            : delete_track_tuple,
           add_track               : add_track_tuple,
           update_track            : update_track_tuple,
           tdd_filter              : set_track_filter,
           tdd_constraints         : set_monitor_constraints,
           tcd_transmit_command    : transmit_command,
           terminate_trans         : boolean,
           tdd_archive_setup       : archive_setup,
           initiate_trans          : initiate_transmission_
                                     sequence,
           tcd_network_setup       : network_setup,
           tcd_emission_control    : emissions_control_
                                     command
   DESCRIPTION { interfaces with the user }
END
IMPLEMENTATION
   GRAPH
```


manage user interface
tdm_resolution_notice
resolution notice panel
weapons_emrep
emergency status screen
comms_email
message arrival panel
sensor_intrep
intelligence report panel
tcd_emission_ control
tcd_network_ setup
get modification data
add_track
update_track
delete_track
tdd_archive_ setup
modify_ database
weapons_statrep
tdd_ constraints
tcd_status_ query
status screen
get user inputs
tdd_filter
editor_selected
tcd_transmit_ command
message editor
terminate_ trans
td_graphic_ track_ request
td_track_ request
initiate_ trans
out_ tracks
display graphic tracks
display tracks
out_ tracks

```
DATA STREAMS
  modify_database          : modify_command;,
  tcd_status_query         : boolean,
  editor_selected          : boolean,
  td_track_request         : database_request,
  td_graphic_track_request : database_request

CONTROL CONSTRAINTS
  OPERATOR manage_user_interface
  OPERATOR get_user_inputs
  OPERATOR resolution_notice_panel
    TRIGGERED BY SOME tdm_resolution_notice
  OPERATOR emergency_status_screen
    TRIGGERED BY SOME weapons_emrep
  OPERATOR message_arrival_panel
    TRIGGERED BY SOME comms_email
  OPERATOR intelligence_report_panel
    TRIGGERED BY SOME sensor_intrep
  OPERATOR get_modification_data
    TRIGGERED BY SOME modify_database
  OPERATOR status_screen
    TRIGGERED IF tcd_status_query
  OPERATOR message_editor
    TRIGGERED IF editor_selected
  OPERATOR display_tracks
    TRIGGERED BY SOME out_tracks
  OPERATOR display_graphic_tracks
    TRIGGERED BY SOME out_tracks
END
```

1. Operator manage_user_interface

This is the operator which controls all of interactions with the user. It opens and closes the panels, senses and receives the user inputs, and assigns these inputs to related variables. This operator does not have any input or output variable

specified in the PSDL description since the external input and output is not allowed in the current version of CAPS.

First it displays the main menu panel and gets the user's choice. The main menu panel provides the user following options:

- Archive Setup
- Track Filter
- Display Tracks
- Display Graphic Tracks
- Weapon Status
- Network setup
- EMCON Status
- Message Editor
- Read Message Panel
- Initiate Track Report
- Terminate Track Report
- Modify Database

When user selects one of the options above, the specified panel is displayed, and user enters related data as shown on the panel. These data is used either to set the variables provided by the operator *get_user_inputs*, or triggering other operators in the user interface module.

Operator *manage_user_interface* is the atomic operator which controls other TAE Plus modules. It makes use of many other functions and procedures which are not described by PSDL. The explanation for these functions and how the user interface runs will be provided in the next chapter. The PSDL description follows:

```
OPERATOR manage_user_interface
SPECIFICATION
   DESCRIPTION { controls other user interface modules }
END
IMPLEMENTATION Ada manage_user_interface
END
```

2. Operator get_user_inputs

This atomic operator puts the values supplied by the manage user interface to the output data streams. If the user does not specify some values, default values are used. The PSDL description follows:

```
OPERATOR get_user_inputs
 SPECIFICATION
    OUTPUT modify_database           : modify_command,
           tcd_status_query          : boolean,
           editor_selected           : boolean,
           td_track_request          : database_request,
           td_graphic_track_request  : database_request,
           initiate_trans            : initiate_transmission_
                                       sequence,
           terminate_trans           : boolean,
           tdd_filter                : set_track_filter,
           tdd_constraints           : set_monitor_
                                       constraints,
           tdd_archive_setup         : archive_setup,
           tcd_network_setup         : network_setup,
           tcd_emission_control      : emissions_control_
                                       command
    DESCRIPTION { assigns the values of the out parameters }
 END
 IMPLEMENTATION Ada get_user_inputs
 END
```

3. Operator resolution_notice_panel

This atomic operator displays the notice sent by the track database manager. The PSDL description follows:

```
OPERATOR resolution_notice_panel
SPECIFICATION
```

```
   INPUT  tdm_resolution_notice : resolution_notice
   DESCRIPTION { displays the resolution notice }
END
IMPLEMENTATION Ada resolution_notice_panel
END
```

4. Operator emergency_status_screen

This atomic operator displays the emergency weapon status sent by the weapons interface. The PSDL description follows:

```
OPERATOR emergency_status_screen
SPECIFICATION
   INPUT  weapons_emrep : weapon_status_report
   DESCRIPTION { displays emergency weapon status }
END
IMPLEMENTATION Ada emergency_status_screen
END
```

5. Operator message_arrival_panel

This atomic operator displays the name of the file which consists a the newly arrived message. If user choices to sec the message, it will be displayed for browsing. The PSDL description follows:

```
OPERATOR message_arrival_panel
SPECIFICATION
   INPUT  comms_email : filename
   DESCRIPTION { displays the name of the file which has
                 the newly arrived message in it }
END
IMPLEMENTATION Ada message_arrival_panel
END
```

6. Operator intelligence_report_panel

This atomic operator displays the intelligence report supplied by the sensors interface. The PSDL description follows:

```
OPERATOR intelligence_report_panel
SPECIFICATION
   INPUT  sensor_intrep : intelligence_report
   DESCRIPTION { displays the intelligence report }
END
IMPLEMENTATION Ada intelligence_report_panel
END
```

7. Operator get_modification_data

This atomic operator displays a panel with three options in it. By these options, the user can add a track into the database manually, or he can delete or update an existing track in the database. The PSDL description follows:

```
OPERATOR get_modification_data
SPECIFICATION
   INPUT  modify_database : modify_command
   OUTPUT add_track       : add_track_tuple,
          update_track    : update_track_tuple,
          delete_track    : delete_track_tuple
   DESCRIPTION { gets track database modification values }
END
IMPLEMENTATION Ada get_modification_data
END
```

8. Operator status_screen

This atomic operator displays the weapon status supplied by the weapons interface. The PSDL description follows:

```
OPERATOR status_screen
SPECIFICATION
   INPUT  weapons_statrep  : weapon_status_report,
          tcd_status_query : boolean
   DESCRIPTION { displays the weapon status }
END
IMPLEMENTATION Ada status_screen
END
```

9. Operator message_editor

This atomic operator provides the user an empty message template. The user can type in the fields of the template to complete the message. The PSDL description follows:

```
OPERATOR message_editor
SPECIFICATION
   INPUT  editor_selected       : boolean
   OUTPUT tcd_transmit_command : transmit_command
   DESCRIPTION { provides the message template }
END
IMPLEMENTATION Ada message_editor
END
```

10. *Operator display_tracks*

This atomic operator first gets the filtering values, such as track type, IFF type, maximum range from the user, and then displays the tracks satisfying the filtering conditions. User can change these filtering conditions any time by querying this panel. The PSDL description follows:

```
OPERATOR display_tracks
SPECIFICATION
```

```
   INPUT  td_track_request : database_request
   DESCRIPTION { displays the tracks textually }
END
IMPLEMENTATION Ada display_tracks
END
```

11. *Operator display_graphic_tracks*

This atomic operator displays the tracks graphically as explained above. The PSDL description follows:

```
OPERATOR display_graphic_tracks
SPECIFICATION
   INPUT  td_graphic_track_request : database_request
   DESCRIPTION { displays the tracks graphically }
END
IMPLEMENTATION Ada display_graphic_tracks
END
```

F. WEAPONS INTERFACE

In the actual C^3I workstation, the Weapons Systems Interface is designed to monitor current weapons systems statuses, including operational availability, reload status, weapons loadout, and any additional information that may be deemed relevant [Ref. 7].

For the prototype workstation, in keeping with the provisional shipboard example, the primary weapons connected to the system will be a PHALANX CIWS, a 5"/54 gun, TOMAHAWK cruise missiles, and Mk 46 torpedoes. It will be assumed that these weapon systems provide real time status data at every 1 second as specified in the timing constraints given in previous chapter.

According to this given timing constraint, there will be 4 weapon status data coming to the interface in 1 second, assuming that the interface shall process an

individual data in 500 ms, the minimum calling period for this interface is 250 ms. Maximum response time is 500 ms. These timing constraints were specified at the first level PSDL description with the guard conditions. The PSDL description follows:

```
OPERATOR weapons_interface
SPECIFICATION
   INPUT  weapon_status_data : weapon_status
   OUTPUT weapons_emrep      : weapon_status_report,
          weapons_statrep    : weapon_status_report
   STATES ciws_status,
          gun_status,
          tws_status,
          mk_48_status       : weapon_status_type
   INITIALLY ready, ready, ready, ready
   MAXIMUM EXECUTION TIME 500 ms
   DESCRIPTION { interfaces with weapon systems, and for-
                 wards weapon status reports }
END
IMPLEMENTATION Ada weapons_interface
END
```

G. EXTERNAL SYSTEMS

These are separate atomic operators simulating the external systems producing data for the prototype C^3I workstation. In general, all of them use a random number generator package to produce random data.

Having these external systems included in the C^3I prototype actually slows down the execution, and effects the timing constraints specified for the other operators. But for the completeness of the prototype, these are included within the C^3I prototype.

Maximum execution times for these operators are given arbitrarily at this moment, and they will be minimized during the execution process through CAPS.

1. Operator comms_links

This operator simulates the communications links attached to the prototype workstation. It is defined as a non time critical operator, so the dynamic scheduler will schedule this operator to produce messages. The PSDL description follows:

```
OPERATOR comms_links
SPECIFICATION
   OUTPUT input_link_message : filename
   DESCRIPTION { produces the input messages }
   MAXIMUM EXECUTION TIME 100 ms
END
IMPLEMENTATION Ada comms_links
END
```

2. Operator sensors

This is an operator which simulates sensors connected to the prototype C^3I workstation. It creates a certain number of tracks initially, and then each time it reports the current position of a track chosen randomly. The initial positions and speed, course values of the tracks are decided randomly by using a random number generator. The operator keeps track of the tracks within a database which is implemented as a linked list, and updates the values as time passes. The PSDL description follows:

```
OPERATOR sensors
SPECIFICATION
   OUTPUT sensor_data : sensor_record
   DESCRIPTION { creates sensor data }
   MAXIMUM EXECUTION TIME 100 ms
END
IMPLEMENTATION Ada sensors
END
```

3. Operator navigation_system

This operator simulates the navigation subsystem of the platform, and produces position data. For the prototyping effort, ownship is assumed to start navigation from a certain geographic position, and steams to a constant route with constant speed. This operator is defined to be a periodic operator to simulate the real navigation system. The PSDL description follows:

```
OPERATOR navigation_system
SPECIFICATION
   OUTPUT position_data : ownship_navigation_info
   DESCRIPTION { creates ownship position data }
   MAXIMUM EXECUTION TIME 100 ms
END
IMPLEMENTATION Ada navigation_system
END
```

4. Operator weapons_systems

This operator simulates the weapons systems attached to the prototype. It produces weapon status data periodically. The PSDL description follows:

```
OPERATOR weapons_systems
SPECIFICATION
   OUTPUT weapon_status_data : weapon_status
   DESCRIPTION { creates weapon status data }
   MAXIMUM EXECUTION TIME 100 ms
END
IMPLEMENTATION Ada weapons_systems
END
```

G. DATA TYPES

A variety of data types are used in the prototype. The definitions of these data types, their associations with the data streams, and the names of the operators using these data streams are provided in Table 4 -1.

TABLE 4-1. User Defined Data Types

DATA TYPE	DATA STREAMS	OPERATORS
add_track_tuple Ada record type that contains track information which will be inserted to the track database.	comms_add_track sensor_add_track add_track filtered_comms_track filtered_sensor_track	c3i_system comms_interface resolve_incoming_messages extract_tracks sensor_interface prepare_sensor_track track_database_manager update_tracks filter_comms_tracks filter_sensor_tracks add_comms_track add_sensor_track add_user_track user_interface get_modification_data
archive_setup Ada record type that contains information to set up the message archiving flags.	tdd_archive_setup	c3i_system comms_interface resolve_incoming_messages decide_for_archiving user_interface get_user_inputs
database_request Ada record type that contains the desired type and range values of the tracks for retrieval from the database.	td_track_request td_graphic_track_request	user_interface get_user_inputs display_tracks display_graphic_tracks
delete_track_tuple Ada record type that contains information for the tracks	delete_track	c3i_system track_database_manager update_tracks delete_the_track

TABLE 4-1. User Defined Data Types

DATA TYPE	DATA STREAMS	OPERATORS
message_list Ada linked list type that contains communications messages.	output_messages waiting_messages	resolve_outgoing_ messages forward_for_transmission convert_to_text_file
modify_command Ada enumeration type which consists of databse modification types (add, update, delete).	modify_database	user_interface get_user_inputs get_modification_data
network_setup Ada array type that contains the names of the stations in four different communications links.	tcd_network_setup	c3i_system comms_interface resolve_incoming_ messages decide_for_relaying resolve_outgoing_ messages make_routing user_interface get_user_inputs
ownship_navigation_info Ada record type that contains the position, velocity and course values for own ship.	position_data	c3i_system sensor_interface prepare_sensor_track track_database_manager update_tracks monitor_ownship_position navigation_system
resolution_notice Ada record type that contains the information about the tracks which are found to be similar by the track database manager.	tdm_resolution_notice	c3i_system track_database_manager monitor_database identify_similarities user_interface resolution_notice_panel
sensor_record Ada record type that contains the track data which is generated by the sensors.	sensor_data	c3i_system sensor_interface analyze_sensor_data sensors

TABLE 4-1. User Defined Data Types

DATA TYPE	DATA STREAMS	OPERATORS
		monitor_database check_for_timeout_and_range identify_similarities user_interface get_modification_data
emissions_control_command Ada record type that contains the current EMCON status.	tcd_emission_control	c3i_system comms_interface resolve_outgoing_messages forward_for_transmission user_interface get_user_inputs
filename Ada string type that contains the name of a message.	input_link_message comms_email	c3i_system comms_interface resolve_incoming_messages parse_input_file decide_for_archiving user_interface message_arrival_panel comms_links
initiate_transmission_sequence Ada record type that contains the track report which will be transmitted to the other stations periodically.	initiate_trans	c3i_system comms_interface prepare_periodic_report user_interface get_user_inputs
intelligence_report Ada record type that contains the intelligence data and the source of the information.	sensor_intrep	c3i_system sensor_interface analyze_sensor_data user_interface intelligence_report_panel
local_track_info Ada record type that contains the track information.	sensor_contact_data normalized_contact_data	sensor_interface analyze_sensor_data normalize_sensor_information prepare_sensor_track

TABLE 4-1. User Defined Data Types

DATA TYPE	DATA STREAMS	OPERATORS
set_monitor_constraints Ada record type which determines monitor constraints.	tdd_constraints	c3i_system track_database_manager monitor_database check_for_timeout_and_range identify_similarities user_interface get_user_inputs
set_track_filter Ada record type which contains filtering data for the tracks.	tdd_filter	c3i_system track_database_manager update_tracks filter_comms_tracks filter_sensor_tracks add_comms_track, add_sensor_track user_interface get_user_inputs
text_record Ada record type that contains a communications message.	input_text_record comms_text_file	resolve_incoming_messages parse_input_file decide_for_relaying decide_for_archiving extract_tracks
track_tuple Ada linked list type of track records.	out_tracks tracks	c3i_system track_database_manager update_tracks add_comms_track add_sensor_track monitor_ownship_position update_the_track delete_the_track add_user_track monitor_database check_for_timeout_and_range identify_similarities user_interface display_tracks display_graphic_tracks

TABLE 4-1. User Defined Data Types

DATA TYPE	DATA STREAMS	OPERATORS
translation_command Ada array type which contains the message text which will be translated.	raw_message	comms_interface resolve_outgoing_messages forward_for_translation translate_message
transmit_command Ada array type which contains message records which will be transmitted	tcd_transmit_command	c3i_system comms_interface resolve_incoming_messages decide_for_relaying resolve_outgoing_messages make_routing prepare_track_report user_interface message_editor
update_track_tuple Ada record type that contains the information for the track which will be updated in the track database.	update_track	c3i_system track_database_manager update_tracks update_the_track user_interface get_modification_data
weapon_status Ada record type that contains status of the weapon systems.	weapon_status_data	c3i_system weapons_interface weapons_systems
weapon_status_report Ada array type that contains current status of the weapon systems	weapons_emrep weapons_statrep	c3i_system user_interface emergency_status_screen status_screen weapons_interface

CHAPTER V

IMPLEMENTATION OF THE C^3I WORKSTATION PROTOTYPE THROUGH CAPS

The prototyping process in the CAPS environment is done by following the steps given below:

- The designer draws the graphs by using the graphic editor.
- The graphic editor provides the skeleton PSDL code.
- The designer modifies the skeleton PSDL code by using the syntax directed editor, and finally system produces *psdl.txt* file which contains the PSDL description of the prototype.
- The translator produces an Ada package, *tl.a*, which handles the instantiation of the data streams, reading data from the data streams, writing data to data streams, executing the atomic operators, and etc. CAPS uses psdl.txt file to produce this driver package.
- The static scheduler tries to find a schedule for the time critical operators, and if a feasible schedule is found, the static scheduler produces *ss.a*, Ada package that contains the static schedule.
- Once the static schedule is found, the dynamic scheduler produces the file *ds.a* which contains the dynamic schedule for the non-time critical operators in the psdl.txt.
- CAPS provides the user with matching Ada reusable components for the atomic operators. If a reusable component can not be found, then the designer either decomposes the operator and tries to find Ada component for them, or he writes the Ada component for that operator. When all the components are found, they are placed in Ada package *sb.a*.
- CAPS compiles and loads sb.a, tl.a, ds.a, and ss.a and starts to execute the prototype.

CAPS environment is still an ongoing research study, and the functions listed above were not all ready when we started prototyping. This chapter explains how these steps are accomplished for C^3I workstation prototype.

A. PREPARING PSDL DESCRIPTION FILE, *PSDL.TXT*

When we started modular decomposition, and defining PSDL descriptions, the graphic and syntax directed editors were not ready to use for a multi level PSDL example. For this reason we have used framemaker to draw the graphs and write the PSDL code.

After completing the multi level decomposition, we prepared the psdl.txt file which includes only the atomic operators in the bottom level of the decomposition. The consistency checking between the levels, and the different modules was done manually.

The psdl.txt file is provided in Appendix B.

B. PRODUCING DRIVER PACKAGE, *TL.A*

The translator gets psdl.txt file as input, and produces tl.a as output.

Package tl.a consists of instantiations of generic package SAMPLED_BUFFER (or FIFO_BUFFER) for each data streams used, and driver procedures for each atomic operator.

During the execution of the prototype, the system continuously gave bus errors at a certain point of the program. After a long debugging effort, we figured out that this error occurs for the data stream which was instantiated as the last package (package c3i_system_package in tl.a). We could not find any meaningful reason for this error, and solved it by adding a new sampled_buffer instantiation at the end of the package, which we actually did not use at all. This problem appears to be caused by a fault in the Verdix Ada Compiler, Version 6.0.

The other problems we encountered either during the translation of psdl.txt or compiling tl.a were mostly solved by Prof. Berzins, V.

The driver package tl.a is provided in Appendix C.

C. BUILDING THE STATIC SCHEDULE, *SS.A*

The static scheduler that we started with constrained the designer to specify minimum calling periods and maximum response times together with the maximum execution times. Later, we thought that it would be better if the static scheduler calculated the undefined mrt and mcp values, depending on the other timing constraints supplied by the designer, and the static scheduler was modified to do so. The algorithm and the corresponding Ada code is provided in Appendix G.

Another problem with the static scheduler occurred during the execution of the prototype. Because of the amount of variables used, the default storage size for the static schedule task was not long enough. For this reason the scheduler package was modified to produce the static schedule package, ss.a, with sufficient storage size. For this modification an Ada representation clause is used.

The static schedule package ss.a is given in Appendix D.

D. BUILDING THE DYNAMIC SCHEDULE, *DS.A*

Similar to static schedule, we encountered a storage size error for the dynamic schedule. The dynamic scheduler was also modified to overcome this error.

The dynamic schedule package produced by the updated dynamic scheduler is provided in Appendix E.

E. BUILDING THE SOFTWARE BASE, *SB.A*

This step of searching, finding and retrieving matching Ada components for the atomic PSDL operators was not yet implemented. In addition, we could not find any matching Ada components manually for almost all of the operators used in C3I. For this reason we had to write Ada components for all of the atomic operators, and test them.

As it was mentioned before, we have used TAE Plus to create the user interface modules. TAE Plus provides the interface designer either Ada or C code for the designed interface. But this tool generated code is not ready to execute, and the designer still has to modify it. The TAE Plus resource file created by the TAE Plus Workbench, user.res, must be placed into the directory where the prototype runs. We thought that it would be very useful to explain how TAE Plus code is modified to get an executable interface code. This is done in Appendix H.

The Ada components are placed in the software base package, sb.a, written in Ada. This is provided in Appendix F. Due to the time limitation of our research, some of the functions could not be integrated into the running version of the prototype. But the Ada code for these functions were also prepared, and tested. This code is given in Appendix I.

CHAPTER VI

CONCLUSIONS AND RECOMMENDATIONS

A. CONCLUSIONS

Due to its complexity and size, it was impossible to realize this prototype by using traditional software engineering techniques and programming languages only. Our research has contributed towards the development of a rapid prototype of a C^3I system software by using the CAPS environment. The required features, characteristics and difficulties with the development of such a software system have been refined as a result of the long term effort which this thesis describes.

This thesis research incorporated the first attempt to make use of a rapid prototyping environment, CAPS, to prototype a hard-real-time software system. This required comprehensive understanding of the major tools provided by the CAPS environment, besides the knowledge of C^3I systems, the PSDL language and other system design issues. As was explained in the previous chapter, using the environment for the first time, we encountered many problems that slowed down the main research effort, and only two thirds of the prototype could be implemented.

The major emphasis of the prototype designed is to support C^3I information management functions such as track information correlation, message generation and information display. Due to the limited research time, some of the minor functions, such as message format translation, automatic message relaying and part of the track database manager is excluded from the prototype. The complete list of these functions is provided in the following recommendations section.

Hard-real-time constraints imposed by the requirements analysis are reflected in the design of the prototype. These timing constraints were feasible for a stand alone

Sun Microsystems Workstation. The prototyping research has been carried out on a multi user Sun system which is much slower than the proposed hardware. This speed difference and the dependency of the system speed on the number of users at the execution time forced us to use longer maximum execution times, and periods to make the prototype run.

This thesis provides a running prototype of a C^3I system with major functions. This work is the first in a series of steps leading toward a complete C^3I software system. Additional research and development is required to identify weaknesses and areas of improvement.

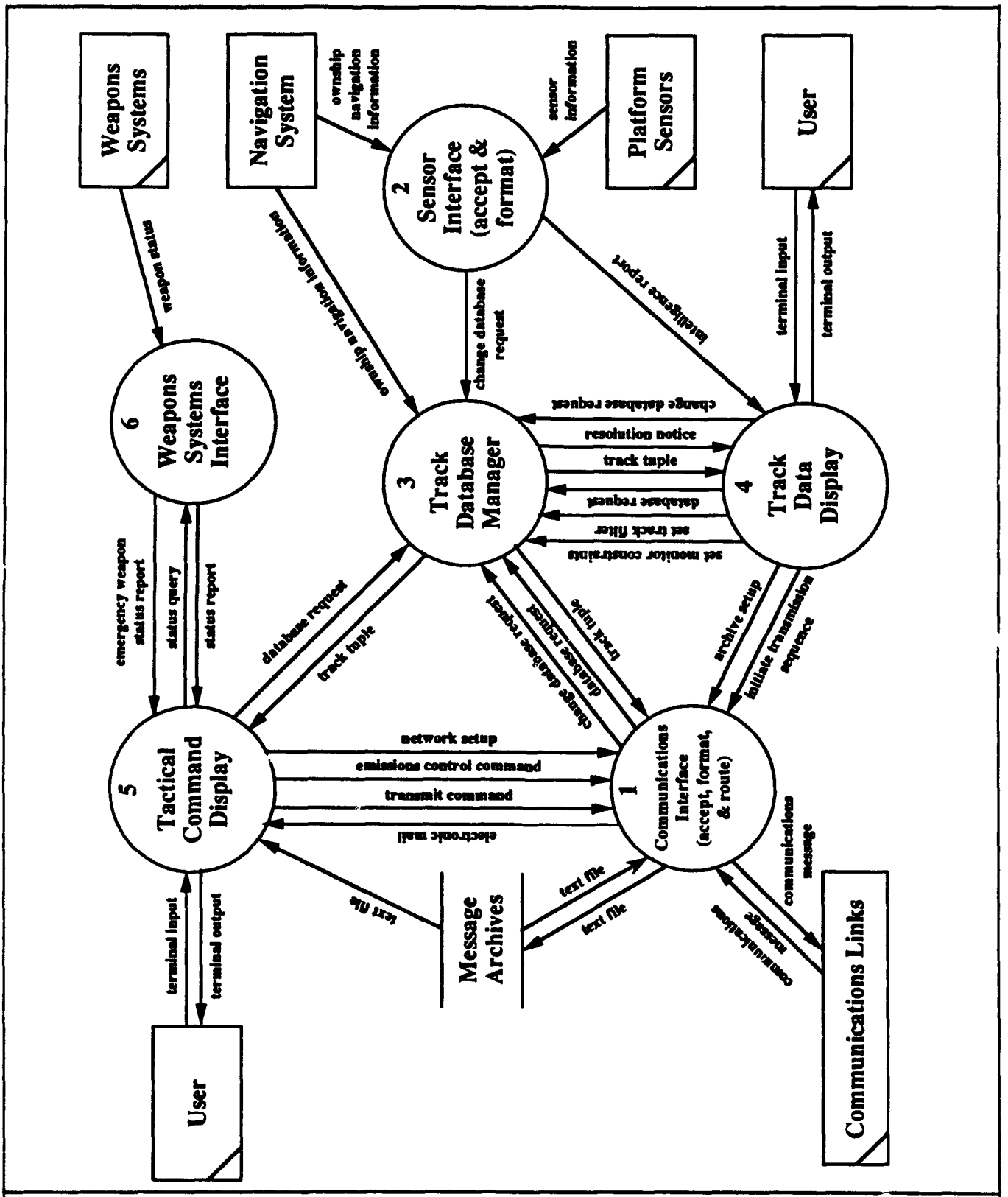
B. RECOMMENDATIONS

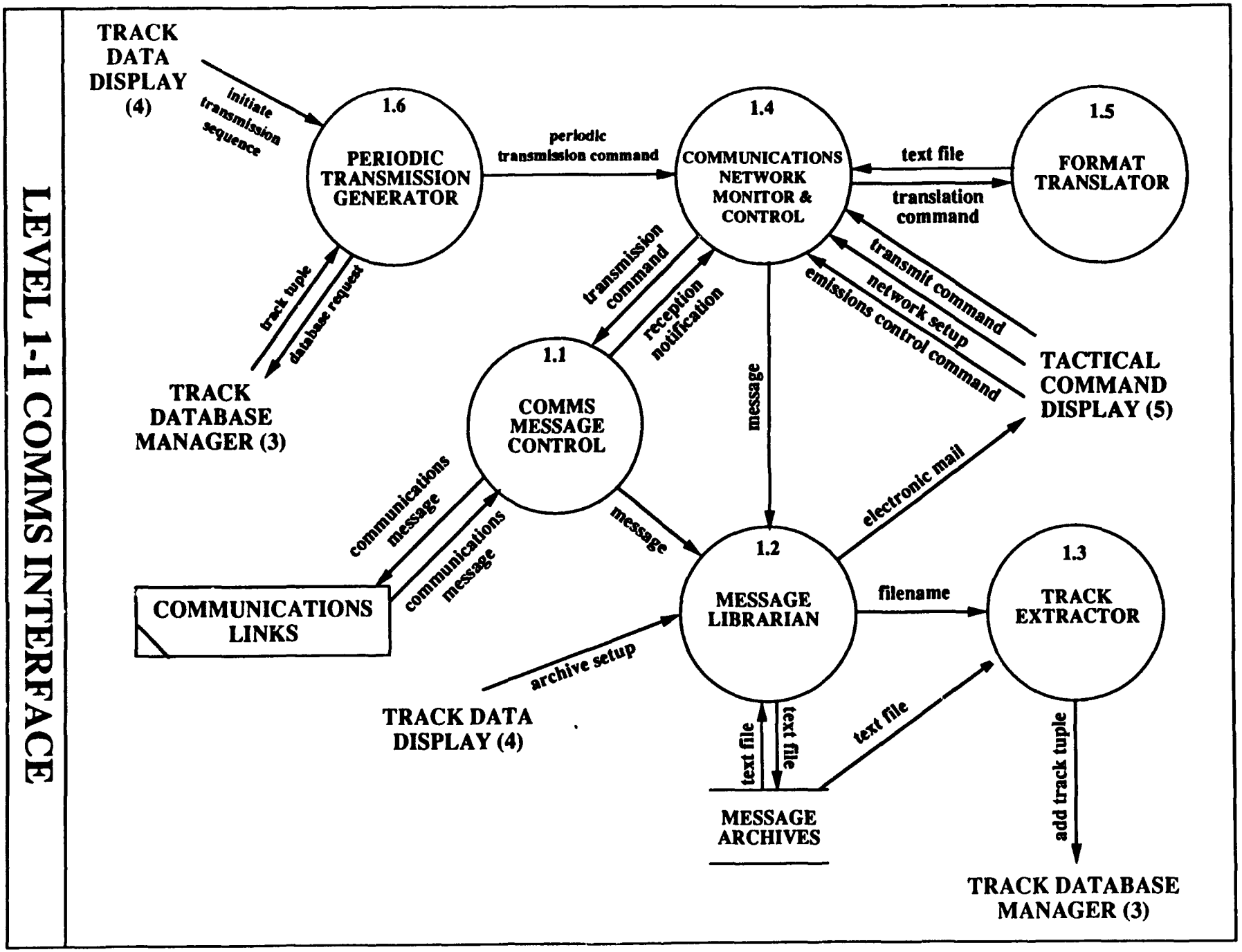
Following is the list of operators which are not implemented in this thesis. Since most of the problems encountered during the prototyping of this version are solved, we believe that adding these features to the prototype is relatively easy.

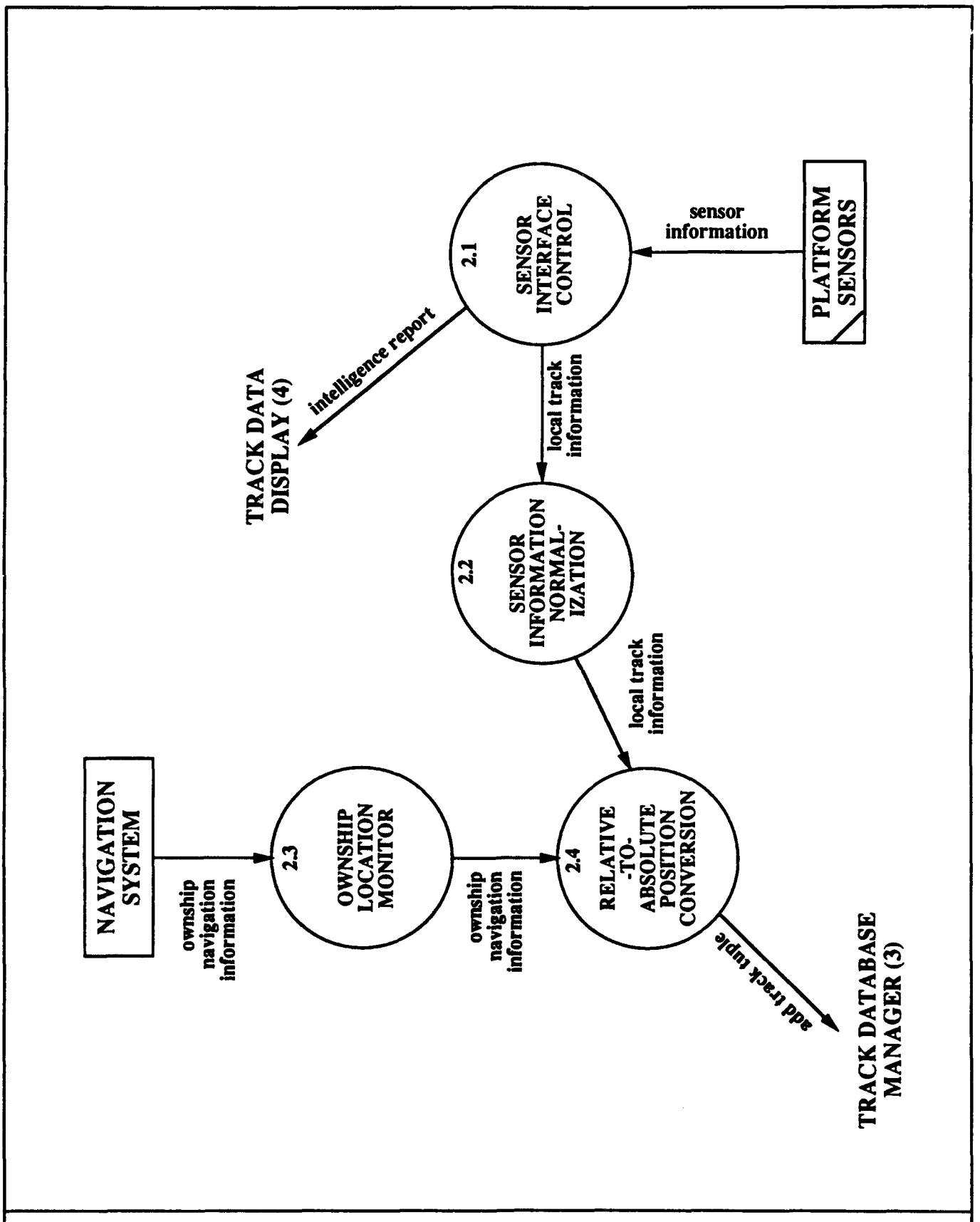
- Operator translate_message
- Operator decide_for_relaying
- Operator forward_for_translation
- Operator normalize_sensor_information
- Operator update_the_track
- Operator delete_the_track
- Operator add_user_track
- Operator check_for_timeout_and_range
- Operator identify_similarities
- Operator resolution_notice_panel
- Operator intelligence_report_panel
- Operator get_modification_data
- Operator display_graphic_tracks

As mentioned earlier, timing constraints used in this version were limited by the hardware. When faster and stand-alone hardware is available, this prototype should be modified to reflect the timing constraints specified by the user.

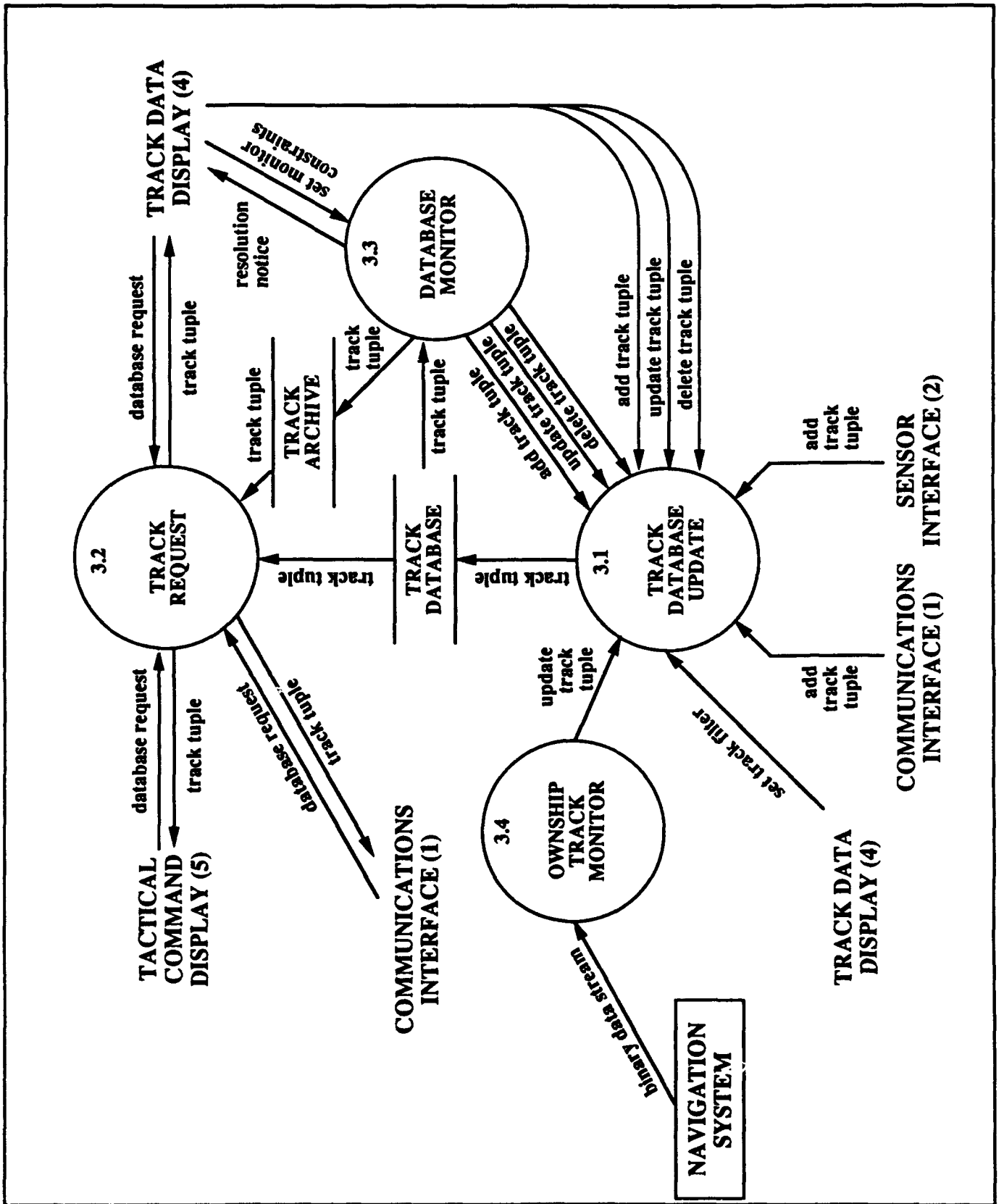
APPENDIX A
DATA FLOW DIAGRAMS



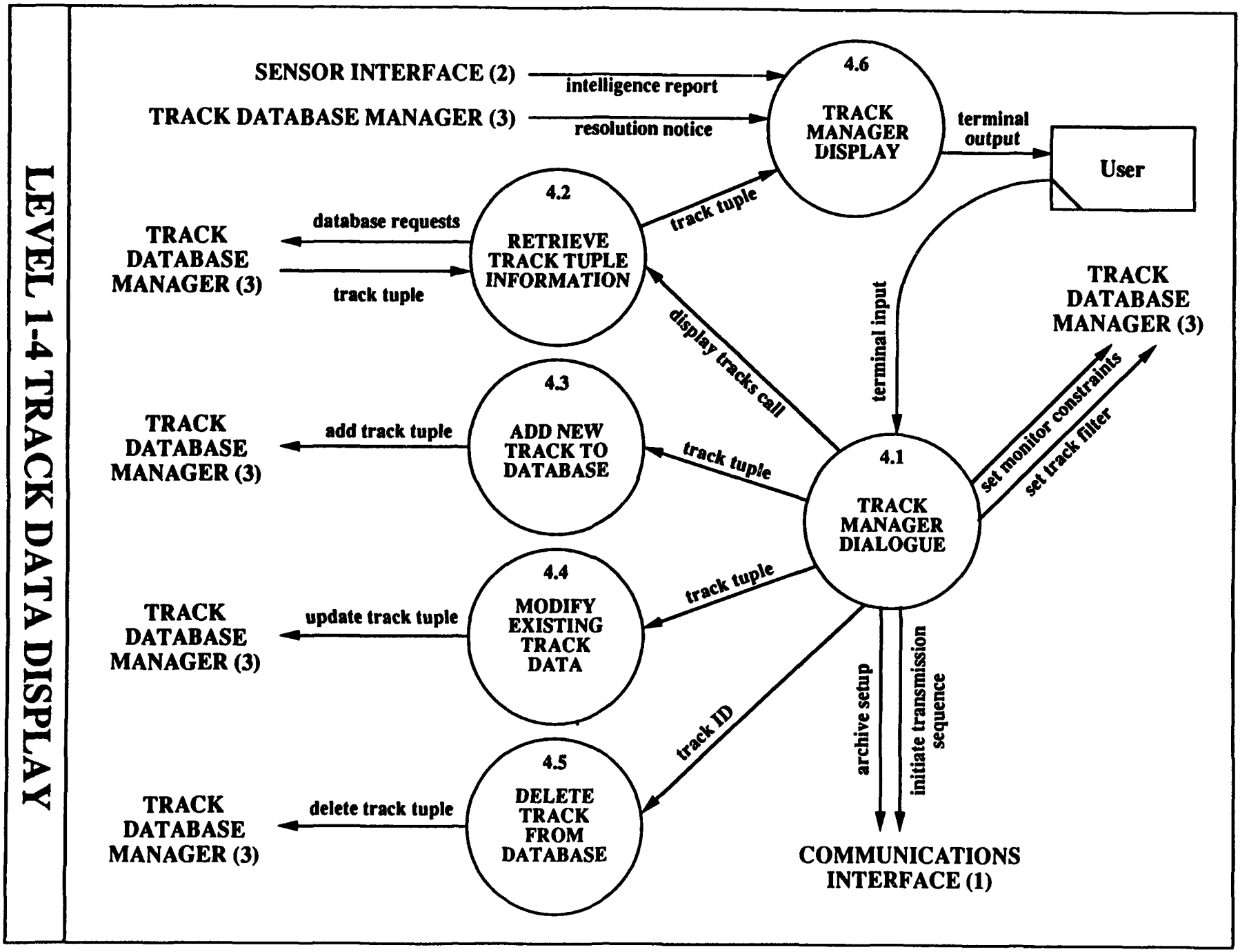

LEVEL 1-1 COMMS INTERFACE
TRACK DATA DISPLAY (4)
initiate transmission sequence
1.6
PERIODIC TRANSMISSION GENERATOR
periodic transmission command
1.4
COMMUNICATIONS NETWORK MONITOR & CONTROL
text file
translation command
1.5
FORMAT TRANSLATOR
track tuple
database request
TRACK DATABASE MANAGER (3)
transmission command
reception notification
transmit command
network setup
emissions control command
TACTICAL COMMAND DISPLAY (5)
1.1
COMMS MESSAGE CONTROL
message
communications message
communications message
COMMUNICATIONS LINKS
message
electronic mail
1.2
MESSAGE LIBRARIAN
filename
1.3
TRACK EXTRACTOR
archive setup
TRACK DATA DISPLAY (4)
text file
text file
text file
MESSAGE ARCHIVES
add track tuple
TRACK DATABASE MANAGER (3)

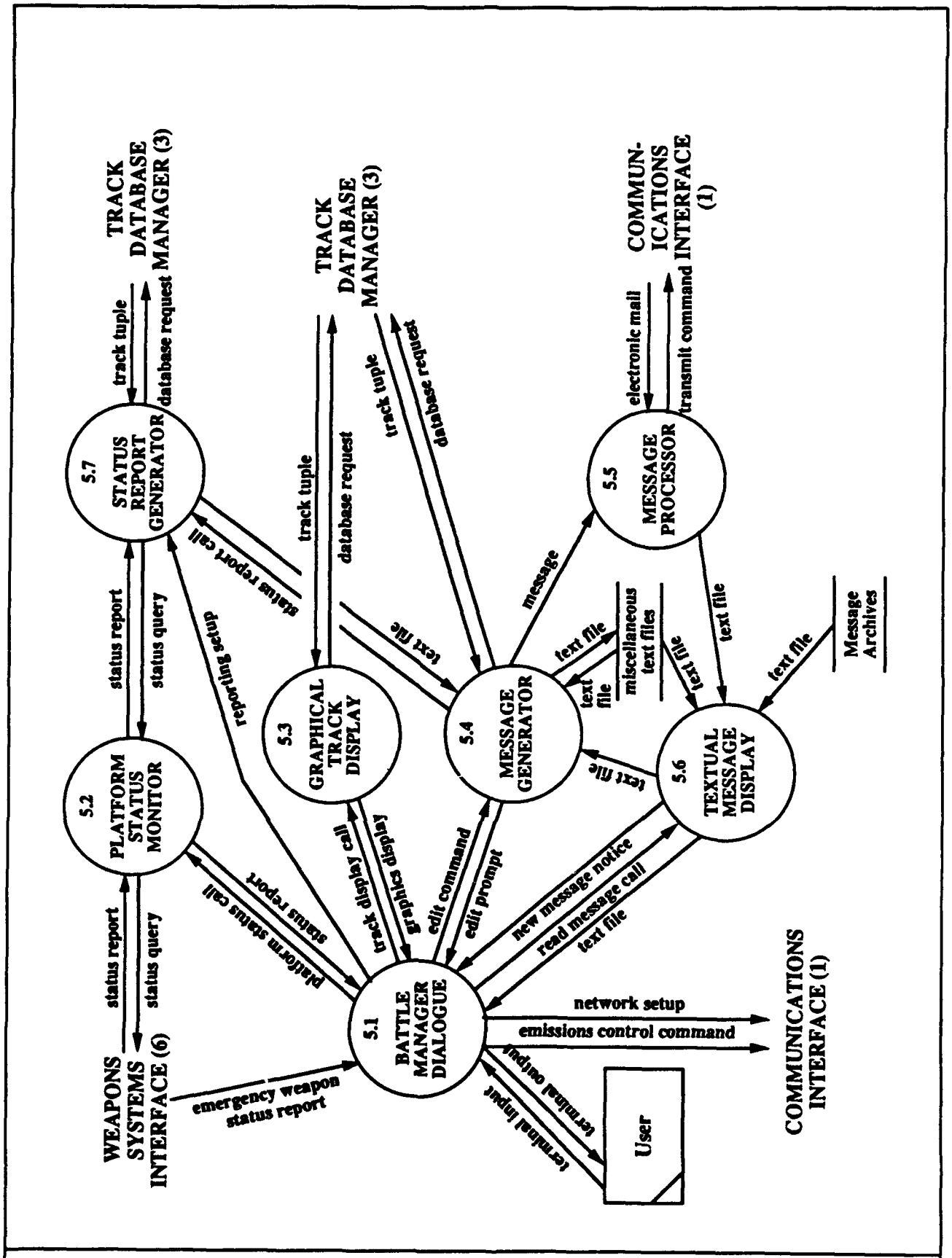


LEVEL 1-2 SENSOR INTERFACE

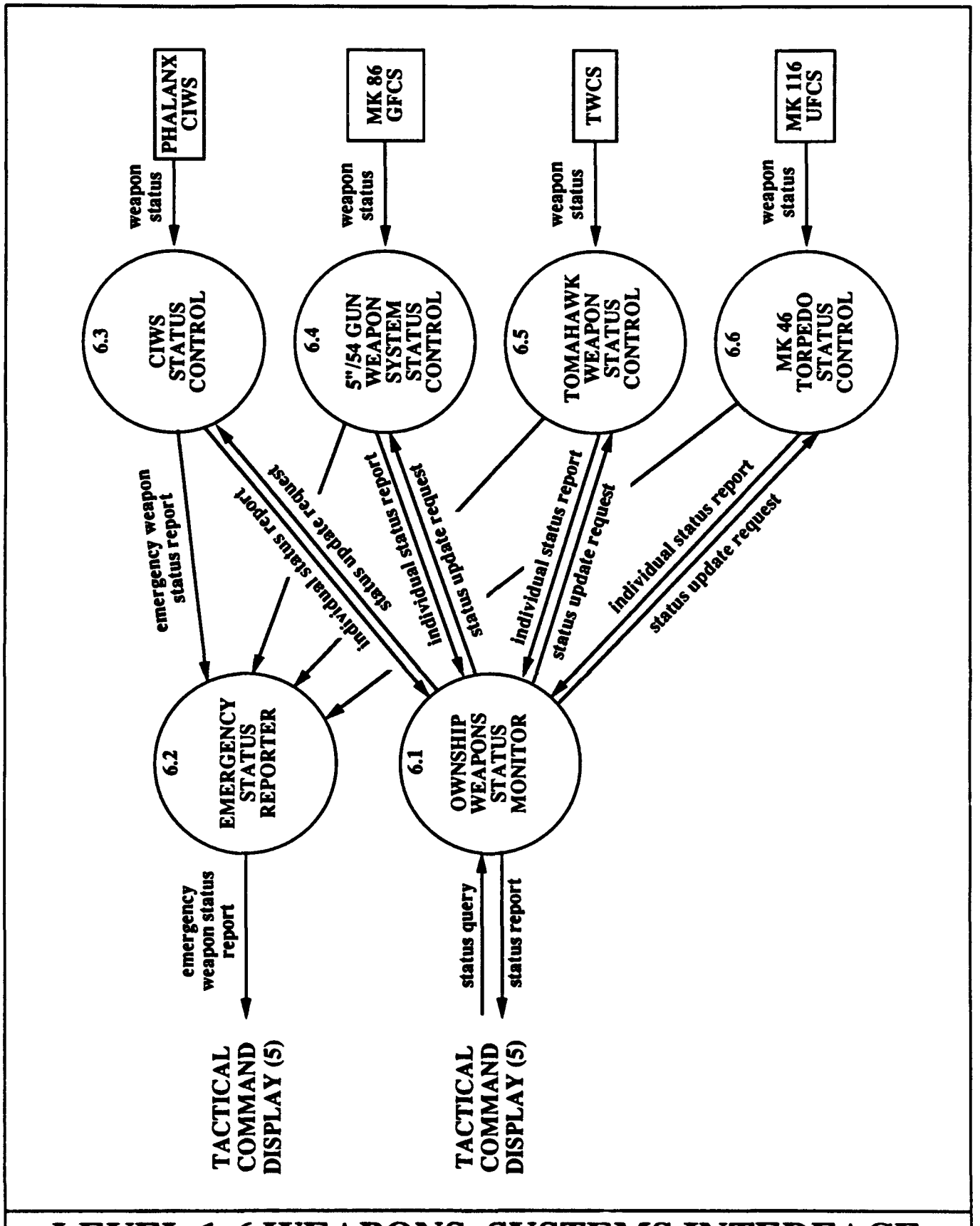


LEVEL 1-3 TRACK DATABASE MANAGER


LEVEL 1-4 TRACK DATA DISPLAY
SENSOR INTERFACE (2)
intelligence report
TRACK DATABASE MANAGER (3)
resolution notice
4.6
TRACK MANAGER DISPLAY
terminal output
User
4.2
RETRIEVE TRACK TUPLE INFORMATION
database requests
track tuple
TRACK DATABASE MANAGER (3)
track tuple
display tracks call
terminal input
TRACK DATABASE MANAGER (3)
4.3
ADD NEW TRACK TO DATABASE
add track tuple
TRACK DATABASE MANAGER (3)
track tuple
4.1
TRACK MANAGER DIALOGUE
set monitor constraints
set track filter
4.4
MODIFY EXISTING TRACK DATA
track tuple
update track tuple
TRACK DATABASE MANAGER (3)
archive setup
initiate transmission sequence
track ID
4.5
DELETE TRACK FROM DATABASE
delete track tuple
TRACK DATABASE MANAGER (3)
COMMUNICATIONS INTERFACE (1)



LEVEL 1-5 TACTICAL COMMAND DISPLAY



LEVEL 1-6 WEAPONS SYSTEMS INTERFACE

APPENDIX B

PSDL DESCRIPTION OF PROTOTYPE

PSDL.TXT

In Chapter IV, we have provided the multi-level PSDL description of the C^3I prototype with the graphical representations. But, the current version of the translator and the static scheduler can handle only flat level PSDL files. This Appendix provides the flat level PSDL description of the C^3I prototype. The graph portion of the PSDL code below is the internal representation of the flat level PSDL graph which is provided at the end of this appendix (Figure B-1).

```
OPERATOR C3I_SYSTEM
 SPECIFICATION
 END
 IMPLEMENTATION
     GRAPH
     VERTEX COMMS_LINKS : 1200 MS
     VERTEX PARSE_INPUT_FILE : 500 MS
     VERTEX DECIDE_FOR_ARCHIVING : 500 MS
     VERTEX EXTRACT_TRACKS : 500 MS
     VERTEX FILTER_COMMS_TRACKS : 500 MS
     VERTEX ADD_COMMS_TRACK : 500 MS
     VERTEX SENSORS : 800 MS
     VERTEX NAVIGATION_SYSTEM : 800 MS
     VERTEX ANALYZE_SENSOR_DATA : 500 MS
     VERTEX PREPARE_SENSOR_TRACK : 500 MS
     VERTEX FILTER_SENSOR_TRACKS : 500 MS
     VERTEX ADD_SENSOR_TRACK : 500 MS
     VERTEX MONITOR_OWNSHIP_POSITION : 500 MS
     VERTEX DISPLAY_TRACKS
     VERTEX GET_USER_INPUTS
     VERTEX MANAGE_USER_INTERFACE
     VERTEX WEAPONS_SYSTEMS : 500 MS
     VERTEX WEAPONS_INTERFACE : 500 MS
     VERTEX STATUS_SCREEN
     VERTEX EMERGENCY_STATUS_SCREEN
```

```
    VERTEX MESSAGE_EDITOR
    VERTEX MAKE_ROUTING : 500 MS
    VERTEX FORWARD_FOR_TRANSMISSION : 500 MS
    VERTEX CONVERT_TO_TEXT_FILE : 800 MS
    VERTEX MESSAGE_ARRIVAL_PANEL
    VERTEX PREPARE_PERIODIC_REPORT : 800 MS
    EDGE INPUT_LINK_MESSAGE COMMS_LINKS -> PARSE_INPUT_FILE
    EDGE INPUT_TEXT_RECORD PARSE_INPUT_FILE -> DECIDE_FOR_ARCHIVING
    EDGE TDD_ARCHIVE_SETUP GET_USER_INPUTS -> DECIDE_FOR_ARCHIVING
    EDGE COMMS_TEXT_FILE DECIDE_FOR_ARCHIVING -> EXTRACT_TRACKS
    EDGE COMMS_EMAIL DECIDE_FOR_ARCHIVING -> MESSAGE_ARRIVAL_PANEL
    EDGE COMMS_ADD_TRACK EXTRACT_TRACKS -> FILTER_COMMS_TRACKS
    EDGE TDD_FILTER GET_USER_INPUTS -> FILTER_COMMS_TRACKS
    EDGE FILTERED_COMMS_TRACK FILTER_COMMS_TRACKS -> ADD_COMMS_TRACK
    EDGE TDD_FILTER GET_USER_INPUTS -> ADD_COMMS_TRACK
    EDGE OUT_TRACKS ADD_COMMS_TRACK -> DISPLAY_TRACKS
    EDGE SENSOR_DATA SENSORS -> ANALYZE_SENSOR_DATA
    EDGE SENSOR_CONTACT_DATA ANALYZE_SENSOR_DATA ->
                                              PREPARE_SENSOR_TRACK
    EDGE POSITION_DATA NAVIGATION_SYSTEM -> PREPARE_SENSOR_TRACK
    EDGE SENSOR_ADD_TRACK PREPARE_SENSOR_TRACK -> FILTER_SENSOR_TRACKS
    EDGE TDD_FILTER GET_USER_INPUTS -> FILTER_SENSOR_TRACKS
    EDGE FILTERED_SENSOR_TRACK FILTER_SENSOR_TRACKS ->
                                                 ADD_SENSOR_TRACK
    EDGE TDD_FILTER GET_USER_INPUTS -> ADD_SENSOR_TRACK
    EDGE OUT_TRACKS ADD_SENSOR_TRACK -> DISPLAY_TRACKS
    EDGE POSITION_DATA NAVIGATION_SYSTEM ->
                                           MONITOR_OWNSHIP_POSITION
    EDGE TD_TRACK_REQUEST GET_USER_INPUTS -> DISPLAY_TRACKS
    EDGE OUT_TRACKS MONITOR_OWNSHIP_POSITION -> DISPLAY_TRACKS
    EDGE WEAPON_STATUS_DATA WEAPONS_SYSTEMS -> WEAPONS_INTERFACE
    EDGE WEAPONS_STATREP WEAPONS_INTERFACE -> STATUS_SCREEN
    EDGE TCD_STATUS_QUERY GET_USER_INPUTS -> STATUS_SCREEN
    EDGE WEAPONS_EMREP WEAPONS_INTERFACE -> EMERGENCY_STATUS_SCREEN
    EDGE EDITOR_SELECTED GET_USER_INPUTS -> MESSAGE_EDITOR
    EDGE TCD_TRANSMIT_COMMAND MESSAGE_EDITOR -> MAKE_ROUTING
    EDGE TCD_NETWORK_SETUP GET_USER_INPUTS -> MAKE_ROUTING
    EDGE TRANSMISSION_MESSAGE MAKE_ROUTING -> FORWARD_FOR_TRANSMISSION
    EDGE TCD_EMISSION_CONTROL GET_USER_INPUTS ->
                                           FORWARD_FOR_TRANSMISSION
    EDGE OUTPUT_MESSAGES FORWARD_FOR_TRANSMISSION ->
                                             CONVERT_TO_TEXT_FILE
    EDGE INITIATE_TRANS GET_USER_INPUTS -> PREPARE_PERIODIC_REPORT
    EDGE TERMINATE_TRANS GET_USER_INPUTS -> PREPARE_PERIODIC_REPORT
    EDGE TCD_TRANSMIT_COMMAND PREPARE_PERIODIC_REPORT -> MAKE_ROUTING

DATA STREAM
  INPUT_LINK_MESSAGE      : FILENAME,
  INPUT_TEXT_RECORD       : TEXT_RECORD,
  TDD_ARCHIVE_SETUP       : ARCHIVE_SETUP,
```

```
  COMMS_TEXT_FILE          : TEXT_RECORD,
  COMMS_ADD_TRACK          : ADD_TRACK_TUPLE,
  TDD_FILTER               : SET_TRACK_FILTER,
  FILTERED_COMMS_TRACK     : ADD_TRACK_TUPLE,
  OUT_TRACKS               : TRACK_TUPLE,
  SENSOR_DATA              : SENSOR_RECORD,
  SENSOR_CONTACT_DATA      : LOCAL_TRACK_INFO,
  POSITION_DATA            : OWNSHIP_NAVIGATION_INFO,
  SENSOR_ADD_TRACK         : ADD_TRACK_TUPLE,
  FILTERED_SENSOR_TRACK    : ADD_TRACK_TUPLE,
  TD_TRACK_REQUEST         : DATABASE_REQUEST,
  WEAPON_STATUS_DATA       : WEAPON_STATUS,
  WEAPONS_STATREP          : WEAPON_STATUS_REPORT,
  WEAPONS_EMREP            : WEAPON_STATUS_REPORT,
  TCD_STATUS_QUERY         : BOOLEAN,
  TCD_TRANSMIT_COMMAND     : TRANSMIT_COMMAND,
  TCD_NETWORK_SETUP        : NETWORK_SETUP,
  TRANSMISSION_MESSAGE     : TRANSMISSION_COMMAND,
  TCD_EMISSION_CONTROL     : EMISSIONS_CONTROL_COMMAND,
  OUTPUT_MESSAGES          : MESSAGE_LIST,
  EDITOR_SELECTED          : BOOLEAN,
  COMMS_EMAIL              : FILENAME,
  INITIATE_TRANS           : INITIATE_TRANSMISSION_SEQUENCE,
  TERMINATE_TRANS          : BOOLEAN

CONTROL CONSTRAINTS
  OPERATOR COMMS_LINKS
    PERIOD 50 sec
  OPERATOR PARSE_INPUT_FILE
    TRIGGERED BY SOME INPUT_LINK_MESSAGE
  OPERATOR DECIDE_FOR_ARCHIVING
    TRIGGERED BY SOME INPUT_TEXT_RECORD
    OUTPUT COMMS_TEXT_FILE IF COMMS_TEXT_FILE.ARCHIVE
    OUTPUT COMMS_EMAIL IF NOT COMMS_TEXT_FILE.IS_TRACK
  OPERATOR EXTRACT_TRACKS
    TRIGGERED IF COMMS_TEXT_FILE.IS_TRACK
  OPERATOR FILTER_COMMS_TRACKS
    TRIGGERED BY SOME COMMS_ADD_TRACK
  OPERATOR ADD_COMMS_TRACK
    TRIGGERED BY SOME FILTERED_COMMS_TRACK
  OPERATOR SENSORS
    PERIOD 50 sec
  OPERATOR NAVIGATION_SYSTEM
    PERIOD 50 sec
```

```
    OPERATOR ANALYZE_SENSOR_DATA
      TRIGGERED BY SOME SENSOR_DATA
    OPERATOR PREPARE_SENSOR_TRACK
      TRIGGERED BY SOME SENSOR_CONTACT_DATA, POSITION_DATA
    OPERATOR FILTER_SENSOR_TRACKS
      TRIGGERED BY SOME SENSOR_ADD_TRACK
    OPERATOR ADD_SENSOR_TRACK
      TRIGGERED BY SOME FILTERED_SENSOR_TRACK
    OPERATOR MONITOR_OWNSHIP_POSITION
      TRIGGERED BY SOME POSITION_DATA
    OPERATOR DISPLAY_TRACKS
      TRIGGERED BY SOME OUT_TRACKS
    OPERATOR GET_USER_INPUTS
      OPERATOR MANAGE_USER_INTERFACE
    OPERATOR WEAPONS_SYSTEMS
      PERIOD 50 sec
    OPERATOR WEAPONS_INTERFACE
      TRIGGERED BY SOME WEAPON_STATUS_DATA
      OUTPUT WEAPONS_EMREP IF WEAPON_STATUS_DATA.STATUS = DAMAGED OR
                           WEAPON_STATUS_DATA.STATUS = SERVICE_REQUIRED OR
                           WEAPON_STATUS_DATA.STATUS = OUT_OF_AMMUNITION
    OPERATOR STATUS_SCREEN
      TRIGGERED IF TCD_STATUS_QUERY
    OPERATOR EMERGENCY_STATUS_SCREEN
      TRIGGERED BY SOME WEAPONS_EMREP
    OPERATOR MESSAGE_EDITOR
      TRIGGERED IF EDITOR_SELECTED
    OPERATOR MAKE_ROUTING
      TRIGGERED BY SOME TCD_TRANSMIT_COMMAND
    OPERATOR FORWARD_FOR_TRANSMISSION
      TRIGGERED BY SOME TRANSMISSION_MESSAGE
      OUTPUT OUTPUT_MESSAGES IF DUMMY
    OPERATOR CONVERT_TO_TEXT_FILE
      TRIGGERED BY SOME OUTPUT_MESSAGES
    OPERATOR MESSAGE_ARRIVAL_PANEL
      TRIGGERED BY SOME COMMS_EMAIL
    OPERATOR PREPARE_PERIODIC_REPORT
      TRIGGERED IF NOT TERMINATE_TRANS
      PERIOD 50 sec
  END

OPERATOR MANAGE_USER_INTERFACE
  SPECIFICATION
  END
```

```
  IMPLEMENTATION ADA MANAGE_USER_INTERFACE
  END

OPERATOR COMMS_LINKS
  SPECIFICATION
    OUTPUT INPUT_LINK_MESSAGE : FILENAME
    MAXIMUM EXECUTION TIME 1200 ms
  END
  IMPLEMENTATION ADA COMMS_LINKS
  END

OPERATOR PARSE_INPUT_FILE
  SPECIFICATION
    INPUT  INPUT_LINK_MESSAGE : FILENAME
    OUTPUT INPUT_TEXT_RECORD  : TEXT_RECORD
    MAXIMUM EXECUTION TIME 500 MS
  END
  IMPLEMENTATION ADA PARSE_INPUT_FILE
  END

OPERATOR DECIDE_FOR_ARCHIVING
  SPECIFICATION
    INPUT  INPUT_TEXT_RECORD    : TEXT_RECORD,
           TDD_ARCHIVE_SETUP    : ARCHIVE_SETUP
    OUTPUT COMMS_TEXT_FILE      : TEXT_RECORD,
           COMMS_EMAIL          : FILENAME
    MAXIMUM EXECUTION TIME 500 ms
  END
  IMPLEMENTATION ADA DECIDE_FOR_ARCHIVING
  END

OPERATOR EXTRACT_TRACKS
  SPECIFICATION
    INPUT  COMMS_TEXT_FILE : TEXT_RECORD
    OUTPUT COMMS_ADD_TRACK : ADD_TRACK_TUPLE
    MAXIMUM EXECUTION TIME 500 ms
  END
  IMPLEMENTATION ADA EXTRACT_TRACKS
  END

OPERATOR FILTER_COMMS_TRACKS
  SPECIFICATION
    INPUT  COMMS_ADD_TRACK      : ADD_TRACK_TUPLE,
           TDD_FILTER           : SET_TRACK_FILTER
```

```
    OUTPUT FILTERED_COMMS_TRACK : ADD_TRACK_TUPLE
    MAXIMUM EXECUTION TIME 500 ms
  END
  IMPLEMENTATION ADA FILTER_COMMS_TRACKS
  END

OPERATOR ADD_COMMS_TRACK
  SPECIFICATION
    INPUT  FILTERED_COMMS_TRACK : ADD_TRACK_TUPLE,
    TDD_FILTER           : SET_TRACK_FILTER
    OUTPUT OUT_TRACKS           : TRACK_TUPLE
    MAXIMUM EXECUTION TIME 500 ms
  END
  IMPLEMENTATION ADA ADD_COMMS_TRACK
  END

OPERATOR SENSORS
  SPECIFICATION
    OUTPUT SENSOR_DATA : SENSOR_RECORD
    MAXIMUM EXECUTION TIME 800 MS
  END
  IMPLEMENTATION ADA SENSORS
  END

OPERATOR ANALYZE_SENSOR_DATA
  SPECIFICATION
    INPUT  SENSOR_DATA         : SENSOR_RECORD
    OUTPUT SENSOR_CONTACT_DATA : LOCAL_TRACK_INFO
    MAXIMUM EXECUTION TIME 500 MS
  END
  IMPLEMENTATION ADA ANALYZE_SENSOR_DATA
  END


OPERATOR PREPARE_SENSOR_TRACK
  SPECIFICATION
    INPUT  SENSOR_CONTACT_DATA : LOCAL_TRACK_INFO,
           POSITION_DATA       : OWNSHIP_NAVIGATION_INFO
    OUTPUT SENSOR_ADD_TRACK    : ADD_TRACK_TUPLE
    MAXIMUM EXECUTION TIME 500 ms
  END
  IMPLEMENTATION ADA PREPARE_SENSOR_TRACK
  END
```

```
OPERATOR FILTER_SENSOR_TRACKS
  SPECIFICATION
    INPUT  SENSOR_ADD_TRACK      : ADD_TRACK_TUPLE,
           TDD_FILTER            : SET_TRACK_FILTER
    OUTPUT FILTERED_SENSOR_TRACK : ADD_TRACK_TUPLE
    STATES APPROVED : BOOLEAN INITIALLY FALSE
    MAXIMUM EXECUTION TIME 500 MS
  END
  IMPLEMENTATION ADA FILTER_SENSOR_TRACKS
  END

OPERATOR ADD_SENSOR_TRACK
  SPECIFICATION
    INPUT  FILTERED_SENSOR_TRACK : ADD_TRACK_TUPLE,
           TDD_FILTER            : SET_TRACK_FILTER
    OUTPUT OUT_TRACKS            : TRACK_TUPLE
    MAXIMUM EXECUTION TIME 500 ms
  END
  IMPLEMENTATION ADA ADD_SENSOR_TRACK
  END

OPERATOR NAVIGATION_SYSTEM
  SPECIFICATION
    OUTPUT POSITION_DATA     : OWNSHIP_NAVIGATION_INFO
    MAXIMUM EXECUTION TIME 800 MS
  END
  IMPLEMENTATION ADA NAVIGATION_SYSTEM
  END

OPERATOR MONITOR_OWNSHIP_POSITION
  SPECIFICATION
    INPUT  POSITION_DATA     : OWNSHIP_NAVIGATION_INFO
    OUTPUT OUT_TRACKS        : TRACK_TUPLE
    MAXIMUM EXECUTION TIME 500 MS
  END
  IMPLEMENTATION ADA MONITOR_OWNSHIP_POSITION
  END

OPERATOR DISPLAY_TRACKS
  SPECIFICATION
    INPUT   OUT_TRACKS       : TRACK_TUPLE,
            TD_TRACK_REQUEST : DATABASE_REQUEST
  END
  IMPLEMENTATION ADA DISPLAY_TRACKS
```

```
  END


OPERATOR GET_USER_INPUTS
  SPECIFICATION
    OUTPUT TDD_ARCHIVE_SETUP     : ARCHIVE_SETUP,
           TDD_FILTER            : SET_TRACK_FILTER,
           TD_TRACK_REQUEST      : DATABASE_REQUEST,
           TCD_STATUS_QUERY      : BOOLEAN,
           TCD_NETWORK_SETUP     : NETWORK_SETUP,
           TCD_EMISSION_CONTROL  : EMISSIONS_CONTROL_COMMAND,
           EDITOR_SELECTED       : BOOLEAN,
           INITIATE_TRANS        : INITIATE_TRANSMISSION_SEQUENCE,
           TERMINATE_TRANS       : BOOLEAN
  END
  IMPLEMENTATION ADA GET_USER_INPUTS
  END

CPERATOR WEAPONS_INTERFACE
  SPECIFICATION
    INPUT  WEAPON_STATUS_DATA   : WEAPON_STATUS
    OUTPUT WEAPONS_EMREP        :WEAPON_STATUS_REPORT,
           WEAPONS_STATREP      :WEAPON_STATUS_REPORT
    STATES CIWS_STATUS,
           GUN_STATUS,
           TWS_STATUS,
           MK48_STATUS : WEAPON_STATUS_TYPE
    INITIALLY READY, READY, READY, READY
    MAXIMUM EXECUTION TIME 500 ms
  END
  IMPLEMENTATION ADA WEAPONS_INTERFACE
  END

OPERATOR WEAPONS_SYSTEMS
  SPECIFICATION
    OUTPUT WEAPON_STATUS_DATA : WEAPON_STATUS
    MAXIMUM EXECUTION TIME 500 ms
  END
  IMPLEMENTATION ADA WEAPONS_SYSTEMS
  END

OPERATOR STATUS_SCREEN
  SPECIFICATION
    INPUT  WEAPONS_STATREP    : WEAPON_STATUS_REPORT,
```

```
            TCD_STATUS_QUERY   : BOOLEAN
    END
    IMPLEMENTATION ADA STATUS_SCREEN
    END

OPERATOR EMERGENCY_STATUS_SCREEN
    SPECIFICATION
      INPUT WEAPONS_EMREP      : WEAPON_STATUS_REPORT
    END
    IMPLEMENTATION ADA EMERGENCY_STATUS_SCREEN
    END

OPERATOR MESSAGE_EDITOR
    SPECIFICATION
      INPUT  EDITOR_SELECTED       : BOOLEAN
      OUTPUT TCD_TRANSMIT_COMMAND  : TRANSMIT_COMMAND
    END
    IMPLEMENTATION ADA MESSAGE_EDITOR
    END

OPERATOR MAKE_ROUTING
    SPECIFICATION
      INPUT  TCD_TRANSMIT_COMMAND : TRANSMIT_COMMAND,
             TCD_NETWORK_SETUP    : NETWORK_SETUP
      OUTPUT TRANSMISSION_MESSAGE : TRANSMISSION_COMMAND
      MAXIMUM EXECUTION TIME 500 MS
    END
    IMPLEMENTATION ADA MAKE_ROUTING
    END

OPERATOR FORWARD_FOR_TRANSMISSION
    SPECIFICATION
      INPUT  TRANSMISSION_MESSAGE : TRANSMISSION_COMMAND,
             TCD_EMISSION_CONTROL : EMISSIONS_CONTROL_COMMAND
      OUTPUT OUTPUT_MESSAGES      : MESSAGE_LIST
      STATES WAITING_MESSAGES     : MESSAGE_LIST INITIALLY NULL
      MAXIMUM EXECUTION TIME 500 MS
    END
    IMPLEMENTATION ADA FORWARD_FOR_TRANSMISSION
    END

OPERATOR CONVERT_TO_TEXT_FILE
    SPECIFICATION
      INPUT  OUTPUT_MESSAGES : MESSAGE_LIST
```

```
    MAXIMUM EXECUTION TIME 800 MS
  END
  IMPLEMENTATION ADA CONVERT_TO_TEXT_FILE
  END

OPERATOR MESSAGE_ARRIVAL_PANEL
  SPECIFICATION
    INPUT  COMMS_EMAIL : FILENAME
  END
  IMPLEMENTATION ADA MESSAGE_ARRIVAL_PANEL
  END

OPERATOR PREPARE_PERIODIC_REPORT
  SPECIFICATION
    INPUT  INITIATE_TRANS       : INITIATE_TRANSMISSION_SEQUENCE,
           TERMINATE_TRANS      : BOOLEAN
    OUTPUT TCD_TRANSMIT_COMMAND : TRANSMIT_COMMAND
    MAXIMUM EXECUTION TIME 800 MS
  END
  IMPLEMENTATION ADA PREPARE_PERIODIC_REPORT
  END
```

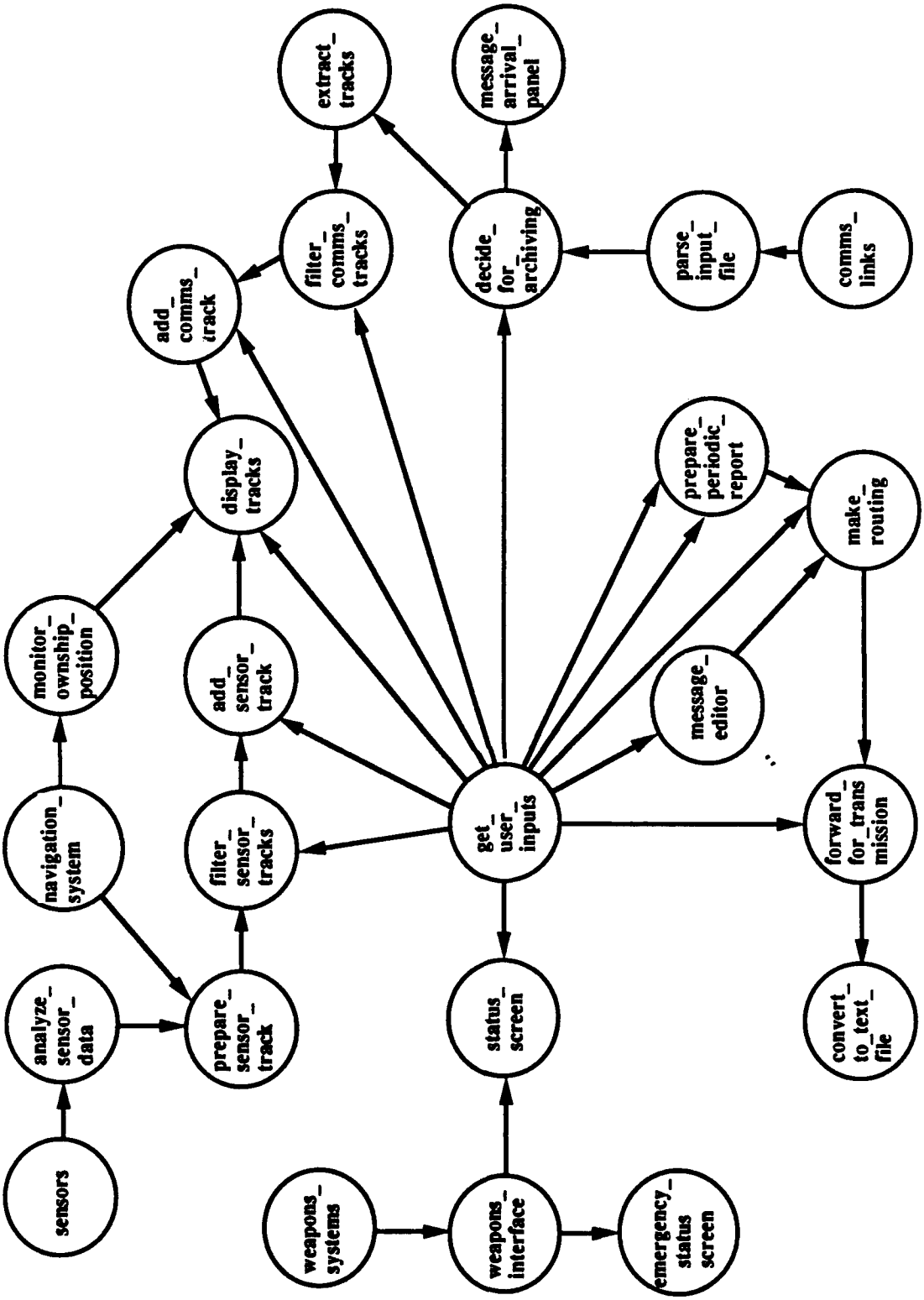


Figure B-1. Flat Level PSDL Graph

APPENDIX C

DRIVER PACKAGE

TL.A

This appendix contains the Ada package tl.a. The translator gets the psdl.txt file provided in Appendix B as input, and creates this driver package.

Package tl.a includes a buffer instantiation for each data stream, and a driver procedure for each atomic operator.

```
package TL is
  procedure PREPARE_PERIODIC_REPORT_DRIVER;
  procedure MESSAGE_ARRIVAL_PANEL_DRIVER;
  procedure CONVERT_TO_TEXT_FILE_DRIVER;
  procedure FORWARD_FOR_TRANSMISSION_DRIVER;
  procedure MAKE_ROUTING_DRIVER;
  procedure MESSAGE_EDITOR_DRIVER;
  procedure EMERGENCY_STATUS_SCREEN_DRIVER;
  procedure STATUS_SCREEN_DRIVER;
  procedure WEAPONS_INTERFACE_DRIVER;
  procedure WEAPONS_SYSTEMS_DRIVER;
  procedure MANAGE_USER_INTERFACE_DRIVER;
  procedure GET_USER_INPUTS_DRIVER;
  procedure DISPLAY_TRACKS_DRIVER;
  procedure MONITOR_OWNSHIP_POSITION_DRIVER;
  procedure ADD_SENSOR_TRACK_DRIVER;
  procedure FILTER_SENSOR_TRACKS_DRIVER;
  procedure PREPARE_SENSOR_TRACK_DRIVER;
  procedure ANALYZE_SENSOR_DATA_DRIVER;
  procedure NAVIGATION_SYSTEM_DRIVER;
  procedure SENSORS_DRIVER;
  procedure ADD_COMMS_TRACK_DRIVER;
  procedure FILTER_COMMS_TRACKS_DRIVER;
  procedure EXTRACT_TRACKS_DRIVER;
```

```
    procedure DECIDE_FOR_ARCHIVING_DRIVER;
    procedure PARSE_INPUT_FILE_DRIVER;
    procedure COMMS_LINKS_DRIVER;
  end TL;



    with SB; use SB;
    with PSDL_STREAMS; use PSDL_STREAMS;
    with DS_Debug_PKG; use DS_Debug_PKG;
    with PSDL_TIMER_PKG;
  package body TL is
    type PSDL_EXCEPTION is (UNDECLARED_ADA_EXCEPTION);
    package C3I_SYSTEM_SPEC is
      package DS_TERMINATE_TRANS is new SAMPLED_BUFFER(BOOLEAN);
      package DS_INITIATE_TRANS is new
                          SAMPLED_BUFFER(INITIATE_TRANSMISSION_SEQUENCE);
      package DS_COMMS_EMAIL is new SAMPLED_BUFFER(FILENAME);
      package DS_EDITOR_SELECTED is new SAMPLED_BUFFER(BOOLEAN);
      package DS_OUTPUT_MESSAGES is new SAMPLED_BUFFER(MESSAGE_LIST);
      package DS_TCD_EMISSION_CONTROL is new
                                SAMPLED_BUFFER(EMISSIONS_CONTROL_COMMAND);
      package DS_TRANSMISSION_MESSAGE is new
                                     SAMPLED_BUFFER(TRANSMISSION_COMMAND);
      package DS_TCD_NETWORK_SETUP is new SAMPLED_BUFFER(NETWORK_SETUP);
      package DS_TCD_TRANSMIT_COMMAND is new
                                        SAMPLED_BUFFER(TRANSMIT_COMMAND);
      package DS_TCD_STATUS_QUERY is new SAMPLED_BUFFER(BOOLEAN);
      package DS_WEAPONS_EMREP is new
                                     SAMPLED_BUFFER(WEAPON_STATUS_REPORT);
      package DS_WEAPONS_STATREP is new
                                     SAMPLED_BUFFER(WEAPON_STATUS_REPORT);
      package DS_WEAPON_STATUS_DATA is new SAMPLED_BUFFER(WEAPON_STATUS);
      package DS_TD_TRACK_REQUEST is new SAMPLED_BUFFER(DATABASE_REQUEST);
      package DS_FILTERED_SENSOR_TRACK is new
                                       SAMPLED_BUFFER(ADD_TRACK_TUPLE);
      package DS_SENSOR_ADD_TRACK is new SAMPLED_BUFFER(ADD_TRACK_TUPLE);
      package DS_POSITION_DATA is new
                                 SAMPLED_BUFFER(OWNSHIP_NAVIGATION_INFO);
      package DS_SENSOR_CONTACT_DATA is new
```

```
                                        SAMPLED_BUFFER(LOCAL_TRACK_INFO);
  package DS_SENSOR_DATA is new SAMPLED_BUFFER(SENSOR_RECORD);
  package DS_OUT_TRACKS is new SAMPLED_BUFFER(TRACK_TUPLE);
  package DS_FILTERED_COMMS_TRACK is new
                                         SAMPLED_BUFFER(ADD_TRACK_TUPLE);
  package DS_TDD_FILTER is new SAMPLED_BUFFER(SET_TRACK_FILTER);
  package DS_COMMS_ADD_TRACK is new SAMPLED_BUFFER(ADD_TRACK_TUPLE);
  package DS_COMMS_TEXT_FILE is new SAMPLED_BUFFER(TEXT_RECORD);
  package DS_TDD_ARCHIVE_SETUP is new SAMPLED_BUFFER(ARCHIVE_SETUP);
  package DS_INPUT_TEXT_RECORD is new SAMPLED_BUFFER(TEXT_RECORD);
  package DS_INPUT_LINK_MESSAGE is new SAMPLED_BUFFER(FILENAME);
  package JUNK is new SAMPLED_BUFFER( ARCHIVE_SETUP);
end C3I_SYSTEM_SPEC;

procedure PREPARE_PERIODIC_REPORT_DRIVER is
  LV_INITIATE_TRANS: INITIATE_TRANSMISSION_SEQUENCE;
  LV_TERMINATE_TRANS: BOOLEAN;
  LV_TCD_TRANSMIT_COMMAND: TRANSMIT_COMMAND;
  EXCEPTION_HAS_OCCURRED: boolean := false;
  EXCEPTION_ID: PSDL_EXCEPTION;
begin
  if true then
    begin
      C3I_SYSTEM_SPEC.DS_INITIATE_TRANS.BUFFER.READ
         (LV_INITIATE_TRANS);
    exception
      when BUFFER_UNDERFLOW =>
        DS_Debug.Buffer_Underflow
         ("INITIATE_TRANS", "PREPARE_PERIODIC_REPORT");
    end;
    begin
     C3I_SYSTEM_SPEC.DS_TERMINATE_TRANS.BUFFER.READ
         (LV_TERMINATE_TRANS);
    exception
      when BUFFER_UNDERFLOW =>
        DS_Debug.Buffer_Underflow
         ("TERMINATE_TRANS", "PREPARE_PERIODIC_REPORT");
    end;
    if not LV_TERMINATE_TRANS then
       begin
         PREPARE_PERIODIC_REPORT
```

```
    (LV_INITIATE_TRANS, LV_TERMINATE_TRANS, LV_TCD_TRANSMIT_COMMAND);
       exception
         when others =>
           DS_Debug.Undeclared_Exception("PREPARE_PERIODIC_REPORT");
           EXCEPTION_HAS_OCCURRED := true;
           EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
       end;
        begin
          C3I_SYSTEM_SPEC.DS_TCD_TRANSMIT_COMMAND.BUFFER.WRITE
        (LV_TCD_TRANSMIT_COMMAND);
        exception
          when BUFFER_OVERFLOW =>
            DS_Debug.Buffer_Overflow
        ("TCD_TRANSMIT_COMMAND", "PREPARE_PERIODIC_REPORT");
        end;
       if EXCEPTION_HAS_OCCURRED then
          DS_Debug.Unhandled_Exception
     ("PREPARE_PERIODIC_REPORT", PSDL_EXCEPTION'image(EXCEPTION_ID));
       end if;
    end if;
  end if;
end PREPARE_PERIODIC_REPORT_DRIVER;

procedure MESSAGE_ARRIVAL_PANEL_DRIVER is
  LV_COMMS_EMAIL: FILENAME;
  EXCEPTION_HAS_OCCURRED: boolean := false;
  EXCEPTION_ID: PSDL_EXCEPTION;
begin
  if C3I_SYSTEM_SPEC.DS_COMMS_EMAIL.NEW_DATA then
    begin
      C3I_SYSTEM_SPEC.DS_COMMS_EMAIL.BUFFER.READ(LV_COMMS_EMAIL);
    exception
      when BUFFER_UNDERFLOW =>
        DS_Debug.Buffer_Underflow
         ("COMMS_EMAIL", "MESSAGE_ARRIVAL_PANEL");
    end;
    if true then
       begin
         MESSAGE_ARRIVAL_PANEL(LV_COMMS_EMAIL);
       exception
         when others =>
```

```
            DS_Debug.Undeclared_Exception("MESSAGE_ARRIVAL_PANEL");
            EXCEPTION_HAS_OCCURRED := true;
            EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
         end;
         if EXCEPTION_HAS_OCCURRED then
            DS_Debug.Unhandled_Exception
         ("MESSAGE_ARRIVAL_PANEL", PSDL_EXCEPTION'image(EXCEPTION_ID));
         end if;
      end if;
   end if;
end MESSAGE_ARRIVAL_PANEL_DRIVER;

procedure CONVERT_TO_TEXT_FILE_DRIVER is
   LV_OUTPUT_MESSAGES: MESSAGE_LIST;
   EXCEPTION_HAS_OCCURRED: boolean := false;
   EXCEPTION_ID: PSDL_EXCEPTION;
begin
   if C3I_SYSTEM_SPEC.DS_OUTPUT_MESSAGES.NEW_DATA then
      begin
       C3I_SYSTEM_SPEC.DS_OUTPUT_MESSAGES.BUFFER.READ
           (LV_OUTPUT_MESSAGES);
      exception
        when BUFFER_UNDERFLOW =>
          DS_Debug.Buffer_Underflow
           ("OUTPUT_MESSAGES", "CONVERT_TO_TEXT_FILE");
      end;
      if true then
         begin
           CONVERT_TO_TEXT_FILE(LV_OUTPUT_MESSAGES);
         exception
           when others =>
             DS_Debug.Undeclared_Exception("CONVERT_TO_TEXT_FILE");
             EXCEPTION_HAS_OCCURRED := true;
             EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
         end;
         if EXCEPTION_HAS_OCCURRED then
            DS_Debug.Unhandled_Exception
          ("CONVERT_TO_TEXT_FILE", PSDL_EXCEPTION'image(EXCEPTION_ID));
         end if;
      end if;
   end if;
```

```
end CONVERT_TO_TEXT_FILE_DRIVER;

procedure FORWARD_FOR_TRANSMISSION_DRIVER is
  LV_TRANSMISSION_MESSAGE: TRANSMISSION_COMMAND;
  LV_TCD_EMISSION_CONTROL: EMISSIONS_CONTROL_COMMAND;
  LV_OUTPUT_MESSAGES: MESSAGE_LIST;
  EXCEPTION_HAS_OCCURRED: boolean := false;
  EXCEPTION_ID: PSDL_EXCEPTION;
begin
  if C3I_SYSTEM_SPEC.DS_TRANSMISSION_MESSAGE.NEW_DATA then
    begin
      C3I_SYSTEM_SPEC.DS_TRANSMISSION_MESSAGE.BUFFER.READ
         (LV_TRANSMISSION_MESSAGE);
    exception
      when BUFFER_UNDERFLOW =>
        DS_Debug.Buffer_Underflow
         ("TRANSMISSION_MESSAGE", "FORWARD_FOR_TRANSMISSION");
    end;
    begin
      C3I_SYSTEM_SPEC.DS_TCD_EMISSION_CONTROL.BUFFER.READ
         (LV_TCD_EMISSION_CONTROL);
    exception
      when BUFFER_UNDERFLOW =>
        DS_Debug.Buffer_Underflow
         ("TCD_EMISSION_CONTROL", "FORWARD_FOR_TRANSMISSION");
    end;
    if true then
       begin
         FORWARD_FOR_TRANSMISSION(LV_TRANSMISSION_MESSAGE,
        LV_TCD_EMISSION_CONTROL, LV_OUTPUT_MESSAGES);
       exception
         when others =>
           DS_Debug.Undeclared_Exception("FORWARD_FOR_TRANSMISSION");
           EXCEPTION_HAS_OCCURRED := true;
           EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
       end;
       if LV_TCD_EMISSION_CONTROL = UNRESTRICTED
       then
         begin
           C3I_SYSTEM_SPEC.DS_OUTPUT_MESSAGES.BUFFER.WRITE
         (LV_OUTPUT_MESSAGES);
```

```
          exception
            when BUFFER_OVERFLOW =>
              DS_Debug.Buffer_Overflow
          ("OUTPUT_MESSAGES", "FORWARD_FOR_TRANSMISSION");
          end;
        end if;
        if EXCEPTION_HAS_OCCURRED then
          DS_Debug.Unhandled_Exception
    ("FORWARD_FOR_TRANSMISSION", PSDL_EXCEPTION'image(EXCEPTION_ID));
        end if;
      end if;
  end if;
end FORWARD_FOR_TRANSMISSION_DRIVER;

procedure MAKE_ROUTING_DRIVER is
  LV_TCD_TRANSMIT_COMMAND: TRANSMIT_COMMAND;
  LV_TCD_NETWORK_SETUP: NETWORK_SETUP;
  LV_TRANSMISSION_MESSAGE: TRANSMISSION_COMMAND;
  EXCEPTION_HAS_OCCURRED: boolean := false;
  EXCEPTION_ID: PSDL_EXCEPTION;
begin
  if C3I_SYSTEM_SPEC.DS_TCD_TRANSMIT_COMMAND.NEW_DATA then
    begin
      C3I_SYSTEM_SPEC.DS_TCD_TRANSMIT_COMMAND.BUFFER.READ
        (LV_TCD_TRANSMIT_COMMAND);
    exception
      when BUFFER_UNDERFLOW =>
        DS_Debug.Buffer_Underflow
        ("TCD_TRANSMIT_COMMAND", "MAKE_ROUTING");
    end;
    begin
      C3I_SYSTEM_SPEC.DS_TCD_NETWORK_SETUP.BUFFER.READ
        (LV_TCD_NETWORK_SETUP);
    exception
      when BUFFER_UNDERFLOW =>
        DS_Debug.Buffer_Underflow
        ("TCD_NETWORK_SETUP", "MAKE_ROUTING");
    end;
    if true then
      begin
        MAKE_ROUTING(LV_TCD_TRANSMIT_COMMAND, LV_TCD_NETWORK_SETUP,
```

```
                    LV_TRANSMISSION_MESSAGE);
      exception
        when others =>
          DS_Debug.Undeclared_Exception("MAKE_ROUTING");
          EXCEPTION_HAS_OCCURRED := true;
          EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
      end;
       begin
         C3I_SYSTEM_SPEC.DS_TRANSMISSION_MESSAGE.BUFFER.WRITE
        (LV_TRANSMISSION_MESSAGE);
        exception
          when BUFFER_OVERFLOW =>
            DS_Debug.Buffer_Overflow
        ("TRANSMISSION_MESSAGE", "MAKE_ROUTING");
        end;
      if EXCEPTION_HAS_OCCURRED then
         DS_Debug.Unhandled_Exception
        ("MAKE_ROUTING", PSDL_EXCEPTION'image(EXCEPTION_ID));
      end if;
    end if;
  end if;
end MAKE_ROUTING_DRIVER;

procedure MESSAGE_EDITOR_DRIVER is
  LV_EDITOR_SELECTED: BOOLEAN;
  LV_TCD_TRANSMIT_COMMAND: TRANSMIT_COMMAND;
  EXCEPTION_HAS_OCCURRED: boolean := false;
  EXCEPTION_ID: PSDL_EXCEPTION;
begin
  if true then
    begin
      C3I_SYSTEM_SPEC.DS_EDITOR_SELECTED.BUFFER.READ
        (LV_EDITOR_SELECTED);
    exception
      when BUFFER_UNDERFLOW =>
        DS_Debug.Buffer_Underflow
        ("EDITOR_SELECTED", "MESSAGE_EDITOR");
    end;
    if LV_EDITOR_SELECTED then
       begin
        MESSAGE_EDITOR(LV_EDITOR_SELECTED, LV_TCD_TRANSMIT_COMMAND);
```

```
        exception
          when others =>
            DS_Debug.Undeclared_Exception("MESSAGE_EDITOR");
            EXCEPTION_HAS_OCCURRED := true;
            EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
        end;
         begin
           C3I_SYSTEM_SPEC.DS_TCD_TRANSMIT_COMMAND.BUFFER.WRITE
          (LV_TCD_TRANSMIT_COMMAND);
          exception
            when BUFFER_OVERFLOW =>
              DS_Debug.Buffer_Overflow
          ("TCD_TRANSMIT_COMMAND", "MESSAGE_EDITOR");
          end;
        if EXCEPTION_HAS_OCCURRED then
           DS_Debug.Unhandled_Exception
          ("MESSAGE_EDITOR", PSDL_EXCEPTION'image(EXCEPTION_ID));
        end if;
    end if;
  end if;
end MESSAGE_EDITOR_DRIVER;

procedure EMERGENCY_STATUS_SCREEN_DRIVER is
  LV_WEAPONS_EMREP: WEAPON_STATUS_REPORT;
  EXCEPTION_HAS_OCCURRED: boolean := false;
  EXCEPTION_ID: PSDL_EXCEPTION;
begin
  if C3I_SYSTEM_SPEC.DS_WEAPONS_EMREP.NEW_DATA then
    begin
      C3I_SYSTEM_SPEC.DS_WEAPONS_EMREP.BUFFER.READ(LV_WEAPONS_EMREP);
    exception
      when BUFFER_UNDERFLOW =>
        DS_Debug.Buffer_Underflow
         ("WEAPONS_EMREP", "EMERGENCY_STATUS_SCREEN");
    end;
    if true then
       begin
         EMERGENCY_STATUS_SCREEN(LV_WEAPONS_EMREP);
       exception
         when others =>
           DS_Debug.Undeclared_Exception("EMERGENCY_STATUS_SCREEN");
```

```
            EXCEPTION_HAS_OCCURRED := true;
            EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
        end;
        if EXCEPTION_HAS_OCCURRED then
           DS_Debug.Unhandled_Exception
      ("EMERGENCY_STATUS_SCREEN", PSDL_EXCEPTION'image(EXCEPTION_ID));
        end if;
    end if;
  end if;
end EMERGENCY_STATUS_SCREEN_DRIVER;

procedure STATUS_SCREEN_DRIVER is
  LV_WEAPONS_STATREP: WEAPON_STATUS_REPORT;
  LV_TCD_STATUS_QUERY: BOOLEAN;
  EXCEPTION_HAS_OCCURRED: boolean := false;
  EXCEPTION_ID: PSDL_EXCEPTION;
begin
  if true then
    begin
      C3I_SYSTEM_SPEC.DS_WEAPONS_STATREP.BUFFER.READ
         (LV_WEAPONS_STATREP);
    exception
      when BUFFER_UNDERFLOW =>
        DS_Debug.Buffer_Underflow("WEAPONS_STATREP", "STATUS_SCREEN");
    end;
    begin
      C3I_SYSTEM_SPEC.DS_TCD_STATUS_QUERY.BUFFER.READ
         (LV_TCD_STATUS_QUERY);
    exception
      when BUFFER_UNDERFLOW =>
        DS_Debug.Buffer_Underflow
         ("TCD_STATUS_QUERY", "STATUS_SCREEN");
    end;
    if LV_TCD_STATUS_QUERY then
       begin
         STATUS_SCREEN(LV_WEAPONS_STATREP, LV_TCD_STATUS_QUERY);
       exception
         when others =>
           DS_Debug.Undeclared_Exception("STATUS_SCREEN");
           EXCEPTION_HAS_OCCURRED := true;
           EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
```

```
        end;
        if EXCEPTION_HAS_OCCURRED then
           DS_Debug.Unhandled_Exception
          ("STATUS_SCREEN", PSDL_EXCEPTION'image(EXCEPTION_ID));
        end if;
      end if;
    end if;
  end STATUS_SCREEN_DRIVER;
  procedure WEAPONS_INTERFACE_DRIVER is
    LV_WEAPON_STATUS_DATA: WEAPON_STATUS;
    LV_WEAPONS_STATREP: WEAPON_STATUS_REPORT;
    LV_WEAPONS_EMREP: WEAPON_STATUS_REPORT;
    EXCEPTION_HAS_OCCURRED: boolean := false;
    EXCEPTION_ID: PSDL_EXCEPTION;
  begin
    if C3I_SYSTEM_SPEC.DS_WEAPON_STATUS_DATA.NEW_DATA then
      begin
        C3I_SYSTEM_SPEC.DS_WEAPON_STATUS_DATA.BUFFER.READ
          (LV_WEAPON_STATUS_DATA);
      exception
        when BUFFER_UNDERFLOW =>
          DS_Debug.Buffer_Underflow
          ("WEAPON_STATUS_DATA", "WEAPONS_INTERFACE");
      end;
      if true then
         begin
           WEAPONS_INTERFACE(LV_WEAPON_STATUS_DATA, LV_WEAPONS_EMREP,
                                        LV_WEAPONS_STATREP);
         exception
           when others =>
             DS_Debug.Undeclared_Exception("WEAPONS_INTERFACE");
             EXCEPTION_HAS_OCCURRED := true;
             EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
         end;
         if LV_WEAPON_STATUS_DATA.STATUS = DAMAGED or
            LV_WEAPON_STATUS_DATA.STATUS = SERVICE_REQUIRED or
            LV_WEAPON_STATUS_DATA.STATUS = OUT_OF_AMMUNITION
         then
           begin
             C3I_SYSTEM_SPEC.DS_WEAPONS_EMREP.BUFFER.WRITE
           (LV_WEAPONS_EMREP);
```

```
          exception
            when BUFFER_OVERFLOW =>
              DS_Debug.Buffer_Overflow
          ("WEAPONS_EMREP", "WEAPONS_INTERFACE");
          end;
        end if;
         begin
           C3I_SYSTEM_SPEC.DS_WEAPONS_STATREP.BUFFER.WRITE
          (LV_WEAPONS_STATREP);
          exception
            when BUFFER_OVERFLOW =>
              DS_Debug.Buffer_Overflow
          ("WEAPONS_STATREP", "WEAPONS_INTERFACE");
          end;
        if EXCEPTION_HAS_OCCURRED then
           DS_Debug.Unhandled_Exception
          ("WEAPONS_INTERFACE", PSDL_EXCEPTION'image(EXCEPTION_ID));
        end if;
    end if;
  end if;
end WEAPONS_INTERFACE_DRIVER;

procedure WEAPONS_SYSTEMS_DRIVER is
  LV_WEAPON_STATUS_DATA: WEAPON_STATUS;
  EXCEPTION_HAS_OCCURRED: boolean := false;
  EXCEPTION_ID: PSDL_EXCEPTION;
begin
  if true then
    if true then
       begin
         WEAPONS_SYSTEMS(LV_WEAPON_STATUS_DATA);
       exception
         when others =>
           DS_Debug.Undeclared_Exception("WEAPONS_SYSTEMS");
           EXCEPTION_HAS_OCCURRED := true;
           EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
       end;
        begin
          C3I_SYSTEM_SPEC.DS_WEAPON_STATUS_DATA.BUFFER.WRITE
         (LV_WEAPON_STATUS_DATA);
          exception
```

```
            when BUFFER_OVERFLOW =>
              DS_Debug.Buffer_Overflow
           ("WEAPON_STATUS_DATA", "WEAPONS_SYSTEMS");
           end;
        if EXCEPTION_HAS_OCCURRED then
           DS_Debug.Unhandled_Exception
          ("WEAPONS_SYSTEMS", PSDL_EXCEPTION'image(EXCEPTION_ID));
        end if;
    end if;
  end if;
end WEAPONS_SYSTEMS_DRIVER;

procedure MANAGE_USER_INTERFACE_DRIVER is
  EXCEPTION_HAS_OCCURRED: boolean := false;
  EXCEPTION_ID: PSDL_EXCEPTION;
begin
  if true then
    if true then
       begin
         MANAGE_USER_INTERFACE;
       exception
         when others =>
           DS_Debug.Undeclared_Exception("MANAGE_USER_INTERFACE");
           EXCEPTION_HAS_OCCURRED := true;
           EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
       end;
       if EXCEPTION_HAS_OCCURRED then
          DS_Debug.Unhandled_Exception
       ("MANAGE_USER_INTERFACE", PSDL_EXCEPTION'image(EXCEPTION_ID));
       end if;
    end if;
  end if;
end MANAGE_USER_INTERFACE_DRIVER;

procedure GET_USER_INPUTS_DRIVER is
  LV_TDD_ARCHIVE_SETUP: ARCHIVE_SETUP;
  LV_TDD_FILTER: SET_TRACK_FILTER;
  LV_TD_TRACK_REQUEST: DATABASE_REQUEST;
  LV_TCD_STATUS_QUERY: BOOLEAN;
  LV_EDITOR_SELECTED: BOOLEAN;
  LV_TCD_NETWORK_SETUP: NETWORK_SETUP;
```

```
  LV_TCD_EMISSION_CONTROL: EMISSIONS_CONTROL_COMMAND;
  LV_INITIATE_TRANS: INITIATE_TRANSMISSION_SEQUENCE;
  LV_TERMINATE_TRANS: BOOLEAN;
  EXCEPTION_HAS_OCCURRED: boolean := false;
  EXCEPTION_ID: PSDL_EXCEPTION;
begin
  if true then
    if true then
       begin
         GET_USER_INPUTS(LV_TDD_ARCHIVE_SETUP, LV_TDD_FILTER,
            LV_TD_TRACK_REQUEST,LV_TCD_STATUS_QUERY,
            LV_TCD_NETWORK_SETUP, LV_TCD_EMISSION_CONTROL,
            LV_EDITOR_SELECTED, LV_INITIATE_TRANS,
            LV_TERMINATE_TRANS);
       exception
         when others =>
           DS_Debug.Undeclared_Exception("GET_USER_INPUTS");
           EXCEPTION_HAS_OCCURRED := true;
           EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
       end;
        begin
          C3I_SYSTEM_SPEC.DS_TERMINATE_TRANS.BUFFER.WRITE
        (LV_TERMINATE_TRANS);
        exception
          when BUFFER_OVERFLOW =>
            DS_Debug.Buffer_Overflow
        ("TERMINATE_TRANS", "GET_USER_INPUTS");
        end;
        begin
          C3I_SYSTEM_SPEC.DS_INITIATE_TRANS.BUFFER.WRITE
        (LV_INITIATE_TRANS);
        exception
          when BUFFER_OVERFLOW =>
            DS_Debug.Buffer_Overflow
        ("INITIATE_TRANS", "GET_USER_INPUTS");
        end;
        begin
          C3I_SYSTEM_SPEC.DS_EDITOR_SELECTED.BUFFER.WRITE
        (LV_EDITOR_SELECTED);
        exception
          when BUFFER_OVERFLOW =>
```

```
      DS_Debug.Buffer_Overflow
 ("EDITOR_SELECTED", "GET_USER_INPUTS");
 end;
begin
  C3I_SYSTEM_SPEC.DS_TCD_EMISSION_CONTROL.BUFFER.WRITE
 (LV_TCD_EMISSION_CONTROL);
 exception
   when BUFFER_OVERFLOW =>
     DS_Debug.Buffer_Overflow
 ("TCD_EMISSION_CONTROL", "GET_USER_INPUTS");
 end;
begin
  C3I_SYSTEM_SPEC.DS_TCD_NETWORK_SETUP.BUFFER.WRITE
 (LV_TCD_NETWORK_SETUP);
 exception
   when BUFFER_OVERFLOW =>
     DS_Debug.Buffer_Overflow
 ("TCD_NETWORK_SETUP", "GET_USER_INPUTS");
 end;
begin
  C3I_SYSTEM_SPEC.DS_TCD_STATUS_QUERY.BUFFER.WRITE
 (LV_TCD_STATUS_QUERY);
 exception
   when BUFFER_OVERFLOW =>
     DS_Debug.Buffer_Overflow
 ("TCD_STATUS_QUERY", "GET_USER_INPUTS");
 end;
begin
  C3I_SYSTEM_SPEC.DS_TD_TRACK_REQUEST.BUFFER.WRITE
 (LV_TD_TRACK_REQUEST);
 exception
   when BUFFER_OVERFLOW =>
     DS_Debug.Buffer_Overflow
 ("TD_TRACK_REQUEST", "GET_USER_INPUTS");
 end;
begin
  C3I_SYSTEM_SPEC.DS_TDD_FILTER.BUFFER.WRITE
 (LV_TDD_FILTER);
 exception
   when BUFFER_OVERFLOW =>
     DS_Debug.Buffer_Overflow
```

```
        ("TDD_FILTER", "GET_USER_INPUTS");
        end;
       begin
         C3I_SYSTEM_SPEC.DS_TDD_ARCHIVE_SETUP.BUFFER.WRITE
        (LV_TDD_ARCHIVE_SETUP);
        exception
          when BUFFER_OVERFLOW =>
            DS_Debug.Buffer_Overflow
        ("TDD_ARCHIVE_SETUP", "GET_USER_INPUTS");
        end;
      if EXCEPTION_HAS_OCCURRED then
         DS_Debug.Unhandled_Exception
        ("GET_USER_INPUTS", PSDL_EXCEPTION'image(EXCEPTION_ID));
      end if;
    end if;
  end if;
end GET_USER_INPUTS_DRIVER;

procedure DISPLAY_TRACKS_DRIVER is
  LV_OUT_TRACKS: TRACK_TUPLE;
  LV_TD_TRACK_REQUEST: DATABASE_REQUEST;
  EXCEPTION_HAS_OCCURRED: boolean := false;
  EXCEPTION_ID: PSDL_EXCEPTION;
begin
  if C3I_SYSTEM_SPEC.DS_OUT_TRACKS.NEW_DATA then
    begin
      C3I_SYSTEM_SPEC.DS_OUT_TRACKS.BUFFER.READ(LV_OUT_TRACKS);
    exception
      when BUFFER_UNDERFLOW =>
        DS_Debug.Buffer_Underflow("OUT_TRACKS", "DISPLAY_TRACKS");
    end;
    begin
      C3I_SYSTEM_SPEC.DS_TD_TRACK_REQUEST.BUFFER.READ
        (LV_TD_TRACK_REQUEST);
    exception
      when BUFFER_UNDERFLOW =>
        DS_Debug.Buffer_Underflow
        ("TD_TRACK_REQUEST", "DISPLAY_TRACKS");
    end;
    if true then
       begin
```

```
        DISPLAY_TRACKS(LV_OUT_TRACKS, LV_TD_TRACK_REQUEST);
      exception
        when others =>
          DS_Debug.Undeclared_Exception("DISPLAY_TRACKS");
          EXCEPTION_HAS_OCCURRED := true;
          EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
      end;
      if EXCEPTION_HAS_OCCURRED then
         DS_Debug.Unhandled_Exception
        ("DISPLAY_TRACKS", PSDL_EXCEPTION'image(EXCEPTION_ID));
      end if;
    end if;
  end if;
end DISPLAY_TRACKS_DRIVER;

procedure MONITOR_OWNSHIP_POSITION_DRIVER is
  LV_POSITION_DATA: OWNSHIP_NAVIGATION_INFO;
  LV_OUT_TRACKS: TRACK_TUPLE;
  EXCEPTION_HAS_OCCURRED: boolean := false;
  EXCEPTION_ID: PSDL_EXCEPTION;
begin
  if C3I_SYSTEM_SPEC.DS_POSITION_DATA.NEW_DATA then
    begin
      C3I_SYSTEM_SPEC.DS_POSITION_DATA.BUFFER.READ(LV_POSITION_DATA);
    exception
      when BUFFER_UNDERFLOW =>
        DS_Debug.Buffer_Underflow
        ("POSITION_DATA", "MONITOR_OWNSHIP_POSITION");
    end;
    if true then
       begin
         MONITOR_OWNSHIP_POSITION(LV_POSITION_DATA, LV_OUT_TRACKS);
       exception
         when others =>
           DS_Debug.Undeclared_Exception("MONITOR_OWNSHIP_POSITION");
           EXCEPTION_HAS_OCCURRED := true;
           EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
       end;
        begin
          C3I_SYSTEM_SPEC.DS_OUT_TRACKS.BUFFER.WRITE(LV_OUT_TRACKS);
         exception
```

```
          when BUFFER_OVERFLOW =>
            DS_Debug.Buffer_Overflow
         ("OUT_TRACKS", "MONITOR_OWNSHIP_POSITION");
          end;
       if EXCEPTION_HAS_OCCURRED then
          DS_Debug.Unhandled_Exception
    ("MONITOR_OWNSHIP_POSITION", PSDL_EXCEPTION'image(EXCEPTION_ID));
       end if;
    end if;
  end if;
end MONITOR_OWNSHIP_POSITION_DRIVER;

procedure ADD_SENSOR_TRACK_DRIVER is
  LV_FILTERED_SENSOR_TRACK: ADD_TRACK_TUPLE;
  LV_TDD_FILTER: SET_TRACK_FILTER;
  LV_OUT_TRACKS: TRACK_TUPLE;
  EXCEPTION_HAS_OCCURRED: boolean := false;
  EXCEPTION_ID: PSDL_EXCEPTION;
begin
  if C3I_SYSTEM_SPEC.DS_FILTERED_SENSOR_TRACK.NEW_DATA then
    begin
      C3I_SYSTEM_SPEC.DS_FILTERED_SENSOR_TRACK.BUFFER.READ
        (LV_FILTERED_SENSOR_TRACK);
    exception
      when BUFFER_UNDERFLOW =>
        DS_Debug.Buffer_Underflow
         ("FILTERED_SENSOR_TRACK", "ADD_SENSOR_TRACK");
    end;
    begin
      C3I_SYSTEM_SPEC.DS_TDD_FILTER.BUFFER.READ(LV_TDD_FILTER);
    exception
      when BUFFER_UNDERFLOW =>
        DS_Debug.Buffer_Underflow("TDD_FILTER", "ADD_SENSOR_TRACK");
    end;
    if true then
       begin
         ADD_SENSOR_TRACK(LV_FILTERED_SENSOR_TRACK,
                          LV_TDD_FILTER, LV_OUT_TRACKS);
       exception
         when others =>
           DS_Debug.Undeclared_Exception("ADD_SENSOR_TRACK");
```

```
            EXCEPTION_HAS_OCCURRED := true;
            EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
          end;
           begin
             C3I_SYSTEM_SPEC.DS_OUT_TRACKS.BUFFER.WRITE(LV_OUT_TRACKS);
           exception
             when BUFFER_OVERFLOW =>
               DS_Debug.Buffer_Overflow
           ("OUT_TRACKS", "ADD_SENSOR_TRACK");
           end;
          if EXCEPTION_HAS_OCCURRED then
             DS_Debug.Unhandled_Exception
           ("ADD_SENSOR_TRACK", PSDL_EXCEPTION'image(EXCEPTION_ID));
          end if;
        end if;
      end if;
    end ADD_SENSOR_TRACK_DRIVER;

    procedure FILTER_SENSOR_TRACKS_DRIVER is
      LV_SENSOR_ADD_TRACK: ADD_TRACK_TUPLE;
      LV_TDD_FILTER: SET_TRACK_FILTER;
      LV_FILTERED_SENSOR_TRACK: ADD_TRACK_TUPLE;
      EXCEPTION_HAS_OCCURRED: boolean := false;
      EXCEPTION_ID: PSDL_EXCEPTION;
    begin
      if C3I_SYSTEM_SPEC.DS_SENSOR_ADD_TRACK.NEW_DATA then
        begin

  C3I_SYSTEM_SPEC.DS_SENSOR_ADD_TRACK.BUFFER.READ(LV_SENSOR_ADD_TRACK);
        exception
          when BUFFER_UNDERFLOW =>
            DS_Debug.Buffer_Underflow
             ("SENSOR_ADD_TRACK", "FILTER_SENSOR_TRACKS");
        end;
        begin
          C3I_SYSTEM_SPEC.DS_TDD_FILTER.BUFFER.READ(LV_TDD_FILTER);
        exception
          when BUFFER_UNDERFLOW =>
            DS_Debug.Buffer_Underflow
             ("TDD_FILTER", "FILTER_SENSOR_TRACKS");
        end;
```

```
    if true then
       begin
         FILTER_SENSOR_TRACKS(LV_SENSOR_ADD_TRACK, LV_TDD_FILTER,
          LV_FILTERED_SENSOR_TRACK);
       exception
         when others =>
           DS_Debug.Undeclared_Exception("FILTER_SENSOR_TRACKS");
           EXCEPTION_HAS_OCCURRED := true;
           EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
       end;
        begin
          C3I_SYSTEM_SPEC.DS_FILTERED_SENSOR_TRACK.BUFFER.WRITE
         (LV_FILTERED_SENSOR_TRACK);
         exception
           when BUFFER_OVERFLOW =>
             DS_Debug.Buffer_Overflow
         ("FILTERED_SENSOR_TRACK", "FILTER_SENSOR_TRACKS");
         end;
       if EXCEPTION_HAS_OCCURRED then
          DS_Debug.Unhandled_Exception
         ("FILTER_SENSOR_TRACKS", PSDL_EXCEPTION'image(EXCEPTION_ID));
       end if;
    end if;
  end if;
end FILTER_SENSOR_TRACKS_DRIVER;

procedure PREPARE_SENSOR_TRACK_DRIVER is
  LV_SENSOR_CONTACT_DATA: LOCAL_TRACK_INFO;
  LV_POSITION_DATA: OWNSHIP_NAVIGATION_INFO;
  LV_SENSOR_ADD_TRACK: ADD_TRACK_TUPLE;
  EXCEPTION_HAS_OCCURRED: boolean := false;
  EXCEPTION_ID: PSDL_EXCEPTION;
begin
  if C3I_SYSTEM_SPEC.DS_POSITION_DATA.NEW_DATA OR
           C3I_SYSTEM_SPEC.DS_SENSOR_CONTACT_DATA.NEW_DATA then
    begin
      C3I_SYSTEM_SPEC.DS_SENSOR_CONTACT_DATA.BUFFER.READ
        (LV_SENSOR_CONTACT_DATA);
    exception
      when BUFFER_UNDERFLOW =>
       DS_Debug.Buffer_Underflow
```

```
        ("SENSOR_CONTACT_DATA", "PREPARE_SENSOR_TRACK");
    end;
    begin
      C3I_SYSTEM_SPEC.DS_POSITION_DATA.BUFFER.READ(LV_POSITION_DATA);
    exception
      when BUFFER_UNDERFLOW =>
        DS_Debug.Buffer_Underflow
         ("POSITION_DATA", "PREPARE_SENSOR_TRACK");
    end;
    if true then
       begin
         PREPARE_SENSOR_TRACK(LV_SENSOR_CONTACT_DATA,
          LV_POSITION_DATA, LV_SENSOR_ADD_TRACK);
       exception
         when others =>
           DS_Debug.Undeclared_Exception("PREPARE_SENSOR_TRACK");
           EXCEPTION_HAS_OCCURRED := true;
           EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
       end;
        begin
          C3I_SYSTEM_SPEC.DS_SENSOR_ADD_TRACK.BUFFER.WRITE
        (LV_SENSOR_ADD_TRACK);
         exception
           when BUFFER_OVERFLOW =>
             DS_Debug.Buffer_Overflow
        ("SENSOR_ADD_TRACK", "PREPARE_SENSOR_TRACK");
         end;
       if EXCEPTION_HAS_OCCURRED then
          DS_Debug.Unhandled_Exception
       ("PREPARE_SENSOR_TRACK", PSDL_EXCEPTION'image(EXCEPTION_ID));
       end if;
    end if;
  end if;
end PREPARE_SENSOR_TRACK_DRIVER;

procedure ANALYZE_SENSOR_DATA_DRIVER is
  LV_SENSOR_DATA: SENSOR_RECORD;
  LV_SENSOR_CONTACT_DATA: LOCAL_TRACK_INFO;
  EXCEPTION_HAS_OCCURRED: boolean := false;
  EXCEPTION_ID: PSDL_EXCEPTION;
begin
```

```
  if C3I_SYSTEM_SPEC.DS_SENSOR_DATA.NEW_DATA then
    begin
      C3I_SYSTEM_SPEC.DS_SENSOR_DATA.BUFFER.READ(LV_SENSOR_DATA);
    exception
      when BUFFER_UNDERFLOW =>
        DS_Debug.Buffer_Underflow
         ("SENSOR_DATA", "ANALYZE_SENSOR_DATA");
    end;
    if true then
       begin
         ANALYZE_SENSOR_DATA(LV_SENSOR_DATA, LV_SENSOR_CONTACT_DATA);
       exception
         when others =>
           DS_Debug.Undeclared_Exception("ANALYZE_SENSOR_DATA");
           EXCEPTION_HAS_OCCURRED := true;
           EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
       end;
        begin
          C3I_SYSTEM_SPEC.DS_SENSOR_CONTACT_DATA.BUFFER.WRITE
        (LV_SENSOR_CONTACT_DATA);
         exception
           when BUFFER_OVERFLOW =>
             DS_Debug.Buffer_Overflow
        ("SENSOR_CONTACT_DATA", "ANALYZE_SENSOR_DATA");
         end;
       if EXCEPTION_HAS_OCCURRED then
          DS_Debug.Unhandled_Exception
         ("ANALYZE_SENSOR_DATA", PSDL_EXCEPTION'image(EXCEPTION_ID));
       end if;
    end if;
  end if;
end ANALYZE_SENSOR_DATA_DRIVER;

procedure NAVIGATION_SYSTEM_DRIVER is
  LV_POSITION_DATA: OWNSHIP_NAVIGATION_INFO;
  EXCEPTION_HAS_OCCURRED: boolean := false;
  EXCEPTION_ID: PSDL_EXCEPTION;
begin
  if true then
    if true then
       begin
```

```
          NAVIGATION_SYSTEM(LV_POSITION_DATA);
        exception
          when others =>
            DS_Debug.Undeclared_Exception("NAVIGATION_SYSTEM");
            EXCEPTION_HAS_OCCURRED := true;
            EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
        end;
         begin
           C3I_SYSTEM_SPEC.DS_POSITION_DATA.BUFFER.WRITE
          (LV_POSITION_DATA);
          exception
            when BUFFER_OVERFLOW =>
              DS_Debug.Buffer_Overflow
          ("POSITION_DATA", "NAVIGATION_SYSTEM");
          end;
        if EXCEPTION_HAS_OCCURRED then
           DS_Debug.Unhandled_Exception
          ("NAVIGATION_SYSTEM", PSDL_EXCEPTION'image(EXCEPTION_ID));
        end if;
    end if;
  end if;
end NAVIGATION_SYSTEM_DRIVER;

procedure SENSORS_DRIVER is
  LV_SENSOR_DATA: SENSOR_RECORD;
  EXCEPTION_HAS_OCCURRED: boolean := false;
  EXCEPTION_ID: PSDL_EXCEPTION;
begin
  if true then
    if true then
       begin
         SENSORS(LV_SENSOR_DATA);
       exception
         when others =>
           DS_Debug.Undeclared_Exception("SENSORS");
           EXCEPTION_HAS_OCCURRED := true;
           EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
       end;
        begin
          C3I_SYSTEM_SPEC.DS_SENSOR_DATA.BUFFER.WRITE(LV_SENSOR_DATA);
          exception
```

```
            when BUFFER_OVERFLOW =>
              DS_Debug.Buffer_Overflow
           ("SENSOR_DATA", "SENSORS");
           end;
         if EXCEPTION_HAS_OCCURRED then
            DS_Debug.Unhandled_Exception
           ("SENSORS", PSDL_EXCEPTION'image(EXCEPTION_ID));
         end if;
      end if;
    end if;
end SENSORS_DRIVER;

procedure ADD_COMMS_TRACK_DRIVER is
  LV_FILTERED_COMMS_TRACK: ADD_TRACK_TUPLE;
  LV_TDD_FILTER: SET_TRACK_FILTER;
  LV_OUT_TRACKS: TRACK_TUPLE;
  EXCEPTION_HAS_OCCURRED: boolean := false;
  EXCEPTION_ID: PSDL_EXCEPTION;
begin
  if C3I_SYSTEM_SPEC.DS_FILTERED_COMMS_TRACK.NEW_DATA then
    begin
      C3I_SYSTEM_SPEC.DS_FILTERED_COMMS_TRACK.BUFFER.READ
        (LV_FILTERED_COMMS_TRACK);
    exception
      when BUFFER_UNDERFLOW =>
        DS_Debug.Buffer_Underflow
        ("FILTERED_COMMS_TRACK", "ADD_COMMS_TRACK");
    end;
    begin
      C3I_SYSTEM_SPEC.DS_TDD_FILTER.BUFFER.READ(LV_TDD_FILTER);
    exception
      when BUFFER_UNDERFLOW =>
        DS_Debug.Buffer_Underflow("TDD_FILTER", "ADD_COMMS_TRACK");
    end;
    if true then
       begin
         ADD_COMMS_TRACK(LV_FILTERED_COMMS_TRACK, LV_TDD_FILTER,
                         LV_OUT_TRACKS);
       exception
         when others =>
           DS_Debug.Undeclared_Exception("ADD_COMMS_TRACK");
```

```
            EXCEPTION_HAS_OCCURRED := true;
            EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
          end;
           begin
             C3I_SYSTEM_SPEC.DS_OUT_TRACKS.BUFFER.WRITE(LV_OUT_TRACKS);
           exception
             when BUFFER_OVERFLOW =>
               DS_Debug.Buffer_Overflow
           ("OUT_TRACKS", "ADD_COMMS_TRACK");
           end;
          if EXCEPTION_HAS_OCCURRED then
            DS_Debug.Unhandled_Exception
           ("ADD_COMMS_TRACK", PSDL_EXCEPTION'image(EXCEPTION_ID));
          end if;
      end if;
    end if;
  end ADD_COMMS_TRACK_DRIVER;

  procedure FILTER_COMMS_TRACKS_DRIVER is
    LV_COMMS_ADD_TRACK: ADD_TRACK_TUPLE;
    LV_TDD_FILTER: SET_TRACK_FILTER;
    LV_FILTERED_COMMS_TRACK: ADD_TRACK_TUPLE;
    EXCEPTION_HAS_OCCURRED: boolean := false;
    EXCEPTION_ID: PSDL_EXCEPTION;
  begin
    if C3I_SYSTEM_SPEC.DS_COMMS_ADD_TRACK.NEW_DATA then
      begin
        C3I_SYSTEM_SPEC.DS_COMMS_ADD_TRACK.BUFFER.READ
          (LV_COMMS_ADD_TRACK);
      exception
        when BUFFER_UNDERFLOW =>
          DS_Debug.Buffer_Underflow
           ("COMMS_ADD_TRACK", "FILTER_COMMS_TRACKS");
      end;
      begin
        C3I_SYSTEM_SPEC.DS_TDD_FILTER.BUFFER.READ(LV_TDD_FILTER);
      exception
        when BUFFER_UNDERFLOW =>
          DS_Debug.Buffer_Underflow
           ("TDD_FILTER", "FILTER_COMMS_TRACKS");
      end;
```

```
    if true then
       begin
         FILTER_COMMS_TRACKS(LV_COMMS_ADD_TRACK, LV_TDD_FILTER,
          LV_FILTERED_COMMS_TRACK);
       exception
         when others =>
           DS_Debug.Undeclared_Exception("FILTER_COMMS_TRACKS");
           EXCEPTION_HAS_OCCURRED := true;
           EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
       end;
        begin
          C3I_SYSTEM_SPEC.DS_FILTERED_COMMS_TRACK.BUFFER.WRITE
         (LV_FILTERED_COMMS_TRACK);
         exception
           when BUFFER_OVERFLOW =>
             DS_Debug.Buffer_Overflow
         ("FILTERED_COMMS_TRACK", "FILTER_COMMS_TRACKS");
         end;
       if EXCEPTION_HAS_OCCURRED then
          DS_Debug.Unhandled_Exception
         ("FILTER_COMMS_TRACKS", PSDL_EXCEPTION'image(EXCEPTION_ID));
       end if;
    end if;
  end if;
end FILTER_COMMS_TRACKS_DRIVER;

procedure EXTRACT_TRACKS_DRIVER is
  LV_COMMS_TEXT_FILE: TEXT_RECORD;
  LV_COMMS_ADD_TRACK: ADD_TRACK_TUPLE;
  EXCEPTION_HAS_OCCURRED: boolean := false;
  EXCEPTION_ID: PSDL_EXCEPTION;
begin
  if true then
    begin
      C3I_SYSTEM_SPEC.DS_COMMS_TEXT_FILE.BUFFER.READ
         (LV_COMMS_TEXT_FILE);
    exception
      when BUFFER_UNDERFLOW =>
        DS_Debug.Buffer_Underflow
         ("COMMS_TEXT_FILE", "EXTRACT_TRACKS");
    end;
```

```
    if LV_COMMS_TEXT_FILE.IS_TRACK then
       begin
         EXTRACT_TRACKS(LV_COMMS_TEXT_FILE, LV_COMMS_ADD_TRACK);
       exception
         when others =>
           DS_Debug.Undeclared_Exception("EXTRACT_TRACKS");
           EXCEPTION_HAS_OCCURRED := true;
           EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
       end;
        begin
          C3I_SYSTEM_SPEC.DS_COMMS_ADD_TRACK.BUFFER.WRITE
        (LV_COMMS_ADD_TRACK);
        exception
          when BUFFER_OVERFLOW =>
            DS_Debug.Buffer_Overflow
        ("COMMS_ADD_TRACK", "EXTRACT_TRACKS");
        end;
       if EXCEPTION_HAS_OCCURRED then
         DS_Debug.Unhandled_Exception
        ("EXTRACT_TRACKS", PSDL_EXCEPTION'image(EXCEPTION_ID));
       end if;
    end if;
  end if;
end EXTRACT_TRACKS_DRIVER;

procedure DECIDE_FOR_ARCHIVING_DRIVER is
  LV_INPUT_TEXT_RECORD: TEXT_RECORD;
  LV_TDD_ARCHIVE_SETUP: ARCHIVE_SETUP;
  LV_COMMS_TEXT_FILE: TEXT_RECORD;
  LV_COMMS_EMAIL: FILENAME;
  EXCEPTION_HAS_OCCURRED: boolean := false;
  EXCEPTION_ID: PSDL_EXCEPTION;
begin
  if C3I_SYSTEM_SPEC.DS_INPUT_TEXT_RECORD.NEW_DATA then
    begin
      C3I_SYSTEM_SPEC.DS_INPUT_TEXT_RECORD.BUFFER.READ
         (LV_INPUT_TEXT_RECORD);
    exception
      when BUFFER_UNDERFLOW =>
        DS_Debug.Buffer_Underflow
        ("INPUT_TEXT_RECORD", "DECIDE_FOR_ARCHIVING");
```

```
end;
begin
  C3I_SYSTEM_SPEC.DS_TDD_ARCHIVE_SETUP.BUFFER.READ
     (LV_TDD_ARCHIVE_SETUP);
exception
  when BUFFER_UNDERFLOW =>
    DS_Debug.Buffer_Underflow
     ("TDD_ARCHIVE_SETUP", "DECIDE_FOR_ARCHIVING");
end;
if true then
   begin
     DECIDE_FOR_ARCHIVING(LV_INPUT_TEXT_RECORD,
        LV_TDD_ARCHIVE_SETUP,LV_COMMS_TEXT_FILE, LV_COMMS_EMAIL);
   exception
     when others =>
       DS_Debug.Undeclared_Exception("DECIDE_FOR_ARCHIVING");
       EXCEPTION_HAS_OCCURRED := true;
       EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
   end;
   if not LV_COMMS_TEXT_FILE.IS_TRACK
   then
     begin
       C3I_SYSTEM_SPEC.DS_COMMS_EMAIL.BUFFER.WRITE
     (LV_COMMS_EMAIL);
     exception
       when BUFFER_OVERFLOW =>
         DS_Debug.Buffer_Overflow
     ("COMMS_EMAIL", "DECIDE_FOR_ARCHIVING");
     end;
   end if;
   if LV_COMMS_TEXT_FILE.ARCHIVE
   then
     begin
       C3I_SYSTEM_SPEC.DS_COMMS_TEXT_FILE.BUFFER.WRITE
     (LV_COMMS_TEXT_FILE);
     exception
       when BUFFER_OVERFLOW =>
         DS_Debug.Buffer_Overflow
     ("COMMS_TEXT_FILE", "DECIDE_FOR_ARCHIVING");
     end;
   end if;
```

```
        if EXCEPTION_HAS_OCCURRED then
           DS_Debug.Unhandled_Exception
          ("DECIDE_FOR_ARCHIVING", PSDL_EXCEPTION'image(EXCEPTION_ID));
        end if;
      end if;
    end if;
end DECIDE_FOR_ARCHIVING_DRIVER;

procedure PARSE_INPUT_FILE_DRIVER is
  LV_INPUT_LINK_MESSAGE: FILENAME;
  LV_INPUT_TEXT_RECORD: TEXT_RECORD;
  EXCEPTION_HAS_OCCURRED: boolean := false;
  EXCEPTION_ID: PSDL_EXCEPTION;
begin
  if C3I_SYSTEM_SPEC.DS_INPUT_LINK_MESSAGE.NEW_DATA then
    begin
      C3I_SYSTEM_SPEC.DS_INPUT_LINK_MESSAGE.BUFFER.READ
        (LV_INPUT_LINK_MESSAGE);
    exception
      when BUFFER_UNDERFLOW =>
       DS_Debug.Buffer_Underflow
        ("INPUT_LINK_MESSAGE", "PARSE_INPUT_FILE");
    end;
    if true then
       begin
        PARSE_INPUT_FILE(LV_INPUT_LINK_MESSAGE,
                         LV_INPUT_TEXT_RECORD);
       exception
         when others =>
           DS_Debug.Undeclared_Exception("PARSE_INPUT_FILE");
           EXCEPTION_HAS_OCCURRED := true;
           EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
       end;
        begin
          C3I_SYSTEM_SPEC.DS_INPUT_TEXT_RECORD.BUFFER.WRITE
        (LV_INPUT_TEXT_RECORD);
        exception
          when BUFFER_OVERFLOW =>
            DS_Debug.Buffer_Overflow
        ("INPUT_TEXT_RECORD", "PARSE_INPUT_FILE");
        end;
```

```
        if EXCEPTION_HAS_OCCURRED then
           DS_Debug.Unhandled_Exception
          ("PARSE_INPUT_FILE", PSDL_EXCEPTION'image(EXCEPTION_ID));
        end if;
      end if;
    end if;
  end PARSE_INPUT_FILE_DRIVER;

  procedure COMMS_LINKS_DRIVER is
    LV_INPUT_LINK_MESSAGE: FILENAME;
    EXCEPTION_HAS_OCCURRED: boolean := false;
    EXCEPTION_ID: PSDL_EXCEPTION;
  begin
    if true then
      if true then
        begin
          COMMS_LINKS(LV_INPUT_LINK_MESSAGE);
        exception
          when others =>
            DS_Debug.Undeclared_Exception("COMMS_LINKS");
            EXCEPTION_HAS_OCCURRED := true;
            EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
        end;
         begin
           C3I_SYSTEM_SPEC.DS_INPUT_LINK_MESSAGE.BUFFER.WRITE
          (LV_INPUT_LINK_MESSAGE);
          exception
            when BUFFER_OVERFLOW =>
              DS_Debug.Buffer_Overflow
          ("INPUT_LINK_MESSAGE", "COMMS_LINKS");
          end;
        if EXCEPTION_HAS_OCCURRED then
           DS_Debug.Unhandled_Exception
          ("COMMS_LINKS", PSDL_EXCEPTION'image(EXCEPTION_ID));
        end if;
      end if;
    end if;
  end COMMS_LINKS_DRIVER;

end TL;
```

APPENDIX D

STATIC SCHEDULE

SS.A

This appendix contains the static schedule package ss.a. The static scheduler uses the PSDL description file, psdl.txt, as input, and prepares the static schedule for the time critical operators.

```
with GLOBAL_DECLARATIONS; use GLOBAL_DECLARATIONS;
with DS_DEBUG_PKG; use DS_DEBUG_PKG;
with TL; use TL;
with DS_PACKAGE; use DS_PACKAGE;
with PRIORITY_DEFINITIONS; use PRIORITY_DEFINITIONS;
with CALENDAR; use CALENDAR;
with TEXT_IO; use TEXT_IO;
procedure STATIC_SCHEDULE is
  PREPARE_PERIODIC_REPORT_TIMING_ERROR : exception;
  CONVERT_TO_TEXT_FILE_TIMING_ERROR : exception;
  FORWARD_FOR_TRANSMISSION_TIMING_ERROR : exception;
  MAKE_ROUTING_TIMING_ERROR : exception;
  WEAPONS_SYSTEMS_TIMING_ERROR : exception;
  WEAPONS_INTERFACE_TIMING_ERROR : exception;
  MONITOR_OWNSHIP_POSITION_TIMING_ERROR : exception;
  NAVIGATION_SYSTEM_TIMING_ERROR : exception;
  ADD_SENSOR_TRACK_TIMING_ERROR : exception;
  FILTER_SENSOR_TRACKS_TIMING_ERROR : exception;
  PREPARE_SENSOR_TRACK_TIMING_ERROR : exception;
  ANALYZE_SENSOR_DATA_TIMING_ERROR : exception;
  SENSORS_TIMING_ERROR : exception;
  ADD_COMMS_TRACK_TIMING_ERROR : exception;
  FILTER_COMMS_TRACKS_TIMING_ERROR : exception;
  EXTRACT_TRACKS_TIMING_ERROR : exception;
  DECIDE_FOR_ARCHIVING_TIMING_ERROR : exception;
  PARSE_INPUT_FILE_TIMING_ERROR : exception;
```

```
COMMS_LINKS_TIMING_ERROR : exception;
task type SCHEDULE_TYPE is
  pragma priority (STATIC_SCHEDULE_PRIORITY);
end SCHEDULE_TYPE;
for SCHEDULE_TYPE'STORAGE_SIZE use 200_000;
SCHEDULE : SCHEDULE_TYPE;
task body SCHEDULE_TYPE is
  PERIOD : duration := duration( 5.00000000000000E+01);
  COMMS_LINKS_STOP_TIME1 : duration :=
                                        duration( 1.20000000000000E+00);
  PARSE_INPUT_FILE_STOP_TIME2 : duration :=
                                       duration( 1.70000000000000E+00);
  DECIDE_FOR_ARCHIVING_STOP_TIME3 : duration :=
                                        duration( 2.20000000000000E+00);
  EXTRACT_TRACKS_STOP_TIME4 : duration :=
                                        duration( 2.70000000000000E+00);
  FILTER_COMMS_TRACKS_STOP_TIME5 : duration :=
                                        duration( 3.20000000000000E+00);
  ADD_COMMS_TRACK_STOP_TIME6 : duration :=
                                        duration( 3.70000000000000E+00);
  SENSORS_STOP_TIME7 : duration :=
                                        duration( 4.50000000000000E+00);
  ANALYZE_SENSOR_DATA_STOP_TIME8 : duration :=
                                        duration( 5.00000000000000E+00);
  PREPARE_SENSOR_TRACK_STOP_TIME9 : duration :=
                                        duration( 5.50000000000000E+00);
  FILTER_SENSOR_TRACKS_STOP_TIME10 : duration :=
                                        duration( 6.00000000000000E+00);
  ADD_SENSOR_TRACK_STOP_TIME11 : duration :=
                                        duration( 6.50000000000000E+00);
  NAVIGATION_SYSTEM_STOP_TIME12 : duration :=
                                        duration( 7.30000000000000E+00);
  MONITOR_OWNSHIP_POSITION_STOP_TIME13 : duration :=
                                        duration( 7.80000000000000E+00);
  WEAPONS_INTERFACE_STOP_TIME14 : duration :=
                                        duration( 8.30000000000000E+00);
  WEAPONS_SYSTEMS_STOP_TIME15 : duration :=
                                        duration( 8.80000000000000E+00);
  MAKE_ROUTING_STOP_TIME16 : duration :=
                                        duration( 9.30000000000000E+00);
  FORWARD_FOR_TRANSMISSION_STOP_TIME17 : duration :=
```

```
                                        duration( 9.80000000000000E+00);
    CONVERT_TO_TEXT_FILE_STOP_TIME18 : duration :=
                                        duration( 1.06000000000000E+01);
    PREPARE_PERIODIC_REPORT_STOP_TIME19 : duration :=
                                        duration( 1.14000000000000E+01);
    SLACK_TIME : duration;
    START_OF_PERIOD : time := clock;
    CURRENT_TIME : duration;
begin
  loop
    begin
      COMMS_LINKS_DRIVER;
      SLACK_TIME := START_OF_PERIOD + COMMS_LINKS_STOP_TIME1 - CLOCK;
      if SLACK_TIME >= 0.0 then
        delay (SLACK_TIME);
      else
        raise COMMS_LINKS_TIMING_ERROR;
      end if;
      delay (START_OF_PERIOD + COMMS_LINKS_STOP_TIME1 - CLOCK);

      PARSE_INPUT_FILE_DRIVER;
      SLACK_TIME :=
          START_OF_PERIOD + PARSE_INPUT_FILE_STOP_TIME2 - CLOCK;
      if SLACK_TIME >= 0.0 then
        delay (SLACK_TIME);
      else
        raise PARSE_INPUT_FILE_TIMING_ERROR;
      end if;
      delay (START_OF_PERIOD + PARSE_INPUT_FILE_STOP_TIME2 - CLOCK);

      DECIDE_FOR_ARCHIVING_DRIVER;
      SLACK_TIME :=
          START_OF_PERIOD + DECIDE_FOR_ARCHIVING_STOP_TIME3 - CLOCK;
      if SLACK_TIME >= 0.0 then
        delay (SLACK_TIME);
      else
        raise DECIDE_FOR_ARCHIVING_TIMING_ERROR;
      end if;
      delay (START_OF_PERIOD + DECIDE_FOR_ARCHIVING_STOP_TIME3 - CLOCK);

      EXTRACT_TRACKS_DRIVER;
```

```
SLACK_TIME := START_OF_PERIOD + EXTRACT_TRACKS_STOP_TIME4 - CLOCK;
if SLACK_TIME >= 0.0 then
  delay (SLACK_TIME);
else
  raise EXTRACT_TRACKS_TIMING_ERROR;
end if;
delay (START_OF_PERIOD + EXTRACT_TRACKS_STOP_TIME4 - CLOCK);

FILTER_COMMS_TRACKS_DRIVER;
SLACK_TIME :=
     START_OF_PERIOD + FILTER_COMMS_TRACKS_STOP_TIME5 - CLOCK;
if SLACK_TIME >= 0.0 then
  delay (SLACK_TIME);
else
  raise FILTER_COMMS_TRACKS_TIMING_ERROR;
end if;
delay (START_OF_PERIOD + FILTER_COMMS_TRACKS_STOP_TIME5 - CLOCK);

ADD_COMMS_TRACK_DRIVER;
SLACK_TIME :=
     START_OF_PERIOD + ADD_COMMS_TRACK_STOP_TIME6 - CLOCK;
if SLACK_TIME >= 0.0 then
  delay (SLACK_TIME);
else
  raise ADD_COMMS_TRACK_TIMING_ERROR;
end if;
delay (START_OF_PERIOD + ADD_COMMS_TRACK_STOP_TIME6 - CLOCK);


SENSORS_DRIVER;
SLACK_TIME :=
     START_OF_PERIOD + SENSORS_STOP_TIME7 - CLOCK;
if SLACK_TIME >= 0.0 then
  delay (SLACK_TIME);
else
  raise SENSORS_TIMING_ERROR;
end if;
delay (START_OF_PERIOD + SENSORS_STOP_TIME7 - CLOCK);

ANALYZE_SENSOR_DATA_DRIVER;
SLACK_TIME :=
```

```
            START_OF_PERIOD + ANALYZE_SENSOR_DATA_STOP_TIME8 - CLOCK;
        if SLACK_TIME >= 0.0 then
          delay (SLACK_TIME);
        else
          raise ANALYZE_SENSOR_DATA_TIMING_ERROR;
        end if;
        delay (START_OF_PERIOD + ANALYZE_SENSOR_DATA_STOP_TIME8 - CLOCK);

        PREPARE_SENSOR_TRACK_DRIVER;
        SLACK_TIME :=
            START_OF_PERIOD + PREPARE_SENSOR_TRACK_STOP_TIME9 - CLOCK;
        if SLACK_TIME >= 0.0 then
          delay (SLACK_TIME);
        else
          raise PREPARE_SENSOR_TRACK_TIMING_ERROR;
        end if;
        delay (START_OF_PERIOD + PREPARE_SENSOR_TRACK_STOP_TIME9 - CLOCK);

        FILTER_SENSOR_TRACKS_DRIVER;
        SLACK_TIME :=
            START_OF_PERIOD + FILTER_SENSOR_TRACKS_STOP_TIME10 - CLOCK;
        if SLACK_TIME >= 0.0 then
          delay (SLACK_TIME);
        else
          raise FILTER_SENSOR_TRACKS_TIMING_ERROR;
        end if;
            delay  (START_OF_PERIOD  +  FILTER_SENSOR_TRACKS_STOP_TIME10  -
CLOCK);

        ADD_SENSOR_TRACK_DRIVER;
        SLACK_TIME :=
            START_OF_PERIOD + ADD_SENSOR_TRACK_STOP_TIME11 - CLOCK;
        if SLACK_TIME >= 0.0 then
          delay (SLACK_TIME);
        else
          raise ADD_SENSOR_TRACK_TIMING_ERROR;
        end if;
        delay (START_OF_PERIOD + ADD_SENSOR_TRACK_STOP_TIME11 - CLOCK);

        NAVIGATION_SYSTEM_DRIVER;
        SLACK_TIME :=
```

```
    START_OF_PERIOD + NAVIGATION_SYSTEM_STOP_TIME12 - CLOCK;
if SLACK_TIME >= 0.0 then
  delay (SLACK_TIME);
else
  raise NAVIGATION_SYSTEM_TIMING_ERROR;
end if;
delay (START_OF_PERIOD + NAVIGATION_SYSTEM_STOP_TIME12 - CLOCK);

MONITOR_OWNSHIP_POSITION_DRIVER;
SLACK_TIME :=
  START_OF_PERIOD + MONITOR_OWNSHIP_POSITION_STOP_TIME13 - CLOCK;
if SLACK_TIME >= 0.0 then
  delay (SLACK_TIME);
else
  raise MONITOR_OWNSHIP_POSITION_TIMING_ERROR;
end if;
delay
(START_OF_PERIOD + MONITOR_OWNSHIP_POSITION_STOP_TIME13 - CLOCK);

WEAPONS_INTERFACE_DRIVER;
SLACK_TIME :=
    START_OF_PERIOD + WEAPONS_INTERFACE_STOP_TIME14 - CLOCK;
if SLACK_TIME >= 0.0 then
  delay (SLACK_TIME);
else
  raise WEAPONS_INTERFACE_TIMING_ERROR;
end if;
delay (START_OF_PERIOD + WEAPONS_INTERFACE_STOP_TIME14 - CLOCK);

WEAPONS_SYSTEMS_DRIVER;
SLACK_TIME :=
    START_OF_PERIOD + WEAPONS_SYSTEMS_STOP_TIME15 - CLOCK;
if SLACK_TIME >= 0.0 then
  delay (SLACK_TIME);
else
  raise WEAPONS_SYSTEMS_TIMING_ERROR;
end if;
delay (START_OF_PERIOD + WEAPONS_SYSTEMS_STOP_TIME15 - CLOCK);


MAKE_ROUTING_DRIVER;
```

```
SLACK_TIME := START_OF_PERIOD + MAKE_ROUTING_STOP_TIME16 - CLOCK;
if SLACK_TIME >= 0.0 then
  delay (SLACK_TIME);
else
  raise MAKE_ROUTING_TIMING_ERROR;
end if;
delay (START_OF_PERIOD + MAKE_ROUTING_STOP_TIME16 - CLOCK);

FORWARD_FOR_TRANSMISSION_DRIVER;
SLACK_TIME :=
  START_OF_PERIOD + FORWARD_FOR_TRANSMISSION_STOP_TIME17 - CLOCK;
if SLACK_TIME >= 0.0 then
  delay (SLACK_TIME);
else
  raise FORWARD_FOR_TRANSMISSION_TIMING_ERROR;
end if;
delay
(START_OF_PERIOD + FORWARD_FOR_TRANSMISSION_STOP_TIME17 - CLOCK);

CONVERT_TO_TEXT_FILE_DRIVER;
SLACK_TIME :=
    START_OF_PERIOD + CONVERT_TO_TEXT_FILE_STOP_TIME18 - CLOCK;
if SLACK_TIME >= 0.0 then
  delay (SLACK_TIME);
else
  raise CONVERT_TO_TEXT_FILE_TIMING_ERROR;
end if;
delay
    (START_OF_PERIOD + CONVERT_TO_TEXT_FILE_STOP_TIME18 - CLOCK);

PREPARE_PERIODIC_REPORT_DRIVER;
SLACK_TIME :=
   START_OF_PERIOD + PREPARE_PERIODIC_REPORT_STOP_TIME19 - CLOCK;
if SLACK_TIME >= 0.0 then
  delay (SLACK_TIME);
else
  raise PREPARE_PERIODIC_REPORT_TIMING_ERROR;
end if;
START_OF_PERIOD := START_OF_PERIOD + PERIOD;
delay (START_OF_PERIOD - clock);
```

```
exception
   when PREPARE_PERIODIC_REPORT_TIMING_ERROR =>
     PUT_LINE
      ("timing error from operator PREPARE_PERIODIC_REPORT");
     START_OF_PERIOD := clock;
   when CONVERT_TO_TEXT_FILE_TIMING_ERROR =>
     PUT_LINE("timing error from operator CONVERT_TO_TEXT_FILE");
     START_OF_PERIOD := clock;
   when FORWARD_FOR_TRANSMISSION_TIMING_ERROR =>
     PUT_LINE
      ("timing error from operator FORWARD_FOR_TRANSMISSION");
     START_OF_PERIOD := clock;
   when MAKE_ROUTING_TIMING_ERROR =>
     PUT_LINE("timing error from operator MAKE_ROUTING");
     START_OF_PERIOD := clock;
   when WEAPONS_SYSTEMS_TIMING_ERROR =>
     PUT_LINE("timing error from operator WEAPONS_SYSTEMS");
     START_OF_PERIOD := clock;
   when WEAPONS_INTERFACE_TIMING_ERROR =>
     PUT_LINE("timing error from operator WEAPONS_INTEPFACE");
     START_OF_PERIOD := clock;
   when MONITOR_OWNSHIP_POSITION_TIMING_ERROR =>
     PUT_LINE
      ("timing error from operator MONITOR_OWNSHIP_POSITION");
     START_OF_PERIOD := clock;
   when NAVIGATION_SYSTEM_TIMING_ERROR =>
     PUT_LINE("timing error from operator NAVIGATION_SYSTEM");
     START_OF_PERIOD := clock;
   when ADD_SENSOR_TRACK_TIMING_ERROR =>
     PUT_LINE("timing error from operator ADD_SENSOR_TRACK");
     START_OF_PERIOD := clock;
   when FILTER_SENSOR_TRACKS_TIMING_ERROR =>
     PUT_LINE("timing error from operator FILTER_SENSOR_TRACKS");
     START_OF_PERIOD := clock;
   when PREPARE_SENSOR_TRACK_TIMING_ERROR =>
     PUT_LINE("timing error from operator PREPARE_SENSOR_TRACK");
     START_OF_PERIOD := clock;
   when ANALYZE_SENSOR_DATA_TIMING_ERROR =>
     PUT_LINE("timing error from operator ANALYZE_SENSOR_DATA");
     START_OF_PERIOD := clock;
   when SENSORS_TIMING_ERROR =>
```

```
            PUT_LINE("timing error from operator SENSORS");
            START_OF_PERIOD := clock;
          when ADD_COMMS_TRACK_TIMING_ERROR =>
            PUT_LINE("timing error from operator ADD_COMMS_TRACK");
            START_OF_PERIOD := clock;
          when FILTER_COMMS_TRACKS_TIMING_ERROR =>
            PUT_LINE("timing error from operator FILTER_COMMS_TRACKS");
            START_OF_PERIOD := clock;
          when EXTRACT_TRACKS_TIMING_ERROR =>
            PUT_LINE("timing error from operator EXTRACT_TRACKS");
            START_OF_PERIOD := clock;
          when DECIDE_FOR_ARCHIVING_TIMING_ERROR =>
            PUT_LINE("timing error from operator DECIDE_FOR_ARCHIVING");
            START_OF_PERIOD := clock;
          when PARSE_INPUT_FILE_TIMING_ERROR =>
            PUT_LINE("timing error from operator PARSE_INPUT_FILE");
            START_OF_PERIOD := clock;
          when COMMS_LINKS_TIMING_ERROR =>
            PUT_LINE("timing error from operator COMMS_LINKS");
            START_OF_PERIOD := clock;
        end;
      end loop;
    end SCHEDULE_TYPE;

begin
  null;
end STATIC_SCHEDULE;
```

APPENDIX E

DYNAMIC SCHEDULE

DS.A

This appendix contains the dynamic schedule package, ds.a. The dynamic scheduler generates this code automatically for the non-time critical operators in the prototype.

```
with TL; use TL;
with PRIORITY_DEFINITIONS; use PRIORITY_DEFINITIONS;
package DS_PACKAGE is
  task type DYNAMIC_SCHEDULE_TYPE is
    pragma priority (DYNAMIC_SCHEDULE_PRIORITY);
  end DYNAMIC_SCHEDULE_TYPE;
  for DYNAMIC_SCHEDULE_TYPE'STORAGE_SIZE use 100_000;
  DYNAMIC_SCHEDULE : DYNAMIC_SCHEDULE_TYPE;
end DS_PACKAGE;

package body DS_PACKAGE is
  task body DYNAMIC_SCHEDULE_TYPE is
  begin
    delay (1.0);
    loop
      MANAGE_USER_INTERFACE_DRIVER;
      DISPLAY_TRACKS_DRIVER;
      GET_USER_INPUTS_DRIVER;
      STATUS_SCREEN_DRIVER;
      EMERGENCY_STATUS_SCREEN_DRIVER;
      MESSAGE_EDITOR_DRIVER;
      MESSAGE_ARRIVAL_PANEL_DRIVER;
      null;
```

```
    end loop;
  end DYNAMIC_SCHEDULE_TYPE;
end DS_PACKAGE;
```

APPENDIX F

SOFTWARE BASE

SB.A

This appendix includes the set of reusable Ada components for C^3I workstation prototype. An index is provided at the end of this appendix for quick referencing.

```
package SB is

--************  TYPE DECLARATIONS ************************************

  TRACK_CAPACITY : constant NATURAL := 1024;
  OWN_ADDRESS    : constant STRING(1..6) := "NAME_0";

  subtype FILENAME is STRING(1..10);
  type SECURITY_CLASS is ( U, C, S, TS );
  type PRECEDENCE_CLASS is ( R, P, O, Z );
  type ADDRESS_TYPE;
  type ADDRESS_LINK is access ADDRESS_TYPE;
  type ADDRESS_TYPE is record
    NAME : STRING(1..6):= "      ";
    NEXT : ADDRESS_LINK:= null;
  end record;
  type VIA_RECORD;
  type VIA_RECORD_LINK is access VIA_RECORD;
  type VIA_RECORD is record
    RELAY_BY : STRING(1..6) := "      ";
    RELAY_TO : ADDRESS_LINK := null;
    NEXT     : VIA_RECORD_LINK := null;
  end record;
  type HEADER_FORMAT is record
    CLASSIFICATION : SECURITY_CLASS := U;
    PRECEDENCE     : PRECEDENCE_CLASS := R;
    ORIGIN         : STRING(1..6) := "NAME_0";
```

```
  ADDRESS         : ADDRESS_LINK := null;
  INFO            : ADDRESS_LINK := null;
  VIA_LINE        : VIA_RECORD_LINK := null;
  SUBJECT         : STRING(1..60)
   := "                                                              ";
end record;
type LINKS_TYPE is ( JTIDS, LINK11, LINK16, OTCIXS );
subtype TEXT_STRING is STRING(1..2400);
type TEXT_RECORD is record
  NAME     : FILENAME := "          ";
  HEADER   : HEADER_FORMAT;
  LINK_ID  : LINKS_TYPE := LINK11;
  RELAYED  : BOOLEAN := false;
  ARCHIVE  : BOOLEAN := true;
  IS_TRACK: BOOLEAN := false;
  FORMAT   : STRING(1..6) := "      ";
  TEXT     : TEXT_STRING;
end record;
type ARCHIVE_SETUP is record
  ALL_SHIPS : BOOLEAN := true;
  OWNSHIP   : BOOLEAN := true;
  JTIDS     : BOOLEAN := true;
  LINK16    : BOOLEAN := true;
  LINK11    : BOOLEAN := true;
  OTCIXS    : BOOLEAN := true;
end record;
type TIME is record
  HOURS       : NATURAL range 0..23;
  MINUTES     : NATURAL range 0..59;
  SECONDS     : NATURAL range 0..59;
  MILISECONDS : NATURAL range 0..99;
end record;
type TRACK_CLASS_TYPE is ( AIR, SURFACE, SUBSURFACE );
type IFF_CLASS_TYPE is   ( FRIENDLY, HOSTILE, NEUTRAL, UNKNOWN );
type ARCHIVE_CLASS is    ( C, N, A, S );
OWNSHIP_IFF_CLASS   : constant IFF_CLASS_TYPE := FRIENDLY;
OWNSHIP_TRACK_CLASS : constant TRACK_CLASS_TYPE := SURFACE;
type TRACK_RECORD;
type TRACK_TUPLE is access TRACK_RECORD;
type TRACK_RECORD is record
  ID               : NATURAL range 0..TRACK_CAPACITY;
```

```
  OBSERVER          : STRING(1..8);
  OBSERVATION_TIME  : TIME;
  TRACK_CLASS       : TRACK_CLASS_TYPE;
  IFF_CLASS         : IFF_CLASS_TYPE;
  LATITUDE          : FLOAT range -90.0..90.0;
  LONGITUDE         : FLOAT range -180.0..180.0;
  ALTITUDE          : FLOAT range -10000.0..99999.9;
  COURSE            : FLOAT range 0.0..360.0;
  VELOCITY          : FLOAT; --in knots
  THE_RANGE         : FLOAT range 0.0..9999.99; --in miles
  ARCHIVE_FLAG      : ARCHIVE_CLASS;
  NEXT              : TRACK_TUPLE;
end record;
type ADD_TRACK_TUPLE is record
  ORIGIN : STRING(1..8);
  TRACK  : TRACK_TUPLE;
end record;
type LOCAL_TRACK_INFO is record
  ORIGIN       : STRING(1..8);
  ID           : NATURAL range 0..TRACK_CAPACITY;
  THE_TIME     : TIME;
  AZIMUTH      : FLOAT range -180.0 .. 180.0;
  ELEVATION    : FLOAT range -90.0  .. 90.0 := 0.0;
  THE_RANGE    : FLOAT := 0.0; --in miles
  VELOCITY     : FLOAT := 0.0; --in knots
  COURSE       : FLOAT range 0.0..360.0;
  IFF_CLASS    : IFF_CLASS_TYPE;
  TRACK_CLASS  : TRACK_CLASS_TYPE;
  ARCHIVE_FLAG : ARCHIVE_CLASS;
end record;
type SENSOR_RECORD is record
  INTELLIGENCE : STRING(1..80);
  CONTACT      : LOCAL_TRACK_INFO;
end record;
type OWNSHIP_NAVIGATION_INFO is record
  COURSE    : FLOAT range 0.0..360.0;
  VELOCITY  : FLOAT; --in knots;
  LATITUDE  : FLOAT range -90.0..90.0 := 35.0;
  LONGITUDE : FLOAT range -180.0..180.0 := 125.0;
  THE_TIME  : TIME;
end record;
```

```
type DESIRED_CLASS_ARRAY is array ( TRACK_CLASS_TYPE ) of BOOLEAN;
type DESIRED_RANGE_ARRAY is array ( TRACK_CLASS_TYPE ) of FLOAT;
type SET_TRACK_FILTER is record
  MAX_NUMBER    : NATURAL range 0..TRACK_CAPACITY := TRACK_CAPACITY;
  DESIRED_CLASS : DESIRED_CLASS_ARRAY := ( true, true, true );
  DESIRED_RANGE : DESIRED_RANGE_ARRAY := ( 10000.0,10000.0,10000.0 );
end record;
type UPDATE_TRACK_TUPLE is record
  ORIGIN  : STRING(1..8);
  TRACK   : TRACK_TUPLE;
end record;
type REQUEST_TYPE is ( ADD, DELETE, UPDATE );
type CHANGE_DATABASE_REQUEST is record
  ORIGIN  : STRING(1..8);
  REQUEST : REQUEST_TYPE;
  TRACK   : TRACK_TUPLE;
end record;
type TRACK_CLASS_ARRAY is array(TRACK_CLASS_TYPE) of BOOLEAN;
type IFF_CLASS_ARRAY is array(IFF_CLASS_TYPE) of BOOLEAN;
type DATABASE_REQUEST is record
  THE_RANGE   : FLOAT := 10000.0;
  TRACK_CLASS : TRACK_CLASS_ARRAY := ( true, true, true );
  IFF_CLASS   : IFF_CLASS_ARRAY   := ( true, true, true, true );
end record;
type INITIATE_TRANSMISSION_SEQUENCE is record
  LINK_ID        : LINKS_TYPE := JTIDS;
  HEADER         : HEADER_FORMAT;
  DESIRED_FORMAT : STRING(1..6) := "      ";
  DBASE_REQUEST  : DATABASE_REQUEST;
end record;
type WEAPONS_TYPE is ( CIWS, GUN, TWS, MK48 );
type WEAPON_STATUS_TYPE is(
   DAMAGED, RELOADING, LAUNCHING,   READY,    SERVICE_REQUIRED,
   SLEWING, SECURED,   MAINTANENCE, ENGAGING, OUT_OF_AMMUNITION );
type WEAPON_STATUS is record
  SYS_TYPE : WEAPONS_TYPE;
  STATUS   : WEAPON_STATUS_TYPE;
end record;
type WEAPON_STATUS_REPORT is array ( WEAPONS_TYPE ) of
                                        WEAPON_STATUS_TYPE;
type TRANSMIT_RECORD is record
```

```
    ROUTE_ADDR : ADDRESS_LINK;
    ROUTED     : BOOLEAN := false;
    FULL       : BOOLEAN := false;
    TEXT       : TEXT_RECORD;
  end record;
  type TRANSMIT_COMMAND is array ( LINKS_TYPE ) of TRANSMIT_RECORD;
  type NETWORK_SETUP is array ( LINKS_TYPE ) of ADDRESS_LINK;
  type TRANSMISSION_RECORD is record
    ROUTE_ADDR : ADDRESS_LINK;
    FULL       : BOOLEAN := false;
    TEXT       : TEXT_RECORD;
  end record;
  type TRANSMISSION_COMMAND is array ( LINKS_TYPE ) of
                                          TRANSMISSION_RECORD;
  type EMISSIONS_CONTROL_COMMAND is ( EMCON, UNRESTRICTED );
  type MESSAGE_RECORD;
  type MESSAGE_LIST is access MESSAGE_RECORD;
  type MESSAGE_RECORD is record
    LINK_ID    : LINKS_TYPE;
    MAIN       : TRANSMISSION_RECORD;
    NEXT       : MESSAGE_LIST;
  end record;


-- ******************** PROCEDURE DECLARATIONS ********************


  procedure COMMS_LINKS(
          INPUT_LINK_MESSAGE : out FILENAME );

  procedure PARSE_INPUT_FILE(
                INPUT_LINK_MESSAGE : in     FILENAME;
          INPUT_TEXT_RECORD  : in out TEXT_RECORD ) ;

  procedure DECIDE_FOR_ARCHIVING(
          INPUT_TEXT_RECORD : in  out TEXT_RECORD;
          TDD_ARCHIVE_SETUP : in      ARCHIVE_SETUP;
          COMMS_TEXT_FILE   : out     TEXT_RECORD;
          COMMS_EMAIL       : out     FILENAME      );

  procedure EXTRACT_TRACKS(
```

```
        COMMS_TEXT_FILE : in  TEXT_RECORD;
        COMMS_ADD_TRACK : out ADD_TRACK_TUPLE );

procedure FILTER_COMMS_TRACKS(
        COMMS_ADD_TRACK      : in out ADD_TRACK_TUPLE;
        TDD_FILTER           : in     SET_TRACK_FILTER;
        FILTERED_COMMS_TRACK : out    ADD_TRACK_TUPLE );

procedure ADD_COMMS_TRACK(
        FILTERED_COMMS_TRACK : in     ADD_TRACK_TUPLE;
        TDD_FILTER           : in     SET_TRACK_FILTER;
        OUT_TRACKS           : in out TRACK_TUPLE );


procedure SENSORS(
        SENSOR_DATA : out SENSOR_RECORD );

procedure ANALYZE_SENSOR_DATA(
        SENSOR_DATA         : in  SENSOR_RECORD;
        SENSOR_CONTACT_DATA : out LOCAL_TRACK_INFO );

procedure PREPARE_SENSOR_TRACK(
        SENSOR_CONTACT_DATA : in  LOCAL_TRACK_INFO;
        POSITION_DATA       : in  OWNSHIP_NAVIGATION_INFO;
        SENSOR_ADD_TRACK    : out ADD_TRACK_TUPLE );

procedure FILTER_SENSOR_TRACKS(
        SENSOR_ADD_TRACK      : in  out ADD_TRACK_TUPLE;
        TDD_FILTER            : in      SET_TRACK_FILTER;
        FILTERED_SENSOR_TRACK : out     ADD_TRACK_TUPLE );

procedure ADD_SENSOR_TRACK(
        FILTERED_SENSOR_TRACK  : in     ADD_TRACK_TUPLE;
        TDD_FILTER             : in     SET_TRACK_FILTER;
        OUT_TRACKS             : in out TRACK_TUPLE );

procedure NAVIGATION_SYSTEM(
        POSITION_DATA : out OWNSHIP_NAVIGATION_INFO );

procedure MONITOR_OWNSHIP_POSITION(
        POSITION_DATA : in     OWNSHIP_NAVIGATION_INFO;
```

```
        OUT_TRACKS     : in out TRACK_TUPLE );

procedure DISPLAY_TRACKS(
        OUT_TRACKS        : in TRACK_TUPLE;
        TD_TRACK_REQUEST  : in DATABASE_REQUEST );

procedure GET_USER_INPUTS(
        TDD_ARCHIVE_SETUP     : out ARCHIVE_SETUP;
        TDD_FILTER            : out SET_TRACK_FILTER;
        TD_TRACK_REQUEST      : out DATABASE_REQUEST;
        TCD_STATUS_QUERY      : out BOOLEAN;
        TCD_NETWORK_SETUP     : out NETWORK_SETUP;
        TCD_EMISSION_CONTROL  : out EMISSIONS_CONTROL_COMMAND;
        EDITOR_SELECTED       : out BOOLEAN;
        INITIATE_TRANS        : out INITIATE_TRANSMISSION_SEQUENCE;
        TERMINATE_TRANS       : out BOOLEAN                      );

procedure MANAGE_USER_INTERFACE;

procedure WEAPONS_SYSTEMS(
        WEAPON_STATUS_DATA : out WEAPON_STATUS );

procedure WEAPONS_INTERFACE(
        WEAPON_STATUS_DATA : in WEAPON_STATUS;
        WEAPONS_EMREP      : out WEAPON_STATUS_REPORT;
        WEAPONS_STATREP    : out WEAPON_STATUS_REPORT );

procedure STATUS_SCREEN(
        WEAPONS_STATREP    : in out WEAPON_STATUS_REPORT;
        TCD_STATUS_QUERY   : in     BOOLEAN              );

procedure EMERGENCY_STATUS_SCREEN(
        WEAPONS_EMREP      : in out WEAPON_STATUS_REPORT ) ;

procedure MAKE_ROUTING(
        TCD_TRANSMIT_COMMAND : in     TRANSMIT_COMMAND;
        TCD_NETWORK_SETUP    : in     NETWORK_SETUP;
        TRANSMISSION_MESSAGE : in out TRANSMISSION_COMMAND );

procedure MESSAGE_EDITOR(
        EDITOR_SELECTED      : in  BOOLEAN;
```

```
        TCD_TRANSMIT_COMMAND : out TRANSMIT_COMMAND );

  procedure FORWARD_FOR_TRANSMISSION(
        TRANSMISSION_MESSAGE : in     TRANSMISSION_COMMAND;
        TCD_EMISSION_CONTROL : in     EMISSIONS_CONTROL_COMMAND;
        OUTPUT_MESSAGES      : in out MESSAGE_LIST              );

  procedure CONVERT_TO_TEXT_FILE(
        OUTPUT_MESSAGES      : in  MESSAGE_LIST);

  procedure MESSAGE_ARRIVAL_PANEL(
        COMMS_EMAIL          : in  FILENAME );

  procedure PREPARE_PERIODIC_REPORT(
              INITIATE_TRANS       : in
                                     INITIATE_TRANSMISSION_SEQUENCE;
              TERMINATE_TRANS      : in  BOOLEAN;
              TCD_TRANSMIT_COMMAND : in out TRANSMIT_COMMAND );

end SB;

with TEXT_IO; use TEXT_IO;
with MATH;    use MATH;
with TAE;     use TAE;
with CALENDAR, RANDOM, X_WINDOWS;


package body SB is

  USE TAE.TAE_MISC;
  USE TAE.TAE_WPT;
  USE TAE.TAE_VM;
  USE TAE.TAE_CO;
  package enum_io is new enumeration_io( boolean ); use enum_io;


  --************************************************************************
  --***common global variable&type declarations                          ***
  --************************************************************************
  --number of characters in text body of separate records
  TEXT_LENGTH : constant NATURAL := 2400;
```

```
--************************************************************************
--*** procedure COMMS_LINKS global variable&type declarations      ***
--************************************************************************
COMMS_TRACK_CAPACITY : constant NATURAL := 10;
type COMMS_TRACK_RECORD;
type COMMS_TRACK_LIST is access COMMS_TRACK_RECORD;
type COMMS_TRACK_RECORD is record
  ID           : NATURAL range 1 .. COMMS_TRACK_CAPACITY;
  ORIGIN       : STRING(1..8);
  THE_TIME     : TIME;
  AZIMUTH      : FLOAT range -180.0 .. 180.0;
  ELEVATION    : FLOAT range -90.0 .. 90.0;
  THE_RANGE    : FLOAT;
  VELOCITY     : FLOAT;
  COURSE       : FLOAT range 0.0 .. 360.0;
  IFF_CLASS    : IFF_CLASS_TYPE;
  TRACK_CLASS  : TRACK_CLASS_TYPE;
  ARCHIVE_FLAG : ARCHIVE_CLASS;
  LATITUDE     : FLOAT range -90.0 .. 90.0;
  LONGITUDE    : FLOAT range -180.0 .. 180.0;
  ALTITUDE     : FLOAT range -10000.0 .. 99999.99;
  REPORTED     : BOOLEAN := false;
  NEXT         : COMMS_TRACK_LIST;
end record;
COMMS_TRACKS   : COMMS_TRACK_LIST;
COMMS_TRACKS_ARE_CREATED : BOOLEAN := false;

--************************************************************************
--*** procedure SENSORS                                                ***
--*** variable&type declarations                                       ***
--************************************************************************
type SENSOR_TRACK_RECORD;
type SENSOR_TRACK_LIST is access SENSOR_TRACK_RECORD;
type SENSOR_TRACK_RECORD is record
  INTELLIGENCE : STRING(1..80); --contains intelligence report
  CONTACT      : LOCAL_TRACK_INFO;
  LATITUDE     : FLOAT range -90.0 .. 90.0;
  LONGITUDE    : FLOAT range -180.0 .. 180.0;
  NEXT         : SENSOR_TRACK_LIST;
end record;
SENSOR_TRACKS  : SENSOR_TRACK_LIST;
```

```
SENSOR_TRACKS_ARE_CREATED : BOOLEAN := false;

--*************************************************************************
--*** procedure NAVIGATION_SYSTEM                                       ***
--*** global variable&type declarations                                 ***
--*************************************************************************
START_TIME : NATURAL := 0;

--*************************************************************************
--*** procedure FILTER_TRACKS                                           ***
--*** global variable&type declarations                                 ***
--*************************************************************************
APPROVED : BOOLEAN := true;

--*************************************************************************
--*** procedure PREPARE_PERIODIC_REPORT                                 ***
--*** global variable&type declarations                                 ***
--*************************************************************************
TRACK_FILE_NAME : constant FILENAME := "TRACK_FILE";

--*************************************************************************
--*** procedure MODIFY_TRACKS                                           ***
--*** global variable&type declarations                                 ***
--*************************************************************************
NUMBER_OF_TRACKS : NATURAL := 0;
TRACKS           : TRACK_TUPLE := null;

--*************************************************************************
--*** procedure WEAPONS_INTERFACE                                       ***
--*** global variable&type declarations                                 ***
--*************************************************************************
CIWS_STATUS : WEAPON_STATUS_TYPE := READY;
GUN_STATUS  : WEAPON_STATUS_TYPE := READY;
TWS_STATUS  : WEAPON_STATUS_TYPE := READY;
MK48_STATUS : WEAPON_STATUS_TYPE := READY;

--*************************************************************************
--*** procedure FORWARD_FOR_TRANSMISSION                                ***
--*** global variable&type declarations                                 ***
--*************************************************************************
WAITING_MESSAGES : MESSAGE_LIST;
```

```
--*************************************************************************
--*** procedures DISPLAY_TRACKS & GET_USER_INPUTS &                     ***
--*** MANAGE_USER_INTERFACE                                             ***
--*** global variable&type declarations                                 ***
--*************************************************************************
TAE_SOURCE_FILE : constant STRING(1..8) := "user.res";
--tae panel pointers
main_info     : event_context_ptr;
filter_info   : event_context_ptr;
archive_info  : event_context_ptr;
contrnts_info : event_context_ptr;
display_info  : event_context_ptr;
weapon_info   : event_context_ptr;
emerg_info    : event_context_ptr;
network_info  : event_context_ptr;
emcon_info    : event_context_ptr;
editor_info   : event_context_ptr;
messcame_info : event_context_ptr;
messdisp_info : event_context_ptr;
messret_info  : event_context_ptr;
trackrep_info : event_context_ptr;
tertrrep_info : event_context_ptr;

--values of these variables are changed globaly by different
--procedures, and these variables are used by
--procedure GET_USER_INPUTS when it is triggered.
new_archive_setup    : archive_setup;
new_track_filter     : set_track_filter;
new_track_request    : database_request;
new_status_query     : boolean;
new_emission_control : emissions_control_command := EMCON;
new_network_setup    : network_setup;
new_transmit_command : transmit_command;
new_editor_selected  : boolean := false;
new_initiate_trans   : initiate_transmission_sequence;
new_terminate_trans  : boolean := true;


--old values for user inputs. these variables are used to restore
--the old values of each variable in case that some components of
--the variable are begun to be changed, but user has changed
```

```
--his/her mind then. For Example, user opened the panel and changed
--some components of a record variable, but then requested
--CANCEL in the panel.
old_archive_setup    : archive_setup;
old_track_filter     : set_track_filter;
old_track_request    : database_request;
old_emission_control : emissions_control_command := EMCON;
old_network_setup    : network_setup;
old_transmit_command : transmit_command;
old_initiate_trans   : initiate_transmission_sequence;

--shows if initial configuration is already set
panels_are_initialized : boolean := false;
--shows if related panel is already displayed
track_display_panel_is_displayed          : boolean := false;
weapon_status_panel_is_displayed          : boolean := false;
emergency_weapon_status_panel_is_displayed : boolean := false;
--X Window display pointer
theDisplay : X_Windows.Display;
--main program pointer
user_ptr : event_context_ptr;
--event data types & pointers
etype    : wpt_eventtype;
wptEvent : wpt_eventptr;
--used by the manage_user_interface
value : array (1..1) of string (1..tae_taeconf.STRINGSIZE);


--************************************************************************
--*** Procedure COMMS_LINKS                                            ***
--************************************************************************
procedure COMMS_LINKS( INPUT_LINK_MESSAGE : out FILENAME ) is

  package P_CLASS_IO is new ENUMERATION_IO( PRECEDENCE_CLASS );
  use P_CLASS_IO;
  package S_CLASS_IO is new ENUMERATION_IO( SECURITY_CLASS );
  use S_CLASS_IO;
  package LINKS_IO   is new ENUMERATION_IO( LINKS_TYPE );
  use LINKS_IO;
  package INT_IO     is new INTEGER_IO( INTEGER );
  use INT_IO;
```

```
package FLT_IO     is new FLOAT_IO( FLOAT );
use FLT_IO;

PI                  : constant FLOAT := 3.1415926536;
RADIANS_PER_DEGREE  : constant FLOAT := PI / 180.0;
DEGREES_PER_RADIAN  : constant FLOAT := 1.0 / RADIANS_PER_DEGREE;
SECONDS_SINCE_MIDNIGHT : NATURAL;
--name of the origin of the simulated message
FROM_NAME : STRING(1..6);
--physical file for the file which to be created and filled
--with message
FILE_NAME : FILENAME;
--logical file variable for the file which to be created and
--filled with message
FILE      : TEXT_IO.FILE_TYPE;
--declare types and variables about name of the current stations
--in the environment
NUMBER_OF_ADDRESSES : constant NATURAL := 10;
type NAME_ARRAY_RECORD is record
  NAME    : STRING(1..6);
  USED    : BOOLEAN := false;
  REUSED  : BOOLEAN := false;
end record;
type NAME_ARRAY is ARRAY(0..NUMBER_OF_ADDRESSES-1) of
                                        NAME_ARRAY_RECORD;
NAMES     : NAME_ARRAY;
--link id of the message
LINK_ID        : LINKS_TYPE;
--float numbers which are used to store the last RANDOM.NUMBER
--for the case same random value is needed more than once
TO_NUMBER      : FLOAT;
INFO_NUMBER    : FLOAT;
RANDOM_NUMBER  : FLOAT;


--************************************************************************
--*** Function CHARACTER_OF                                            ***
--************************************************************************
function CHARACTER_OF( N : in NATURAL ) return CHARACTER is
begin
  return( CHARACTER'VAL( N + 48 ) );
```

```
end CHARACTER_OF;


--*********************************************************************
--*** FUNCTION GET_FILE_NAME                                        ***
--*** Algorithm : creates a file name by using current time         ***
--***             and returns it                                   ***
--*********************************************************************
function GET_FILE_NAME return FILENAME is
  YEAR      : CALENDAR.YEAR_NUMBER;
  MONTH     : CALENDAR.MONTH_NUMBER;
  DAY       : CALENDAR.DAY_NUMBER;
  HOURS     : NATURAL;
  MINUTES   : NATURAL;
  SECONDS   : CALENDAR.DAY_DURATION;
  FILE_NAME : FILENAME;


--*********************************************************************
--*** Function NATURAL_TO_STRING_2                                  ***
--*** Algorithm : converts 2-digit natural number to                ***
--***             2-character string                               ***
--*********************************************************************
  function NATURAL_TO_STRING_2( NUMBER : in NATURAL )
                                          return STRING is
    N : NATURAL      := NUMBER;
    S : STRING(1..2) := "00";
  begin
    if N > 9 then
      S(1) := CHARACTER_OF( N/10 );
      N := N - ( (N/10) * 10 );
    end if;
    S(2) := CHARACTER_OF( N );
    return( S );
  end NATURAL_TO_STRING_2;


begin --GET_FILE_NAME;
  CALENDAR.SPLIT( CALENDAR.CLOCK, YEAR, MONTH, DAY, SECONDS );
  HOURS   :=  INTEGER(SECONDS) / 3600;
  MINUTES := (INTEGER(SECONDS) mod 3600) / 60;
```

```
  FILE_NAME(1..2)  := "c_";
  FILE_NAME(3..4)  := NATURAL_TO_STRING_2(NATURAL(DAY           ) );
  FILE_NAME(5..6)  := NATURAL_TO_STRING_2(NATURAL(HOURS         ) );
  FILE_NAME(7..8)  := NATURAL_TO_STRING_2(NATURAL(MINUTES       ) );
  FILE_NAME(9..10) := NATURAL_TO_STRING_2(NATURAL(SECONDS) mod 60 );
  return( FILE_NAME );
end GET_FILE_NAME;


--*********************************************************************
--*** Procedure CREATE_COMMS_TRACKS                                 ***
--*** Algorithm : creates and initializes all of the                ***
--***              comms tracks                                     ***
--*********************************************************************
procedure CREATE_COMMS_TRACKS(
          COMMS_TRACKS           : in out COMMS_TRACK_LIST;
          SECONDS_SINCE_MIDNIGHT : in     NATURAL ) is

  NEW_TRACK     : COMMS_TRACK_LIST;
  RANDOM_NUMBER : FLOAT;
  --track information are reported by using 10 tracks
  COMMS_TRACK_CAPACITY : constant NATURAL := 10;
  --constant attributes for the tracks
  MIN_AIR_HEIGHT              : constant FLOAT := 3000.0;
  MIN_AIR_VELOCITY            : constant FLOAT := 300.0;
  MAX_AIR_VELOCITY            : constant FLOAT := 800.0;
  AIR_VELOCITY_DIFF           : constant FLOAT
                        := MAX_AIR_VELOCITY - MIN_AIR_VELOCITY;
  MIN_SURFACE_VELOCITY        : constant FLOAT := 0.0;
  MAX_SURFACE_VELOCITY        : constant FLOAT := 40.0;
  MAX_HEADING_ANGLE           : constant FLOAT := 360.0;
  MAX_COMMS_RANGE             : constant FLOAT := 500.0;
  MIN_SUBSURFACE_DEPTH        : constant FLOAT := 0.0;
  MIN_SUBSURFACE_VELOCITY     : constant FLOAT := 0.0;
  MAX_SUBSURFACE_VELOCITY     : constant FLOAT := 30.0;
  MIN_ELEVATION               : constant FLOAT := -90.0;
  MAX_ELEVATION               : constant FLOAT :=  90.0;
  MIN_AZIMUTH                 : constant FLOAT := -180.0;
  MAX_AZIMUTH                 : constant FLOAT :=  180.0;
  MAX_BEARING_ANGLE           : constant FLOAT
                        := MAX_AZIMUTH - MIN_AZIMUTH;
```

```
  MIN_ALTITUDE                 : constant FLOAT := -10000.0;
  MAX_ALTITUDE                 : constant FLOAT :=  99999.99;
  AIR_TRACK_REBIRTH_RATIO      : constant FLOAT := 0.2;
  SURFACE_TRACK_REBIRTH_RATIO  : constant FLOAT := 0.8;
  MILES_PER_DEGREE             : constant FLOAT := 60.0;

begin --CREATE_COMMS_TRACKS
  for I in 1..COMMS_TRACK_CAPACITY loop
    NEW_TRACK      := new COMMS_TRACK_RECORD;
    NEW_TRACK.ID   := I;
    RANDOM_NUMBER := RANDOM.NUMBER;

    if RANDOM_NUMBER < AIR_TRACK_REBIRTH_RATIO then
      --create air track
      NEW_TRACK.TRACK_CLASS := AIR;
      NEW_TRACK.ELEVATION   := MAX_ELEVATION * RANDOM.NUMBER;
      NEW_TRACK.VELOCITY    := MIN_AIR_VELOCITY +
       ( RANDOM.NUMBER*AIR_VELOCITY_DIFF );
      NEW_TRACK.ALTITUDE    := MAX_ALTITUDE * RANDOM.NUMBER;

    elsif RANDOM_NUMBER < SURFACE_TRACK_REBIRTH_RATIO then
      --create surface track
      NEW_TRACK.TRACK_CLASS := SURFACE;
      NEW_TRACK.ELEVATION   := 0.0;
      NEW_TRACK.VELOCITY    := MIN_SURFACE_VELOCITY +
       ( RANDOM.NUMBER * MAX_SURFACE_VELOCITY );
      NEW_TRACK.ALTITUDE    := 0.0;

    else
      --create subsurface track
      NEW_TRACK.TRACK_CLASS := SUBSURFACE;
      NEW_TRACK.ELEVATION   := MIN_ELEVATION * RANDOM.NUMBER ;
      NEW_TRACK.VELOCITY    := MIN_SUBSURFACE_VELOCITY +
       ( RANDOM.NUMBER * MAX_SUBSURFACE_VELOCITY );
      NEW_TRACK.ALTITUDE    := MIN_ALTITUDE * RANDOM.NUMBER;
    end if;

    NEW_TRACK.COURSE    := RANDOM.NUMBER * MAX_HEADING_ANGLE;
    NEW_TRACK.THE_RANGE := RANDOM.NUMBER * MAX_COMMS_RANGE;
    NEW_TRACK.AZIMUTH   := MIN_AZIMUTH +
       ( RANDOM.NUMBER * MAX_BEARING_ANGLE );
```

```
    NEW_TRACK.ORIGIN    := "COMMS   ";
    NEW_TRACK.THE_TIME.HOURS   := SECONDS_SINCE_MIDNIGHT / 3600;
    NEW_TRACK.THE_TIME.MINUTES :=
       (SECONDS_SINCE_MIDNIGHT mod 3600) / 60;
    NEW_TRACK.THE_TIME.SECONDS := SECONDS_SINCE_MIDNIGHT mod 60;
    RANDOM_NUMBER := RANDOM.NUMBER;
    if    RANDOM_NUMBER <0.25 then  NEW_TRACK.IFF_CLASS := FRIENDLY;
    elsif RANDOM_NUMBER <0.50 then  NEW_TRACK.IFF_CLASS := HOSTILE;
    elsif RANDOM_NUMBER <0.75 then  NEW_TRACK.IFF_CLASS := NEUTRAL;
    else                            NEW_TRACK.IFF_CLASS := UNKNOWN;
    end if;
    NEW_TRACK.LATITUDE := 35.0 + (NEW_TRACK.THE_RANGE *
       COS(NEW_TRACK.AZIMUTH*RADIANS_PER_DEGREE))/
       MILES_PER_DEGREE;
    if NEW_TRACK.AZIMUTH > 0.0 then
      NEW_TRACK.LONGITUDE := 125.0 + (NEW_TRACK.THE_RANGE *
       SIN(NEW_TRACK.AZIMUTH*RADIANS_PER_DEGREE))/
       MILES_PER_DEGREE;
    else
      NEW_TRACK.LONGITUDE := 125.0 - (NEW_TRACK.THE_RANGE *
       SIN(NEW_TRACK.AZIMUTH*RADIANS_PER_DEGREE))/
       MILES_PER_DEGREE;
    end if;
    NEW_TRACK.NEXT := COMMS_TRACKS;
    COMMS_TRACKS   := NEW_TRACK;
  end loop;
end CREATE_COMMS_TRACKS;


--*********************************************************************
--*** Procedure UPDATE_COMMS_TRACK                                  ***
--*** Algorithm : Updates the position of the track                 ***
--***              according to its previous position, course       ***
--***              & velocity                                       ***
--*********************************************************************
procedure UPDATE_COMMS_TRACK(
           SELECTED_TRACK         : in out COMMS_TRACK_LIST;
           SECONDS_SINCE_MIDNIGHT : in     NATURAL ) is

  PREVIOUS_SECONDS_SINCE_MIDNIGHT : NATURAL;
  SECONDS_PASSED                  : NATURAL;
```

```
   MILES_PER_DEGREE                  : constant FLOAT := 60.0;

   --ownship information variables
   OWN_LAT       : FLOAT range -90.0 .. 90.0 := 35.0;
   OWN_LONG      : constant FLOAT := 125.0;
   OWN_COURSE    : FLOAT := 0.0;
   OWN_VELOCITY : FLOAT := 25.0;

   --distance between current & previous position of the track
   DISTANCE     : FLOAT;

   --latitude & longitude distances between current & previous
   --position of the track
   LAT_DIST     : FLOAT;
   LONG_DIST    : FLOAT;
 begin --UPDATE_COMMS_TRACK
   --get the the time difference between consecutive echos
   --of the track in seconds
   PREVIOUS_SECONDS_SINCE_MIDNIGHT := 3600 *
        SELECTED_TRACK.THE_TIME.HOURS + 60 *
        SELECTED_TRACK.THE_TIME.MINUTES +
        SELECTED_TRACK.THE_TIME.SECONDS;
   SECONDS_PASSED := SECONDS_SINCE_MIDNIGHT -
        PREVIOUS_SECONDS_SINCE_MIDNIGHT;

   --write the new observation time of the track to the
   --track's record
   SELECTED_TRACK.THE_TIME.HOURS    := SECONDS_SINCE_MIDNIGHT/3600;
   SELECTED_TRACK.THE_TIME.MINUTES := (SECONDS_SINCE_MIDNIGHT
        mod 3600) / 60;
   SELECTED_TRACK.THE_TIME.SECONDS := (SECONDS_SINCE_MIDNIGHT
        mod 3600) mod  60;

   --change the position values of the track & write to the
   --track's record
   OWN_LAT := 35.0 + ((FLOAT(SECONDS_PASSED)*OWN_VELOCITY) /
         3600.0)/MILES_PER_DEGREE;
   DISTANCE := (FLOAT(SECONDS_PASSED)*SELECTED_TRACK.VELOCITY)/
        3600.0;
   SELECTED_TRACK.LATITUDE := SELECTED_TRACK.LATITUDE +
        (DISTANCE * COS(SELECTED_TRACK.COURSE*RADIANS_PER_DEGREE))/
```

```
      MILES_PER_DEGREE;
  SELECTED_TRACK.LONGITUDE := SELECTED_TRACK.LONGITUDE +
       (DISTANCE * SIN(SELECTED_TRACK.COURSE*RADIANS_PER_DEGREE))/
       MILES_PER_DEGREE;

  --change the range of the track & write to the track's record
  LAT_DIST := (SELECTED_TRACK.LATITUDE - OWN_LAT) *
        MILES_PER_DEGREE;
  LONG_DIST := (SELECTED_TRACK.LONGITUDE - OWN_LONG) *
        MILES_PER_DEGREE;
  SELECTED_TRACK.THE_RANGE := SQRT(LAT_DIST**2 + LONG_DIST**2);

  --change the azimuth of the track & write to the track's record
  SELECTED_TRACK.AZIMUTH :=
        DEGREES_PER_RADIAN *
       ARCCOS(LAT_DIST/SELECTED_TRACK.THE_RANGE);
end UPDATE_COMMS_TRACK;


--*********************************************************************
--*** procedure PRINT_ONE_TRACK                                     ***
--*** Algorithm : Prints out components of one track to             ***
--***              the file                                         ***
--*********************************************************************
procedure PRINT_ONE_TRACK( TRACK : in out COMMS_TRACK_LIST ) is


  --*********************************************************************
  --*** procedure PUT_LEADING_ZEROS                                   ***
  --*** Algorithm : Puts leading ZEROS in front of the value         ***
  --***              into text                                        ***
  --*********************************************************************
  procedure PUT_LEADING_ZEROS(
           FILE       : in FILE_TYPE;
           OUT_NUMBER : in FLOAT;
           BEFORE     : in NATURAL ) is

    --BEFORE is number of original digits to be filled by both
    --LEADING ZEROS and the number itself

    NUMBER : FLOAT := FLOAT( 10**( BEFORE-1 ));
```

```
  begin --PUT_LEADING_ZEROS
    while NUMBER /= 1.0 loop
      if OUT_NUMBER < NUMBER then
        PUT( FILE, '0' );
      end if;
      NUMBER := NUMBER / 10.0;
    end loop;
  end PUT_LEADING_ZEROS;

begin --PRINT_ONE_TRACK
  --put "beginning of track information" indicator
  PUT( FILE, "POS/" );
  --put TRACK.ID
  PUT_LEADING_ZEROS( FILE, FLOAT(TRACK.ID), 4 );
  PUT( FILE, TRACK.ID, 1 ); PUT( FILE, '/' );
  --put TRACK.ORIGIN
  PUT( FILE, TRACK.ORIGIN ); PUT( FILE, '/' );
  --put TRACK.THE_TIME
  PUT_LEADING_ZEROS( FILE, FLOAT(TRACK.THE_TIME.HOURS), 2 );
  PUT( FILE, TRACK.THE_TIME.HOURS, 1 );
  PUT_LEADING_ZEROS( FILE, FLOAT(TRACK.THE_TIME.MINUTES), 2 );
  PUT( FILE, TRACK.THE_TIME.MINUTES, 1 );
  PUT_LEADING_ZEROS( FILE, FLOAT(TRACK.THE_TIME.SECONDS), 2 );
  PUT( FILE, TRACK.THE_TIME.SECONDS, 1 ); PUT( FILE, '/' );
  --put TRACK.TRACK_CLASS
  if    TRACK.TRACK_CLASS = AIR     then PUT( FILE, "AA/" );
  elsif TRACK.TRACK_CLASS = SURFACE then PUT( FILE, "SU/" );
  else                                   PUT( FILE, "SS/" );
  end if;
  --put TRACK.IFF_CLASS
  if    TRACK.IFF_CLASS = FRIENDLY then PUT( FILE, "F/" );
  elsif TRACK.IFF_CLASS = HOSTILE  then PUT( FILE, "H/" );
  elsif TRACK.IFF_CLASS = NEUTRAL  then PUT( FILE, "N/" );
  else                                  PUT( FILE, "U/" );
  end if;
  --put TRACK.LATITUDE
  if TRACK.LATITUDE < 0.0 then
    PUT( FILE, '-' );
    PUT_LEADING_ZEROS( FILE, -TRACK.LATITUDE, 2 );
  else
    PUT( FILE, ' ' );
```

```
    PUT_LEADING_ZEROS( FILE, TRACK.LATITUDE, 2 );
  end if;
    PUT( FILE, TRACK.LATITUDE, 1, 1, 0 ); PUT( FILE, '/' );
  --put TRACK.LONGITUDE
  if TRACK.LONGITUDE < 0.0 then
    PUT( FILE, '-' );
    PUT_LEADING_ZEROS( FILE, -TRACK.LONGITUDE, 3 );
   else
     PUT( FILE, ' ' );
    PUT_LEADING_ZEROS( FILE, TRACK.LONGITUDE, 3 );
  end if;
  PUT( FILE, TRACK.LONGITUDE, 1, 1, 0 ); PUT( FILE, '/' );
  --put TRACK.ALTITUDE
  if TRACK.ALTITUDE < 0.0 then
    PUT( FILE, '-' );
    PUT_LEADING_ZEROS( FILE, -TRACK.ALTITUDE, 5 );
    PUT( FILE, -TRACK.ALTITUDE, 1, 1, 0 );
  elsif TRACK.ALTITUDE > 0.0 then
    PUT( FILE, ' ' );
    PUT_LEADING_ZEROS( FILE, TRACK.ALTITUDE, 5 );
    PUT( FILE, TRACK.ALTITUDE, 1, 1, 0 );
  else
    PUT( FILE, " 00000.0" );
  end if;
  PUT( FILE, '/' );
  --put TRACK.COURSE
  PUT_LEADING_ZEROS( FILE, TRACK.COURSE, 3 );
  PUT( FILE, TRACK.COURSE, 1, 1, 0 ); PUT( FILE, '/' );
  --put TRACK.VELOCITY
  PUT_LEADING_ZEROS( FILE, TRACK.VELOCITY, 3 );
  PUT( FILE, TRACK.VELOCITY, 1, 1, 0 ); PUT( FILE, '/' );
  --put TRACK.THE_RANGE
  PUT_LEADING_ZEROS( FILE, TRACK.THE_RANGE, 4 );
  PUT( FILE, TRACK.THE_RANGE, 1, 2, 0 ); PUT( FILE, '/' );
  --put "end of track information" indicator to the file
  PUT_LINE( FILE, "C/" );
end PRINT_ONE_TRACK;
```

```
--*************************************************************************
--*** procedure PRINT_TRACKS                                            ***
--*************************************************************************
procedure PRINT_TRACKS(
          FILE                    : in     FILE_TYPE;
          COMMS_TRACKS            : in out COMMS_TRACK_LIST;
          SECONDS_SINCE_MIDNIGHT  : in out NATURAL ) is

  SELECTED_TRACK_ID : NATURAL;
  SELECTED_TRACK    : COMMS_TRACK_LIST;
  PREVIOUS          : COMMS_TRACK_LIST;
  CURRENT           : COMMS_TRACK_LIST;
  RANDOM_NUMBER     : FLOAT;

begin --PRINT_TRACKS
  CURRENT := COMMS_TRACKS;
  --initialize all track's reported component to befalse
  for I in 1..COMMS_TRACK_CAPACITY loop
    CURRENT.REPORTED := false;
    CURRENT := CURRENT.NEXT;
  end loop;
  --read one random number
  RANDOM_NUMBER := RANDOM.NUMBER;
  --find a number between 0 & COMMS_TRACK_CAPACITY, and then
  --report that amount of tracks
  for I in  0 .. NATURAL
       (RANDOM_NUMBER * FLOAT(COMMS_TRACK_CAPACITY-1)) loop
    --begin from the head of the list each time to search
    SELECTED_TRACK := COMMS_TRACKS;
    PREVIOUS := null;
    --select a track to report
    SELECTED_TRACK_ID := 1 +
      NATURAL( RANDOM.NUMBER * FLOAT(COMMS_TRACK_CAPACITY-1) );
    --search the track list with the SELECTED_TRACK_ID and find
    --the related track
    while SELECTED_TRACK.ID /= SELECTED_TRACK_ID loop
      PREVIOUS := SELECTED_TRACK;
      SELECTED_TRACK := SELECTED_TRACK.NEXT;
    end loop;
    --if selected track has not already been reported,
    --then report it, else pass it
```

```
      if SELECTED_TRACK.REPORTED = false then
        UPDATE_COMMS_TRACK( SELECTED_TRACK, SECONDS_SINCE_MIDNIGHT );
        PRINT_ONE_TRACK( SELECTED_TRACK );
        SELECTED_TRACK.REPORTED := true;
      end if;
    end loop;
   --put "end of all tracks indicator" to the file
   PUT( FILE, '\' );
 end PRINT_TRACKS;


begin --COMMS_LINKS
  SECONDS_SINCE_MIDNIGHT := NATURAL(CALENDAR.SECONDS(CALENDAR.CLOCK));
  if not COMMS_TRACKS_ARE_CREATED then
    --initialize the configuration of the COMMS_TRACKS
    CREATE_COMMS_TRACKS( COMMS_TRACKS, SECONDS_SINCE_MIDNIGHT );
    COMMS_TRACKS_ARE_CREATED := true;
  end if;
  --assign the names of all of the individual stations
  for I in 0 .. (NUMBER_OF_ADDRESSES-1) loop
    NAMES(I).NAME(1..5) := "NAME_";
    NAMES(I).NAME(6)    := CHARACTER_OF( I );
  end loop;
  --open file to which message will be printed
  FILE_NAME := GET_FILE_NAME;
  CREATE( FILE, NAME => FILE_NAME );
  SET_LINE_LENGTH( FILE, 80 );
  --assign the out variable of the procedure
  INPUT_LINK_MESSAGE := FILE_NAME;
  --process FILE_NAME field
  PUT( FILE, "NAME : " );
  PUT( FILE, FILE_NAME );
  NEW_LINE( FILE );
  --process LINK_ID field
  PUT( FILE, "LINK_ID : " );
  RANDOM_NUMBER := RANDOM.NUMBER;
  if    RANDOM_NUMBER < 0.25 then PUT( FILE, JTIDS  );
  elsif RANDOM_NUMBER < 0.50 then PUT( FILE, LINK11 );
  elsif RANDOM_NUMBER < 0.75 then PUT( FILE, LINK16 );
  else                            PUT( FILE, OTCIXS );
  end if;
```

```
NEW_LINE( FILE );
--process CLASSIFICATION field
PUT( FILE, "CLASSIFICATION : " );
RANDOM_NUMBER := RANDOM.NUMBER;
if     RANDOM_NUMBER < 0.25 then PUT( FILE, U  );
elsif RANDOM_NUMBER < 0.50 then PUT( FILE, C  );
elsif RANDOM_NUMBER < 0.75 then PUT( FILE, S  );
else                            PUT( FILE, TS );
end if;
--process PRECEDENCE field
PUT( FILE, " PRECEDENCE : ");
RANDOM_NUMBER := RANDOM.NUMBER;
if     RANDOM_NUMBER < 0.25 then PUT( FILE, R );
elsif RANDOM_NUMBER < 0.50 then PUT( FILE, P );
elsif RANDOM_NUMBER < 0.75 then PUT( FILE, O );
else                            PUT( FILE, Z );
end if;
NEW_LINE( FILE );
--process FM field
PUT( FILE, "FM : " );
RANDOM_NUMBER := RANDOM.NUMBER;
PUT( FILE, NAMES
       (NATURAL(RANDOM_NUMBER*FLOAT(NUMBER_OF_ADDRESSES-1))).NAME );
NAMES
( NATURAL(RANDOM_NUMBER*FLOAT(NUMBER_OF_ADDRESSES-1))).USED := true;
FROM_NAME :=  NAMES
       ( NATURAL(RANDOM_NUMBER*FLOAT(NUMBER_OF_ADDRESSES-1))).NAME;
NEW_LINE( FILE );
--process TO field
PUT( FILE, "TO : " );
TO_NUMBER := RANDOM.NUMBER;
for I in 0..NATURAL(TO_NUMBER*FLOAT(NUMBER_OF_ADDRESSES/3)) loop
  RANDOM_NUMBER := RANDOM.NUMBER;
  while NAMES(NATURAL
       (RANDOM_NUMBER*FLOAT(NUMBER_OF_ADDRESSES-1))).USED loop
    RANDOM_NUMBER := RANDOM.NUMBER;
  end loop;
  SET_COL( FILE, 6 );
  PUT( FILE, NAMES
       (NATURAL(RANDOM_NUMBER*FLOAT(NUMBER_OF_ADDRESSES-1))).NAME );
  NEW_LINE( FILE );
```

```
  NAMES
(NATURAL(RANDOM_NUMBER*FLOAT(NUMBER_OF_ADDRESSES-1))).USED := true;
end loop;
NEW_LINE( FILE );
--process INFO field
PUT( FILE, "INFO : " );
INFO_NUMBER := RANDOM.NUMBER;
for I in 0..NATURAL(INFO_NUMBER*FLOAT(NUMBER_OF_ADDRESSES/3)) loop
  RANDOM_NUMBER := RANDOM.NUMBER;
  while NAMES(NATURAL
       (RANDOM_NUMBER*FLOAT(NUMBER_OF_ADDRESSES-1))).USED loop
    RANDOM_NUMBER := RANDOM.NUMBER;
  end loop;
  SET_COL( FILE, 8 );
  PUT( FILE, NAMES
       (NATURAL(RANDOM_NUMBER*FLOAT(NUMBER_OF_ADDRESSES-1))).NAME );
  NEW_LINE( FILE );
  NAMES(NATURAL
        (RANDOM_NUMBER*FLOAT(NUMBER_OF_ADDRESSES-1))).USED := true;
end loop;
NEW_LINE( FILE );

--process VIA field
PUT( FILE, "VIA : " );
RANDOM_NUMBER := RANDOM.NUMBER;
for I in 0..(NATURAL(TO_NUMBER*FLOAT(NUMBER_OF_ADDRESSES/3)) +
             NATURAL(INFO_NUMBER*FLOAT(NUMBER_OF_ADDRESSES/3))) loop
  RANDOM_NUMBER := RANDOM.NUMBER;
  while not ( NAMES(NATURAL
       (RANDOM_NUMBER*FLOAT(NUMBER_OF_ADDRESSES-1))).USED
  and NAMES(NATURAL(RANDOM_NUMBER*
       FLOAT(NUMBER_OF_ADDRESSES-1))).NAME /= FROM_NAME
  and not NAMES(NATURAL(RANDOM_NUMBER*
       FLOAT(NUMBER_OF_ADDRESSES-1))).REUSED)loop
    RANDOM_NUMBER := RANDOM.NUMBER;
  end loop;
  SET_COL( FILE, 7 );
  PUT( FILE, NAMES
       (NATURAL(RANDOM_NUMBER*FLOAT(NUMBER_OF_ADDRESSES-1))).NAME );
  NEW_LINE( FILE );
  NAMES(NATURAL
```

```
        (RANDOM_NUMBER*FLOAT(NUMBER_OF_ADDRESSES-1))).REUSED := true;
    end loop;
    --process BY field
    SET_COL( FILE, 7 );
    PUT( FILE, "BY " );
    while not ( NAMES(NATURAL(RANDOM_NUMBER*
          FLOAT(NUMBER_OF_ADDRESSES-1))).USED
    and NAMES(NATURAL(RANDOM_NUMBER*
          FLOAT(NUMBER_OF_ADDRESSES-1))).NAME /= FROM_NAME
    and not NAMES(NATURAL(RANDOM_NUMBER*
          FLOAT(NUMBER_OF_ADDRESSES-1))).REUSED ) loop
      RANDOM_NUMBER := RANDOM.NUMBER;
    end loop;
    PUT_LINE( FILE, NAMES
          (NATURAL(RANDOM_NUMBER*FLOAT(NUMBER_OF_ADDRESSES-1))).NAME );
    NAMES(NATURAL
         (RANDOM_NUMBER*FLOAT(NUMBER_OF_ADDRESSES-1))).REUSED := true;
    NEW_LINE( FILE );
    --process SUBJECT field
    PUT( FILE, "SUBJECT : THIS IS THE SUBJECT" );
    NEW_LINE( FILE, 4 );
    if RANDOM.NUMBER < 0.25 then
      --approximately one fourth of the messages will contain
      --plain text data
      PUT( FILE, "THIS IS THE TEXT");
    else
      --approximately three fourth of the messages will contain
      --track data
      PRINT_TRACKS( FILE, COMMS_TRACKS, SECONDS_SINCE_MIDNIGHT );
    end if;
    CLOSE( FILE );
end COMMS_LINKS;


--*********************************************************************
--*** procedure PREPARE_PERIODIC_REPORT                             ***
--*********************************************************************
procedure PREPARE_PERIODIC_REPORT(
                    INITIATE_TRANS      : in
                                        INITIATE_TRANSMISSION_SEQUENCE;
                   TERMINATE_TRANS      : in     BOOLEAN;
```

```
                TCD_TRANSMIT_COMMAND : in out TRANSMIT_COMMAND ) is

PERIODIC_REPORT_FILE : constant FILENAME := "TRACK_FILE";
LINK : LINKS_TYPE;


--*********************************************************************
--*** Function PUT_TRACKS                                           ***
--*********************************************************************
function PUT_TRACKS (
      TRACK          : TRACK_TUPLE;
      INITIATE_TRANS : INITIATE_TRANSMISSION_SEQUENCE )
                                          return TEXT_STRING is

  TEXT      : TEXT_STRING;
  I         : NATURAL := 1; --index of TEXT variable
  CURRENT   : TRACK_TUPLE;
  STRING_4  : STRING(1..4) := "    ";
  STRING_8  : STRING(1..8) := "        ";


  --*********************************************************************
  --*** Function CHARACTER_OF                                         ***
  --*********************************************************************
  function CHARACTER_OF( N : in NATURAL ) return CHARACTER is
  begin
    return( CHARACTER'VAL( N + 48 ) );
  end CHARACTER_OF;


  --*********************************************************************
  --*** Function FLOAT_TO_STRING_8                                    ***
  --*** Algorithm : converts 8-digit float number to                  ***
  --***              8-character string                               ***
  --*********************************************************************
  function FLOAT_TO_STRING_8( NUMBER : in FLOAT ) return STRING is
    S   : STRING(1..8) := " 00000.0";
    N   : NATURAL;
    F   : FLOAT := NUMBER;
    R   : FLOAT;
  begin
```

```
  if F < 0.0 then
    S(1) := '-';
    F := -F;
  end if;
  N := NATURAL( F );
  R := F - FLOAT( N );
  if R < 0.0 then
    N := N - 1;
    R := R + 1.0;
  end if;
  if N > 9 then
    if N > 99 then
      if N > 999 then
        if N > 9999 then
          S(2) := CHARACTER_OF( N/10000 );
          N := N - ( (N/10000) * 10000);
        end if;
        S(3) := CHARACTER_OF( N/1000 );
        N := N - ( (N/1000) * 1000 );
      end if;
      S(4) := CHARACTER_OF( N/100 );
      N := N - ( (N/100) * 100 );
    end if;
    S(5) := CHARACTER_OF( N/10 );
    N := N - ( (N/10) * 10 );
  end if;
  S(6) := CHARACTER_OF( N );
  N := NATURAL( R * 10.0 );
  S(8) := CHARACTER_OF( N );
  return(S);
end FLOAT_TO_STRING_8;


--*********************************************************************
--*** Function NATURAL_TO_STRING_4                                  ***
--*** Algorithm : converts 4-digit natural number to               ***
--***              4-character string                              ***
--*********************************************************************
function NATURAL_TO_STRING_4( NUMBER : NATURAL ) return STRING is
  S : STRING(1..4) := "0000";
  N : NATURAL := NUMBER;
```

```
  begin
    if N > 9 then
      if N > 99 then
        if N > 999 then
          S(1) := CHARACTER_OF( N/1000 );
          N := N - ( (N/1000) * 1000 );
        end if;
        S(2) := CHARACTER_OF( N/100 );
        N := N - ( (N/100) * 100 );
      end if;
      S(3) := CHARACTER_OF( N/10 );
      N := N - ( (N/10) * 10 );
    end if;
    S(4) := CHARACTER_OF(N);
    return(S);
  end NATURAL_TO_STRING_4;

begin --PUT_TRACKS
  CURRENT := TRACK;
  while CURRENT /= null loop
    if (INITIATE_TRANS.DBASE_REQUEST.TRACK_CLASS
                              (CURRENT.TRACK_CLASS)) and
       (INITIATE_TRANS.DBASE_REQUEST.IFF_CLASS
                        (CURRENT.IFF_CLASS)     ) and
       (INITIATE_TRANS.DBASE_REQUEST.THE_RANGE>=CURRENT.THE_RANGE)
                                                                  then
      TEXT(I..I+3) := "POS/";
      TEXT(I+4..I+7) := NATURAL_TO_STRING_4(CURRENT.ID);
      TEXT(I+8) := '/';
      TEXT(I+9..I+16) := CURRENT.OBSERVER;
      TEXT(I+17) := '/';
      STRING_4 := NATURAL_TO_STRING_4
                  (CURRENT.OBSERVATION_TIME.HOURS);
      TEXT(I+18..I+19) := STRING_4(3..4);
      STRING_4 := NATURAL_TO_STRING_4
                  (CURRENT.OBSERVATION_TIME.MINUTES);
      TEXT(I+20..I+21) := STRING_4(3..4);
      STRING_4 := NATURAL_TO_STRING_4
                  (CURRENT.OBSERVATION_TIME.SECONDS);
      TEXT(I+22..I+23) := STRING_4(3..4);
      TEXT(I+24) := '/';
```

```
if    CURRENT.TRACK_CLASS = SURFACE then
  TEXT(I+25..I+26) := "SU";
elsif CURRENT.TRACK_CLASS = SUBSURFACE then
  TEXT(I+25..I+26) := "SS";
else
  TEXT(I+25..I+26) := "AA";
end if;
TEXT(I+27) := '/';
if    CURRENT.IFF_CLASS = FRIENDLY then TEXT(I+28) := 'F';
elsif CURRENT.IFF_CLASS = HOSTILE  then TEXT(I+28) := 'H';
elsif CURRENT.IFF_CLASS = NEUTRAL  then TEXT(I+28) := 'N';
else                                    TEXT(I+28) := 'U';
end if;
TEXT(I+29) := '/';
STRING_8 := FLOAT_TO_STRING_8(CURRENT.LATITUDE);
if STRING_8(1) = '-' then
  TEXT(I+30) := '-';
else
  TEXT(I+30) := ' ';
end if;
TEXT(I+31..I+34) := STRING_8(5..8);
TEXT(I+35) := '/';
STRING_8 := FLOAT_TO_STRING_8(CURRENT.LONGITUDE);
if STRING_8(1) = '-' then
  TEXT(I+36) := '-';
else
  TEXT(I+36) := ' ';
end if;
TEXT(I+37..I+41) := STRING_8(4..8);
TEXT(I+42) := '/';
TEXT(I+43..I+50) := FLOAT_TO_STRING_8(CURRENT.ALTITUDE);
TEXT(I+51) := '/';
STRING_8 := FLOAT_TO_STRING_8(CURRENT.COURSE);
TEXT(I+52..I+56) := STRING_8(4..8);
TEXT(I+57) := '/';
STRING_8 := FLOAT_TO_STRING_8(CURRENT.VELOCITY);
TEXT(I+58..I+62) := STRING_8(4..8);
TEXT(I+63) := '/';
STRING_8 := FLOAT_TO_STRING_8(CURRENT.THE_RANGE);
TEXT(I+64..I+70) := STRING_8(2..8);
TEXT(I+71) := '/';
```

```
        if    CURRENT.ARCHIVE_FLAG = C then TEXT(I+72) := 'C';
        elsif CURRENT.ARCHIVE_FLAG = N then TEXT(I+72) := 'N';
        elsif CURRENT.ARCHIVE_FLAG = A then TEXT(I+72) := 'A';
        else                                TEXT(I+72) := 'S';
        end if;
        TEXT(I+73) := '/';
      end if;
      CURRENT := CURRENT.NEXT;
      I := I + 80;
    end loop;
    text(I) := '\';
    for J in I+1..TEXT_LENGTH loop
      TEXT(J) := ' ';
    end loop;
    return(TEXT);
  end PUT_TRACKS;

begin --PREPARE_PERIODIC_REPORT
  LINK := INITIATE_TRANS.LINK_ID;
  TCD_TRANSMIT_COMMAND(LINK).ROUTE_ADDR     := null;
  TCD_TRANSMIT_COMMAND(LINK).ROUTED         := false;
  TCD_TRANSMIT_COMMAND(LINK).FULL           := true;
  TCD_TRANSMIT_COMMAND(LINK).TEXT.NAME      := PERIODIC_REPORT_FILE;
  TCD_TRANSMIT_COMMAND(LINK).TEXT.HEADER    := INITIATE_TRANS.HEADER;
  TCD_TRANSMIT_COMMAND(LINK).TEXT.LINK_ID   := LINK;
  TCD_TRANSMIT_COMMAND(LINK).TEXT.RELAYED   := false;
  TCD_TRANSMIT_COMMAND(LINK).TEXT.ARCHIVE   := true;
  TCD_TRANSMIT_COMMAND(LINK).TEXT.IS_TRACK  := true;
  TCD_TRANSMIT_COMMAND(LINK).TEXT.FORMAT    :=
                                      INITIATE_TRANS.DESIRED_FORMAT;
  TCD_TRANSMIT_COMMAND(LINK).TEXT.TEXT      := PUT_TRACKS
                                       ( TRACKS, INITIATE_TRANS );
end PREPARE_PERIODIC_REPORT;


--*************************************************************************
--*** procedure PARSE_INPUT_FILE                                        ***
--*************************************************************************
procedure PARSE_INPUT_FILE(
                  INPUT_LINK_MESSAGE : in     FILENAME;
                  INPUT_TEXT_RECORD  : in out TEXT_RECORD ) is
```

```
  FILE          : TEXT_IO.FILE_TYPE;
  LINE          : STRING(1..80);
  I             : NATURAL;
  ADDRESS_HEAD  : ADDRESS_LINK;
  NEW_ADDRESS   : ADDRESS_LINK;
  VIA_HEAD      : VIA_RECORD_LINK;
  NEW_VIA       : VIA_RECORD_LINK;
  LINE_LENGTH   : NATURAL;


  --*********************************************************************
  --*** procedure GET_ONE_LINE                                        ***
  --*** algorithm : this procedure reads one line from the           ***
  --***              pointed file and returns the content of the     ***
  --***              file. The difference between this procedure     ***
  --***              and TEXT_IO.GET_LINE is that, this procedure    ***
  --***              adds BLANK character to the empty places in     ***
  --***              the line                                        ***
  --*********************************************************************
  procedure GET_ONE_LINE(
              FILE        : in  TEXT_IO.FILE_TYPE;
              LINE        : out STRING;
              LINE_LENGTH : out NATURAL ) is

    THE_LINE   : STRING(1..80); --dummy line variable is used
    THE_LENGTH : NATURAL;       --dummy line_length variable is used

  begin --GET_ONE_LINE
    --read the line from the input file
    TEXT_IO.GET_LINE( FILE, THE_LINE, THE_LENGTH );
    --add BLANK character to the empty places
    for I in THE_LENGTH+1 .. 80 loop
      THE_LINE( I ) := ' ';
    end loop;
    --assign the OUT variables to local variables
    LINE := THE_LINE;
    LINE_LENGTH := THE_LENGTH;
  end GET_ONE_LINE;

begin --PARSE_INPUT_FILE
  --open the file to be parsed
```

```
OPEN( FILE, NAME => INPUT_LINK_MESSAGE, MODE => IN_FILE );
--process filename field
INPUT_TEXT_RECORD.NAME := INPUT_LINK_MESSAGE;
SKIP_LINE( FILE );
--process link_id field
GET_ONE_LINE( FILE, LINE, LINE_LENGTH );
if    LINE(11..15) = "JTIDS"  then
  INPUT_TEXT_RECORD.LINK_ID := JTIDS;
elsif LINE(11..16) = "LINK11" then
  INPUT_TEXT_RECORD.LINK_ID := LINK11;
elsif LINE(11..16) = "LINK16" then
  INPUT_TEXT_RECORD.LINK_ID := LINK16;
else
  INPUT_TEXT_RECORD.LINK_ID := OTCIXS;
end if;
--skip route field & blank lines
while LINE(1..14) /= "CLASSIFICATION" loop
  GET_ONE_LINE( FILE, LINE, LINE_LENGTH );
end loop;
--process classification field
if    LINE(18) = 'U' then
  INPUT_TEXT_RECORD.HEADER.CLASSIFICATION:=U; I := 33;
elsif LINE(18) = 'C' then
  INPUT_TEXT_RECORD.HEADER.CLASSIFICATION:=C; I := 33;
elsif LINE(18) = 'S' then
  INPUT_TEXT_RECORD.HEADER.CLASSIFICATION:=S; I := 33;
else
  INPUT_TEXT_RECORD.HE' )ER.CLASSIFICATION:=TS;I := 34;
end if;
--process precedence field
if    LINE(I) = 'R' then INPUT_TEXT_RECORD.HEADER.PRECEDENCE := R;
elsif LINE(I) = 'P' then INPUT_TEXT_RECORD.HEADER.PRECEDENCE := P;
elsif LINE(I) = 'O' then INPUT_TEXT_RECORD.HEADER.PRECEDENCE := O;
else                     INPUT_TEXT_RECORD.HEADER.PRECEDENCE := Z;
end if;
--process origin field
GET_ONE_LINE( FILE, LINE, LINE_LENGTH );
INPUT_TEXT_RECORD.HEADER.ORIGIN := LINE(6..11);
--process address field
GET_ONE_LINE( FILE, LINE, LINE_LENGTH );
ADDRESS_HEAD := null;
```

```
while LINE_LENGTH > 1 loop --there exist another address
  NEW_ADDRESS      := new ADDRESS_TYPE;
  NEW_ADDRESS.NAME := LINE(6..11);
  NEW_ADDRESS.NEXT := ADDRESS_HEAD;
  ADDRESS_HEAD     := NEW_ADDRESS;
  GET_ONE_LINE( FILE, LINE, LINE_LENGTH );
end loop;
INPUT_TEXT_RECORD.HEADER.ADDRESS := ADDRESS_HEAD;
--process info field
GET_ONE_LINE( FILE, LINE, LINE_LENGTH );
ADDRESS_HEAD := null;
while LINE_LENGTH > 1 loop --there exist another address
  NEW_ADDRESS      := new ADDRESS_TYPE;
  NEW_ADDRESS.NAME := LINE(8..13);
  NEW_ADDRESS.NEXT := ADDRESS_HEAD;
  ADDRESS_HEAD     := NEW_ADDRESS;
  GET_ONE_LINE( FILE, LINE, LINE_LENGTH );
end loop;
INPUT_TEXT_RECORD.HEADER.INFO := ADDRESS_HEAD;
--process via field
GET_ONE_LINE( FILE, LINE, LINE_LENGTH );
VIA_HEAD := null;
while LINE_LENGTH > 1 loop --there exist another address
  NEW_VIA   := new VIA_RECORD;
  ADDRESS_HEAD := null;
  while LINE(7..8) /= "BY" loop
    --process relay_to field
    NEW_ADDRESS      := new ADDRESS_TYPE;
    NEW_ADDRESS.NAME := LINE(7..12);
    NEW_ADDRESS.NEXT := ADDRESS_HEAD;
    ADDRESS_HEAD     := NEW_ADDRESS;
    GET_ONE_LINE( FILE, LINE, LINE_LENGTH );
  end loop;
  --process relay_by field
  NEW_VIA.RELAY_BY := LINE(10..15);
  NEW_VIA.RELAY_TO := ADDRESS_HEAD;
  NEW_VIA.NEXT     := VIA_HEAD;
  VIA_HEAD         := NEW_VIA;
  GET_ONE_LINE( FILE, LINE, LINE_LENGTH );
end loop;
INPUT_TEXT_RECORD.HEADER.VIA_LINE := VIA_HEAD;
```

```
  --process subject field
  GET_ONE_LINE( FILE, LINE, LINE_LENGTH );
  INPUT_TEXT_RECORD.HEADER.SUBJECT(1..60) := LINE(11..70);
  SKIP_LINE( FILE, 3 );
  --fill in relay field
  INPUT_TEXT_RECORD.RELAYED := false;
  --fill in archive field
  INPUT_TEXT_RECORD.ARCHIVE := false;
  --process text field
  I := 1;
  while ( not END_OF_FILE ( FILE ) and I <= TEXT_LENGTH ) loop
    GET_ONE_LINE( FILE, LINE, LINE_LENGTH );
    INPUT_TEXT_RECORD.TEXT(I..I+LINE_LENGTH-1) :=
                                              LINE(1..LINE_LENGTH);
    I := I + 80;
  end loop;
  if INPUT_TEXT_RECORD.TEXT(1..4) = "POS/" then
    --file consists of track info
    INPUT_TEXT_RECORD.IS_TRACK := true;
    close( FILE );
  else --file consists of message
    INPUT_TEXT_RECORD.IS_TRACK := false;
    close( FILE );
  end if;
end PARSE_INPUT_FILE;


--************************************************************************
--*** procedure DECIDE_FOR_ARCHIVING                                   ***
--************************************************************************
procedure DECIDE_FOR_ARCHIVING(
            INPUT_TEXT_RECORD  : in  out TEXT_RECORD;
            TDD_ARCHIVE_SETUP  : in      ARCHIVE_SETUP;
            COMMS_TEXT_FILE    : out     TEXT_RECORD;
            COMMS_EMAIL        : out     FILENAME ) is

  CURRENT : ADDRESS_LINK;

begin
  if TDD_ARCHIVE_SETUP.ALL_SHIPS then
    --we are done, since all incoming messages are to be archived
```

```
    INPUT_TEXT_RECORD.ARCHIVE := true;
    COMMS_TEXT_FILE := INPUT_TEXT_RECORD;
  elsif ((TDD_ARCHIVE_SETUP.JTIDS) and
        (INPUT_TEXT_RECORD.LINK_ID = JTIDS)) or
        ((TDD_ARCHIVE_SETUP.LINK16) and
        (INPUT_TEXT_RECORD.LINK_ID = LINK16)) or
        ((TDD_ARCHIVE_SETUP.LINK11) and
        (INPUT_TEXT_RECORD.LINK_ID = LINK11)) or
        ((TDD_ARCHIVE_SETUP.OTCIXS) and
        (INPUT_TEXT_RECORD.LINK_ID = OTCIXS)) then
    --link_id of the incoming message matches with user choice
    --specified in by the archive setup.
    INPUT_TEXT_RECORD.ARCHIVE := true;
    COMMS_TEXT_FILE := INPUT_TEXT_RECORD;
  else
    --we want to archive only the files which are directed to us
    CURRENT := INPUT_TEXT_RECORD.HEADER.ADDRESS;
    while CURRENT.NEXT /= null and not INPUT_TEXT_RECORD.ARCHIVE loop
      if CURRENT.NAME = OWN_ADDRESS then
        INPUT_TEXT_RECORD.ARCHIVE := true;
        COMMS_TEXT_FILE := INPUT_TEXT_RECORD;
      else
        CURRENT := CURRENT.NEXT;
      end if;
    end loop;
  end if;
  COMMS_EMAIL := INPUT_TEXT_RECORD.NAME;
end DECIDE_FOR_ARCHIVING;


--************************************************************************
--*** procedure EXTRACT_TRACKS                                         ***
--************************************************************************
procedure EXTRACT_TRACKS(
             COMMS_TEXT_FILE : in  TEXT_RECORD;
                COMMS_ADD_TRACK : out ADD_TRACK_TUPLE ) is
  NEW_TRACK    : TRACK_TUPLE;
  HEAD         : TRACK_TUPLE;
  TEXT         : TEXT_STRING;
  I            : NATURAL := 1; --index of TEXT variable
  END_OF_TEXT : BOOLEAN := false;
```

```
ID          : STRING(1..6) := "000000";
HOUR        : STRING(1..6) := "000000";
MINUTE      : STRING(1..6) := "000000";
SECOND      : STRING(1..6) := "000000";
LATITUDE    : STRING(1..8) := "000000.0";
LONGITUDE   : STRING(1..8) := "000000.0";
ALTITUDE    : STRING(1..8) := "000000.0";
COURSE      : STRING(1..8) := "000000.0";
VELOCITY    : STRING(1..8) := "000000.0";
THE_RANGE   : STRING(1..9) := "000000.00";


--*********************************************************************
--*** function NATURAL_OF                                           ***
--*********************************************************************
function NATURAL_OF( CHAR : in CHARACTER ) return NATURAL is
begin --NATURAL_OF
  return ( CHARACTER'POS( CHAR ) - 48 );
end NATURAL_OF;


--*********************************************************************
--*** function STRING_6_TO_NATURAL                                  ***
--*********************************************************************
function STRING_6_TO_NATURAL( STR : in STRING ) return NATURAL is
  COEFFICIENT : NATURAL := 100000;
  OUT_NUMBER  : NATURAL := 0;
begin --STRING_6_TO_NATURAL
  for I in 1..6 loop
    OUT_NUMBER  := OUT_NUMBER + NATURAL_OF( STR(I) ) * COEFFICIENT;
    COEFFICIENT := COEFFICIENT / 10;
  end loop;
  return ( OUT_NUMBER );
end STRING_6_TO_NATURAL;


--*********************************************************************
--*** function STRING_8_TO_FLOAT                                    ***
--*********************************************************************
function STRING_8_TO_FLOAT( STR : in STRING ) return FLOAT is
  PREFIX      : NATURAL;
```

```
    POSTFIX    : NATURAL;
    OUT_NUMBER : FLOAT;
  begin
    PREFIX     := STRING_6_TO_NATURAL( STR(1..6) );
    POSTFIX    := NATURAL_OF ( STR(8) );
    OUT_NUMBER := FLOAT( PREFIX ) + 0.1 * FLOAT( POSTFIX );
    return( OUT_NUMBER );
  end STRING_8_TO_FLOAT;


  --*********************************************************************
  --*** function STRING_9_TO_FLOAT                                    ***
  --*********************************************************************
  function STRING_9_TO_FLOAT( STR : in STRING ) return FLOAT is
    OUT_NUMBER : FLOAT;
  begin --STRING_9_TO_FLOAT
    OUT_NUMBER := STRING_8_TO_FLOAT
                  (STR(1..8)) + 0.01 * FLOAT(NATURAL_OF(STR(9)));
    return( OUT_NUMBER );
  end STRING_9_TO_FLOAT;

begin --EXTRACT_TRACKS
  HEAD := null;
  TEXT := COMMS_TEXT_FILE.TEXT;
  while not END_OF_TEXT loop
    NEW_TRACK := new TRACK_RECORD;
    --process ID field
    ID(3..6)      := TEXT(I+4..I+7);
    NEW_TRACK.ID := STRING_6_TO_NATURAL(ID);
    --process OBSERVER field
    NEW_TRACK.OBSERVER := TEXT(I+9..I+16);
    --process TIME field
    HOUR(5..6)    := TEXT(I+18..I+19);
    MINUTE(5..6) := TEXT(I+20..I+21);
    SECOND(5..6) := TEXT(I+22..I+23);
    NEW_TRACK.OBSERVATION_TIME.HOURS   := STRING_6_TO_NATURAL(HOUR  );
    NEW_TRACK.OBSERVATION_TIME.MINUTES := STRING_6_TO_NATURAL(MINUTE);
    NEW_TRACK.OBSERVATION_TIME.SECONDS := STRING_6_TO_NATURAL(SECOND);
    --process TRACK_CLASS field
    if    TEXT(I+25..I+26) = "SU" then
      NEW_TRACK.TRACK_CLASS := SURFACE;
```

```
elsif TEXT(I+25..I+26) = "SS" then
  NEW_TRACK.TRACK_CLASS := SUBSURFACE;
else
  NEW_TRACK.TRACK_CLASS := AIR;
end if;
--process IFF_CLASS field
if    TEXT(I+28) = 'F' then NEW_TRACK.IFF_CLASS := FRIENDLY;
elsif TEXT(I+28) = 'H' then NEW_TRACK.IFF_CLASS := HOSTILE;
elsif TEXT(I+28) = 'N' then NEW_TRACK.IFF_CLASS := NEUTRAL;
else                        NEW_TRACK.IFF_CLASS := UNKNOWN;
end if;
--process LATITUDE field
LATITUDE(5..8) := TEXT(I+31..I+34);
NEW_TRACK.LATITUDE := STRING_8_TO_FLOAT(LATITUDE);
if TEXT(I+30) = '-' then
  NEW_TRACK.LATITUDE := -NEW_TRACK.LATITUDE;
end if;
--process LONGITUDE field
LONGITUDE(4..8) := TEXT(I+37..I+41);
NEW_TRACK.LONGITUDE := STRING_8_TO_FLOAT(LONGITUDE);
if TEXT(I+36) = '-' then
  NEW_TRACK.LONGITUDE := -NEW_TRACK.LONGITUDE;
end if;
--process ALTITUDE field
ALTITUDE(2..8) := TEXT(I+44..I+50);
NEW_TRACK.ALTITUDE := STRING_8_TO_FLOAT(ALTITUDE);
if TEXT(I+43) = '-' then
  NEW_TRACK.ALTITUDE := -NEW_TRACK.ALTITUDE;
end if;
--process COURSE field
COURSE(4..8) := TEXT(I+52..I+56);
NEW_TRACK.COURSE := STRING_8_TO_FLOAT(COURSE);
--process VELOCITY field
VELOCITY(4..8) := TEXT(I+58..I+62);
NEW_TRACK.VELOCITY := STRING_8_TO_FLOAT(VELOCITY);
--process THE_RANGE field
THE_RANGE(3..9) := TEXT(I+64..I+70);
NEW_TRACK.THE_RANGE := STRING_9_TO_FLOAT(THE_RANGE);
--process ARCHIVE_FLAG field
if    TEXT(I+72) = 'C' then NEW_TRACK.ARCHIVE_FLAG := C;
elsif TEXT(I+72) = 'N' then NEW_TRACK.ARCHIVE_FLAG := N;
```

```
    elsif TEXT(I+72) = 'A' then NEW_TRACK.ARCHIVE_FLAG := A;
    else                        NEW_TRACK.ARCHIVE_FLAG := S;
    end if;
    --link NEW_TRACK to head of tracks
    NEW_TRACK.NEXT := HEAD;
    HEAD := NEW_TRACK;
    --go to the begininng of the next line
    I := I + 80;
    if TEXT(I) = '\' then
      --end of tracks sign is encountered
      END_OF_TEXT := true;
    end if;
  end loop;
  COMMS_ADD_TRACK.TRACK := HEAD;
  COMMS_ADD_TRACK.ORIGIN := "COMMS    ";
end EXTRACT_TRACKS;


--************************************************************************
--*** procedure FILTER_COMMS_TRACKS                                    ***
--************************************************************************
procedure FILTER_COMMS_TRACKS(
             COMMS_ADD_TRACK      : in out ADD_TRACK_TUPLE;
             TDD_FILTER           : in     SET_TRACK_FILTER;
             FILTERED_COMMS_TRACK : out    ADD_TRACK_TUPLE ) is

  HEAD     : TRACK_TUPLE := COMMS_ADD_TRACK.TRACK;
  CURRENT  : TRACK_TUPLE := COMMS_ADD_TRACK.TRACK;
  PREVIOUS : TRACK_TUPLE := null;

begin --FILTER_COMMS_TRACKS
  while CURRENT /= null loop
    --search through comms_add_track in order to find the place
    --for the new track according to the precedence

    if (TDD_FILTER.DESIRED_CLASS(CURRENT.TRACK_CLASS)) and
       (TDD_FILTER.DESIRED_RANGE(CURRENT.TRACK_CLASS) >=
                                              CURRENT.THE_RANGE) then
       PREVIOUS := CURRENT;
       CURRENT := CURRENT.NEXT;
    else
```

```
      if PREVIOUS = null then
        HEAD := CURRENT.NEXT;
      else
        PREVIOUS.NEXT := CURRENT.NEXT;
      end if;
      CURRENT := CURRENT.NEXT;
    end if;
  end loop;
  if HEAD = null then
    APPROVED := false;
  end if;
  FILTERED_COMMS_TRACK.TRACK := HEAD;
end FILTER_COMMS_TRACKS;


  --*********************************************************************
  --*** procedure DELETE_TRACK                                        ***
  --*********************************************************************
  procedure DELETE_TRACK(
              TRACKS    : in out TRACK_TUPLE;
              TARGET_ID : in     NATURAL ) is

    TRACK_IS_FOUND : BOOLEAN := false;
    POSITION       : TRACK_TUPLE;
    PREVIOUS       : TRACK_TUPLE;
    CURRENT        : TRACK_TUPLE;

  begin --DELETE_TRACK
    POSITION := TRACKS;
    PREVIOUS := null;
    while  POSITION /= null and not TRACK_IS_FOUND loop
      if POSITION.ID = TARGET_ID then
        TRACK_IS_FOUND := true;
        NUMBER_OF_TRACKS := NUMBER_OF_TRACKS - 1;
        if PREVIOUS = null then
          TRACKS := POSITION.NEXT;
        else
          PREVIOUS.NEXT := POSITION.NEXT;
        end if;
      else
        PREVIOUS := POSITION;
```

```
      POSITION := POSITION.NEXT;
    end if;
  end loop;
end DELETE_TRACK;


--*********************************************************************
--*** procedure PLACE_TRACK                                         ***
--*** algorithm : this procedure adds the new_track to TRACKS       ***
--***             so that the resultant TRACKS link list have       ***
--***             AIR tracks followed by SURFACE tracks,            ***
--***             followed by SUBSURFACE tracks. Also the track     ***
--***             with the same TRACK_CLASS but with smaller        ***
--***             RANGE comes before the other track  in TRACKS     ***
--***             link list                                         ***
--*********************************************************************
procedure PLACE_TRACK(
           TRACKS    : in out TRACK_TUPLE;
              NEW_TRACK : in     TRACK_TUPLE ) is

  CURRENT  : TRACK_TUPLE := TRACKS;
  PREVIOUS : TRACK_TUPLE := null;

begin --PLACE_TRACK
  if CURRENT = null then
    --currently, there is no track in TRACKS, so assign it to
    --new track
    TRACKS := NEW_TRACK;
  elsif CURRENT.TRACK_CLASS > NEW_TRACK.TRACK_CLASS then
    --new track's position is before the head of the tracks list
    --according to their track_class in the track_class_type,
    --so add new_track to the head of the TRACKS
    NEW_TRACK.NEXT := TRACKS;
    TRACKS := NEW_TRACK;
  elsif (CURRENT.TRACK_CLASS <= NEW_TRACK.TRACK_CLASS) then
    --new track's position is after the head of the tracks list
    --according to their track_class in the track_class_type,
    --so find a suitable position for the new_track, so that it
    -- will satisfy the condition, described at the algorithm
    --of the procedure
    while ( CURRENT /= null ) and then
```

```
            ( CURRENT.TRACK_CLASS < NEW_TRACK.TRACK_CLASS ) loop
          PREVIOUS := CURRENT;
          CURRENT := CURRENT.NEXT;
        end loop;
        if CURRENT = null then
          PREVIOUS.NEXT := NEW_TRACK;
        else
          while ( CURRENT /= null ) and then
          ( CURRENT.TRACK_CLASS = NEW_TRACK.TRACK_CLASS ) loop
            if CURRENT.THE_RANGE < NEW_TRACK.THE_RANGE then
              PREVIOUS := CURRENT;
              CURRENT := CURRENT.NEXT;
            else
              exit;
            end if;
          end loop;
          if CURRENT = null then
            PREVIOUS.NEXT := NEW_TRACK;
          elsif PREVIOUS = null then
            NEW_TRACK.NEXT := TRACKS;
            TRACKS := NEW_TRACK;
          else
            PREVIOUS.NEXT := NEW_TRACK;
            NEW_TRACK.NEXT := CURRENT;
          end if;
        end if;
      end if;
    end PLACE_TRACK;


    --*************************************************************************
    --*** procedure DELETE_LAST                                             ***
    --*************************************************************************
    procedure DELETE_LAST( TRACKS : in out TRACK_TUPLE ) is

      CURRENT  : TRACK_TUPLE := TRACKS;
      PREVIOUS : TRACK_TUPLE := null;

    begin --DELETE_LAST
      while CURRENT.NEXT /= null loop
        PREVIOUS := CURRENT;
```

```
      CURRENT := CURRENT.NEXT;
    end loop;
    PREVIOUS.NEXT := null;
  end DELETE_LAST;


--*************************************************************************
--*** procedure ADD_COMMS_TRACK                                         ***
--*************************************************************************
procedure ADD_COMMS_TRACK(
             FILTERED_COMMS_TRACK : in     ADD_TRACK_TUPLE;
             TDD_FILTER           : in     SET_TRACK_FILTER;
             OUT_TRACKS           : in out TRACK_TUPLE ) is

  CURRENT   : TRACK_TUPLE := FILTERED_COMMS_TRACK.TRACK;
  NEW_TRACK : TRACK_TUPLE;

begin --ADD_COMMS_TRACK
  while CURRENT /= null loop
    NEW_TRACK := new TRACK_RECORD;
    NEW_TRACK.ALL := CURRENT.ALL;
    CURRENT := CURRENT.NEXT;
    NEW_TRACK.NEXT := null;
    if NUMBER_OF_TRACKS < TDD_FILTER.MAX_NUMBER then
      NUMBER_OF_TRACKS := NUMBER_OF_TRACKS + 1;
    else
      DELETE_LAST( TRACKS );
    end if;
    DELETE_TRACK( TRACKS, NEW_TRACK.ID );
    PLACE_TRACK( TRACKS, NEW_TRACK );
  end loop;
  OUT_TRACKS := new TRACK_RECORD;
  OUT_TRACKS.ALL := TRACKS.ALL;
end ADD_COMMS_TRACK;


--*************************************************************************
--*** procedure SENSORS                                                 ***
--*************************************************************************
procedure SENSORS(    SENSOR_DATA : out SENSOR_RECORD ) is
```

```
SECONDS_SINCE_MIDNIGHT : NATURAL;
NEW_ECHO               : SENSOR_RECORD;
PREVIOUS               : SENSOR_TRACK_LIST;
SELECTED_TRACK         : SENSOR_TRACK_LIST;
SELECTED_TRACK_ID      : NATURAL;
SENSOR_TRACK_CAPACITY  : constant NATURAL := 15;
PI                     : constant FLOAT   := 3.1415926536;
RADIANS_PER_DEGREE     : constant FLOAT   := PI / 180.0;
DEGREES_PER_RADIAN     : constant FLOAT := 1.0 / RADIANS_PER_DEGREE;


--*********************************************************************
--*** procedure CREATE_SENSOR_TRACKS                                ***
--*********************************************************************
procedure CREATE_SENSOR_TRACKS(
          SENSOR_TRACKS          : in out SENSOR_TRACK_LIST;
          SECONDS_SINCE_MIDNIGHT : in     NATURAL ) is

  NEW_TRACK                   : SENSOR_TRACK_LIST;
  RANDOM_NUMBER               : FLOAT;
  --constant attributes for the tracks
  MIN_AIR_HEIGHT              : constant FLOAT := 3000.0;
  MIN_AIR_VELOCITY            : constant FLOAT := 300.0;
  MAX_AIR_VELOCITY            : constant FLOAT := 800.0;
  AIR_VELOCITY_DIFF           : constant FLOAT
                             := MAX_AIR_VELOCITY - MIN_AIR_VELOCITY;
  MIN_SURFACE_VELOCITY        : constant FLOAT := 0.0;
  MAX_SURFACE_VELOCITY        : constant FLOAT := 40.0;
  MAX_HEADING_ANGLE           : constant FLOAT := 360.0;
  MAX_SENSOR_RANGE            : constant FLOAT := 500.0;
  MIN_SUBSURFACE_DEPTH        : constant FLOAT := 0.0;
  MIN_SUBSURFACE_VELOCITY     : constant FLOAT := 0.0;
  MAX_SUBSURFACE_VELOCITY     : constant FLOAT := 30.0;
  MIN_ELEVATION               : constant FLOAT := -90.0;
  MAX_ELEVATION               : constant FLOAT :=  90.0;
  MIN_AZIMUTH                 : constant FLOAT := -180.0;
  MAX_AZIMUTH                 : constant FLOAT :=  180.0;
  MAX_BEARING_ANGLE           : constant FLOAT
                                  := MAX_AZIMUTH - MIN_AZIMUTH;
  AIR_TRACK_REBIRTH_RATIO     : constant FLOAT := 0.2;
  SURFACE_TRACK_REBIRTH_RATIO : constant FLOAT := 0.8;
```

```
  MILES_PER_DEGREE              : constant FLOAT := 60.0;

begin --CREATE_SENSOR_TRACKS
  for I in 11 .. SENSOR_TRACK_CAPACITY loop
    NEW_TRACK             := new SENSOR_TRACK_RECORD;
    NEW_TRACK.CONTACT.ID := I;
    RANDOM_NUMBER         := RANDOM.NUMBER;
    if RANDOM_NUMBER < AIR_TRACK_REBIRTH_RATIO then
      --create air track
      NEW_TRACK.CONTACT.TRACK_CLASS := AIR;
      NEW_TRACK.CONTACT.ELEVATION   := MAX_ELEVATION *
                                                    RANDOM.NUMBER;
      NEW_TRACK.CONTACT.VELOCITY    := MIN_AIR_VELOCITY +
                              ( RANDOM.NUMBER * AIR_VELOCITY_DIFF );
    elsif RANDOM_NUMBER < SURFACE_TRACK_REBIRTH_RATIO then
      --create surface track
      NEW_TRACK.CONTACT.TRACK_CLASS := SURFACE;
      NEW_TRACK.CONTACT.ELEVATION   := 0.0;
      NEW_TRACK.CONTACT.VELOCITY    := MIN_SURFACE_VELOCITY +
                           ( RANDOM.NUMBER * MAX_SURFACE_VELOCITY );
    else
      --create subsurface track
      NEW_TRACK.CONTACT.TRACK_CLASS := SUBSURFACE;
      NEW_TRACK.CONTACT.ELEVATION   := MIN_ELEVATION *
                                                  RANDOM.NUMBER ;
      NEW_TRACK.CONTACT.VELOCITY    := MIN_SUBSURFACE_VELOCITY +
                         ( RANDOM.NUMBER * MAX_SUBSURFACE_VELOCITY );
    end if;
    NEW_TRACK.CONTACT.COURSE    := RANDOM.NUMBER *
                                                MAX_HEADING_ANGLE;
    NEW_TRACK.CONTACT.THE_RANGE := RANDOM.NUMBER * MAX_SENSOR_RANGE;

    NEW_TRACK.CONTACT.AZIMUTH   := MIN_AZIMUTH +
                               (RANDOM.NUMBER * MAX_BEARING_ANGLE);
    NEW_TRACK.CONTACT.ORIGIN    := "SENSOR  ";
    NEW_TRACK.CONTACT.THE_TIME.HOURS   :=
                                        SECONDS_SINCE_MIDNIGHT/3600;
    NEW_TRACK.CONTACT.THE_TIME.MINUTES :=
                             (SECONDS_SINCE_MIDNIGHT mod 3600) / 60;
    NEW_TRACK.CONTACT.THE_TIME.SECONDS :=
                           (SECONDS_SINCE_MIDNIGHT mod 3600) mod 60;
```

```
    RANDOM_NUMBER := RANDOM.NUMBER;
    if    RANDOM_NUMBER < 0.25 then
      NEW_TRACK.CONTACT.IFF_CLASS := FRIENDLY;
    elsif RANDOM_NUMBER < 0.50 then
      NEW_TRACK.CONTACT.IFF_CLASS := HOSTILE;
    elsif RANDOM_NUMBER < 0.75 then
      NEW_TRACK.CONTACT.IFF_CLASS := NEUTRAL;
    else
      NEW_TRACK.CONTACT.IFF_CLASS := UNKNOWN;
    end if;
    NEW_TRACK.CONTACT.ARCHIVE_FLAG := C;
    NEW_TRACK.INTELLIGENCE( 1..40) :=
                        "THIS IS INTELLIGENCE                    ";
    NEW_TRACK.INTELLIGENCE(41..80) :=
                        "                                        ";
    NEW_TRACK.LATITUDE := 35.0 + (NEW_TRACK.CONTACT.THE_RANGE *
                COS(NEW_TRACK.CONTACT.AZIMUTH*RADIANS_PER_DEGREE))/
                MILES_PER_DEGREE;
    if NEW_TRACK.CONTACT.AZIMUTH > 0.0 then
      NEW_TRACK.LONGITUDE := 125.0 + (NEW_TRACK.CONTACT.THE_RANGE *
                SIN(NEW_TRACK.CONTACT.AZIMUTH*RADIANS_PER_DEGREE))/
                MILES_PER_DEGREE;
    else
      NEW_TRACK.LONGITUDE := 125.0 - (NEW_TRACK.CONTACT.THE_RANGE *
                SIN(NEW_TRACK.CONTACT.AZIMUTH*RADIANS_PER_DEGREE))/
                MILES_PER_DEGREE;
    end if;
    --add  NEW_TRACK to the head of the SENSOR_TRACKS
    NEW_TRACK.NEXT := SENSOR_TRACKS;
    SENSOR_TRACKS  := NEW_TRACK;
  end loop;
end CREATE_SENSOR_TRACKS;


--*************************************************************************
--*** procedure UPDATE_SENSOR_TRACK                                     ***
--*************************************************************************
procedure UPDATE_SENSOR_TRACK(
          SELECTED_TRACK         : in out SENSOR_TRACK_LIST;
          SECONDS_SINCE_MIDNIGHT : in     NATURAL ) is
```

```
PREVIOUS_SECONDS_SINCE_MIDNIGHT : NATURAL;
SECONDS_PASSED                  : NATURAL;
OWN_LAT          : FLOAT range -90.0..90.0 := 35.0;
OWN_LONG         : constant FLOAT := 125.0;
OWN_COURSE       : FLOAT := 0.0;
OWN_VELOCITY     : FLOAT := 25.0;
DISTANCE         : FLOAT;
LAT_DIST         : FLOAT;
LONG_DIST        : FLOAT;
MILES_PER_DEGREE : constant FLOAT := 60.0;

begin --UPDATE_SENSOR_TRACK
--get the the time difference between consecutive echos of
--the track in seconds
PREVIOUS_SECONDS_SINCE_MIDNIGHT   :=
                3600  * SELECTED_TRACK.CONTACT.THE_TIME.HOURS +
                 60   * SELECTED_TRACK.CONTACT.THE_TIME.MINUTES +
                        SELECTED_TRACK.CONTACT.THE_TIME.SECONDS;
SECONDS_PASSED := SECONDS_SINCE_MIDNIGHT -
                                  PREVIOUS_SECONDS_SINCE_MIDNIGHT;
--write the new observation time of the track to the
--track's record
SELECTED_TRACK.CONTACT.THE_TIME.HOURS   :=
                                       SECONDS_SINCE_MIDNIGHT/3600;
SELECTED_TRACK.CONTACT.THE_TIME.MINUTES :=
                           (SECONDS_SINCE_MIDNIGHT mod 3600) / 60;
SELECTED_TRACK.CONTACT.THE_TIME.SECONDS :=
                          (SECONDS_SINCE_MIDNIGHT mod 3600) mod 60;
--update own_ship's latitude
OWN_LAT := 35.0 +
((FLOAT(SECONDS_PASSED)*OWN_VELOCITY) / 3600.0)/MILES_PER_DEGREE;
--calculate the distance that the contact has went
DISTANCE :=
  (FLOAT(SECONDS_PASSED)*SELECTED_TRACK.CONTACT.VELOCITY)/3600.0;
--update selected_track's latitude
SELECTED_TRACK.LATITUDE := SELECTED_TRACK.LATITUDE + (DISTANCE*
    COS(SELECTED_TRACK.CONTACT.COURSE*RADIANS_PER_DEGREE))/
     MILES_PER_DEGREE;
--update selected_track's longitude
SELECTED_TRACK.LONGITUDE := SELECTED_TRACK.LONGITUDE + (DISTANCE*
    SIN(SELECTED_TRACK.CONTACT.COURSE*RADIANS_PER_DEGREE))/
```

```
        MILES_PER_DEGREE;
    --change the range of the track & write to the track's record
    LAT_DIST  := ( SELECTED_TRACK.LATITUDE - OWN_LAT ) *
        MILES_PER_DEGREE;
    LONG_DIST := ( SELECTED_TRACK.LONGITUDE - OWN_LONG ) *
        MILES_PER_DEGREE;
    SELECTED_TRACK.CONTACT.THE_RANGE :=
                                    SQRT( LAT_DIST**2 + LONG_DIST**2 );
    --update selected_track's range
    SELECTED_TRACK.CONTACT.AZIMUTH := DEGREES_PER_RADIAN *
        ARCCOS(LAT_DIST/SELECTED_TRACK.CONTACT.THE_RANGE);
  end UPDATE_SENSOR_TRACK;

begin --SENSORS
  SECONDS_SINCE_MIDNIGHT := NATURAL(CALENDAR.SECONDS(CALENDAR.CLOCK));
  if not SENSOR_TRACKS_ARE_CREATED then
    CREATE_SENSOR_TRACKS( SENSOR_TRACKS, SECONDS_SINCE_MIDNIGHT );
    SENSOR_TRACKS_ARE_CREATED := true;
  end if;
  --begin from the head of the list each time to search
  SELECTED_TRACK_ID := 0;
  SELECTED_TRACK    := SENSOR_TRACKS;
  PREVIOUS          := null;
  while SELECTED_TRACK_ID < 11 loop
    -- because sensor tracks start with id 11
    SELECTED_TRACK_ID :=
              NATURAL( RANDOM.NUMBER*FLOAT( SENSOR_TRACK_CAPACITY ) );
  end loop;
  while SELECTED_TRACK.NEXT /= null and
       SELECTED_TRACK.CONTACT.ID /= SELECTED_TRACK_ID loop
    PREVIOUS := SELECTED_TRACK;
    SELECTED_TRACK := SELECTED_TRACK.NEXT;
  end loop;
  UPDATE_SENSOR_TRACK( SELECTED_TRACK, SECONDS_SINCE_MIDNIGHT );
  SENSOR_DATA.INTELLIGENCE := SELECTED_TRACK.INTELLIGENCE;
  SENSOR_DATA.CONTACT := SELECTED_TRACK.CONTACT;
end SENSORS;
```

```
--*************************************************************************
--*** procedure ANALYZE_SENSOR_DATA                                     ***
--*************************************************************************
procedure ANALYZE_SENSOR_DATA(
            SENSOR_DATA         : in  SENSOR_RECORD;
            SENSOR_CONTACT_DATA : out LOCAL_TRACK_INFO ) is
begin --ANALYZE_SENSOR_DATA
  SENSOR_CONTACT_DATA := SENSOR_DATA.CONTACT;
end ANALYZE_SENSOR_DATA;


--*************************************************************************
--*** procedure PREPARE_SENSOR_TRACK                                    ***
--*************************************************************************
procedure PREPARE_SENSOR_TRACK(
        SENSOR_CONTACT_DATA : in  LOCAL_TRACK_INFO;
        POSITION_DATA       : in  OWNSHIP_NAVIGATION_INFO;
        SENSOR_ADD_TRACK    : out ADD_TRACK_TUPLE ) is

  HEAD               : TRACK_TUPLE;
  NEW_TRACK          : TRACK_TUPLE;
  PI                 : constant FLOAT := 3.1415926536;
  RADIANS_PER_DEGREE : constant FLOAT := PI / 180.0;
  MILES_PER_DEGREE   : constant FLOAT := 60.0;


begin --PREPARE_SENSOR_TRACK
  SENSOR_ADD_TRACK.ORIGIN     := "SENSOR  ";
  HEAD                        := null;
  NEW_TRACK                   := new TRACK_RECORD;
  NEW_TRACK.ID                := SENSOR_CONTACT_DATA.ID;
  NEW_TRACK.OBSERVER          := SENSOR_CONTACT_DATA.ORIGIN;
  NEW_TRACK.OBSERVATION_TIME  := SENSOR_CONTACT_DATA.THE_TIME;
  NEW_TRACK.TRACK_CLASS       := SENSOR_CONTACT_DATA.TRACK_CLASS;
  NEW_TRACK.IFF_CLASS         := SENSOR_CONTACT_DATA.IFF_CLASS;

  NEW_TRACK.LATITUDE          := POSITION_DATA.LATITUDE +
        (SENSOR_CONTACT_DATA.THE_RANGE *
        COS(RADIANS_PER_DEGREE*SENSOR_CONTACT_DATA.AZIMUTH)) /
        MILES_PER_DEGREE;
  NEW_TRACK.LONGITUDE         := POSITION_DATA.LONGITUDE +
```

```
        ( SENSOR_CONTACT_DATA.THE_RANGE *
        SIN(RADIANS_PER_DEGREE*SENSOR_CONTACT_DATA.AZIMUTH))/
        MILES_PER_DEGREE;
  NEW_TRACK.ALTITUDE          := SENSOR_CONTACT_DATA.THE_RANGE *
        SIN(RADIANS_PER_DEGREE*SENSOR_CONTACT_DATA.ELEVATION);
  NEW_TRACK.COURSE            := SENSOR_CONTACT_DATA.COURSE;
  NEW_TRACK.VELOCITY          := SENSOR_CONTACT_DATA.VELOCITY;
  NEW_TRACK.THE_RANGE         := SENSOR_CONTACT_DATA.THE_RANGE;
  NEW_TRACK.ARCHIVE_FLAG      := SENSOR_CONTACT_DATA.ARCHIVE_FLAG;
  NEW_TRACK.NEXT              := HEAD;
  HEAD                        := NEW_TRACK;
  SENSOR_ADD_TRACK.TRACK      := HEAD;
end PREPARE_SENSOR_TRACK;


--************************************************************************
--*** procedure FILTER_SENSOR_TRACKS                                   ***
--************************************************************************
procedure FILTER_SENSOR_TRACKS(
            SENSOR_ADD_TRACK      : in out  ADD_TRACK_TUPLE;
            TDD_FILTER            : in      SET_TRACK_FILTER;
            FILTERED_SENSOR_TRACK : out     ADD_TRACK_TUPLE ) is

  HEAD     : TRACK_TUPLE := SENSOR_ADD_TRACK.TRACK;
  CURRENT  : TRACK_TUPLE := SENSOR_ADD_TRACK.TRACK;
  PREVIOUS : TRACK_TUPLE := null;

begin --FILTER_SENSOR_TRACKS
  while CURRENT /= null loop
    if (TDD_FILTER.DESIRED_CLASS(CURRENT.TRACK_CLASS)) and
       (TDD_FILTER.DESIRED_RANGE(CURRENT.TRACK_CLASS) >=
                                          CURRENT.THE_RANGE) then
       PREVIOUS := CURRENT;
       CURRENT := CURRENT.NEXT;
    else
      if PREVIOUS = null then
        HEAD := CURRENT.NEXT;
      else
        PREVIOUS.NEXT := CURRENT.NEXT;
      end if;
      CURRENT := CURRENT.NEXT;
```

```
    end if;
  end loop;
  if HEAD = null then
    APPROVED := false;
  end if;
  FILTERED_SENSOR_TRACK.TRACK := HEAD;
end FILTER_SENSOR_TRACKS;


--************************************************************************
--*** procedure ADD_SENSOR_TRACK                                       ***
--************************************************************************
procedure ADD_SENSOR_TRACK(
             FILTERED_SENSOR_TRACK : in     ADD_TRACK_TUPLE;
             TDD_FILTER            : in     SET_TRACK_FILTER;
             OUT_TRACKS            : in out TRACK_TUPLE ) is

  CURRENT   : TRACK_TUPLE;
  NEW_TRACK : TRACK_TUPLE;

begin --ADD_SENSOR_TRACK
  CURRENT := FILTERED_SENSOR_TRACK.TRACK;
  while CURRENT /= null loop
    NEW_TRACK      := new TRACK_RECORD;
    NEW_TRACK.ALL  := CURRENT.ALL;
    CURRENT        := CURRENT.NEXT;
    NEW_TRACK.NEXT := null;
    if NUMBER_OF_TRACKS < TDD_FILTER.MAX_NUMBER then
      NUMBER_OF_TRACKS := NUMBER_OF_TRACKS + 1;
    else
      DELETE_LAST( TRACKS );
    end if;
    DELETE_TRACK( TRACKS, NEW_TRACK.ID );
    PLACE_TRACK( TRACKS, NEW_TRACK );
  end loop;
  OUT_TRACKS := new TRACK_RECORD;
  OUT_TRACKS.ALL := TRACKS.ALL;
end ADD_SENSOR_TRACK;
```

```
--*************************************************************************
--*** procedure NAVIGATION_SYSTEM                                       ***
--*************************************************************************
procedure NAVIGATION_SYSTEM(
        POSITION_DATA : out OWNSHIP_NAVIGATION_INFO ) is

CONSTANT_COURSE         : FLOAT := 0.0;
CONSTANT_VELOCITY       : FLOAT := 25.0;
START_LATITUDE          : FLOAT := 35.0;
START_LONGITUDE         : FLOAT := 125.0;
SECONDS_PASSED          : NATURAL;
SECONDS_SINCE_MIDNIGHT  : NATURAL;
MILES_PER_DEGREE        : constant FLOAT := 60.0;

begin --NAVIGATION_SYSTEM
  SECONDS_SINCE_MIDNIGHT:=
              NATURAL( FLOAT( CALENDAR.SECONDS( CALENDAR.CLOCK ) ) );
  if START_TIME = 0 then
    START_TIME := SECONDS_SINCE_MIDNIGHT;
  end if;
  SECONDS_PASSED           := SECONDS_SINCE_MIDNIGHT - START_TIME;
  POSITION_DATA.COURSE     := CONSTANT_COURSE;
  POSITION_DATA.VELOCITY   := CONSTANT_VELOCITY;
  POSITION_DATA.LONGITUDE  := START_LONGITUDE;
  POSITION_DATA.LATITUDE   := START_LATITUDE +
        (( FLOAT(SECONDS_PASSED)/3600.0) * CONSTANT_VELOCITY)/
        MILES_PER_DEGREE;
  POSITION_DATA.THE_TIME.HOURS   := SECONDS_SINCE_MIDNIGHT/3600;
  POSITION_DATA.THE_TIME.MINUTES :=
                              ( SECONDS_SINCE_MIDNIGHT mod 3600 )/ 60;
  POSITION_DATA.THE_TIME.SECONDS :=
                             (SECONDS_SINCE_MIDNIGHT mod 3600) mod 60;
end NAVIGATION_SYSTEM;


--*************************************************************************
--*** procedure MONITOR_OWNSHIP_POSITION                                ***
--*************************************************************************
procedure MONITOR_OWNSHIP_POSITION(
            POSITION_DATA : in     OWNSHIP_NAVIGATION_INFO;
            OUT_TRACKS    : in out TRACK_TUPLE ) is
```

```
OWNSHIP : TRACK_TUPLE;


--*********************************************************************
--*** procedure DELETE_OWNSHIP                                      ***
--*********************************************************************
procedure DELETE_OWNSHIP is
  OWNSHIP_IS_FOUND : BOOLEAN := false;
  CURRENT          : TRACK_TUPLE;
  PREVIOUS         : TRACK_TUPLE;
begin --MONITOR_OWNSHIP_POSITION
  CURRENT := TRACKS;
  PREVIOUS := null;
  OWNSHIP_IS_FOUND := false;
  while CURRENT /= null and not OWNSHIP_IS_FOUND loop
    if CURRENT.ID = 0 then
      OWNSHIP_IS_FOUND := true;
      NUMBER_OF_TRACKS := NUMBER_OF_TRACKS - 1;
      if PREVIOUS = null then
        TRACKS := CURRENT.NEXT;
      else
        PREVIOUS.NEXT := CURRENT.NEXT;
      end if;
    else
      PREVIOUS := CURRENT;
      CURRENT := CURRENT.NEXT;
    end if;
  end loop;
end DELETE_OWNSHIP;


--*********************************************************************
--*** procedure PLACE_OWNSHIP                                       ***
--*********************************************************************
procedure PLACE_OWNSHIP(
           TRACKS    : in out TRACK_TUPLE;
           NEW_TRACK : in     TRACK_TUPLE ) is

  CURRENT   : TRACK_TUPLE := TRACKS;
  PREVIOUS  : TRACK_TUPLE := null;
```

```
begin --PLACE_OWNSHIP
  if CURRENT = null then
    --currently, there is no track in TRACKS, so assign it
    --to ownship
    TRACKS := NEW_TRACK;
  elsif (CURRENT.TRACK_CLASS > NEW_TRACK.TRACK_CLASS) then
    --ownship's position is before the head of the tracks list
    --according to their track_class in the track_class_type,
    --so add new_track to the head of the TRACKS
    NEW_TRACK.NEXT := TRACKS;
    TRACKS := NEW_TRACK;
  elsif (CURRENT.TRACK_CLASS <= NEW_TRACK.TRACK_CLASS) then
    --ownship's position is after the head of the tracks list
    --according to their track_class in the track_class_type,
    --so find a suitable position for the ownship, so that it
    --will satisfy the condition, described at the algorithm
    --of the procedure
    while ( CURRENT /= null ) and then
          ( CURRENT.TRACK_CLASS < NEW_TRACK.TRACK_CLASS ) loop
      PREVIOUS := CURRENT;
      CURRENT := CURRENT.NEXT;
    end loop;
    if CURRENT = null then
      PREVIOUS.NEXT := NEW_TRACK;
    else
      while ( CURRENT /= null ) and then
            ( CURRENT.TRACK_CLASS = NEW_TRACK.TRACK_CLASS ) loop
        if CURRENT.THE_RANGE < NEW_TRACK.THE_RANGE then
          PREVIOUS := CURRENT;
          CURRENT := CURRENT.NEXT;
        else
          exit;
        end if;
      end loop;
      if CURRENT = null then
        PREVIOUS.NEXT := NEW_TRACK;
      elsif PREVIOUS = null then
        NEW_TRACK.NEXT := TRACKS;
        TRACKS := NEW_TRACK;
      else
        PREVIOUS.NEXT := NEW_TRACK;
```

```
          NEW_TRACK.NEXT := CURRENT;
        end if;
      end if;
    end if;
  end PLACE_OWNSHIP;

begin --MONITOR_OWNSHIP_POSITION

  DELETE_OWNSHIP;
  OWNSHIP                    := new TRACK_RECORD;
  OWNSHIP.ID                 := 0;
  OWNSHIP.OBSERVER           := "NVSYSTEM";
  OWNSHIP.OBSERVATION_TIME   := POSITION_DATA.THE_TIME;
  OWNSHIP.TRACK_CLASS        := OWNSHIP_TRACK_CLASS;
  OWNSHIP.IFF_CLASS          := OWNSHIP_IFF_CLASS;
  OWNSHIP.LATITUDE           := POSITION_DATA.LATITUDE;
  OWNSHIP.LONGITUDE          := POSITION_DATA.LONGITUDE;
  OWNSHIP.COURSE             := POSITION_DATA.COURSE;
  OWNSHIP.VELOCITY           := POSITION_DATA.VELOCITY;
  OWNSHIP.ARCHIVE_FLAG       := C;
  OWNSHIP.NEXT               := null;
  PLACE_OWNSHIP( TRACKS, OWNSHIP );
  NUMBER_OF_TRACKS := NUMBER_OF_TRACKS + 1;
  OUT_TRACKS       := new TRACK_RECORD;
  OUT_TRACKS.ALL   := TRACKS.ALL;
end MONITOR_OWNSHIP_POSITION;


--************************************************************************
--*** procedure WEAPONS_SYSTEMS                                        ***
--************************************************************************
procedure WEAPONS_SYSTEMS(
        WEAPON_STATUS_DATA  : out WEAPON_STATUS ) is

  RANDOM_NUMBER : FLOAT;

begin --WEAPONS_SYSTEMS
  RANDOM_NUMBER := RANDOM.NUMBER;
  if    RANDOM_NUMBER < 0.001 then
    WEAPON_STATUS_DATA.STATUS := DAMAGED;
  elsif RANDOM_NUMBER < 0.01  then
```

```
    WEAPON_STATUS_DATA.STATUS := SERVICE_REQUIRED;
  elsif RANDOM_NUMBER < 0.1   then
    WEAPON_STATUS_DATA.STATUS := LAUNCHING;
  elsif RANDOM_NUMBER < 0.2   then
    WEAPON_STATUS_DATA.STATUS := OUT_OF_AMMUNITION;
  elsif RANDOM_NUMBER < 0.3   then
    WEAPON_STATUS_DATA.STATUS := SLEWING;
  elsif RANDOM_NUMBER < 0.4   then
    WEAPON_STATUS_DATA.STATUS := SECURED;
  elsif RANDOM_NUMBER < 0.5   then
    WEAPON_STATUS_DATA.STATUS := MAINTANENCE;
  elsif RANDOM_NUMBER < 0.55  then
    WEAPON_STATUS_DATA.STATUS := ENGAGING;
  elsif RANDOM_NUMBER < 0.6   then
    WEAPON_STATUS_DATA.STATUS := RELOADING;
  else
    WEAPON_STATUS_DATA.STATUS := READY;
  end if;
  RANDOM_NUMBER := RANDOM.NUMBER;
  if    RANDOM_NUMBER < 0.25 then WEAPON_STATUS_DATA.SYS_TYPE := CIWS;
  elsif RANDOM_NUMBER < 0.50 then WEAPON_STATUS_DATA.SYS_TYPE := GUN;
  elsif RANDOM_NUMBER < 0.75 then WEAPON_STATUS_DATA.SYS_TYPE := TWS;
  else                            WEAPON_STATUS_DATA.SYS_TYPE := MK48;
  end if;
end WEAPONS_SYSTEMS;


--************************************************************************
--*** procedure WEAPONS_INTERFACE                                      ***
--************************************************************************
procedure WEAPONS_INTERFACE(
             WEAPON_STATUS_DATA : in  WEAPON_STATUS;
             WEAPONS_EMREP      : out WEAPON_STATUS_REPORT;
             WEAPONS_STATREP    : out WEAPON_STATUS_REPORT ) is
begin --WEAPONS_INTERFACE
  --assign the status to the related weapons states
  if   WEAPON_STATUS_DATA.SYS_TYPE = CIWS then
    CIWS_STATUS := WEAPON_STATUS_DATA.STATUS;
  elsif WEAPON_STATUS_DATA.SYS_TYPE = GUN  then
    GUN_STATUS := WEAPON_STATUS_DATA.STATUS;
  elsif WEAPON_STATUS_DATA.SYS_TYPE = TWS  then
```

```
    TWS_STATUS := WEAPON_STATUS_DATA.STATUS;
  else
    MK48_STATUS := WEAPON_STATUS_DATA.STATUS;
  end if;
  --assign the weapons states to the related status report component
  WEAPONS_STATREP(CIWS) := CIWS_STATUS;
  WEAPONS_STATREP(GUN)  := GUN_STATUS;
  WEAPONS_STATREP(TWS)  := TWS_STATUS;
  WEAPONS_STATREP(MK48) := MK48_STATUS;
  --check whetеher there is an emeregency situation
  if ( WEAPON_STATUS_DATA.STATUS = DAMAGED           ) or
     ( WEAPON_STATUS_DATA.STATUS = SERVICE_REQUIRED  ) or
     ( WEAPON_STATUS_DATA.STATUS = OUT_OF_AMMUNITION ) then
    --build the emergency weapons status
    WEAPONS_EMREP(CIWS) := CIWS_STATUS;
    WEAPONS_EMREP(GUN)  := GUN_STATUS;
    WEAPONS_EMREP(TWS)  := TWS_STATUS;
    WEAPONS_EMREP(MK48) := MK48_STATUS;
  end if;
end WEAPONS_INTERFACE;


--************************************************************************
--*** procedure MAKE_ROUTING                                           ***
--************************************************************************
procedure MAKE_ROUTING(
             TCD_TRANSMIT_COMMAND : in     TRANSMIT_COMMAND;
             TCD_NETWORK_SETUP    : in     NETWORK_SETUP;
             TRANSMISSION_MESSAGE : in out TRANSMISSION_COMMAND ) is

  ADDRESS_FOUND_IN_NETWORK_SETUP : BOOLEAN := false;
  TRANSMIT_CURRENT               : ADDRESS_LINK;
  NETWORK_CURRENT                : ADDRESS_LINK;
  TRANSMISSION_LINK              : LINKS_TYPE;
  NEW_ADDRESS                    : ADDRESS_LINK;

begin --MAKE_ROUTING
  for TRANSMIT_LINK in LINKS_TYPE loop
    --search through all array components of the TCD_TRANSMIT_COMMAND
    if TCD_TRANSMIT_COMMAND(TRANSMIT_LINK).FULL
    and not TCD_TRANSMIT_COMMAND(TRANSMIT_LINK).ROUTED then
```

```
--there is a file in the current component of
--TCD_TRANSMIT_COMMAND, and it needs to be routed
--first, process the ADDRESS field
TRANSMIT_CURRENT :=
       TCD_TRANSMIT_COMMAND(TRANSMIT_LINK).TEXT.HEADER.ADDRESS;
while TRANSMIT_CURRENT /= null loop
  --search through TCD_TRANSMIT_COMMAND to find the
  TRANSMISSION_LINK := JTIDS;
  ADDRESS_FOUND_IN_NETWORK_SETUP  := false;
  while not ADDRESS_FOUND_IN_NETWORK_SETUP loop
    NETWORK_CURRENT := TCD_NETWORK_SETUP(TRANSMISSION_LINK);
    while NETWORK_CURRENT /= null and
          not ADDRESS_FOUND_IN_NETWORK_SETUP loop
      if NETWORK_CURRENT.NAME = TRANSMIT_CURRENT.NAME then
        ADDRESS_FOUND_IN_NETWORK_SETUP := true;
        NEW_ADDRESS       := new ADDRESS_TYPE;
        NEW_ADDRESS.NAME := TRANSMIT_CURRENT.NAME;
        NEW_ADDRESS.NEXT :=
             TRANSMISSION_MESSAGE(TRANSMISSION_LINK).ROUTE_ADDR;
        TRANSMISSION_MESSAGE(TRANSMISSION_LINK).ROUTE_ADDR :=
                                                  NEW_ADDRESS;
        TRANSMISSION_MESSAGE(TRANSMISSION_LINK).FULL := true;
        TRANSMISSION_MESSAGE(TRANSMISSION_LINK).TEXT :=
                        TCD_TRANSMIT_COMMAND(TRANSMIT_LINK).TEXT;
      end if;
      NETWORK_CURRENT := NETWORK_CURRENT.NEXT;
    end loop;
    if TRANSMISSION_LINK = OTCIXS then
      exit;
    else
      TRANSMISSION_LINK := LINKS_TYPE'SUCC(TRANSMISSION_LINK);
    end if;
  end loop;
  TRANSMIT_CURRENT := TRANSMIT_CURRENT.NEXT;
end loop;
--second, process the INFO field
TRANSMIT_CURRENT :=
          TCD_TRANSMIT_COMMAND(TRANSMIT_LINK).TEXT.HEADER.INFO;
while TRANSMIT_CURRENT /= null loop
  --search through TCD_TRANSMIT_COMMAND to find the
  TRANSMISSION_LINK := JTIDS;
```

```
        --initialize the boolean value again
        ADDRESS_FOUND_IN_NETWORK_SETUP  := false;
        while not ADDRESS_FOUND_IN_NETWORK_SETUP loop
          NETWORK_CURRENT := TCD_NETWORK_SETUP(TRANSMISSION_LINK);
          while NETWORK_CURRENT /= null and
                not ADDRESS_FOUND_IN_NETWORK_SETUP loop
            if NETWORK_CURRENT.NAME = TRANSMIT_CURRENT.NAME then
              ADDRESS_FOUND_IN_NETWORK_SETUP := true;
              --address is found, so add the file and other necessary
              --information to the related LINK component of
              --TRANSMISSION_MESSAGE where the address is staying
              NEW_ADDRESS      := new ADDRESS_TYPE;
              NEW_ADDRESS.NAME := TRANSMIT_CURRENT.NAME;
              NEW_ADDRESS.NEXT :=
                  TRANSMISSION_MESSAGE(TRANSMISSION_LINK).ROUTE_ADDR;
              TRANSMISSION_MESSAGE(TRANSMISSION_LINK).ROUTE_ADDR :=
                                                        NEW_ADDRESS;
              TRANSMISSION_MESSAGE(TRANSMISSION_LINK).FULL := true;
              TRANSMISSION_MESSAGE(TRANSMISSION_LINK).TEXT :=
                              TCD_TRANSMIT_COMMAND(TRANSMIT_LINK).TEXT;
            end if;
            --move forward in the NETWORK_SETUP
            NETWORK_CURRENT := NETWORK_CURRENT.NEXT;
          end loop;
          if TRANSMISSION_LINK = OTCIXS then
            --we traversed the whole TRANSMISSION_MESSAGE, so
            --exit from the loop
            exit;
          else
            --there is still remaining component in
            --TRANSMISSION_MESSAGE,
            TRANSMISSION_LINK := LINKS_TYPE'SUCC(TRANSMISSION_LINK);
          end if;
        end loop;
        TRANSMIT_CURRENT := TRANSMIT_CURRENT.NEXT;
      end loop;
    end if;
  end loop;
end MAKE_ROUTING;
```

```
--*************************************************************************
--*** procedure FORWARD_FOR_TRANSMISSION                                ***
--*************************************************************************
procedure FORWARD_FOR_TRANSMISSION(
              TRANSMISSION_MESSAGE : in     TRANSMISSION_COMMAND;
              TCD_EMISSION_CONTROL : in     EMISSIONS_CONTROL_COMMAND;
              OUTPUT_MESSAGES      : in out MESSAGE_LIST         ) is

  LINK        : LINKS_TYPE := JTIDS;
  NEW_MESSAGE : MESSAGE_LIST;

begin --FORWARD_FOR_TRANSMISSION
  for I in LINKS_TYPE loop
    if TRANSMISSION_MESSAGE(I).FULL then
      NEW_MESSAGE         := new MESSAGE_RECORD;
      NEW_MESSAGE.LINK_ID := I;
      NEW_MESSAGE.MAIN    := TRANSMISSION_MESSAGE(I);
      NEW_MESSAGE.NEXT    := WAITING_MESSAGES;
      WAITING_MESSAGES    := NEW_MESSAGE;
    end if;
  end loop;
  if TCD_EMISSION_CONTROL = UNRESTRICTED then
    OUTPUT_MESSAGES     :=  new MESSAGE_RECORD;
    OUTPUT_MESSAGES.ALL := WAITING_MESSAGES.ALL;
    WAITING_MESSAGES    := null;
  end if;
end FORWARD_FOR_TRANSMISSION;


--*************************************************************************
--*** procedure CONVERT_TO_TEXT_FILE                                    ***
--*************************************************************************
procedure CONVERT_TO_TEXT_FILE
        ( OUTPUT_MESSAGES : in  MESSAGE_LIST ) is

  package P_CLASS_IO is new ENUMERATION_IO( PRECEDENCE_CLASS );
  use P_CLASS_IO;
  package S_CLASS_IO is new ENUMERATION_IO( SECURITY_CLASS );
  use S_CLASS_IO;
  package LINKS_IO   is new ENUMERATION_IO( LINKS_TYPE );
  use LINKS_IO;
```

```
  CURRENT_MESSAGE : MESSAGE_LIST := OUTPUT_MESSAGES;
  FILE            : FILE_TYPE;
  ADDRESS_HEAD    : ADDRESS_LINK;
  VIA_HEAD        : VIA_RECORD_LINK;
  FILE_NAME       : STRING(1..17) := "                ";

begin --CONVERT_TO_TEXT_FILE
  while CURRENT_MESSAGE /= null loop
    --there is still at least one message to convert to text file
    --first, prepare the file name. the resultant file name
    --should have the form
    --"CURRENT_MESSAGE.LINK_ID & CURRENT_MESSAGE.MAIN.TEXT.NAME"
    if CURRENT_MESSAGE.LINK_ID = JTIDS then
      FILE_NAME(1..6) := "JTIDS.";
      FILE_NAME(7..16) := CURRENT_MESSAGE.MAIN.TEXT.NAME;
    elsif CURRENT_MESSAGE.LINK_ID = LINK11 then
      FILE_NAME(1..7) := "LINK11.";
      FILE_NAME(8..17) := CURRENT_MESSAGE.MAIN.TEXT.NAME;
    elsif CURRENT_MESSAGE.LINK_ID = LINK16 then
      FILE_NAME(1..7) := "LINK16.";
      FILE_NAME(8..17) := CURRENT_MESSAGE.MAIN.TEXT.NAME;
    else
      FILE_NAME(1..7) := "OTCIXS.";
      FILE_NAME(8..17) := CURRENT_MESSAGE.MAIN.TEXT.NAME;
    end if;
    --open the file to be filled in, and set the line length
    CREATE( FILE, NAME => FILE_NAME );
    SET_LINE_LENGTH( FILE, 80 );
    --print FILENAME info
    PUT ( FILE,"NAME : " );
    PUT( FILE, CURRENT_MESSAGE.MAIN.TEXT.NAME ); NEW_LINE( FILE );
    --print LINK_ID info
    PUT( FILE, "LINK ID : " );
    PUT( FILE, CURRENT_MESSAGE.LINK_ID ); NEW_LINE( FILE );
    --print ROUTE info
    PUT( FILE, "ROUTE : ");
    ADDRESS_HEAD := CURRENT_MESSAGE.MAIN.ROUTE_ADDR;
    while ADDRESS_HEAD /= null loop
      PUT( FILE, ADDRESS_HEAD.NAME );
      PUT( FILE, ", " );
      NEW_LINE( FILE );
```

```
  PUT( FILE, "        ");
  ADDRESS_HEAD := ADDRESS_HEAD.NEXT;
end loop;
NEW_LINE( FILE );
--print CLASSIFICATION info
PUT( FILE, "CLASSIFICATION : ");
PUT( FILE, CURRENT_MESSAGE.MAIN.TEXT.HEADER.CLASSIFICATION );
--print PRESEDENCE info
PUT( FILE, "  PRECEDENCE : " );
PUT( FILE, CURRENT_MESSAGE.MAIN.TEXT.HEADER.PRECEDENCE );
NEW_LINE( FILE );
--print origin(FM) info
PUT( FILE, "FM : " );
PUT( FILE, CURRENT_MESSAGE.MAIN.TEXT.HEADER.ORIGIN );
NEW_LINE( FILE );
--print address(TO) info
PUT( FILE, "TO : " );
ADDRESS_HEAD := CURRENT_MESSAGE.MAIN.TEXT.HEADER.ADDRESS;
while ADDRESS_HEAD /= null loop
  PUT( FILE, ADDRESS_HEAD.NAME );
  PUT( FILE, ", " );
  NEW_LINE( FILE );
  SET_COL( FILE, 6 );
  ADDRESS_HEAD := ADDRESS_HEAD.NEXT;
end loop;
NEW_LINE( FILE );
--print INFO info
PUT( FILE, "INFO : " );
ADDRESS_HEAD := CURRENT_MESSAGE.MAIN.TEXT.HEADER.INFO;
while ADDRESS_HEAD /= null loop
  PUT( FILE, ADDRESS_HEAD.NAME );
  PUT( FILE, ", " );
  NEW_LINE( FILE );
  SET_COL( FILE, 8 );
  ADDRESS_HEAD := ADDRESS_HEAD.NEXT;
end loop;
NEW_LINE( FILE );
--print VIA info
PUT( FILE, "VIA : " );
VIA_HEAD := CURRENT_MESSAGE.MAIN.TEXT.HEADER.VIA_LINE;
while VIA_HEAD /= null loop
```

```
      ADDRESS_HEAD := VIA_HEAD.RELAY_TO;
      while ADDRESS_HEAD /= null loop
        PUT( FILE, ADDRESS_HEAD.NAME );
        PUT( FILE, ", ");
        NEW_LINE( FILE );
        SET_COL( FILE, 7 );
        ADDRESS_HEAD := ADDRESS_HEAD.NEXT;
      end loop;
      PUT( FILE, "BY " );
      PUT( FILE, VIA_HEAD.RELAY_BY ); NEW_LINE( FILE );
      PUT( FILE, "      " );
      VIA_HEAD := VIA_HEAD.NEXT;
    end loop;
    NEW_LINE( FILE );
    --print SUBJECT info
    PUT( FILE, "SUBJECT : " );
    PUT( FILE, CURRENT_MESSAGE.MAIN.TEXT.HEADER.SUBJECT );
    NEW_LINE( FILE, 4 );
    --print body(TEXT) of the message
    PUT( FILE, CURRENT_MESSAGE.MAIN.TEXT.TEXT );
    NEW_LINE( FILE );
    --close the file
    CLOSE( FILE );
    --Do everyting for the all of the messages in the list
    CURRENT_MESSAGE := CURRENT_MESSAGE.NEXT;
  end loop;
end CONVERT_TO_TEXT_FILE;


--************************************************************************
--*** procedure MESSAGE_EDITOR                                         ***
--************************************************************************
procedure MESSAGE_EDITOR(
            EDITOR_SELECTED      : in  BOOLEAN;
            TCD_TRANSMIT_COMMAND : out TRANSMIT_COMMAND ) is
begin --MESSAGE_EDITOR
  TCD_TRANSMIT_COMMAND := NEW_TRANSMIT_COMMAND;
end MESSAGE_EDITOR;
```

```
--*************************************************************************
--*** procedure DISPLAY_TRACKS                                          ***
--*************************************************************************
procedure display_tracks(
            out_tracks       : in track_tuple;
            td_track_request : in database_request ) is

  current        : track_tuple := out_tracks;  --track to be displayed

  --item id's of the related panels
  id_id          : string(1..5)  := "id_00";
  observer_id    : string(1..11) := "observer_00";
  time_id        : string(1..7)  := "time_00";
  track_class_id : string(1..14) := "track_class_00";
  iff_class_id   : string(1..12) := "iff_class_00";
  latitude_id    : string(1..11) := "latitude_00";
  longitude_id   : string(1..12) := "longitude_00";
  altitude_id    : string(1..11) := "altitude_00";
  course_id      : string(1..9)  := "course_00";
  velocity_id    : string(1..11) := "velocity_00";
  range_id       : string(1..8)  := "range_00";

  --row of related track
  postfix        : string(1..2)  := "00";

  --string correspondance of time
  time_string    : string(1..8)  := "00:00:00";


  --*************************************************************************
  --*** Function CHARACTER_OF                                             ***
  --*************************************************************************
  function CHARACTER_OF( N : in NATURAL ) return CHARACTER is
  begin
    return( CHARACTER'VAL( N + 48 ) );
  end CHARACTER_OF;
```

```
--*************************************************************************
--*** procedure display_ownship                                         ***
--*************************************************************************
  procedure display_ownship( current: in track_tuple ) is

    time_string : string(1..8) := "00:00:00";
    --string correspondance of time

  begin --display_ownship
    if current.observation_time.hours > 9 then
      time_string(1) :=
                  character_of( current.observation_time.hours/10  );
      time_string(2) :=
              character_of( current.observation_time.hours mod 10  );
    else
      time_string(2) :=
                    character_of( current.observation_time.hours  );
    end if;

    if current.observation_time.minutes > 9 then
      time_string(4) :=
                character_of( current.observation_time.minutes/10  );
      time_string(5) :=
            character_of( current.observation_time.minutes mod 10  );
    else
      time_string(5) :=
                  character_of( current.observation_time.minutes  );
    end if;

    if current.observation_time.seconds > 9 then
      time_string(7) :=
                character_of( current.observation_time.seconds/10  );

      time_string(8) :=
            character_of( current.observation_time.seconds mod 10  );
    else
      time_string(8) :=
                  character_of( current.observation_time.seconds  );
    end if;

    Wpt_SetString(display_info.panel_id, "time_00", time_string);
```

```
    Wpt_SetReal(display_info.panel_id, "latitude_00",
      taefloat(float(integer(current.latitude*10.0))/10.0));

    Wpt_SetReal(display_info.panel_id, "longitude_00",
      taefloat(float(integer(current.longitude*10.0))/10.0));

    Wpt_SetReal(display_info.panel_id, "course_00",
      taefloat(float(integer(current.course*10.0))/10.0));

    Wpt_SetReal(display_info.panel_id, "velocity_00",
      taefloat(float(integer(current.velocity*10.0))/10.0));

  end display_ownship;


  --*********************************************************************
  --*** function satisfies_database_request                          ***
  --*********************************************************************
  function satisfies_database_request(
           track            : in track_tuple;
           td_track_request : in database_request ) return boolean is

  begin --satisfies_database_request
    if (td_track_request.track_class(track.track_class)) and
       (td_track_request.iff_class(track.iff_class)    ) and
       (td_track_request.the_range >= track.the_range  ) then
     return ( true );
   else
     return ( false );
   end if;
 end satisfies_database_request;


begin --display_tracks
  if track_display_panel_is_displayed then
    while current /= null loop
      --there is another track to be displayed
      --check if current is the ownship
      if current.id = 0 then
        display_ownship( current );
      elsif
```

```
 satisfies_database_request( current, td_track_request ) then
--find the current track's display row
if current.id > 9 then
  postfix(1) := character_of( current.id/10  );
  postfix(2) := character_of( (current.id mod 10)  );
else
  postfix(1) := '0';
  postfix(2) := character_of( current.id  );
end if;

--assign the postfixes of the panel id's and get the
--panel specifications
id_id(4..5)             := postfix;
observer_id(10..11)     := postfix;
time_id(6..7)           := postfix;
track_class_id(13..14) := postfix;
iff_class_id(11..12)    := postfix;
latitude_id(10..11)     := postfix;
longitude_id(11..12)    := postfix;
altitude_id(10..11)     := postfix;
course_id(8..9)         := postfix;
velocity_id(10..11)     := postfix;
range_id(7..8)          := postfix;

--print out the track information into the related row of
--the panel
Wpt_SetIntg(display_info.panel_id, id_id,taeint(current.id));
Wpt_SetString(
      display_info.panel_id, observer_id, current.observer );

if current.observation_time.hours > 9 then
  time_string(1) :=
            character_of(current.observation_time.hours/10 );
  time_string(2) :=
        character_of(current.observation_time.hours mod 10 );
else
  time_string(1) := '0';
  time_string(2) :=
              character_of(current.observation_time.hours );
end if;
```

```
if current.observation_time.minutes > 9 then
  time_string(4) :=
          character_of(current.observation_time.minutes/10 );
  time_string(5) :=
      character_of(current.observation_time.minutes mod 10 );
else
  time_string(4) := '0';
  time_string(5) :=
             character_of(current.observation_time.minutes );
end if;
 if current.observation_time.seconds > 9 then
  time_string(7) :=
          character_of(current.observation_time.seconds/10 );
  time_string(8) :=
      character_of(current.observation_time.seconds mod 10 );
else
  time_string(7) := '0';
  time_string(8) :=
             character_of(current.observation_time.seconds );
end if;

Wpt_SetString( display_info.panel_id, time_id, time_string );

if    current.track_class = AIR then
  Wpt_SetString( display_info.panel_id, track_class_id, "AA");
elsif current.track_class = SURFACE then
  Wpt_SetString( display_info.panel_id, track_class_id, "SU");
else--current.track_class = SUBSURFACE
  Wpt_SetString( display_info.panel_id, track_class_id, "SS");
end if;

if    current.iff_class = FRIENDLY then
  Wpt_SetString( display_info.panel_id, iff_class_id, "F");
elsif current.iff_class = HOSTILE then
  Wpt_SetString( display_info.panel_id, iff_class_id, "H");
elsif current.iff_class = NEUTRAL then
  Wpt_SetString( display_info.panel_id, iff_class_id, "N");
else--current.iff_class = UNKNOWN
  Wpt_SetString( display_info.panel_id, iff_class_id, "U");
end if;
```

```
        if current.id > 10 then
          --truncate the extra zeros because of the limitations
          --of the tae
          Wpt_SetReal(display_info.panel_id, latitude_id,
            taefloat(float(integer(current.latitude*10.0))/10.0));
          Wpt_SetReal(display_info.panel_id, longitude_id,
            taefloat(float(integer(current.longitude*10.0))/10.0));
          Wpt_SetReal(display_info.panel_id, altitude_id,
            taefloat(float(integer(current.altitude*10.0))/10.0));
          Wpt_SetReal(display_info.panel_id, course_id,
            taefloat(float(integer(current.course*10.0))/10.0));
          Wpt_SetReal(display_info.panel_id, velocity_id,
            taefloat(float(integer(current.velocity*10.0))/10.0));
          Wpt_SetReal(display_info.panel_id, range_id,
            taefloat(float(integer(current.the_range*100.0))/100.0));

        else --current.id <= 10
          --no truncation is required
          Wpt_SetReal(display_info.panel_id,latitude_id,
                      taefloat(current.latitude));
          Wpt_SetReal(display_info.panel_id,longitude_id,
                      taefloat(current.longitude));
          Wpt_SetReal(display_info.panel_id,altitude_id,
                      taefloat(current.altitude));
          Wpt_SetReal(display_info.panel_id,course_id,
                      taefloat(current.course));
          Wpt_SetReal(display_info.panel_id,velocity_id,
                      taefloat(current.velocity));
          Wpt_SetReal(display_info.panel_id,range_id,
                      taefloat(current.the_range));
        end if;
      end if;
      current := current.next;
    end loop;
  end if;--track_display_panel_is_displayed
end display_tracks;
```

```
--*************************************************************************
--*** procedure GET_USER_INPUTS                                         ***
--*************************************************************************
procedure get_user_inputs(
            tdd_archive_setup    : out archive_setup;
            tdd_filter           : out set_track_filter;
            td_track_request     : out database_request;
            tcd_status_query     : out boolean;
            tcd_network_setup    : out network_setup;
            tcd_emission_control : out emissions_control_command;
            editor_selected      : out boolean;
            initiate_trans       : out initiate_transmission_sequence;
            terminate_trans      : out boolean ) is

begin --get_inputs;
  tdd_archive_setup    := new_archive_setup;
  tdd_filter           := new_track_filter;
  td_track_request     := new_track_request;
  tcd_status_query     := new_status_query;
  tcd_emission_control := new_emission_control;
  tcd_network_setup    := new_network_setup;
  editor_selected      := new_editor_selected;
  initiate_trans       := new_initiate_trans;
  terminate_trans      := new_terminate_trans;
  new_editor_selected  := false;
end get_user_inputs;


--*************************************************************************
--*** procedure GET_DISPLAY_VALUES                                      ***
--*************************************************************************
procedure get_display_values(
            weapons_statrep : in  weapon_status_report;
            weapon          : in  weapons_type;
            display_item_id : out string;
            status          : out string ) is
begin --get_display_values

  if    weapon = CIWS then display_item_id := "ciws";
  elsif weapon = GUN  then display_item_id := "gun ";
  elsif weapon = TWS  then display_item_id := "tws ";
```

```
  elsif weapon = MK48 then display_item_id := "mk48";
  end if;

  if    weapons_statrep(weapon) = DAMAGED           then
    status := "DAMAGED           ";
  elsif weapons_statrep(weapon) = RELOADING         then
    status := "RELOADING         ";
  elsif weapons_statrep(weapon) = LAUNCHING         then
    status := "LAUNCHING         ";
  elsif weapons_statrep(weapon) = READY             then
    status := "READY             ";
  elsif weapons_statrep(weapon) = SERVICE_REQUIRED  then
    status := "SERVICE_REQUIRED  ";
  elsif weapons_statrep(weapon) = SLEWING           then
    status := "SLEWING           ";
  elsif weapons_statrep(weapon) = SECURED           then
    status := "SECURED           ";
  elsif weapons_statrep(weapon) = MAINTANENCE       then
    status := "MAINTANENCE       ";
  elsif weapons_statrep(weapon) = ENGAGING          then
    status := "ENGAGING          ";
  elsif weapons_statrep(weapon) = OUT_OF_AMMUNITION then
    status := "OUT_OF_AMMUNITION";
  end if;
end get_display_values;


--************************************************************************
--*** procedure STATUS_SCREEN                                          ***
--************************************************************************
procedure status_screen(
            weapons_statrep  : in out weapon_status_report;
            tcd_status_query : in     boolean ) is

  display_item_id : string(1..4);
  status          : string(1..17);
begin --status_screen

  Wpt_NewPanel ("", weapon_info.target, weapon_info.view,
     X_Windows.Null_Window, weapon_info, WPT_DEFAULT,
     weapon_info.panel_id);
```

```
  for weapon in WEAPONS_TYPE loop
    get_display_values(
                  weapons_statrep, weapon, display_item_id, status );
    Wpt_SetString( weapon_info.panel_id, display_item_id, status );
  end loop;

  new_status_query := false;
  weapon_status_panel_is_displayed := true;
end status_screen;


--************************************************************************
--*** procedure EMERGENCY_STATUS_SCREEN                                ***
--************************************************************************
procedure emergency_status_screen(
        weapons_emrep : in out weapon_status_report ) is

  display_item_id : string(1..4);
  status          : string(1..17);

begin --emergency_status_screen
  if not emergency_weapon_status_panel_is_displayed then
    Wpt_NewPanel ("", emerg_info.target, emerg_info.view,
       X_Windows.Null_Window, emerg_info, WPT_DEFAULT,
       emerg_info.panel_id);
    for weapon in WEAPONS_TYPE loop
      get_display_values(
        weapons_emrep, weapon, display_item_id, status );
      Wpt_SetString( emerg_info.panel_id, display_item_id, status );
    end loop;
    emergency_weapon_status_panel_is_displayed := true;
  end if;
end emergency_status_screen;


--************************************************************************
--*** procedure MESSCAME_DISPLAY                                       ***
--************************************************************************
procedure messcame_display ( file_name : in filename ) is

  incoming_file : text_io.file_type;
```

```
  display_file  : text_io.file_type;
  line          : string(1..80);
  length        : natural;

begin --messcame_display

  --open the incoming_file to be displayed
  open( incoming_file, name => file_name, mode => in_file );

  --create display_file, because tae uses a filename which is
  --entered in creation of tae_source_file process
  create( display_file, name => "DISPLAY_FILE");

  --copy content of the incoming_file to display_file
  while not end_of_file( incoming_file ) loop
    get_line( incoming_file, line, length );
    put_line( display_file, line(1..length) );
  end loop;

  close( display_file );

  Wpt_NewPanel ("", messdisp_info.target, messdisp_info.view,
     X_Windows.Null_Window, messdisp_info, WPT_DEFAULT,
     messdisp_info.panel_id);

  loop

    Wpt_NextEvent (wptEvent, etype);

    case etype is

      when Wpt_PARM_EVENT =>

        Wpt_Extract_Context (wptEvent, user_ptr);
        Wpt_Extract_Parm (wptEvent,user_ptr.parm_name);
        Wpt_Extract_Data (wptEvent,user_ptr.datavm_ptr);
        Vm_Find (
         user_ptr.datavm_ptr,user_ptr.parm_name,user_ptr.parm_ptr);

        if s_equal ("save", user_ptr.parm_name) then
          close( incoming_file );
```

```
            Wpt_PanelErase(messdisp_info.panel_id);
            exit;
          elsif s_equal ("delete", user_ptr.parm_name) then
            delete( incoming_file );
            Wpt_PanelErase(messdisp_info.panel_id);
            exit;
          else
            put_line("Enter data to MESSAGE  PANEL Only");
          end if;

        when Wpt_WINDOW_EVENT =>
          null;

        when OTHERS =>
          put("ERROR: Unexpected Wpt_NextEvent. ");

      end case;

    end loop;

    exception
      when NAME_ERROR =>
        begin
          put(file_name); put_line(" DOES NOT EXIST !!!");
        end;

end messcame_display;


--*************************************************************************
--*** procedure NEW_NAME                                                ***
--*************************************************************************
function new_name( file_name : in filename ) return filename is

  new_file_name : filename;

begin --new_name

  for I in 1..10 loop
    if file_name( I ) /= ' ' then
      new_file_name( I ) := file_name( I );
```

```
    else
      exit;
    end if;
  end loop;

  return( new_file_name );

end new_name;


--*************************************************************************
--*** procedure MESSAGE_ARRIVAL_PANEL                                   ***
--*************************************************************************
procedure message_arrival_panel( comms_email : in filename ) is
begin --message_arrival_panel

  Wpt_NewPanel ("", messcame_info.target, messcame_info.view,
     X_Windows.Null_Window, messcame_info, Wpt_DEFAULT,
     messcame_info.panel_id);

  Wpt_SetString( ..s.came_info.panel_id, "filename", comms_email );

  loop

    Wpt_NextEvent (wptEvent, etype);

    case etype is

      when Wpt_PARM_EVENT =>
        Wpt_Extract_Context (wptEvent, user_ptr);
        Wpt_Extract_Parm (wptEvent,user_ptr.parm_name);
        Wpt_Extract_Data (wptEvent,user_ptr.datavm_ptr);
        Vm_Find (
         user_ptr.datavm_ptr,user_ptr.parm_name,user_ptr.parm_ptr);
        if s_equal ("display", user_ptr.parm_name) then
          Wpt_PanelErase(messcame_info.panel_id);
          messcame_display (comms_email);  --display the message
          exit;
        elsif s_equal ("cancel", user_ptr.parm_name) then
          Wpt_PanelErase(messcame_info.panel_id);
          exit;
```

```
        else
          put_line("Enter data to MESSAGE ARRIVAL PANEL Only");
        end if;

      when Wpt_WINDOW_EVENT =>
        null;

      when OTHERS =>
        put("ERROR: Unexpected Wpt_NextEvent. ");

    end case;

  end loop;

end message_arrival_panel;


--************************************************************************
--*** procedure MESSAGE_RETRIEVAL_PANEL                                ***
--************************************************************************
procedure message_retrieval_panel is

  file_name : filename;
  count     : taeint;
  value     : array (1..1) of string (1..tae_taeconf.STRINGSIZE);
  filename_is_entered : boolean := false;

begin --message_retrieval_panel

  Wpt_NewPanel ("", messret_info.target, messret_info.view,
     X_Windows.Null_Window, messret_info, Wpt_DEFAULT,
     messret_info.panel_id);
  loop
    Wpt_NextEvent (wptEvent, etype);
    case etype is

      when Wpt_PARM_EVENT =>
        Wpt_Extract_Context (wptEvent, user_ptr);
        Wpt_Extract_Parm (wptEvent,user_ptr.parm_name);
        Wpt_Extract_Data (wptEvent,user_ptr.datavm_ptr);
        Vm_Find (
```

```
        user_ptr.datavm_ptr,user_ptr.parm_name,user_ptr.parm_ptr);

        if s_equal ("filename", user_ptr.parm_name) then
          --Vm_Extract_Count (info.parm_ptr, count);
          --if count <= 0 then
            Vm_Extract_SVAL (user_ptr.parm_ptr, 1, value(1));
            file_name := new_name(value(1)(1..10));
            filename_is_entered := true;
        elsif s_equal ("display", user_ptr.parm_name) then
          if filename_is_entered = true then
            Wpt_PanelReset(messret_Info.Panel_id);
            Wpt_PanelErase(messret_info.panel_id);
            messcame_display (file_name);
            exit;
          else
            Wpt_ParmReject( messret_info.panel_id, user_ptr.parm_name,
            "Filename should be Entered");
          end if;
        elsif s_equal ("cancel", user_ptr.parm_name) then
          Wpt_PanelReset(messret_Info.Panel_id);
          Wpt_PanelErase(messret_info.panel_id);
          exit;
        else
          put_line("Enter data to MESSAGE RETRIEVAL PANEL Only");
        end if;
      when Wpt_WINDOW_EVENT =>
        null;
      when OTHERS =>
        put("ERROR: Unexpected Wpt_NextEvent. ");
    end case;

  end loop;

end message_retrieval_panel;


--************************************************************************
--*** procedure INITIALIZEPANELS                                       ***
--************************************************************************
procedure initializepanels( file : in string ) is
```

```
  tmp_info : event_context_ptr;

begin --initializePanels

  f_force_lower (false);     -- permit upper/lowercase file names
  Wpt_Init ("",theDisplay);
  Wpt_NewEvent (wptEvent);

  tmp_info := new event_context;
  Co_New (0, tmp_info.collection);
  Co_ReadFile (tmp_info.collection, file, P_CONT);

  -- pair of Co_Finds for each panel in this resource file

  main_info := new event_context;
  main_info.collection := tmp_info.collection;
  Co_Find (main_info.collection, "main_v", main_info.view);
  Co_Find (main_info.collection, "main_t", main_info.target);

  filter_info := new event_context;
  filter_info.collection := tmp_info.collection;
  Co_Find (filter_info.collection, "filter_v", filter_info.view);
  Co_Find (filter_info.collection, "filter_t", filter_info.target);

  archive_info := new event_context;
  archive_info.collection := tmp_info.collection;
  Co_Find (archive_info.collection, "archive_v", archive_info.view);
  Co_Find (archive_info.collection, "archive_t", archive_info.target);

  contrnts_info := new event_context;
  contrnts_info.collection := tmp_info.collection;
  Co_Find (contrnts_info.collection, "contrnts_v",
           contrnts_info.view);
  Co_Find (contrnts_info.collection, "contrnts_t",
           contrnts_info.target);

  display_info := new event_context;
  display_info.collection := tmp_info.collection;
  Co_Find (display_info.collection, "display_v", display_info.view);
  Co_Find (display_info.collection, "display_t", display_info.target);
```

```
weapon_info := new event_context;
weapon_info.collection := tmp_info.collection;
Co_Find (weapon_info.collection, "weapon_v", weapon_info.view);
Co_Find (weapon_info.collection, "weapon_t", weapon_info.target);

emerg_info := new event_context;
emerg_info.collection := tmp_info.collection;
Co_Find (emerg_info.collection, "emerg_v", emerg_info.view);
Co_Find (emerg_info.collection, "emerg_t", emerg_info.target);

network_info := new event_context;
network_info.collection := tmp_info.collection;
Co_Find (network_info.collection, "network_v", network_info.view);
Co_Find (network_info.collection, "network_t", network_info.target);

emcon_info := new event_context;
emcon_info.collection := tmp_info.collection;
Co_Find (emcon_info.collection, "emcon_v", emcon_info.view);
Co_Find (emcon_info.collection, "emcon_t", emcon_info.target);

editor_info := new event_context;
editor_info.collection := tmp_info.collection;
Co_Find (editor_info.collection, "editor_v", editor_info.view);
Co_Find (editor_info.collection, "editor_t", editor_info.target);

messcame_info := new event_context;
messcame_info.collection := tmp_info.collection;
Co_Find (messcame_info.collection, "messcame_v",
         messcame_info.view);
Co_Find (messcame_info.collection, "messcame_t",
         messcame_info.target);

messdisp_info := new event_context;
messdisp_info.collection := tmp_info.collection;
Co_Find (messdisp_info.collection, "messdisp_v",
         messdisp_info.view);
Co_Find (messdisp_info.collection, "messdisp_t",
         messdisp_info.target);

messret_info := new tae_wpt.event_context;
messret_info.collection := tmp_info.collection;
```

```
  Co_Find (messret_info.collection, "messret_v", messret_info.view);
  Co_Find (messret_info.collection, "messret_t", messret_info.target);

  trackrep_info := new tae_wpt.event_context;
  trackrep_info.collection := tmp_info.collection;
  Co_Find (trackrep_info.collection, "trackrep_v",
           trackrep_info.view);
  Co_Find (trackrep_info.collection, "trackrep_t",
           trackrep_info.target);

  tertrrep_info := new tae_wpt.event_context;
  tertrrep_info.collection := tmp_info.collection;
  Co_Find (tertrrep_info.collection, "tertrrep_v",
           tertrrep_info.view);
  Co_Find (tertrrep_info.collection, "tertrrep_t",
           tertrrep_info.target);

  Wpt_NewPanel ("", main_info.target, main_info.view,
     X_Windows.Null_Window, main_info, Wpt_DEFAULT,
     main_info.panel_id);

end initializepanels;


--************************************************************************
--*** procedure MANAGE_USER_INTERFACE                                  ***
--************************************************************************
procedure manage_user_interface is


  --************************************************************************
  --*** procedure GET_TRACK_FILTER_VALUES                                ***
  --************************************************************************
  procedure get_track_filter_values is

    value         : array (1..1) of string (1..tae_taeconf.stringsize);
    float_value   : array (1..1) of taefloat;
    integer_value : array (1..1) of taeint;

  begin --get_track_filter_values
```

```
Wpt_NewPanel ("", filter_info.target, filter_info.view,
   X_Windows.Null_Window,filter_info,
   Wpt_DEFAULT,filter_info.panel_id);

loop

  Wpt_NextEvent (wptEvent, etype);

  case etype is

    when Wpt_PARM_EVENT =>

      Wpt_Extract_Context (wptEvent, user_ptr);
      Wpt_Extract_Parm (wptEvent,user_ptr.parm_name);
      Wpt_Extract_Data (wptEvent,user_ptr.datavm_ptr);
      Vm_Find (
     user_ptr.datavm_ptr,user_ptr.parm_name,user_ptr.parm_ptr);

      if s_equal ("max_num_track", user_ptr.parm_name) then
        Vm_Extract_IVAL (user_ptr.parm_ptr, 1, integer_value(1));
        if integer_value(1) >= 0 then
          new_track_filter.max_number :=
                                        integer(integer_value(1));
        else
          Wpt_ParmReject( filter_info.panel_id,
             user_ptr.parm_name, "Number of Track >= 0 required");
        end if;
      elsif s_equal ("class_air", user_ptr.parm_name) then
        Vm_Extract_SVAL (user_ptr.parm_ptr, 1, value(1));
        if value(1)(1..3) = "yes" then
          new_track_filter.desired_class(air) := true;
        else
          new_track_filter.desired_class(air) := false;
        end if;
      elsif s_equal ("class_surface", user_ptr.parm_name) then
        Vm_Extract_SVAL (user_ptr.parm_ptr, 1, value(1));
        if value(1)(1..3) = "yes" then
          new_track_filter.desired_class(surface) := true;
        else
          new_track_filter.desired_class(surface) := false;
        end if;
```

```
elsif s_equal ("class_subsurf", user_ptr.parm_name) then
  Vm_Extract_SVAL (user_ptr.parm_ptr, 1, value(1));
  if value(1)(1..3) = "yes" then
    new_track_filter.desired_class(subsurface) := true;
  else
    new_track_filter.desired_class(subsurface) := false;
  end if;
elsif s_equal ("air_range", user_ptr.parm_name) then
  Vm_Extract_RVAL (user_ptr.parm_ptr, 1, float_value(1));
  if float_value(1) >= 0.0 then
    new_track_filter.desired_range(air) :=
                                        float(float_value(1));
  else
    Wpt_ParmReject( filter_info.panel_id,
              user_ptr.parm_name, "Range >= 0.0 required");
  end if;
elsif s_equal ("surface_range", user_ptr.parm_name) then
  Vm_Extract_RVAL (user_ptr.parm_ptr, 1, float_value(1));
  if float_value(1) >= 0.0 then
    new_track_filter.desired_range(surface) :=
                                        float(float_value(1));
  else
    Wpt_ParmReject( filter_info.panel_id,
              user_ptr.parm_name, "Range >= 0.0 required");
  end if;
elsif s_equal ("subsurf_range", user_ptr.parm_name) then
  Vm_Extract_RVAL (user_ptr.parm_ptr, 1, float_value(1));
  if float_value(1) >= 0.0 then
    new_track_filter.desired_range(subsurface) :=
                                        float(float_value(1));
  else
    Wpt_ParmReject( filter_info.panel_id,
              user_ptr.parm_name, "Range >= 0.0 required");
  end if;
elsif s_equal ("cancel", user_ptr.parm_name) then
  --reassign the new values to old values,
  --changes are not valid
  new_track_filter := old_track_filter;
  Wpt_PanelErase(filter_info.panel_id);
  exit;
elsif s_equal ("ok", user_ptr.parm_name) then
```

```
          Wpt_PanelErase(filter_info.panel_id);
          exit;
        else
          put_line("Enter data to TRACK FILTER PANEL Only");
        end if;

      when Wpt_WINDOW_EVENT =>
        null;

      when OTHERS =>
        put("ERROR: Unexpected Wpt_NextEvent. ");

    end case;

  end loop;

end get_track_filter_values;


--*********************************************************************
--*** procedure GET_ARCHIVE_SETUP_VALUES                          ***
--*********************************************************************
procedure get_archive_setup_values is

  value : array (1..1) of string (1..tae_taeconf.stringsize);

begin

  Wpt_NewPanel ("", archive_info.target, archive_info.view,
     X_Windows.Null_Window,archive_info,
     Wpt_DEFAULT,archive_info.panel_id);

  loop

    Wpt_NextEvent (wptEvent, etype);

    case etype is

      when Wpt_PARM_EVENT =>

        Wpt_Extract_Context (wptEvent, user_ptr);
```

```
Wpt_Extract_Parm (wptEvent,user_ptr.parm_name);
Wpt_Extract_Data (wptEvent,user_ptr.datavm_ptr);
Vm_Find (
  user_ptr.datavm_ptr,user_ptr.parm_name,user_ptr.parm_ptr);

if s_equal ("allships", user_ptr.parm_name) then
   Vm_Extract_SVAL (user_ptr.parm_ptr, 1, value(1));
   if value(1)(1..3) = "yes" then
     new_archive_setup.all_ships := true;
   else
     new_archive_setup.all_ships := false;
   end if;
elsif s_equal ("ownship", user_ptr.parm_name) then
   Vm_Extract_SVAL (user_ptr.parm_ptr, 1, value(1));
   if value(1)(1..3) = "yes" then
     new_archive_setup.ownship := true;
   else
     new_archive_setup.ownship := false;
   end if;
elsif s_equal ("jtids", user_ptr.parm_name) then
   Vm_Extract_SVAL (user_ptr.parm_ptr, 1, value(1));
   if value(1)(1..3) = "yes" then
     new_archive_setup.jtids := true;
   else
     new_archive_setup.jtids := false;
   end if;
elsif s_equal ("link11", user_ptr.parm_name) then
   Vm_Extract_SVAL (user_ptr.parm_ptr, 1, value(1));
   if value(1)(1..3) = "yes" then
     new_archive_setup.link11 := true;
   else
     new_archive_setup.link11 := false;
   end if;
elsif s_equal ("link16", user_ptr.parm_name) then
   Vm_Extract_SVAL (user_ptr.parm_ptr, 1, value(1));
   if value(1)(1..3) = "yes" then
     new_archive_setup.link16 := true;
   else
     new_archive_setup.link16 := false;
   end if;
elsif s_equal ("otcixs", user_ptr.parm_name) then
```

```
            Vm_Extract_SVAL (user_ptr.parm_ptr, 1, value(1));
            if value(1)(1..3) = "yes" then
              new_archive_setup.otcixs := true;
            else
              new_archive_setup.otcixs := false;
            end if;
          elsif s_equal ("cancel", user_ptr.parm_name) then
            --reassign the new values to old values,
            --changes are not valid
            new_archive_setup := old_archive_setup;
            Wpt_PanelErase(archive_info.panel_id);
            exit;
          elsif s_equal ("ok", user_ptr.parm_name) then
            Wpt_PanelErase(archive_info.panel_id);
            exit;
          else
            put_line("Enter data to ARCHIVE SETUP PANEL Only");
          end if;

        when Wpt_WINDOW_EVENT =>
          null;

        when OTHERS =>
          put("ERROR: Unexpected Wpt_NextEvent. ");

      end case;

    end loop;

end get_archive_setup_values;


--*********************************************************************
--*** procedure GET_TRACK_DISPLAY_CONSTRAINTS                      ***
--*********************************************************************
procedure get_track_display_constraints is

  value         : array (1..1) of string (1..tae_taeconf.stringsize);
  float_value   : array (1..1) of taefloat;
  integer_value : array (1..1) of taeint;
```

```
begin

  Wpt_NewPanel ("", contrnts_info.target, contrnts_info.view,
     X_Windows.Null_Window, contrnts_info,
     Wpt_DEFAULT,contrnts_info.panel_id);

  loop

    Wpt_NextEvent (wptEvent, etype);

    case etype is

      when Wpt_PARM_EVENT =>

        Wpt_Extract_Context (wptEvent, user_ptr);
        Wpt_Extract_Parm (wptEvent,user_ptr.parm_name);
        Wpt_Extract_Data (wptEvent,user_ptr.datavm_ptr);
        Vm_Find (
          user_ptr.datavm_ptr,user_ptr.parm_name,user_ptr.parm_ptr);

        if s_equal ("range", user_ptr.parm_name) then
          Vm_Extract_RVAL (user_ptr.parm_ptr, 1, float_value(1));
          if float_value(1) >= 0.0 then
            new_track_request.the_range := float(float_value(1));
          else
            Wpt_ParmReject( contrnts_info.panel_id,
                        user_ptr.parm_name, "Range >= 0 required");
          end if;
        elsif s_equal ("class_air", user_ptr.parm_name) then
          Vm_Extract_SVAL (user_ptr.parm_ptr, 1, value(1));
          if value(1)(1..3) = "yes" then
            new_track_request.track_class(air) := true;
          else
            new_track_request.track_class(air) := false;
          end if;
        elsif s_equal ("class_surface", user_ptr.parm_name) then
          Vm_Extract_SVAL (user_ptr.parm_ptr, 1, value(1));
          if value(1)(1..3) = "yes" then
            new_track_request.track_class(surface) := true;
          else
            new_track_request.track_class(surface) := false;
```

```
  end if;
elsif s_equal ("class_subsurf", user_ptr.parm_name) then
  Vm_Extract_SVAL (user_ptr.parm_ptr, 1, value(1));
  if value(1)(1..3) = "yes" then
    new_track_request.track_class(subsurface) := true;
  else
    new_track_request.track_class(subsurface) := false;
  end if;
elsif s_equal ("friendly", user_ptr.parm_name) then
  Vm_Extract_SVAL (user_ptr.parm_ptr, 1, value(1));
  if value(1)(1..3) = "yes" then
    new_track_request.iff_class(friendly) := true;
  else
    new_track_request.iff_class(friendly) := false;
  end if;
elsif s_equal ("hostile", user_ptr.parm_name) then
  Vm_Extract_SVAL (user_ptr.parm_ptr, 1, value(1));
  if value(1)(1..3) = "yes" then
    new_track_request.iff_class(hostile) := true;
  else
    new_track_request.iff_class(hostile) := false;
  end if;
elsif s_equal ("neutral", user_ptr.parm_name) then
  Vm_Extract_SVAL (user_ptr.parm_ptr, 1, value(1));
  if value(1)(1..3) = "yes" then
    new_track_request.iff_class(neutral) := true;
  else
    new_track_request.iff_class(neutral) := false;
  end if;
elsif s_equal ("unknown", user_ptr.parm_name) then
  Vm_Extract_SVAL (user_ptr.parm_ptr, 1, value(1));
  if value(1)(1..3) = "yes" then
    new_track_request.iff_class(unknown) := true;
  else
    new_track_request.iff_class(unknown) := false;
  end if;
elsif s_equal ("cancel", user_ptr.parm_name) then
  --reassign the new values to old values,
  --changes are not valid
  new_track_request := old_track_request;
  Wpt_PanelErase(contrnts_info.panel_id);
```

```
            exit;
          elsif s_equal ("ok", user_ptr.parm_name) then
            old_track_request := new_track_request;
            Wpt_PanelErase(contrnts_info.panel_id);
            if track_display_panel_is_displayed then
              Wpt_PanelReset(display_Info.Panel_id);
            else
              Wpt_NewPanel ("",display_info.target,display_info.view,
                      X_Windows.Null_Window,display_info,Wpt_DEFAULT,
                      display_info.panel_id);
              track_display_panel_is_displayed := true;
            end if;
            exit;
          else
            put_line(
                "Enter data to TRACK DISPLAY CONSTRAINTS PANEL Only");
          end if;

        when Wpt_WINDOW_EVENT =>
          null;

        when OTHERS =>
          put("ERROR: Unexpected Wpt_NextEvent. ");

      end case;

    end loop;

  end get_track_display_constraints;


  --*********************************************************************
  --*** procedure GET_EMCON_STATUS                                    ***
  --*********************************************************************
  procedure get_emcon_status is

    value : array (1..1) of string (1..tae_taeconf.STRINGSIZE);

   begin --get_emcon_status

    Wpt_NewPanel ("", emcon_info.target, emcon_info.view,
```

```
    X_Windows.Null_Window, emcon_info, WPT_DEFAULT,
    emcon_info.panel_id);

  loop

    Wpt_NextEvent (wptEvent, etype);

    case etype is

      when Wpt_PARM_EVENT =>

        Wpt_Extract_Context (wptEvent, user_ptr);
        Wpt_Extract_Parm (wptEvent,user_ptr.parm_name);
        Wpt_Extract_Data (wptEvent,user_ptr.datavm_ptr);
        Vm_Find (
          user_ptr.datavm_ptr,user_ptr.parm_name,user_ptr.parm_ptr);

        if s_equal ("emcon_status", user_ptr.parm_name) then
          Vm_Extract_SVAL (user_ptr.parm_ptr, 1, value(1));
          if s_equal ("SILENCE", user_ptr.parm_name) then
            new_emission_control := EMCON;
          else
            new_emission_control := UNRESTRICTED;
          end if;
        elsif s_equal ("ok", user_ptr.parm_name) then
          old_emission_control := new_emission_control;
          Wpt_PanelErase(emcon_info.panel_id);
          exit;
        elsif s_equal ("cancel", user_ptr.parm_name) then
          --reassign the new values to old values,
          --changes are not valid
          new_emission_control := old_emission_control;
          Wpt_PanelErase(emcon_info.panel_id);
          exit;
        else
          put_line("Enter data to EMCON STATUS PANEL Only");
        end if;

      when Wpt_WINDOW_EVENT =>
        null;
```

```
      when OTHERS =>
        put("ERROR: Unexpected Wpt_NextEvent. ");

    end case;

  end loop;

end get_emcon_status;


--*********************************************************************
--*** procedure GET_NETWORK_SETUP                                   ***
--*********************************************************************
procedure get_network_setup is

  value : array (1..1) of string (1..tae_taeconf.STRINGSIZE);


  --*********************************************************************
  --*** procedure ATTACH_TO_LINK                                      ***
  --*********************************************************************
  procedure attach_to_link(
           new_network_setup : in out network_setup;
           link_name         : in     links_type;
           address_name      : in     string ) is

    previous    : address_link := null;
    current     : address_link := new_network_setup(link_name);
    new_address : address_link := new address_type;

  begin --attach_to_link
    new_address.name := address_name;
    if current = null then
      new_address.next := new_network_setup(link_name);
      new_network_setup(link_name) := new_address;
    else
      while current /= null loop
        previous := current;
        current := current.next;
      end loop;
      previous.next := new_address;
```

```
    end if;
  end attach_to_link;

begin --get_network_setup

  for I in links_type loop
    new_network_setup(I) := null;
  end loop;

  Wpt_NewPanel ("", network_info.target, network_info.view,
     X_Windows.Null_Window, network_info, WPT_DEFAULT,
     network_info.panel_id);

  loop

    Wpt_NextEvent (wptEvent, etype);

    case etype is

      when Wpt_PARM_EVENT =>

        Wpt_Extract_Context (wptEvent, user_ptr);
        Wpt_Extract_Parm (wptEvent,user_ptr.parm_name);
        Wpt_Extract_Data (wptEvent,user_ptr.datavm_ptr);
        Vm_Find (
          user_ptr.datavm_ptr,user_ptr.parm_name,user_ptr.parm_ptr);

        if s_equal ("jtids_1", user_ptr.parm_name) then
          Vm_Extract_SVAL (user_ptr.parm_ptr, 1, value(1));
          attach_to_link(new_network_setup, jtids, value(1)(1..6));
        elsif s_equal ("jtids_2", user_ptr.parm_name) then
          Vm_Extract_SVAL (user_ptr.parm_ptr, 1, value(1));
          attach_to_link(new_network_setup, jtids, value(1)(1..6));
        elsif s_equal ("jtids_3", user_ptr.parm_name) then
          Vm_Extract_SVAL (user_ptr.parm_ptr, 1, value(1));
          attach_to_link(new_network_setup, jtids, value(1)(1..6));
        elsif s_equal ("jtids_5", user_ptr.parm_name) then
          Vm_Extract_SVAL (user_ptr.parm_ptr, 1, value(1));
          attach_to_link(new_network_setup, jtids, value(1)(1..6));
        elsif s_equal ("jtids_4", user_ptr.parm_name) then
          Vm_Extract_SVAL (user_ptr.parm_ptr, 1, value(1));
```

```
  attach_to_link(new_network_setup, jtids, value(1)(1..6));
elsif s_equal ("link11_1", user_ptr.parm_name) then
  Vm_Extract_SVAL (user_ptr.parm_ptr, 1, value(1));
  attach_to_link(new_network_setup, link11, value(1)(1..6));
elsif s_equal ("link11_2", user_ptr.parm_name) then
  Vm_Extract_SVAL (user_ptr.parm_ptr, 1, value(1));
  attach_to_link(new_network_setup, link11, value(1)(1..6));
elsif s_equal ("link11_3", user_ptr.parm_name) then
  Vm_Extract_SVAL (user_ptr.parm_ptr, 1, value(1));
  attach_to_link(new_network_setup, link11, value(1)(1..6));
elsif s_equal ("link11_4", user_ptr.parm_name) then
  Vm_Extract_SVAL (user_ptr.parm_ptr, 1, value(1));
  attach_to_link(new_network_setup, link11, value(1)(1..6));
elsif s_equal ("link11_5", user_ptr.parm_name) then
  Vm_Extract_SVAL (user_ptr.parm_ptr, 1, value(1));
  attach_to_link(new_network_setup, link11, value(1)(1..6));
elsif s_equal ("link16_1", user_ptr.parm_name) then
  Vm_Extract_SVAL (user_ptr.parm_ptr, 1, value(1));
  attach_to_link(new_network_setup, link16, value(1)(1..6));
elsif s_equal ("link16_2", user_ptr.parm_name) then
  Vm_Extract_SVAL (user_ptr.parm_ptr, 1, value(1));
  attach_to_link(new_network_setup, link16, value(1)(1..6));
elsif s_equal ("link16_3", user_ptr.parm_name) then
  Vm_Extract_SVAL (user_ptr.parm_ptr, 1, value(1));
  attach_to_link(new_network_setup, link16, value(1)(1..6));
elsif s_equal ("link16_4", user_ptr.parm_name) then
  Vm_Extract_SVAL (user_ptr.parm_ptr, 1, value(1));
  attach_to_link(new_network_setup, link16, value(1)(1..6));
elsif s_equal ("link16_5", user_ptr.parm_name) then
  Vm_Extract_SVAL (user_ptr.parm_ptr, 1, value(1));
  attach_to_link(new_network_setup, link16, value(1)(1..6));
elsif s_equal ("otcixs_1", user_ptr.parm_name) then
  Vm_Extract_SVAL (user_ptr.parm_ptr, 1, value(1));
  attach_to_link(new_network_setup, otcixs, value(1)(1..6));
elsif s_equal ("otcixs_2", user_ptr.parm_name) then
  Vm_Extract_SVAL (user_ptr.parm_ptr, 1, value(1));
  attach_to_link(new_network_setup, otcixs, value(1)(1..6));
elsif s_equal ("otcixs_3", user_ptr.parm_name) then
  Vm_Extract_SVAL (user_ptr.parm_ptr, 1, value(1));
  attach_to_link(new_network_setup, otcixs, value(1)(1..6));
elsif s_equal ("otcixs_4", user_ptr.parm_name) then
```

```
          Vm_Extract_SVAL (user_ptr.parm_ptr, 1, value(1));
          attach_to_link(new_network_setup, otcixs, value(1)(1..6));
        elsif s_equal ("otcixs_5", user_ptr.parm_name) then
          Vm_Extract_SVAL (user_ptr.parm_ptr, 1, value(1));
          attach_to_link(new_network_setup, otcixs, value(1)(1..6));
        elsif s_equal ("cancel", user_ptr.parm_name) then
          --reassign the new values to old values,
          --changes are not valid
          new_network_setup := old_network_setup;
          Wpt_PanelReset(network_info.panel_id);
          Wpt_PanelErase(network_info.panel_id);
          exit;
        elsif s_equal ("ok", user_ptr.parm_name) then
          old_network_setup := new_network_setup;
          Wpt_PanelReset(network_info.panel_id);
          Wpt_PanelErase(network_info.panel_id);
          exit;
        else
          put_line("Enter data to NETWORK SETUP PANEL Only");
        end if;

      when Wpt_WINDOW_EVENT =>
        null;

      when OTHERS =>
        put("ERROR: Unexpected Wpt_NextEvent. ");

    end case;

  end loop;
end get_network_setup;


--************************************************************************
--*** procedure GET_USER_MESSAGE                                       ***
--************************************************************************
procedure get_user_message is

  value        : array (1..1) of string (1..tae_taeconf.STRINGSIZE);
  current_relay_to   : integer := 0;
```

```
--following flags are used to determine if at least minimum
--required information is entered into the editor
name_is_entered    : boolean := false;
origin_is_entered  : boolean := false;
address_is_entered : boolean := false;


--*********************************************************************
--*** procedure ATTACH_TO_LIST                                      ***
--*********************************************************************
procedure attach_to_list(
         address_list : in out address_link;
            in_name      : in     string ) is

  previous    : address_link;
  current     : address_link := address_list;
  new_address : address_link := new address_type;

begin --attach_to_list

  new_address.name := in_name;

  if current = null then
    new_address.next := address_list;
    address_list    := new_address;
  else
    while current /= null loop
      previous := current;
      current := current.next;
    end loop;
    previous.next := new_address;
  end if;

end attach_to_list;


--*********************************************************************
--*** procedure GET_BODY_CONTENT                                    ***
--*********************************************************************
procedure get_body_content( text : in out string ) is
```

```
  string_count    : taeint;
  string_vector   : tae.tae_vm.s_vector(1..30);
  entered_string  : tae.variable_ptr;
  temp_string     : string(1..tae_taeconf.STRINGSIZE);
  count           : natural := 1;

begin --get_body_content

  Vm_Find (editor_info.target, "body", entered_string );
  Vm_Extract_COUNT( entered_string, string_count );

  for I in 1 .. TEXT_LENGTH loop
    text(I) := ' ';
  end loop;

  for line in 1..string_count loop
    Vm_Extract_SVAL( entered_string, line, temp_string );
    string_vector(integer(line)) :=
                                 new string(1..temp_string'length);
    string_vector(integer(line)).all := temp_string;
    text(count..count+temp_string'length-1) :=
                                   string_vector(integer(line)).all;
    count := count + 80;
  end loop;

end get_body_content;


--*********************************************************************
--*** procedure CHECK_AND_ATTACH_TO_VIA_LIST                        ***
--*********************************************************************
procedure check_and_attach_to_via_list(
         relay_by : in string;
         to       : in string;
         I        : in integer ) is


  --*******************************************************************
  --*** procedure ATTACH_TO_VIA_LIST                                 ***
  --*******************************************************************
  procedure attach_to_via_list(
```

```
      relay_by : in string;
      to       : in string ) is

  new_address  : address_link;
  new_via_line : via_record_link;
  current      : address_link;
  previous     : address_link := null;

begin --attach_to_via_list
  if current_relay_to = 1 then
    new_via_line := new via_record;
    new_transmit_command(jtids).text.header.via_line :=
                                                  new_via_line;
  elsif current_relay_to = 5 then
    new_via_line := new via_record;
    new_transmit_command(jtids).text.header.via_line.next :=
                                                   new_via_line;
  end if;
  if current_relay_to < 5 then
    new_via_line :=
              new_transmit_command(jtids).text.header.via_line;
  else
    new_via_line :=                                 ;
          new_transmit_command(jtids).text.header.via_line.next;
  end if;
  if relay_by = "      " then   -- relay_to field came
    new_address   := new address_type;
    new_address.name := to;
    current := new_via_line.relay_to;
    if current = null then
      new_address.next := new_via_line.relay_to;
      new_via_line.relay_to := new_address;
    else
      while current /= null loop
        previous := current;
        current := current.next;
      end loop;
      previous.next :=  new_address;
    end if;
  else
    new_via_line.relay_by := relay_by;
```

```
      end if;
    end attach_to_via_list;

  begin --check_and_attach_to_via_list
    if (I >= 1 and I <= 3 and I - current_relay_to = 1) or
       (I >= 5 and I <= 7 and I - current_relay_to = 1) or
       (I=4 and current_relay_to >= 1 and current_relay_to <=3) or
       (I=8 and current_relay_to >= 5 and current_relay_to <=7) then
      current_relay_to := I;
      attach_to_via_list(relay_by,to);
    else
      put_line("Enter data to APPROPRIATE ADDRESS FIELD");
    end if;
  end check_and_attach_to_via_list;

begin --get_user_message

  Wpt_NewPanel ("", editor_info.target, editor_info.view,
     X_Windows.Null_Window, editor_info, WPT_DEFAULT,
     editor_info.panel_id);

  --initialize new_transmit_command
  for I in links_type loop
    new_transmit_command(I).route_addr := null;
    new_transmit_command(I).routed     := false;
    new_transmit_command(I).full       := false;
  end loop;

  --initialize new_transmit_command(jtids)
  new_transmit_command(jtids).full                          := true;
    new_transmit_command(jtids).text.name                           :=
"          ";
  new_transmit_command(jtids).text.header.classification := U;
  new_transmit_command(jtids).text.header.precedence     := R;
  new_transmit_command(jtids).text.header.origin        := "NAME_0";
  new_transmit_command(jtids).text.header.address        := null;
  new_transmit_command(jtids).text.header.info           := null;
  new_transmit_command(jtids).text.header.via_line       := null;
  for I in 1..60 loop
    new_transmit_command(jtids).text.header.subject(I)   := ' ';
  end loop;
```

```
new_transmit_command(jtids).text.link_id                    := jtids;
new_transmit_command(jtids).text.relayed                    := false;
new_transmit_command(jtids).text.archive                    := true;
new_transmit_command(jtids).text.is_track                   := false;
for I in 1..TEXT_LENGTH loop
  new_transmit_command(jtids).text.text(I) := ' ';
end loop;

loop

Wpt_NextEvent (wptEvent, etype);

case etype is


when Wpt_PARM_EVENT =>

  Wpt_Extract_Context (wptEvent, user_ptr);
  Wpt_Extract_Parm (wptEvent,user_ptr.parm_name);
  Wpt_Extract_Data (wptEvent,user_ptr.datavm_ptr);
  Vm_Find (
        user_ptr.datavm_ptr,user_ptr.parm_name,user_ptr.parm_ptr);

  if s_equal ("name", user_ptr.parm_name) then
    Vm_Extract_SVAL (user_ptr.parm_ptr, 1, value(1));
    new_transmit_command(jtids).text.name := value(1)(1..10);
    name_is_entered := true;
  elsif s_equal ("classification", user_ptr.parm_name) then
    Vm_Extract_SVAL (user_ptr.parm_ptr, 1, value(1));
    if s_equal (value(1), "U") then
      new_transmit_command(jtids).text.header.classification := U;
    elsif s_equal (value(1), "C") then
      new_transmit_command(jtids).text.header.classification := C;
    elsif s_equal (value(1), "S") then
      new_transmit_command(jtids).text.header.classification := S;
    else
     new_transmit_command(jtids).text.header.classification := TS;
    end if;
  elsif s_equal ("precedence", user_ptr.parm_name) then
    Vm_Extract_SVAL (user_ptr.parm_ptr, 1, value(1));
    if s_equal (value(1), "R") then
```

```
      new_transmit_command(jtids).text.header.precedence := R;
    elsif s_equal (value(1), "P") then
      new_transmit_command(jtids).text.header.precedence := P;
    elsif s_equal (value(1), "O") then
      new_transmit_command(jtids).text.header.precedence := O;
    else
      new_transmit_command(jtids).text.header.precedence := Z;
    end if;
  elsif s_equal ("origin",user_ptr.parm_name) then
    Vm_Extract_SVAL (user_ptr.parm_ptr, 1, value(1));
    new_transmit_command(jtids).text.header.origin :=
                                                  value(1)(1..6);
    origin_is_entered := true;
  elsif s_equal ("to_1", user_ptr.parm_name) then
    Vm_Extract_SVAL (user_ptr.parm_ptr, 1, value(1));
    attach_to_list(
new_transmit_command(jtids).text.header.address,value(1)(1..6));
    address_is_entered := true;
  elsif s_equal ("to_2", user_ptr.parm_name) then
    Vm_Extract_SVAL (user_ptr.parm_ptr, 1, value(1));
    attach_to_list(
new_transmit_command(jtids).text.header.address,value(1)(1..6));
  elsif s_equal ("to_3", user_ptr.parm_name) then
    Vm_Extract_SVAL (user_ptr.parm_ptr, 1, value(1));
    attach_to_list(
new_transmit_command(jtids).text.header.address,value(1)(1..6));
  elsif s_equal ("to_4", user_ptr.parm_name) then
    Vm_Extract_SVAL (user_ptr.parm_ptr, 1, value(1));
    attach_to_list(
new_transmit_command(jtids).text.header.address,value(1)(1..6));
  elsif s_equal ("info_1", user_ptr.parm_name) then
    Vm_Extract_SVAL (user_ptr.parm_ptr, 1, value(1));
    attach_to_list(
    new_transmit_command(jtids).text.header.info,value(1)(1..6));
  elsif s_equal ("info_2", user_ptr.parm_name) then
    Vm_Extract_SVAL (user_ptr.parm_ptr, 1, value(1));
    attach_to_list(
    new_transmit_command(jtids).text.header.info,value(1)(1..6));
  elsif s_equal ("info_3", user_ptr.parm_name) then
    Vm_Extract_SVAL (user_ptr.parm_ptr, 1, value(1));
    attach_to_list(
```

```
  new_transmit_command(jtids).text.header.info,value(1)(1..6));
elsif s_equal ("info_4", user_ptr.parm_name) then
  Vm_Extract_SVAL (user_ptr.parm_ptr, 1, value(1));
  attach_to_list(
  new_transmit_command(jtids).text.header.info,value(1)(1..6));
elsif s_equal ("via_1_1", user_ptr.parm_name) then
  Vm_Extract_SVAL (user_ptr.parm_ptr, 1, value(1));
  check_and_attach_to_via_list("      ",value(1)(1..6),1);
elsif s_equal ("via_1_2", user_ptr.parm_name) then
  Vm_Extract_SVAL (user_ptr.parm_ptr, 1, value(1));
  check_and_attach_to_via_list("      ",value(1)(1..6),2);
elsif s_equal ("via_1_3", user_ptr.parm_name) then
  Vm_Extract_SVAL (user_ptr.parm_ptr, 1, value(1));
  check_and_attach_to_via_list("      ",value(1)(1..6),3);
elsif s_equal ("by_1", user_ptr.parm_name) then
  Vm_Extract_SVAL (user_ptr.parm_ptr, 1, value(1));
  check_and_attach_to_via_list(value(1)(1..6),"      ",4);
elsif s_equal ("via_2_1", user_ptr.parm_name) then
  Vm_Extract_SVAL (user_ptr.parm_ptr, 1, value(1));
  check_and_attach_to_via_list("      ",value(1)(1..6),5);
elsif s_equal ("via_2_2", user_ptr.parm_name) then
  Vm_Extract_SVAL (user_ptr.parm_ptr, 1, value(1));
  check_and_attach_to_via_list("      ",value(1)(1..6),6);
elsif s_equal ("via_2_3", user_ptr.parm_name) then
  Vm_Extract_SVAL (user_ptr.parm_ptr, 1, value(1));
  check_and_attach_to_via_list("      ",value(1)(1..6),7);
elsif s_equal ("by_2", user_ptr.parm_name) then
  Vm_Extract_SVAL (user_ptr.parm_ptr, 1, value(1));
  check_and_attach_to_via_list(value(1)(1..6),"      ",8);
elsif s_equal ("subject", user_ptr.parm_name) then
  Vm_Extract_SVAL (user_ptr.parm_ptr, 1, value(1));
  new_transmit_command(jtids).text.header.subject :=
                                              value(1)(1..60);
elsif s_equal ("body", user_ptr.parm_name) then
  --actually, because of the type of the item (: pageedit),
  --the flow of the program does not enter here ever.
  --the content of the pageedit item
  --( here: body) will be get after SEND is pressed
  Vm_Extract_SVAL (user_ptr.parm_ptr, 1, value(1));
  new_transmit_command(jtids).text.text :=
                                        value(1)(1..TEXT_LENGTH);
```

```
    elsif s_equal ("cancel", user_ptr.parm_name) then
      --reassign the new values to old values, changes are not valid
      Wpt_PanelReset(editor_info.panel_id);
      Wpt_PanelErase(editor_info.panel_id);
      exit;
    elsif s_equal ("send", user_ptr.parm_name) then
      Wpt_CloseItems( editor_info.panel_id );
      get_body_content(new_transmit_command(jtids).text.text);
      if name_is_entered   and
         origin_is_entered and
         address_is_entered  then
        new_editor_selected := true;
        Wpt_PanelReset(editor_info.panel_id);
        Wpt_PanelErase(editor_info.panel_id);
        exit;
      else
        Wpt_ParmReject( editor_info.panel_id, user_ptr.parm_name,
        "Minimum Required Information is Missing");
      end if;
    else
      put_line("Enter data to MESSAGE EDITOR PANEL Only");
    end if;

  when Wpt_WINDOW_EVENT =>
    null;

  when OTHERS =>
    put("ERROR: Unexpected Wpt_NextEvent. ");

  end case;

  end loop;

end get_user_message;


--*************************************************************************
--*** procedure GET_TRANSMISSION_SEQUENCE                              ***
--*************************************************************************
procedure get_transmission_sequence is
```

```
value        : array (1..1) of string (1..tae_taeconf.stringsize);
float_value      : array (1..1) of taefloat;
current_relay_to : integer := 0;

origin_is_entered  : boolean := false;
address_is_entered : boolean := false;


--*********************************************************************
--*** procedure ATTACH_TO_LIST                                      ***
--*********************************************************************
procedure attach_to_list(
         address_list : in out address_link;
            in_name      : in     string ) is

  previous    : address_link;
  current     : address_link := address_list;
  new_address : address_link := new address_type;

begin --attach_to_list

  new_address.name := in_name;

  if current = null then
    new_address.next := address_list;
    address_list     := new_address;
  else
    while current /= null loop
      previous := current;
      current := current.next;
    end loop;
    previous.next := new_address;
  end if;

end attach_to_list;


--*********************************************************************
--*** procedure CHECK_AND_ATTACH_TO_VIA_LIST                        ***
--*********************************************************************
procedure check_and_attach_to_via_list(
```

```
        relay_by : in string;
        to       : in string;
        I        : in integer ) is


--*********************************************************************
--*** procedure ATTACH_TO_VIA_LIST                                  ***
--*********************************************************************
procedure attach_to_via_list(
        relay_by : in string;
        to       : in string ) is

  new_record   : address_link;
  new_via_line : via_record_link;
  current      : address_link;
  previous     : address_link := null;

begin --attach_to_via_list
  if current_relay_to = 1 then
    new_via_line := new via_record;
    new_initiate_trans.header.via_line := new_via_line;
  elsif current_relay_to = 5 then
    new_via_line := new via_record;
    new_initiate_trans.header.via_line.next := new_via_line;
  end if;
  if current_relay_to < 5 then
    new_via_line := new_initiate_trans.header.via_line;
  else
    new_via_line := new_initiate_trans.header.via_line.next;
  end if;
  if relay_by = "      " then   -- relay_to field came
    new_record   := new address_type;
    new_record.name := to;
    current := new_via_line.relay_to;
    if current = null then
      new_record.next := new_via_line.relay_to;
      new_via_line.relay_to := new_record;
    else
      while current /= null loop
        previous := current;
        current := current.next;
```

```
            end loop;
            previous.next :=  new_record;
          end if;
        else
          new_via_line.relay_by := relay_by;
        end if;
      end attach_to_via_list;

    begin --check_and_attach_to_via_list
      if (I >= 1 and I <= 3 and I - current_relay_to = 1) or
         (I >= 5 and I <= 7 and I - current_relay_to = 1) or
         (I=4 and current_relay_to >= 1 and current_relay_to <= 3 ) or
         (I=8 and current_relay_to >=5 and current_relay_to <= 7) then
        current_relay_to := I;
        attach_to_via_list(relay_by,to);
      else
        put_line("Enter Data to APPROPRIATE ADDRESS FIELD");
      end if;
    end check_and_attach_to_via_list;

begin --get_transmission_sequence

    Wpt_NewPanel ("", trackrep_info.target, trackrep_info.view,
         X_Windows.Null_Window, trackrep_info,
         tae_wpt.WPT_DEFAULT, trackrep_info.panel_id);

    loop

      Wpt_NextEvent (wptEvent, etype);

      case etype is

        when Wpt_PARM_EVENT =>

          Wpt_Extract_Context (wptEvent, user_ptr);
          Wpt_Extract_Parm (wptEvent,user_ptr.parm_name);
          Wpt_Extract_Data (wptEvent,user_ptr.datavm_ptr);
          Vm_Find (
            user_ptr.datavm_ptr,user_ptr.parm_name,user_ptr.parm_ptr);

          if s_equal ("link_id", user_ptr.parm_name) then
```

```
  Vm_Extract_SVAL ( user_ptr.parm_ptr, 1, value(1));
  if s_equal (value(1), "JTIDS") then
    new_initiate_trans.link_id := JTIDS;
  elsif s_equal (value(1), "LINK11") then
    new_initiate_trans.link_id := LINK11;
  elsif s_equal (value(1), "LINK16") then
    new_initiate_trans.link_id := LINK16;
  else --if s_equal (value(1), "OTCIXS") then
    new_initiate_trans.link_id := OTCIXS;
  end if;
elsif s_equal ("classification", user_ptr.parm_name) then
  Vm_Extract_SVAL (user_ptr.parm_ptr, 1, value(1));
  if s_equal (value(1), "U") then
    new_initiate_trans.header.classification := U;
  elsif s_equal (value(1), "C") then
    new_initiate_trans.header.classification := C;
  elsif s_equal (value(1), "S") then
    new_initiate_trans.header.classification := S;
  else
    new_initiate_trans.header.classification := TS;
  end if;
elsif s_equal ("precedence", user_ptr.parm_name) then
  Vm_Extract_SVAL (user_ptr.parm_ptr, 1, value(1));
  if s_equal (value(1), "R") then
    new_initiate_trans.header.precedence := R;
  elsif s_equal (value(1), "P") then
    new_initiate_trans.header.precedence := P;
  elsif s_equal (value(1), "O") then
    new_initiate_trans.header.precedence := O;
  else
    new_initiate_trans.header.precedence := Z;
  end if;
elsif s_equal ("origin", user_ptr.parm_name) then
  Vm_Extract_SVAL (user_ptr.parm_ptr, 1, value(1));
  new_initiate_trans.header.origin := value(1)(1..6);
  origin_is_entered := true;
elsif s_equal ("to_1", user_ptr.parm_name) then
  Vm_Extract_SVAL (user_ptr.parm_ptr, 1, value(1));
  attach_to_list(
        new_initiate_trans.header.address,value(1)(1..6));
  address_is_entered := true;
```

```
elsif s_equal ("to_2", user_ptr.parm_name) then
  Vm_Extract_SVAL (user_ptr.parm_ptr, 1, value(1));
  attach_to_list(
        new_initiate_trans.header.address,value(1)(1..6));
  address_is_entered := true;
elsif s_equal ("to_3", user_ptr.parm_name) then
  Vm_Extract_SVAL (user_ptr.parm_ptr, 1, value(1));
  attach_to_list(
        new_initiate_trans.header.address,value(1)(1..6));
  address_is_entered := true;
elsif s_equal ("to_4", user_ptr.parm_name) then
  Vm_Extract_SVAL (user_ptr.parm_ptr, 1, value(1));
  attach_to_list(
        new_initiate_trans.header.address,value(1)(1..6));
  address_is_entered := true;
elsif s_equal ("info_1", user_ptr.parm_name) then
  Vm_Extract_SVAL (user_ptr.parm_ptr, 1, value(1));
  attach_to_list(
          new_initiate_trans.header.info,value(1)(1..6));
elsif s_equal ("info_2", user_ptr.parm_name) then
  Vm_Extract_SVAL (user_ptr.parm_ptr, 1, value(1));
  attach_to_list(
          new_initiate_trans.header.info,value(1)(1..6));
elsif s_equal ("info_3", user_ptr.parm_name) then
  Vm_Extract_SVAL (user_ptr.parm_ptr, 1, value(1));
  attach_to_list(
          new_initiate_trans.header.info,value(1)(1..6));
elsif s_equal ("info_4", user_ptr.parm_name) then
  Vm_Extract_SVAL (user_ptr.parm_ptr, 1, value(1));
  attach_to_list(
          new_initiate_trans.header.info,value(1)(1..6));
elsif s_equal ("via_1_1", user_ptr.parm_name) then
  Vm_Extract_SVAL (user_ptr.parm_ptr, 1, value(1));
  check_and_attach_to_via_list("      ",value(1)(1..6),1);
elsif s_equal ("via_1_2", user_ptr.parm_name) then
  Vm_Extract_SVAL (user_ptr.parm_ptr, 1, value(1));
  check_and_attach_to_via_list("      ",value(1)(1..6),2);
elsif s_equal ("via_1_3", user_ptr.parm_name) then
  Vm_Extract_SVAL (user_ptr.parm_ptr, 1, value(1));
  check_and_attach_to_via_list("      ",value(1)(1..6),3);
elsif s_equal ("by_1", user_ptr.parm_name) then
```

```
          Vm_Extract_SVAL (user_ptr.parm_ptr, 1, value(1));
          check_and_attach_to_via_list(value(1)(1..6),"      ",4);
        elsif s_equal ("via_2_1", user_ptr.parm_name) then
          Vm_Extract_SVAL (user_ptr.parm_ptr, 1, value(1));
          check_and_attach_to_via_list("      ",value(1)(1..6),5);
        elsif s_equal ("via_2_2", user_ptr.parm_name) then
          Vm_Extract_SVAL (user_ptr.parm_ptr, 1, value(1));
          check_and_attach_to_via_list("      ",value(1)(1..6),6);
        elsif s_equal ("via_2_3", user_ptr.parm_name) then
          Vm_Extract_SVAL (user_ptr.parm_ptr, 1, value(1));
          check_and_attach_to_via_list("      ",value(1)(1..6),7);
        elsif s_equal ("by_2", user_ptr.parm_name) then
          Vm_Extract_SVAL (user_ptr.parm_ptr, 1, value(1));
          check_and_attach_to_via_list(value(1)(1..6),"      ",8);
        elsif s_equal ("range", user_ptr.parm_name) then
          Vm_Extract_RVAL (user_ptr.parm_ptr, 1, float_value(1));
          if float_value(1) >= 0.0 then
            new_initiate_trans.dbase_request.the_range :=
                                          float(float_value(1));
          else
            Wpt_ParmReject(
trackrep_info.panel_id, user_ptr.parm_name, "Range >= 0 required");
          end if;
        elsif s_equal ("friendly", user_ptr.parm_name) then
          Vm_Extract_SVAL (trackrep_info.parm_ptr, 1, value(1));
          if value(1)(1..3) = "yes" then
            new_initiate_trans.dbase_request.iff_class(friendly) :=
                                                          true;
          else
            new_initiate_trans.dbase_request.iff_class(friendly) :=
                                                         false;
          end if;
        elsif s_equal ("hostile", user_ptr.parm_name) then
          Vm_Extract_SVAL (trackrep_info.parm_ptr, 1, value(1));
          if value(1)(1..3) = "yes" then
            new_initiate_trans.dbase_request.iff_class(hostile) :=
                                                          true;
          else
            new_initiate_trans.dbase_request.iff_class(hostile) :=
                                                         false;
          end if;
```

```
elsif s_equal ("neutral", user_ptr.parm_name) then
  Vm_Extract_SVAL (trackrep_info.parm_ptr, 1, value(1));
  if value(1)(1..3) = "yes" then
    new_initiate_trans.dbase_request.iff_class(neutral) :=
                                                        true;
  else
    new_initiate_trans.dbase_request.iff_class(neutral) :=
                                                       false;
  end if;
elsif s_equal ("unknown", user_ptr.parm_name) then
  Vm_Extract_SVAL (trackrep_info.parm_ptr, 1, value(1));
  if value(1)(1..3) = "yes" then
    new_initiate_trans.dbase_request.iff_class(unknown) :=
                                                        true;
  else
    new_initiate_trans.dbase_request.iff_class(unknown) :=
                                                       false;
  end if;
elsif s_equal ("air", user_ptr.parm_name) then
  Vm_Extract_SVAL (trackrep_info.parm_ptr, 1, value(1));
  if value(1)(1..3) = "yes" then
    new_initiate_trans.dbase_request.track_class(air) :=
                                                        true;
  else
    new_initiate_trans.dbase_request.track_class(air) :=
                                                      false;
  end if;
elsif s_equal ("surface", user_ptr.parm_name) then
  Vm_Extract_SVAL (trackrep_info.parm_ptr, 1, value(1));
  if value(1)(1..3) = "yes" then
    new_initiate_trans.dbase_request.track_class(surface)
                                                    := true;
  else
    new_initiate_trans.dbase_request.track_class(surface)
                                                   := false;
  end if;
elsif s_equal ("subsurface", user_ptr.parm_name) then
  Vm_Extract_SVAL (trackrep_info.parm_ptr, 1, value(1));
  if value(1)(1..3) = "yes" then
    new_initiate_trans.dbase_request.track_class(surface)
                                                   := true;
```

```
            else
              new_initiate_trans.dbase_request.track_class(surface)
                                                              := false;
            end if;
          elsif s_equal ("cancel", user_ptr.parm_name) then
            new_initiate_trans := old_initiate_trans;
            Wpt_PanelErase(trackrep_info.panel_id);
            exit;
          elsif s_equal ("ok", user_ptr.parm_name) then
            if ( origin_is_entered and address_is_entered ) then
              old_initiate_trans := new_initiate_trans;
              new_terminate_trans := false;
              Wpt_PanelErase(trackrep_info.panel_id);
              exit;
            else
              Wpt_ParmReject( trackrep_info.panel_id,
       user_ptr.parm_name,"Minimum Required Information is Missing");
            end if;
          else
            put_line(
                    "Enter data to PERIODIC TRACK REPORT PANEL Only");
          end if;

        when Wpt_WINDOW_EVENT =>
          null;

        when OTHERS =>
          put("ERROR: Unexpected Wpt_NextEvent. ");

      end case;

    end loop;

    new_initiate_trans.header.subject(1..12) := "track_report";
    new_initiate_trans.desired_format := "      ";

  end get_transmission_sequence;
```

```
--*********************************************************************
--*** procedure GET_TERMINATE_TRACK_REPORT                          ***
--*********************************************************************
procedure get_terminate_track_report is

begin --get_terminate_track_report

  Wpt_NewPanel ("", tertrrep_info.target, tertrrep_info.view,
     X_Windows.Null_Window, tertrrep_info,
     tae_wpt.WPT_DEFAULT, tertrrep_info.panel_id);

  loop

    Wpt_NextEvent (wptEvent, etype);

    case etype is

      when Wpt_PARM_EVENT =>

        Wpt_Extract_Context (wptEvent, user_ptr);
        Wpt_Extract_Parm (wptEvent,user_ptr.parm_name);
        Wpt_Extract_Data (wptEvent,user_ptr.datavm_ptr);
        Vm_Find (
       user_ptr.datavm_ptr,user_ptr.parm_name,user_ptr.parm_ptr);
        if s_equal ("cancel", user_ptr.parm_name) then
          Wpt_PanelErase(tertrrep_info.panel_id);
          exit;
        elsif s_equal ("ok", user_ptr.parm_name) then
          new_terminate_trans := true;
          Wpt_PanelErase(tertrrep_info.panel_id);
          exit;
        else
          put_line(
              "Enter data to TERMINATE TRACK REPORT PANEL Only");
        end if;

      when Wpt_WINDOW_EVENT =>
        null;

      when OTHERS =>
        put("ERROR: Unexpected Wpt_NextEvent. ");
```

```
      end case;

    end loop;

  end get_terminate_track_report;


begin--MANAGE_USER_INTERFACE


  if not panels_are_initialized then
    InitializePanels( file => tae_source_file );
    panels_are_initialized := true;
  end if;

  while Wpt_Pending loop

    Wpt_NextEvent (wptEvent, etype);

    case etype is

      when Wpt_PARM_EVENT =>

        Wpt_Extract_Context (wptEvent, user_ptr);
        Wpt_Extract_Parm (wptEvent, user_ptr.parm_name);
        Wpt_Extract_Data (wptEvent, user_ptr.datavm_ptr);
        Vm_Find (
         user_ptr.datavm_ptr, user_ptr.parm_name, user_ptr.parm_ptr);

        if "=" (user_ptr, main_info)  then
          if s_equal ("options", user_ptr.parm_name) then
            Vm_Extract_SVAL (user_ptr.parm_ptr, 1, value(1));
            if s_equal (value(1), "Archive Setup") then
              get_archive_setup_values;
            elsif s_equal (value(1), "Track Filter Values") then
              get_track_filter_values;
            elsif s_equal (value(1), "Display Tracks") then
              get_track_display_constraints;
            elsif s_equal (value(1), "Weapon Status") then
              if weapon_status_panel_is_displayed = false then
                new_status_query := true;
```

```
      else
        put_line("WEAPON STATUS PANEL is already displayed");
      end if;
    elsif s_equal (value(1), "EMCON Status" ) then
      get_emcon_status;
    elsif s_equal (value(1), "Network Setup" ) then
      get_network_setup;
    elsif s_equal (value(1), "Message Editor") then
      get_user_message;
    elsif s_equal (value(1), "Read Message") then
      message_retrieval_panel;
    elsif s_equal (value(1), "Periodic Track Report") then
      get_transmission_sequence;
    elsif s_equal (value(1), "Terminate Track Report") then
      get_terminate_track_report;
    end if;
  elsif s_equal ("quit", user_ptr.parm_name) then
    if track_display_panel_is_displayed = true then
      Wpt_PanelErase(display_info.panel_id);
      track_display_panel_is_displayed := false;
    end if;
    if weapon_status_panel_is_displayed = true then
      Wpt_PanelErase(weapon_info.panel_id);
      weapon_status_panel_is_displayed := false;
    end if;
    if emergency_weapon_status_panel_is_displayed = true then
      Wpt_PanelErase(emerg_info.panel_id);
      emergency_weapon_status_panel_is_displayed := false;
    end if;
    Wpt_PanelErase(main_info.panel_id);
  end if;
elsif "=" (user_ptr, display_info)  then
  if s_equal ("change_values", user_ptr.parm_name) then
    get_track_display_constraints;
  elsif s_equal ("quit", user_ptr.parm_name) then
    Wpt_PanelErase(display_info.panel_id);
    track_display_panel_is_displayed := false;
  end if;
elsif "=" (user_ptr, weapon_info)  then
  if s_equal ("ok", user_ptr.parm_name) then
    Wpt_PanelErase(weapon_info.panel_id);
```

```
            weapon_status_panel_is_displayed := false;
          end if;
        elsif "=" (user_ptr, emerg_info)  then
          if s_equal ("em_ok", user_ptr.parm_name) then
            Wpt_PanelErase(emerg_info.panel_id);
            emergency_weapon_status_panel_is_displayed := false;
          end if;
        else
           put_line("unexpected event from wpt!");
        end if;

      when Wpt_WINDOW_EVENT =>
        null;
      when OTHERS =>
        put("ERROR: Unexpected Wpt_NextEvent. ");
    end case;
     end loop;
end manage_user_interface;
end SB;
```

TABLE OF CONTENTS FOR APPENDIX F

Type Declarations 147

Procedure Declarations 151

Variable Declarations 154

Procedure comms_links 158

Simulates the communications links connected to the system, and produces messages.

Procedure prepare_periodic_report 172

Prepares the periodic track reports

Procedure parse_input_file 177

Parses the incoming communications message and writes the contents of the message into an Ada record type for further processing.

Procedure decide_for_archiving 181

Depending on the archive setup value, this procedure archives the input file

Procedure extract_tracks 182

Extracts the track data from the message.

Procedure filter_comms_tracks 186

Filters the tracks from the communication links according to the track filtering values.

Procedure add_comms_track 190

Adds the filtered tracks to the track list.

Procedure sensors 190

Simulates the sensors connected to the system and generates track data.

Procedure analyze_sensor_data 196

Extracts the track information from the input variable.

Procedure prepare_sensor_track 196

Calculates the real position of a track from the relative position.

Procedure filter_sensor_tracks 197

Filters the tracks from the sensors according to the track filtering values.

Procedure add_sensor_track 198

Adds the filtered tracks to the track list.

Procedure navigation_system 199

Simulates the navigation system and generates own position data.

Procedure monitor_ownship_position 199

Modifies own position value in the track database.

Procedure weapons_systems 202

Simulates the weapon systems connected to the C^3I system, and produces random situation reports for each weapon system.

Procedure weapons_interface 203

Provides the last situations of the weapon systems.

Procedure make_routing 204

Prepares the route field of the outgoing messages.

Procedure forward_for_transmission 207

It acts as a buffer to keep the outgoing messages until the EMCON status allows message transmission.

Procedure convert_to_text_file 207

Writes the contents of the input Ada record type into a text file which represents the outgoing message.

Procedure message_editor 210

Assigns the value of the message generated by the user to the output data stream.

Procedure display_tracks 211

Outputs the track values to the screen.

Procedure get_user_inputs 217

Assigns the current values of various variables to the output data streams.

Procedure status_screen 218

Displays the weapon status panel on the screen.

Procedure emergency_status_screen 219

Displays the emergency weapon status panel on the screen.

Procedure message_arrival_panel 222

Displays the name of the message which is received from the communications links.

Procedure manage_user_interface 227

This is the control procedure for the user interface. It senses the user inputs, opens necessary panels, gets the user inputs, and assigns them to the related variables.

APPENDIX G

AN ALGORITHM TO CALCULATE MINIMUM CALLING PERIODS (MCP) AND MAXIMUM RESPONSE TIMES (MRT)

The initial version of the static scheduler required that Minimum Calling Period and the Maximum Response Time must be specified for each time critical operator in the prototype [Ref. 9]. The static scheduler is improved by implementing an algorithm that calculates these timing constraints from the Maximum Execution Time, if they are not specified by the user.

This algorithm calculates the minimum calling periods and the maximum response times of the sporadic operators by making four passes on the graph that includes the data for these time critical operators.

In the first pass the load factor and the greatest common divisor for the periods of the periodic operators is calculated. The load factor is the sum of ratios for maximum execution time over period for all periodic operators.

In the second pass the number of sporadic operators is found, and for the sporadic operators with a defined MCP or MRT, the undefined MCP or MRT value is calculated with the assumption that MCP = MRT - MET, where MET is the maximum Execution time of the operator. Then, for these operators, an equivalent period (= MCP) and a unit factor is calculated. Unit factor is a coefficient which is used to avoid combinatorical explosion in the length of the static schedule. To calculate the unit factor, for each sporadic operator with user defined MCP and/or MRT value, an even ratio of calculated equivalent period over the greatest common divisor of the periods of the peri-

odic operators is found. Then, the least common multiplier of these ratios is defined as the unit factor.

In the third pass equivalent periods for the sporadic operators with MCP and/or MRT value is calculated and inserted in the graph by using the unit factor calculated in the second pass such that,

PERIOD = PERIOD - (PERIOD mod UNIT FACTOR)

Then the load factor calculated in the first pass is modified to include the ratios for these sporadic operators.

In the fourth pass MCP, MRT and PERIOD values for the sporadic operators without user defined MCP or MRT values are calculated and entered to the graph by using following assumptions :

MRT = ALPHA * MET

MCP = PERIOD = MRT - MET

These assumptions guarantee a feasible schedule for some constant ALPHA, which is determined by the Mok's Theorem [Ref. 10]. This theorem states that a schedule is feasible if total load is less than or equal to 0.5. By using the above equations, and the Mok's theorem, ALPHA is determined.

ALPHA = (number of sporadic operators / (C - load factor)) + 1

Where C is 0.5 if load factor is less than 0.5, otherwise the average value of 0.5 and load factor. If load factor exceeds 1, then user is notified that a feasible schedule is not possible with single processor.

The Ada code for the algorithm is as follows, and it is inserted into file processor package which is in fp_b.a file.

```
procedure FIND_PERIODS( THE_GRAPH : in out DIGRAPH.GRAPH ) is

  TARGET      : DIGRAPH.V_LISTS.LIST;
  N           : NATURAL := 0;
  L           : FLOAT := 0.0;
  NEW_PERIOD  : NATURAL := 0;
  OP          : OPERATOR;
  C           : FLOAT;
  FIRST       : BOOLEAN := true;
  FOUND       : BOOLEAN := false;
  FRACTION    : NATURAL;
  FR_GCD      : NATURAL;
  LCM         : NATURAL;
  UNIT        : NATURAL;
  ALPHA       : FLOAT;
  GCD         : NATURAL;
 package I_IO is new TEXT_IO.INTEGER_IO(NATURAL);

  procedure CALCULATE_NEW_PERIOD (O          : in OPERATOR;
                                  NEW_PERIOD : in out NATURAL  ) is
    DIFFERENCE : NATURAL;
    package VALUE_IO is new TEXT_IO.INTEGER_IO(NATURAL);
  begin
    DIFFERENCE := O.THE_MRT - O.THE_MET;
    if DIFFERENCE < O.THE_MCP then
      NEW_PERIOD := DIFFERENCE;
    else
      NEW_PERIOD := O.THE_MCP;
    end if;
    TEXT_IO.put("The new PERIOD is => ");
    VALUE_IO.put(NEW_PERIOD);
```

```
  TEXT_IO.NEW_LINE;
end CALCULATE_NEW_PERIOD;


function FIND_GCD (SMALL : in VALUE; LARGE : in VALUE)
                                                    return VALUE is
  REMAINDER : VALUE := SMALL;
begin
  if LARGE mod SMALL = 0 then
    return REMAINDER;
  else
    REMAINDER := FIND_GCD(LARGE mod SMALL, SMALL);
    return REMAINDER;
  end if;
end FIND_GCD;

function FIND_LCM (NUMBER1, NUMBER2 : VALUE) return VALUE is
begin
  return(NUMBER1 * NUMBER2) / GCD;
end FIND_LCM;

function REDUCE_TO_EVEN_FRACTION( GCD,PERIOD : NATURAL)
                                                  return NATURAL is
  N : NATURAL := GCD / PERIOD;
begin
  if N * PERIOD = GCD then
    return N;
  else
    return N + 1;
  end if;
end REDUCE_TO_EVEN_FRACTION;
```

```
begin
  -- FIRST PASS
  TARGET := THE_GRAPH.VERTICES;
  while DIGRAPH.V_LISTS.NON_EMPTY(TARGET) loop
    OP := DIGRAPH.V_LISTS.VALUE(TARGET);
    if OP.THE_MET = 0 then
      Exception_Operator := OP.THE_OPERATOR_ID;
      raise CRIT_OP_LACKS_MET;
    elsif OP.THE_PERIOD /= 0 then  -- a periodic operator
      L := L + FLOAT(OP.THE_MET)/FLOAT(OP.THE_PERIOD);
      if FIRST then
        GCD := OP.THE_PERIOD;
        FIRST := false;
      else
        if GCD > OP.THE_PERIOD then
          GCD := FIND_GCD(OP.THE_PERIOD,GCD);
        else
          GCD := FIND_GCD(GCD,OP.THE_PERIOD);
        end if;
      end if;
    end if;
    DIGRAPH.V_LISTS.NEXT(TARGET);
  end loop;

  -- SECOND PASS

  TARGET := THE_GRAPH.VERTICES;
  FIRST := true;
  while DIGRAPH.V_LISTS.NON_EMPTY(TARGET) loop
    OP := DIGRAPH.V_LISTS.VALUE(TARGET);
```

```
if OP.THE_PERIOD = 0 then  -- a sporadic operator
  if OP.THE_MCP /= 0 and OP.THE_MRT = 0 then
    OP.THE_MRT := OP.THE_MET + OP.THE_MCP;
    TARGET.ELEMENT.THE_MRT := OP.THE_MRT;
  elsif OP.THE_MCP = 0 and OP.THE_MRT /= 0 then
    OP.THE_MCP := OP.THE_MRT - OP.THE_MET;
    TARGET.ELEMENT.THE_MCP := OP.THE_MCP;
  end if;
  if OP.THE_MCP /= 0 and OP.THE_MRT /= 0 then
    CALCULATE_NEW_PERIOD(OP,NEW_PERIOD);
    if NEW_PERIOD < GCD then
      FOUND := true;
      FRACTION := GCD/REDUCE_TO_EVEN_FRACTION(GCD,NEW_PERIOD);
      if FIRST then
        FR_GCD := FRACTION;
LCM := FRACTION;
        FIRST := false;
      else
        if FRACTION > FR_GCD then
          FR_GCD := FIND_GCD(FR_GCD, FRACTION);
        else
          FR_GCD := FIND_GCD(FRACTION,FR_GCD);
        end if;
        LCM := FIND_LCM(LCM,FRACTION);
      end if;
    end if;
  else
    N := N + 1;
  end if;
end if;
DIGRAPH.V_LISTS.NEXT(TARGET);
```

```
end loop;
if FOUND then
  UNIT := LCM;
else
  UNIT := GCD;
end if;

-- THIRD PASS
TARGET := THE_GRAPH.VERTICES;
while DIGRAPH.V_LISTS.NON_EMPTY(TARGET) loop
  OP := DIGRAPH.V_LISTS.VALUE(TARGET);
  if OP.THE_PERIOD = 0 then  -- a sporadic operator
    if OP.THE_MRT /= 0 and OP.THE_MCP /= 0 then
      CALCULATE_NEW_PERIOD(OP,NEW_PERIOD);
      NEW_PERIOD := NEW_PERIOD - NEW_PERIOD mod UNIT;
OP.THE_PERIOD := NEW_PERIOD;
      TEXT_IO.PUT("New PERIOD for operator ");
      VARSTRING.PUT(OP.THE_OPERATOR_ID);
      TEXT_IO.PUT(" is ");
      I_IO.PUT(NEW_PERIOD,1);
      TEXT_IO.NEW_LINE;
      TARGET.ELEMENT.THE_PERIOD := OP.THE_PERIOD;
      L := L + FLOAT(OP.THE_MET)/FLOAT(NEW_PERIOD);
    end if;
  end if;
  DIGRAPH.V_LISTS.NEXT(TARGET);
end loop;

if L < 0.5 then
  C := 0.5;
elsif L >= 0.5 and L < 1.0 then
```

```
  C := (1.0 + L) / 2.0;
else
  raise NOT_FEASIBLE;
end if;
ALPHA := FLOAT(N)/(C - L) + 1.0;
if ALPHA < 2.0 then
  ALPHA := 2.0;
end if;

-- FOURTH PASS
TARGET := THE_GRAPH.VERTICES;
while DIGRAPH.V_LISTS.NON_EMPTY(TARGET) loop
  OP := DIGRAPH.V_LISTS.VALUE(TARGET);
  if OP.THE_PERIOD = 0 then  -- a sporadic operator
    if OP.THE_MRT = 0 and OP.THE_MCP = 0 then
OP.THE_MRT := NATURAL(ALPHA) * OP.THE_MET;
      OP.THE_MCP := OP.THE_MRT - OP.THE_MET;
      if (OP.THE_MCP / UNIT) * UNIT /= OP.THE_MCP then
        OP.THE_PERIOD := OP.THE_MCP +
                                    UNIT - (OP.THE_MCP mod UNIT);
      else
        OP.THE_PERIOD := OP.THE_MCP;
      end if;
      TEXT_IO.PUT("New PERIOD for operator ");
      VARSTRING.PUT(OP.THE_OPERATOR_ID);
      TEXT_IO.PUT(" is ");
      I_IO.PUT(OP.THE_PERIOD,1);
      TEXT_IO.NEW_LINE;
    end if;
  end if;
  TARGET.ELEMENT.THE_PERIOD := OP.THE_PERIOD;
```

```
      TARGET.ELEMENT.THE_MRT := OP.THE_MRT;
      TARGET.ELEMENT.THE_MCP := OP.THE_MCP;
      DIGRAPH.V_LISTS.NEXT(TARGET);
    end loop;
  end FIND_PERIODS;
```

APPENDIX H

PRODUCING EXECUTABLE

USER INTERFACE CODE

IN ADA

We used ***TAE Plus*** software to create a C^3I workstation User Interface. TAE Plus is a user interface tool based on X Window System, Version 11, release 3, and the X Toolkit.

To create the C^3I Workstation User Interface, we first used *TAE Plus Workbench* tool. We created the *panels* (windows) and *items* needed for the user interface, in the Workbench environment. After creation, the Workbench saves the required information into the *TAE Plus resource file, user.res*. While still in the Workbench environment, we chose the *Generate* selection of the *Utility* option from Workbench, to generate Ada programs for user interface. If you do not change anything in this program, after compiling and loading with TAE Plus library packages on your path, when you run the program, it outputs values you have entered into the panels to the standard output (Screen).

There were differences between the program that *TAE Plus Workbench* tool generated, and the program that we needed. First of all, the program that TAE Plus generated is arranged to be a main program, so that this User Interface program would control the whole environment, and all other utility subprograms would run under control of this main program, in accordance with the user inputs. But in our case, our main program is created by CAPS and we needed only subprograms corresponding to each atomic operator, not a main program that controls the whole environment. Another dif-

ference was, we wanted to store the inputs that the user enters, into the variables that the atomic operators should output, instead of outputting those to the standard output device.

We have changed the program to satisfy our needs. The resulting Ada code became a collection of separate subprograms which match the atomic modules of the C^3I Workstation prototype. These subprograms are included in the software base which is given in Appendix F.

A. REQUIREMENTS IN ORDER TO USE THE TAE Plus PROGRAM

In order to run the TAE Plus program, you have to bring up X-Window with *uwm* window manager. Also the *TAE Plus resource file* that is created by using *Workbench* (user.res in our case) must be in the current directory.

B. WALKING THROUGH TAE PLUS UTILITY FUNCTIONS

Package TAE is the program that contains all of the utility functions. There are four separate packages declared in this package that we have used. These packages and the utility functions are as follows:

1. Package TAE.TAE_WPT

The Window Programming Tools package is a library of application program-callable routines used to define and control elements of the TAE Plus user interface.

The following Utility functions are used in our application:

- Wpt_NewPanel : Displays an Interaction Panel. This function is used every time a previously established panel is needed to either get the user inputs or to output some values.
- Wpt_Extract_Context, Wpt_Extract_Parm, Wpt_Extract_Data, Vm_Find functions are used as a block every time the user inputs

some value to one of the panels, in order to extract the contents of the input.

- Wpt_PanelReset : Resets the parameter values in a specific panel to the values those are initially shown when Wpt_NewPanel was first called to display the panel. This function is used in two cases. First, we may always choose to display the values as initialized, not as made on the previous session. In this case, this function is used to restore the initial values of the items on the panel before removing the panel from the screen, so that when same panel is called another time, it will have the initial values, instead of the values as entered in this session. Second, when we want to clear the information on the items, this procedure allows to clear the items on the panel if their initial values are blank.
- Wpt_PanelErase : Removes the displayed panel from the screen.
- Wpt_SetString : Sets the value of a *TAE Plus string variable* and updates the displayed value of the associated TAE Plus item.
- Wpt_SetReal : Sets the value of a TAE Plus float variable and updates the displayed value of the associated TAE Plus item.
- Wpt_SetIntg: Sets the value of a TAE Plus integer variable and updates the displayed value of the associated TAE Plus item.
- Wpt_ParmReject : Generates a rejection message for a given parameter, and restores the last value before the change on the item.
- Wpt_NextEvent : Reads next panel-related event from Wpt.
- Wpt_Pending : Determines if the user entered a value to any panels.

2. Package TAE.T E_CO

The Collection Package provides the capability to manage groups of TAE objects.

- Co_New : Creates an empty collection. This is used for whole TAE program once, to create the collection of all objects.
- Co_ReadFile : Reads collection of Vm objects from the *TAE resource file*.
- Co_Find : Finds the related information about each panel in the collection of all the objects.

3. Package TAE.TAE_VM

The Collection Package allows the program to manipulate process TAE variables.

- Vm_Find : Finds the related variable in the structure object.
- Vm_Extract_SVAL : Extracts string value of TAE variable structure.
- Vm_Extract_IVAL : Extracts integer value of TAE variable structure.
- Vm_Extract_RVAL : Extracts float value of TAE variable structure.

4. Package TAE.TAE_MISC

The Miscellaneous Functions Package provides some utility functions.

- s_equal (variable, parameter_name) : checks if the user entered a a specific value.
- f_force_lower : enables lower case file names for source files.

C. INITIALIZING THE TAE PLUS APPLICATION ENVIRONMENT

Before using any of the TAE+ subprograms, The panels must be initialized. This is done by procedure InitializePanels in our application. Procedure InitializePanels opens the *TAE resource file* and then associates each *panel pointer* with the related panel.

D. TAE PLUS PROCEDURES

A typical TAE Plus procedure consists of :

- Wpt_NewPanel procedure to open the related panel
- Wpt_NextEvent procedure to get the next panel related event, mostly the ones that the user entered,
- Case statement to choose different types of events,
- Wpt_Extract_Context, Wpt_Extract_Parm, Wpt_Extract_Data, Vm_Find functions to extract the contents of the input, if the eventtype is a Wpt_PARM_EVENT.
- if-clause on `s_equal` function to determine which input the user entered, if the eventtype is a Wpt_PARM_EVENT. The related actions follows related `if section`.
- when eventtype is Wpt_WINDOW_EVENT, nothing is to be done.

No other types of events are expected in this case statement. So if any other type of event arrives, it is handled with a put_line statement following the related *when OTHERS* selection. A typical procedure is shown below:

```
procedure message_retrieval_panel is
   name  : filename;
   count : taeint;
   value : array (1..1) of string(1..tae_taeconf.STRINGSIZE);
   filename_is_entered : boolean := false;
begin --message_retrieval_panel
   Wpt_NewPanel ("", messret_info.target, messret_info.view,
       X_Windows.Null_Window, messret_info, Wpt_DEFAULT,
       messret_info.panel_id);
   loop
      Wpt_NextEvent (wptEvent, etype);
      case etype is
         when Wpt_PARM_EVENT =>
            Wpt_Extract_Context (wptEvent, user_ptr);
            Wpt_Extract_Parm (wptEvent,user_ptr.parm_name);
            Wpt_Extract_Data (wptEvent,user_ptr.datavm_ptr);
            Vm_Find(user_ptr.datavm_ptr,user_ptr.parm_name,
                    user_ptr.parm_ptr);
            if s_equal ("filename", user_ptr.parm_name) then
               Vm_Extract_SVAL (user_ptr.parm_ptr, 1,value(1));
               file_name := new_name(value(1)(1..10));
               filename_is_entered := true;
            elsif s_equal ("display", user_ptr.parm_name) then
              if filename_is_entered = true then
                  Wpt_PanelReset(messret_Info.Panel_id);
                  Wpt_PanelErase(messret_info.panel_id);
                  messcame_display (file_name);
                  exit;
               else
                  Wpt_ParmReject( messret_info.panel_id,
                  user_ptr.parm_name, "Filename should be Entered");
               end if;
            elsif s_equal ("cancel", user_ptr.parm_name) then
                Wpt_PanelReset(messret_Info.Panel_id);
                Wpt_PanelErase(messret_info.panel_id);
                exit;
            else
                put_line("Enter data to MESSAGE RETRIEVAL PANEL Only");
            end if;
         when Wpt_WINDOW_EVENT =>
             null;
         when OTHERS =>
```

```
            put("ERROR: Unexpected Wpt_NextEvent. ");
        end case;
      end loop;
    end message_retrieval_panel;
```

The boolean variable *filename_is_entered* is used in the procedure to check if filename is entered , before trying to retrieve the file. If the filename is not entered yet, then the SEND event is rejected by displaying a related message, by using Wpt_ParmReject.

APPENDIX I

ADA CODE

FOR EXCLUDED MODULES

This appendix includes the reusable Ada components for the atomic operators which are not included in the running version of the C^3I prototype. These modules can be used to improve the features of the prototype by following the prototyping steps explained in previous chapters.

```
-- ************************************************************************
-- * MODULE NUMBER   : 1.1.2
-- * MODULE NAME     : DECIDE FOR RELAYING
-- * DATE            : 9 AUGUST 1990
-- * REVISED         : 5 SEPTEMBER 1990
-- ************************************************************************

with DECLARATIONS;
 use DECLARATIONS;

procedure DECIDE_FOR_RELAYING(
        INPUT_TEXT_RECORD     : in out TEXT_RECORD;
        TCD_NETWORK_SETUP     : in out NETWORK_SETUP;
        TCD_TRANSMIT_COMMAND  : in out TRANSMIT_COMMAND ) is
  VIA_POINTER   : VIA_RECORD_LINK;
  TO_POINTER    : ADDRESS_LINK;
```

```
  ADDR_PTR      : ADDRESS_LINK;
  FOUND         : BOOLEAN := false;
  NEW_ADDRESS   : ADDRESS_LINK;
  LINK          : LINKS_TYPE;
begin
  VIA_POINTER := INPUT_TEXT_RECORD.HEADER.VIA_LINE;
  while VIA_POINTER /= null loop
    if VIA_POINTER.RELAY_BY = OWN_ADDRESS then
      TO_POINTER := VIA_POINTER.RELAY_TO;
      while TO_POINTER /= null loop
        LINK := JTIDS;
        FOUND := false;
        while not FOUND loop
          ADDR_PTR := TCD_NETWORK_SETUP(LINK);
          while ADDR_PTR /= null and not FOUND loop
            if TO_POINTER.NAME = ADDR_PTR.NAME then
              FOUND := true;
              NEW_ADDRESS       := new ADDRESS_TYPE;
              NEW_ADDRESS.NAME  := TO_POINTER.NAME;
              NEW_ADDRESS.NEXT :=
                              TCD_TRANSMIT_COMMAND(LINK).ROUTE_ADDR;
              TCD_TRANSMIT_COMMAND(LINK).ROUTE_ADDR := NEW_ADDRESS;
              TCD_TRANSMIT_COMMAND(LINK).ROUTED     := true;
              TCD_TRANSMIT_COMMAND(LINK).FULL       := true;
              TCD_TRANSMIT_COMMAND(LINK).TEXT       :=
                                                  INPUT_TEXT_RECORD;
            end if;
            ADDR_PTR := ADDR_PTR.NEXT;
          end loop;
          if LINK = OTCIXS then
            exit;
```

```
          else
            LINK := LINKS_TYPE'SUCC(LINK);
          end if;
        end loop;
        TO_POINTER := TO_POINTER.NEXT;
      end loop;
      INPUT_TEXT_RECORD.RELAYED := true;
    end if;
    VIA_POINTER := VIA_POINTER.NEXT;
  end loop;
end DECIDE_FOR_RELAYING;


-- ********************************************************************
-- * MODULE NUMBER : 1.2.2.
-- * MODULE NAME   : FORWARD FOR TRANSLATION
-- * DATE          : 10 AUGUST 1990
-- * REVISED       : 8 SEPTEMBER 1990;
-- ********************************************************************
with DECLARATIONS;
 use DECLARATIONS;

package FORWARD_FOR_TRANSLATION is
  TRANSLATE : boolean := false;
  procedure FORWARD_FOR_TRANSLATION(
               TRANSMISSION_MESSAGE : in out TRANSMISSION_COMMAND;
               READY_MESSAGE        : out    TRANSMISSION_COMMAND;
               RAW_MESSAGE          : in out TRANSLATION_COMMAND );
end FORWARD_FOR_TRANSLATION;

package body FORWARD_FOR_TRANSLATION is
```

```
procedure FORWARD_FOR_TRANSLATION(
              TRANSMISSION_MESSAGE : in out TRANSMISSION_COMMAND;
              READY_MESSAGE        : out    TRANSMISSION_COMMAND;
              RAW_MESSAGE          : in out TRANSLATION_COMMAND) is

  procedure MATCH_FORMAT ( LINK           : in   LINKS_TYPE;
                           SOURCE_FORMAT  : in   STRING;
                           DESIRED_FORMAT : out  STRING     ) is
  -- this procedure looks at a table with the input variables,
  -- and finds the  matching format.
  begin
    null;
  end MATCH_FORMAT;

begin
  for LINK in LINKS_TYPE loop
    if TRANSMISSION_MESSAGE(LINK).FULL then
      MATCH_FORMAT( LINK,
                    TRANSMISSION_MESSAGE(LINK).TEXT.FORMAT,
                    RAW_MESSAGE(LINK).DESIRED_FORMAT);
      if TRANSMISSION_MESSAGE(LINK).TEXT.FORMAT /=
                        RAW_MESSAGE(LINK).DESIRED_FORMAT then
        TRANSLATE := true;
        RAW_MESSAGE(LINK).TEXT := TRANSMISSION_MESSAGE(LINK).TEXT;
        RAW_MESSAGE(LINK).ROUTE_ADDR :=
                          TRANSMISSION_MESSAGE(LINK).ROUTE_ADDR;
        RAW_MESSAGE(LINK).FULL := TRANSMISSION_MESSAGE(LINK).FULL;
        RAW_MESSAGE(LINK).SOURCE_FORMAT :=
                         TRANSMISSION_MESSAGE(LINK).TEXT.FORMAT;
      else
        TRANSLATE := false;
```

```
        end if;
      end if;
    end loop;
    READY_MESSAGE := TRANSMISSION_MESSAGE;
  end FORWARD_FOR_TRANSLATION;
end FORWARD_FOR_TRANSLATION;


-- ***************************************************************
-- * MODULE NUMBER : 1.3
-- * MODULE NAME   : TRANSLATE MESSAGE
-- * DATE          : 13 AUGUST 1990
-- * REVISED       : 7 SEPTEMBER 1990
-- ***************************************************************
with DECLARATIONS, TEXT_IO;
 use DECLARATIONS, TEXT_IO;
procedure TRANSLATE_MESSAGE(
                RAW_MESSAGE   : in  TRANSLATION_COMMAND;
                READY_MESSAGE : out TRANSMISSION_COMMAND ) is

begin
  PUT_LINE("*** MESSAGE TRANSLATION IS IN PROCESS.....");
  for LINK in LINKS_TYPE loop
    READY_MESSAGE(LINK).ROUTE_ADDR  := RAW_MESSAGE(LINK).ROUTE_ADDR;
    READY_MESSAGE(LINK).FULL        := RAW_MESSAGE(LINK).FULL;
    READY_MESSAGE(LINK).TEXT        := RAW_MESSAGE(LINK).TEXT;
    READY_MESSAGE(LINK).TEXT.FORMAT :=
                                    RAW_MESSAGE(LINK).DESIRED_FORMAT;
  end loop;
  PUT_LINE("*** TRANSLATION COMPLETED.....");
end TRANSLATE_MESSAGE;
```

```
-- ***************************************************************************
-- * MODULE NUMBER   : 3.2.1
-- * MODULE NAME     : CHECK_FOR_TIMEOUT_AND_RANGE
-- * DATE            : 17 AUGUST 1990
-- * REVISED         : 13 SEPTEMBER 1990
-- ***************************************************************************

with DECLARATIONS, CALENDAR,TEXT_IO;
 use DECLARATIONS,          TEXT_IO;

package CHECK_FOR_TIMEOUT_AND_RANGE is
  BAD_TRACK_FOUND : BOOLEAN := false;
  procedure CHECK_FOR_TIMEOUT_AND_RANGE(
        TDD_CONSTRAINTS : in     SET_MONITOR_CONSTRAINTS;
        TRACKS          : in     TRACK_TUPLE;
        DELETE_TRACK    : in out DELETE_TRACK_TUPLE );

  end CHECK_FOR_TIMEOUT_AND_RANGE;


package body CHECK_FOR_TIMEOUT_AND_RANGE is
  procedure CHECK_FOR_TIMEOUT_AND_RANGE(
        TDD_CONSTRAINTS : in     SET_MONITOR_CONSTRAINTS;
        TRACKS          : in     TRACK_TUPLE;
        DELETE_TRACK    : in out DELETE_TRACK_TUPLE ) is

    FILE_NAME    : FILENAME;
    CURRENT      : TRACK_TUPLE;
    PREVIOUS     : TRACK_TUPLE;
    NEW_RECORD   : TRACK_TUPLE;
    TRACK_HEAD   : TRACK_TUPLE;
```

```
FILE        : FILE_TYPE;

package INT_IO is new INTEGER_IO( NATURAL );
  use INT_IO;
package FLT_IO is new FLOAT_IO( FLOAT );
  use FLT_IO;
package TRACK_CLASS_IO is new ENUMERATION_IO(TRACK_CLASS_TYPE);
    use TRACK_CLASS_IO;
package IFF_CLASS_IO is new ENUMERATION_IO( IFF_CLASS_TYPE );
    use IFF_CLASS_IO;
package ARCHIVE_CLASS_IO is new ENUMERATION_IO( ARCHIVE_CLASS );
    use ARCHIVE_CLASS_IO;

  function IS_OLD(
    THE_TIME     : DECLARATIONS.TIME;
    TIMEOUT      : ARCHIVE_TIMEOUT;
    TRACK_CLASS : TRACK_CLASS_TYPE ) return BOOLEAN is
    OBS_TIME   : NATURAL;
    SECONDS    : NATURAL;
  begin
    OBS_TIME := 3600*THE_TIME.HOURS + 60*THE_TIME.MINUTES +
                THE_TIME.SECONDS;
    SECONDS  := NATURAL( FLOAT( CALENDAR.SECONDS
                                          ( CALENDAR.CLOCK ) ) );
    if SECONDS - OBS_TIME >= TIMEOUT( TRACK_CLASS ) then
      return( true );
    else
      return( false );
    end if;
  end IS_OLD;
```

```
   function IS_FAR(
     RANGE_1      : FLOAT;
     TRACK_CLASS  : TRACK_CLASS_TYPE;
     RANGE_2      : MONITOR_RANGE     )  return BOOLEAN is
 begin
   if ( RANGE_1 - RANGE_2(TRACK_CLASS) > 0.0001 ) then
      return( true );
   else
      return( false );
   end if;
end IS_FAR;

 function GET_FILE_NAME return STRING is
   CURRENT_TIME : CALENDAR.TIME;
   YEAR         : CALENDAR.YEAR_NUMBER;
   MONTH        : CALENDAR.MONTH_NUMBER;
   DAY          : CALENDAR.DAY_NUMBER;
   HOURS        : NATURAL;
   MINUTES      : NATURAL;
   SECONDS      : CALENDAR.DAY_DURATION;
   FILE_NAME    : FILENAME;

   function NATURAL_TO_STRING_2( NUMBER : INTEGER )
                                                  return STRING is
     N : NATURAL      := NUMBER;
     S : STRING(1..2) := "00";
   begin
     if N > 9 then
       S(1) := CHARACTER'VAL( N/10 + 48 );
       N := N - ( (N/10) * 10 );
     end if;
```

```
        S(2) := CHARACTER'VAL( N + 48 );
        return( S );
      end NATURAL_TO_STRING_2;

    begin
      CURRENT_TIME := CALENDAR.CLOCK;
      CALENDAR.SPLIT( CURRENT_TIME, YEAR, MONTH, DAY, SECONDS );
      HOURS   :=  INTEGER(SECONDS) / 3600;
      MINUTES := (INTEGER(SECONDS) mod 3600) / 60;
      FILE_NAME(1..2)  := "f_";
      FILE_NAME(3..4)  := NATURAL_TO_STRING_2( INTEGER(MONTH  ) );
      FILE_NAME(5..6)  := NATURAL_TO_STRING_2( INTEGER(DAY    ) );
      FILE_NAME(7..8)  := NATURAL_TO_STRING_2( INTEGER(HOURS  ) );
      FILE_NAME(9..10) := NATURAL_TO_STRING_2( INTEGER(MINUTES) );
      return( FILE_NAME );
    end GET_FILE_NAME;

  begin
    DELETE_TRACK.ORIGIN := "MONITOR ";
    TRACK_HEAD := TRACKS;
    while TRACK_HEAD /= null loop
      if ( IS_OLD(TRACK_HEAD.OBSERVATION_TIME,
           TDD_CONSTRAINTS.TIMEOUT,TRACK_HEAD.TRACK_CLASS ) ) or
         ( IS_FAR(TRACK_HEAD.THE_RANGE,
                  TRACK_HEAD.TRACK_CLASS,
                  TDD_CONSTRAINTS.THE_RANGE) ) then
        BAD_TRACK_FOUND     := true;
        NEW_RECORD          := new TRACK_RECORD;
        NEW_RECORD.ID       := TRACK_HEAD.ID;
        NEW_RECORD.NEXT     := DELETE_TRACK.TRACK;
        DELETE_TRACK.TRACK := NEW_RECORD;
```

```
        end if;
        TRACK_HEAD := TRACK_HEAD.NEXT;
      end loop;
      FILE_NAME    := GET_FILE_NAME;
      CREATE( FILE, NAME => FILE_NAME );
      CURRENT := DELETE_TRACK.TRACK;
      while CURRENT /= null loop
        PUT( FILE, CURRENT.ID                          ); NEW_LINE(FILE);
        PUT( FILE, CURRENT.OBSERVER                    ); NEW_LINE(FILE);
        PUT( FILE, CURRENT.OBSERVATION_TIME.HOURS      ); NEW_LINE(FILE);
        PUT( FILE, CURRENT.OBSERVATION_TIME.MINUTES    ); NEW_LINE(FILE);
        PUT( FILE, CURRENT.OBSERVATION_TIME.SECONDS    ); NEW_LINE(FILE);
        PUT( FILE, CURRENT.TRACK_CLASS                 ); NEW_LINE(FILE);
        PUT( FILE, CURRENT.IFF_CLASS                   ); NEW_LINE(FILE);
        PUT( FILE, CURRENT.LATITUDE                    ); NEW_LINE(FILE);
        PUT( FILE, CURRENT.LONGITUDE                   ); NEW_LINE(FILE);
        PUT( FILE, CURRENT.ALTITUDE                    ); NEW_LINE(FILE);
        PUT( FILE, CURRENT.COURSE                      ); NEW_LINE(FILE);
        PUT( FILE, CURRENT.VELOCITY                    ); NEW_LINE(FILE);
        PUT( FILE, CURRENT.THE_RANGE                   ); NEW_LINE(FILE);
        PUT( FILE, CURRENT.ARCHIVE_FLAG                ); NEW_LINE(FILE);
        NEW_LINE(FILE);
        CURRENT := CURRENT.NEXT;
      end loop;
      CLOSE(FILE);
    end CHECK_FOR_TIMEOUT_AND_RANGE;

  end CHECK_FOR_TIMEOUT_AND_RANGE;
```

```
-- ***************************************************************
-- * MODULE NUMBER   : 3.2.2
-- * MODULE NAME     : IDENTIFY_SIMILARITIES
-- * DATE            : 17 AUGUST 1990
-- * REVISED         : 12 SEPTEMBER 1990
-- ***************************************************************

with DECLARATIONS, text_io;
 use DECLARATIONS, text_io;

package IDENTIFY_SIMILARITIES is
  procedure IDENTIFY_SIMILARITIES(
        TDD_CONSTRAINTS        : in     SET_MONITOR_CONSTRAINTS;
        TRACKS                 : in out TRACK_TUPLE;
        TDM_RESOLUTION_NOTICE  : in out RESOLUTION_NOTICE;
        DELETE_TRACK           : in out DELETE_TRACK_TUPLE );

  SIMILARITY_FOUND : BOOLEAN := false;

end IDENTIFY_SIMILARITIES;


package body IDENTIFY_SIMILARITIES is

  procedure IDENTIFY_SIMILARITIES(
        TDD_CONSTRAINTS        : in     SET_MONITOR_CONSTRAINTS;
        TRACKS                 : in out TRACK_TUPLE;
        TDM_RESOLUTION_NOTICE  : in out RESOLUTION_NOTICE;
        DELETE_TRACK           : in out DELETE_TRACK_TUPLE ) is

  CURRENT     : TRACK_TUPLE;
```

```
CURRENT_2  : TRACK_TUPLE;
NEW_RECORD : TRACK_TUPLE;
begin
  if TDD_CONSTRAINTS.MODE = OFF then
    null;
  else
    TDM_RESOLUTION_NOTICE.SIMILAR := false;
    TDM_RESOLUTION_NOTICE.TRACK := null;
    DELETE_TRACK.TRACK := null;
    CURRENT := TRACKS;
    while CURRENT /= null loop
      CURRENT_2 := CURRENT.NEXT;
      while CURRENT_2 /= null loop
        if (CURRENT_2.LATITUDE - CURRENT.LATITUDE < 0.01) and
           (CURRENT_2.LONGITUDE - CURRENT.LONGITUDE < 0.01 ) and
           (CURRENT_2.TRACK_CLASS = CURRENT.TRACK_CLASS ) then
          SIMILARITY_FOUND := true;
          NEW_RECORD := new TRACK_RECORD;
          NEW_RECORD.ID := CURRENT_2.ID;
          if TDD_CONSTRAINTS.MODE = AUTOMATIC then
            NEW_RECORD.NEXT := DELETE_TRACK.TRACK;
            DELETE_TRACK.TRACK := NEW_RECORD;
            DELETE_TRACK.ORIGIN := "TDM     ";
          else
            TDM_RESOLUTION_NOTICE.SIMILAR := true;
            NEW_RECORD.NEXT := TDM_RESOLUTION_NOTICE.TRACK;
            TDM_RESOLUTION_NOTICE.TRACK := NEW_RECORD;
          end if;
        end if;
        CURRENT_2 := CURRENT_2.NEXT;
      end loop;
```

```
        CURRENT := CURRENT.NEXT;
      end loop;
    end if;
  end IDENTIFY_SIMILARITIES;
end IDENTIFY_SIMILARITIES;


-- ***************************************************************
-- * MODULE NAME     : DECLARATIONS
-- * DATE            : 17 AUGUST 1990
-- * REVISED         : 12 OCTOBER 1990
-- ***************************************************************
with TEXT_IO, CALENDAR;
 use TEXT_IO;

package DECLARATIONS is

  type TIME is record
    HOURS        : NATURAL range 0..23;
    MINUTES      : NATURAL range 0..59;
    SECONDS      : NATURAL range 0..59;
    MILISECONDS  : NATURAL range 0..99;
  end record;
  TRACK_CAPACITY : constant NATURAL := 1024;
  type ARCHIVE_CLASS is     ( C, N, A, S );
  type TRACK_CLASS_TYPE is  ( AIR, SURFACE, SUBSURFACE );
  type IFF_CLASS_TYPE is    ( FRIENDLY, HOSTILE, NEUTRAL, UNKNOWN );
  OWNSHIP_IFF_CLASS : constant IFF_CLASS_TYPE := FRIENDLY;
  OWNSHIP_TRACK_CLASS : constant TRACK_CLASS_TYPE := SURFACE;
  type TRACK_RECORD;
  type TRACK_TUPLE is access TRACK_RECORD;
```

```
type TRACK_RECORD is record
  ID               : NATURAL range 0..TRACK_CAPACITY;
  OBSERVER         : STRING(1..8);
  OBSERVATION_TIME : TIME;
  TRACK_CLASS      : TRACK_CLASS_TYPE;
  IFF_CLASS        : IFF_CLASS_TYPE;
  LATITUDE         : FLOAT range -90.0..90.0;
  LONGITUDE        : FLOAT range -180.0..180.0;
  ALTITUDE         : FLOAT range -10000.0..99999.9;
  COURSE           : FLOAT range 0.0..360.0;
  VELOCITY         : FLOAT; --in knots
  THE_RANGE        : FLOAT range 0.0..9999.99; --in miles
  ARCHIVE_FLAG     : ARCHIVE_CLASS;
  NEXT             : TRACK_TUPLE;
end record;
type ADD_TRACK_TUPLE is record
  ORIGIN  : STRING(1..8);
  TRACK   : TRACK_TUPLE;
end record;
type UPDATE_TRACK_TUPLE is record
  ORIGIN  : STRING(1..8);
  TRACK   : TRACK_TUPLE;
end record;
type DELETE_TRACK_TUPLE is record
  ORIGIN : STRING(1..8);
  TRACK  : TRACK_TUPLE;
end record;
type REQUEST_TYPE is ( ADD, DELETE, UPDATE );
type CHANGE_DATABASE_REQUEST is record
  ORIGIN  : STRING(1..8);
  REQUEST : REQUEST_TYPE;
```

```
  TRACK    : TRACK_TUPLE;
end record;
subtype TEXT_STRING is STRING(1..3440);
subtype SENSOR_INFORMATION is STRING(1..132);
type LOCAL_TRACK_INFO is record
  ORIGIN       : STRING(1..8);
  ID           : NATURAL range 0..TRACK_CAPACITY;
  THE_TIME     : TIME;
  AZIMUTH      : FLOAT range -180.0 .. 180.0;
  ELEVATION    : FLOAT range -90.0  .. 90.0 := 0.0;
  THE_RANGE    : FLOAT := 0.0; --in miles
  VELOCITY     : FLOAT := 0.0; --in knots
  COURSE       : FLOAT range 0.0..360.0;
  IFF_CLASS    : IFF_CLASS_TYPE;
  TRACK_CLASS  : TRACK_CLASS_TYPE;
  ARCHIVE_FLAG : ARCHIVE_CLASS;
end record;
type INTELLIGENCE_REPORT is record
  ORIGIN       : STRING(1..8);
  INTELLIGENCE : STRING(1..80);
end record;
type SENSOR_RECORD is record
  INTELLIGENCE : STRING(1..80);
  CONTACT      : LOCAL_TRACK_INFO;
end record;
type OWNSHIP_NAVIGATION_INFO is record
  COURSE    : FLOAT range 0.0..360.0;
  VELOCITY  : FLOAT; --in knots;
  LATITUDE  : FLOAT range -90.0..90.0;
  LONGITUDE : FLOAT range -180.0..180.0;
  THE_TIME  : TIME;
```

```
end record;
OWN_ADDRESS : constant STRING(1..6) := "AAAAAA";
type ADDRESS_TYPE;
type ADDRESS_LINK is access ADDRESS_TYPE;
type ADDRESS_TYPE is record
  NAME : STRING(1..6);
  NEXT : ADDRESS_LINK;
end record;
type LINKS;
type LINKS_LINK is access LINKS;
type LINKS_TYPE is ( JTIDS, LINK11, LINK16, OTCIXS );
type LINKS is record
  NAME : LINKS_TYPE;
  NEXT : LINKS_LINK;
end record;
type SECURITY_CLASS is ( U, C, S, TS );
type PRECEDENCE_CLASS is ( R, P, O, Z );
type VIA_RECORD;
type VIA_RECORD_LINK is access VIA_RECORD;
type VIA_RECORD is record
  RELAY_BY : STRING(1..6);
  RELAY_TO : ADDRESS_LINK;
  NEXT     : VIA_RECORD_LINK;
end record;
type HEADER_FORMAT is record
  CLASSIFICATION : SECURITY_CLASS;
  PRECEDENCE     : PRECEDENCE_CLASS;
  ORIGIN         : STRING(1..6);
  ADDRESS        : ADDRESS_LINK;
  INFO           : ADDRESS_LINK;
  VIA_LINE       : VIA_RECORD_LINK;
```

```
  SUBJECT        : STRING(1..60);
end record;
subtype FILENAME is STRING(1..10);
subtype TEXT_FILE is FILE_TYPE;
type TEXT_RECORD is record
  NAME     : FILENAME;
  HEADER   : HEADER_FORMAT;
  LINK_ID  : LINKS_TYPE;
  RELAYED  : BOOLEAN;
  ARCHIVE  : BOOLEAN;
  IS_TRACK: BOOLEAN;
  FORMAT   : STRING(1..6);
  TEXT     : TEXT_STRING;
end record;
type TRANSMISSION_RECORD is record
  ROUTE_ADDR : ADDRESS_LINK;
  FULL       : BOOLEAN;
  TEXT       : TEXT_RECORD;
end record;
type TRANSMISSION_COMMAND is array ( LINKS_TYPE ) of
     TRANSMISSION_RECORD;
type TRANSMIT_RECORD is record
  ROUTE_ADDR : ADDRESS_LINK;
  ROUTED     : BOOLEAN;
  FULL       : BOOLEAN;
  TEXT       : TEXT_RECORD;
end record;
type TRANSMIT_COMMAND is array ( LINKS_TYPE ) of TRANSMIT_RECORD;
type MESSAGE_RECORD;
type MESSAGE_LIST is access MESSAGE_RECORD;
type MESSAGE_RECORD is record
```

```
  LINK_ID     : LINKS_TYPE;
  MAIN        : TRANSMISSION_RECORD;
  NEXT        : MESSAGE_LIST;
end record;
type TRANSLATION_RECORD is record
  ROUTE_ADDR      : ADDRESS_LINK;
  FULL            : BOOLEAN;
  TEXT            : TEXT_RECORD;
  SOURCE_FORMAT   : STRING(1..6);
  DESIRED_FORMAT  : STRING(1..6);
end record;
type TRANSLATION_COMMAND is array ( LINKS_TYPE ) of
     TRANSLATION_RECORD;
type NETWORK_SETUP is array ( LINKS_TYPE ) of ADDRESS_LINK;
type EMISSIONS_CONTROL_COMMAND is ( EMCON, UNRESTRICTED );
type PERIODIC_NETWORK_TRANSMISSION is record
  LINK_ID : LINKS;
  TEXT    : TEXT_STRING;
end record;
type ARCHIVE_SETUP is record
  ALL_SHIPS : BOOLEAN;
  OWNSHIP   : BOOLEAN;
  JTIDS     : BOOLEAN;
  LINK16    : BOOLEAN;
  LINK11    : BOOLEAN;
  OTCIXS    : BOOLEAN;
end record;
type TRACK_CLASS_ARRAY is array(TRACK_CLASS_TYPE) of BOOLEAN;
type IFF_CLASS_ARRAY is array(IFF_CLASS_TYPE) of BOOLEAN;
type DATABASE_REQUEST is record
  THE_RANGE   : FLOAT;
```

```
  TRACK_CLASS : TRACK_CLASS_ARRAY := ( true, true, true );
  IFF_CLASS   : IFF_CLASS_ARRAY   := ( true, true, true, true );
end record;
type INITIATE_TRANSMISSION_SEQUENCE is record
  LINK_ID        : LINKS_TYPE;
  HEADER         : HEADER_FORMAT;
  DESIRED_FORMAT : STRING(1..6);
  DBASE_REQUEST  : DATABASE_REQUEST;
end record;
subtype REPORT_DATA is STRING(1..20);
type WEAPONS_TYPE is ( CIWS, GUN, TWS, MK48 );
type WEAPON_STATUS_TYPE is(
     DAMAGED, RELOADING, LAUNCHING,   READY,    SERVICE_REQUIRED,
     SLEWING, SECURED,   MAINTANENCE, ENGAGING,
     OUT_OF_AMMUNITION );
type WEAPON_STATUS is record
  SYS_TYPE : WEAPONS_TYPE;
  STATUS   : WEAPON_STATUS_TYPE;
end record;
type WEAPON_STATUS_REPORT is array ( WEAPONS_TYPE ) of
     WEAPON_STATUS_TYPE;
  OWNSHIP_CLASS : constant TRACK_CLASS_TYPE := SURFACE;
type DESIRED_CLASS_ARRAY is array ( TRACK_CLASS_TYPE ) of BOOLEAN;
type DESIRED_RANGE_ARRAY is array ( TRACK_CLASS_TYPE ) of FLOAT;
type SET_TRACK_FILTER is record
  MAX_NUMBER   : NATURAL range 0..TRACK_CAPACITY := TRACK_CAPACITY;
  DESIRED_CLASS : DESIRED_CLASS_ARRAY := ( true, true, true );
  DESIRED_RANGE : DESIRED_RANGE_ARRAY;
end record;
type ARCHIVE_TIMEOUT is array ( TRACK_CLASS_TYPE ) of NATURAL;
type MONITOR_RANGE is array ( TRACK_CLASS_TYPE ) of FLOAT;
```

```
  type MONITOR_MODE is ( AUTOMATIC, ADVISE, OFF );
  type SET_MONITOR_CONSTRAINTS is record
    TIMEOUT    : ARCHIVE_TIMEOUT;
    THE_RANGE  : MONITOR_RANGE;
    MODE       : MONITOR_MODE;
  end record;
  type RESOLUTION_NOTICE is record
    SIMILAR : BOOLEAN;
    TRACK   : TRACK_TUPLE;
  end record;
end DECLARATIONS;
```

LIST OF REFERENCES

1. Berzins, V. and Luqi, *Software with Abstractions*, Addison-Wesley, 1991.

2. Yourdon, E., *Modern Structured Analysis*, Yourdon Press, 1989

3. Luqi and Berzins, V., "Rapidly Prototyping Real-Time Systems", *IEEE Transactions on Software Engineering*, September 1988.

4. Luqi, "Software Evolution Through Rapid Prototyping", *Computer*, v.22, no.5, pp. 13-25, May 1989.

5. White, L., *The Development of a Rapid Prototyping Environment*, M.S. Thesis, Naval Postgraduate School, Monterey, California, December 1990.

6. Luqi and Berzins, V., and Yeh, R., "A Prototyping Language for Real Time Software", *IEEE Transactions on Software Engineering*, October 1988.

7. Anderson, E. S., *Functional Specification For a Generic C3I Station*, M.S. Thesis, Naval Postgraduate School, Monterey, California, September 1990.

8. *Transportable Applications Environment (TAE) Plus*, National Aeronautics and Space Administration, Goddard Space Flight Center, January 1990.

9. Kilic, M., *Static Schedulers for Embedded Real_Time Systems*, M.S. Thesis, Naval Postgraduate School, Monterey, California, December 1989.

10. Mok, A., "A Graph Based Computational Model for Real-Time Systems", *Proceedings of the IEEE International Conference on Parallel Processing*, Pennsylvania State University, 1985.

BIBLIOGRAPHY

Altizer, C., *Implementation of a Language Translator for the Computer Aided Prototyping System*, M.S. Thesis, Naval Postgraduate School, Monterey, California, December 1988.

Beam, W. R., *Command, Control, and Communications Engineering*, McGraw-Hill, 1989.

Booch, G., *Software Engineering with Ada*, 2d ed., Benjamin-Cummings, 1987.

Cohen, N., *Ada as a Second Language*, McGraw-Hill, 1986.

Cummings, M. A., *The Development of User Interface Tool for the Computer-Aided Prototyping System*, M.S. Thesis, Naval Postgraduate School, Monterey, California, December 1990.

Gonzales, D. W., *Ada Programmer's Handbook and Language Reference Manual*, Benjamin-Cummings, 1991.

Luqi, "Handling Timing Constraints in Rapid Prototyping", *Proceedings of the 22nd Annual Hawaii International Conference on System Sciences*, IEEE Computer Society, pp.4-17-424, January 1989.

Luqi, *Rapid Prototyping for Large Software System Design*, Ph.D. Dissertation, University of Minnesota, Minneapolis, Minnesota, May 1986.

Luqi, and Ketabchi, M., "A Computer-Aided Prototyping System", *IEEE Transactions on Software Engineering*, October 1988.

Naval Research Advisory Committee, "Next Generation Computer Resources", Committee Report, February 1989.

Ng, P. A. and Yeh, R. T., *Modern Software Engineering Foundations and Current Perspectives*, Van Nostrand Reinhold, 1990.

Marlowe, L., *A Scheduler for Critical Time Constraints*, M.S. Thesis, Naval Postgraduate School, Monterey, California, December 1988.

Massachusetts Institute of Technology, *X Window System Manual*, 1988.

Orr, George E., *Combat Operations C3I: Fundamentals and Interactions*, Air University Press, Maxwell Air Force Base, Alabama, 1983.

Palazzo, F. V., *Integration of the Execution Support System for the Computer-Aided Prototyping System (CAPS)*, M.S. Thesis, Naval Postgraduate School, Monterey, California, September 1990.

PDW 120-S-00533(Rev.B, Change 4), "Over-the-Horizon Targeting (OTH-T) Gold Reporting Format", Naval Tactical Interoperability Support Activity, 30 June 1989.

Schneiderman, B., *Designing the User Interface: Strategies for Effective Human-Computer Interaction*, Addison-Wesley, 1987.

Skansholm, J., *Ada from the Beginning*, Addison-Wesley, 1988.

Stankovic, J. A. and Ramamritham, K., *Hard-Real Time Systems Tutorial*, Computer Society Press, 1988.

Sun Microsystems, *Sun Documentation Manual*, 1989.

Tanik, M. and Yeh, R.T., "Rapid Prototyping in Software Development", *Computer*, v. 22, n. 5, pp. 9-10, May 1989.

United States Department, *Reference Manual for the Ada Programming Language*, 1983.

INITIAL DISTRIBUTION LIST

Defense Technical Information Center Cameron Station Alexandria, VA 22304-6145	2
Dudley Knox Library Code 52 Naval Postgraduate School Monterey, CA 93943	2
Office of the Chief of Naval Operations (OP-094) Attn: VADM Tuttle Department of the Navy Washington, DC 20350-2000	1
Office of the Chief of Naval Operations (OP-094H) Attn: Dr. John Davis Department of the Navy Washington, DC 20350-2000	1
Office of the Chief of Naval Operations (OP-940) Department of the Navy Washington, DC 20350-2000	1
Office of the Chief of Naval Operations (OP-940C) Department of the Navy Washington, DC 20350-2000	1
Commander, Space and Naval Warfare Systems Command Attn: Captain John Gauss, PMW-162 Washington, DC 20350-2000	1
Office of the Chief of Naval Operations (OP-942) Department of the Navy Washington, DC 20350-2000	1
Office of the Chief of Naval Operations (OP-942E) Department of the Navy Washington, DC 20350-2000	1

Office of the Chief of Naval Operations (OP-942F) 1
Department of the Navy
Washington, DC 20350-2000

Office of the Chief of Naval Operations (OP-942G) 1
Department of the Navy
Washington, DC 20350-2000

Prof. Luqi, Code CS/Lq 35
Naval Postgraduate School
Monterey, CA 93943

CDR Gary Hughes, Code CS/Hu 1
Naval Postgraduate School
Monterey, CA 93943

Chief of Naval Research 1
Attn: RADM W. Miller
800 North Quincy Street
Arlingtion, VA 22217-5000

Commander, Naval Sea Systems Command 1
Attn: LCDR Scott Kelly
Code 06D3131
Washington, DC 20362-5101

Commander, Naval Sea Systems Command 1
Attn: Bill Wilder
PMS-412
Washington, DC 20362-5101

NOSC, Code 805 1
Attn: Mr. Jack Stawiski
San Diego, CA 92152-5000

NOSC, Code 40 1
Director of Computer Technology
Attn: Dr. Waslowski
San Diego, CA 92152-5000

Naval Surface Warfare Center 1
Attn: Philip Q. Hwang (U33)
Silver Spring, MD 20903-5000

Director of Research Administration 1
Code 012
Naval Postgraduate School
Monterey, CA 93943

Chairman, Code CS 1
Computer Science Department
Naval Postgraduate School
Monterey, CA 93943

Prof. Carl R. Jones 1
C3 Academic Group, Code CC
Naval Postgraduate School
Monterey, CA 93943

Prof. Michael G. Sovereign, Code OR/Sm 1
Naval Postgraduate School
Monterey, CA 93943

Office of Naval Technology 1
Attn: CDR Jane Van Fossen, Code 227
800 North Quincy Street
Arlingtion, VA 22217-5000

Office of Naval Research 1
Attn: Dr. Van Tilborg
Computer Science Division, Code 1133
800 North Quincy Street
Arlingtion, VA 22217-5000

Office of Naval Research 1
Attn: Dr. R. Wachter
Computer Science Division, Code 1133
800 North Quincy Street
Arlingtion, VA 22217-5000

National Science Foundation 1
Attn: Dr. K. C. Tai
Division of Computer and Computation Research
Washington, DC 20550

Ada Joint Project Office 1
OUSDRE (D&AT)
Attn: Dr. John Soloman
The Pentagon
Washington, DC 20301

Assistant Secretary of the Navy(RD&A) 1
Attn: Mr. G. Cann
Washington, DC 20310

Deputy Assistant Secretary of the Navy (C3I, EW & S) 1
Attn: Dr. E. Whitman
Washington, DC 20310

Defence Advanced Research Projects Agency (DARPA) 1
Integrated Strategic Technology Office (ISTO)
Attn: Dr. B. Boehm
1400 Wilson Boulevard
Arlingtion, VA 22209-2308

Defence Advanced Research Projects Agency (DARPA) 1
Integrated Strategic Technology Office (ISTO)
Attn: Dr. E. Mettala
1400 Wilson Boulevard
Arlingtion, VA 22209-2308

Department of the Air Force 1
Director of Computer Sciences
Attn: Dr. Charles Holland
Bolling AFB, DC 20332-6448

U.S. Army Research Office Elec. Div. 1
Attn: Dr. David Hislop
4300 S. Miami Boulevard
Research Triangle Park, NC 27709-2211

Dr. Raymond Yeh 1
ISSI, Echelon 4, Suite 250
9420 Research Boulevard
Austin, TX 78759

LTJG Cengiz Kesoglu 4
Nuri Pamir Cad. Duz Sok. 51/19
Kecioren, ANKARA TURKEY

LTJG Vedat Coskun 4
Visnezade Mah. Catlak Cesme Sok. 13/6
Besiktas, 80680 ISTANBUL TURKEY

Kara Harp Okulu Kutuphanesi 1
Kara Harp Okulu Komutanligi
Bakanliklar, ANKARA TURKEY

Deniz Harp Okulu Kutuphanesi 1
Deniz Harp Okulu Komutanligi
Tuzla, ISTANBUL TURKEY

Hava Harp Okulu Kutuphanesi 1
Hava Harp Okulu Komutanligi
Yesilyurt, ISTANBUL TURKEY