DTIC FILE COPY

AD-A229 270

Research Product 90-26

Tactical Planning Workstation Software Description

DTIC
ELECTE
NOV 14 1990
S E D

September 1990

Field Unit at Fort Leavenworth, Kansas
Systems Research Laboratory

U.S. Army Research Institute for the Behavioral and Social Sciences

Approved for public release; distribution is unlimited

90 11 13 176

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS --
2a. SECURITY CLASSIFICATION AUTHORITY --	3. DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE --	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) --	5. MONITORING ORGANIZATION REPORT NUMBER(S) ARI Research Product 90-26
6a. NAME OF PERFORMING ORGANIZATION Science Applications International Corporation 6b. OFFICE SYMBOL (If applicable) --	7a. NAME OF MONITORING ORGANIZATION U.S. Army Research Institute Field Unit at Fort Leavenworth, KS
6c. ADDRESS (City, State, and ZIP Code) 424 Delaware, Suite C3 Leavenworth, KS 66048	7b. ADDRESS (City, State, and ZIP Code) P.O. Box 3407 Fort Leavenworth, KS 66027-0347
8a. NAME OF FUNDING/SPONSORING ORGANIZATION U.S. Army Research Institute for the Behavioral and Social Sciences 8b. OFFICE SYMBOL (If applicable) PERI-S	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER 9-X5E-7825E-1
8c. ADDRESS (City, State, and ZIP Code) 5001 Eisenhower Avenue Alexandria, VA 22333-5600	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. 62785A; PROJECT NO. 790; TASK NO. 1304; WORK UNIT ACCESSION NO. C01

11. TITLE (Include Security Classification)

Tactical Planning Workstation Software Description

12. PERSONAL AUTHOR(S)

Packard, Bruce R.

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Interim	FROM 88/12 TO 90/05	1990, September	756

16. SUPPLEMENTARY NOTATION Dr. Stanley M. Halpin, Contracting Officer's Representative, see related document, Flanagan, J. P. and Fallesen, J. J., Tactical Planning Workstation Functional Description, ARI Research Product.

17. COSATI CODES		
FIELD	GROUP	SUB-GROUP

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)

Command and control	Software design	Workstation
Decision support	Software architecture	Ada
Tactical planning	Scenario development	X Windows

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

This document describes the Tactical Planning Workstation software, including a dBASE scenario development system and a dBASE data recording and analysis system. These systems are integrated into the Experimental Development Demonstration and Integration Center (EDDIC) facility located at the Army Research Institute Field Unit at Fort Leavenworth, Kansas. The Tactical Planning Workstation Functional Description provides a description of the system and definitions of terms. This software document is for programmers who require detailed knowledge of the software, data files, and command files.

The overall system architecture of the Tactical Planning Workstation is in Section 2. Section 3 describes the operations and maintenance procedures. It includes a description of the command files, dBASE operating instructions, installation procedures, and procedures to adapt the system for specific experiments. Section 4 describes the software and contains Ada Utilities, Ada programs, C utilities, and dBASE programs.

(Continued)

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	Unclassified

22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE (Include Area Code)	22c. OFFICE SYMBOL
Jon J. Fallesen	(913) 684-4933	PERI-SL

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

SECURITY CLASSIFICATION OF THIS PAGE

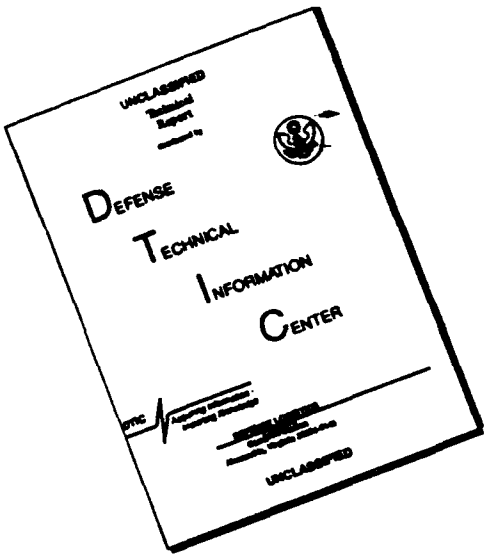
UNCLASSIFIED

ARI Research Product 90-26

19. ABSTRACT (Continued)

Appendixes A, B, and C contain the Ada package specifications for the Ada utilities, the Ada programs, and the Ada and C bindings. Appendixes D and E describe the data base formats for the EDDIC workstation data bases and for the PC-based dBASE data bases. Appendix F describes the Unix environment variables.

DISCLAIMER NOTICE



THIS DOCUMENT IS BEST QUALITY AVAILABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

Research Product 90-26

Tactical Planning Workstation Software Description

Bruce R. Packard
Science Applications International Corp.

Field Unit at Fort Leavenworth, Kansas
Stanley M. Halpin, Chief

Systems Research Laboratory
Robin L. Keesee, Director

U.S. Army Research Institute for the Behavioral and Social Sciences
5001 Eisenhower Avenue, Alexandria, Virginia 22333-5600

Office, Deputy Chief of Staff for Personnel
Department of the Army

September 1990

Army Project Number
2Q162785A790

Human Performance Effectiveness and Simulation

Approved for public release; distribution is unlimited.

U.S. ARMY RESEARCH INSTITUTE
FOR THE BEHAVIORAL AND SOCIAL SCIENCES

A Field Operating Agency Under the Jurisdiction of the Deputy Chief of Staff for Personnel

EDGAR M. JOHNSON
Technical Director

JON W. BLADES
COL, IN
Commanding

Research accomplished under contract for the Department of the Army

Science Applications International Corp.

Technical review by

Jon J. Falleson
Alfred Taylor

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

DTIC COPY INSPECTED 6

NOTICES

DISTRIBUTION: Primary distribution of this report has been made by ARI. Please address correspondence concerning distribution of reports to: U.S. Army Research Institute for the Behavioral and Social Sciences, ATTN: PERI-POX, 5001 Eisenhower Ave., Alexandria, Virginia 22333-5600.

FINAL DISPOSITION: This report may be destroyed when it is no longer needed. Please do not return it to the U.S. Army Research Institute for the Behavioral and Social Sciences.

NOTE: The findings in this report are not to be construed as an official Department of the Army position, unless so designated by other authorized documents.

FOREWORD

This document provides the software description for the Tactical Planning Workstation developed by the Fort Leavenworth Field Unit. The purpose of the workstation is to demonstrate and assess an integrated set of information and decision aids for division-level staff operations. This document is intended for software managers and programmers who are supporting requirements generation for tactical command and control systems. The workstation is being used by the Field Unit in experiments on the utility of decision support systems for staff operations. The software also has been transported to other Army laboratories for their use and demonstrations.

EDGAR M. JOHNSON
Technical Director

TACTICAL PLANNING WORKSTATION SOFTWARE DESCRIPTION

CONTENTS

	Page
INTRODUCTION	1-1
SYSTEM ARCHITECTURE	2-1
File Server	2-2
Workstation	2-4
OPERATIONS AND MAINTENANCE	3-1
Running and Maintaining the System	3-1
Installation Instructions	3-19
Customization Procedures	3-21
SOFTWARE	4-1
Ada Utilities	4-1
Ada Programs	4-52
C Utilities	4-108
dBASE Programs	4-119
APPENDIX A. ADA UTILITY SPECIFICATIONS	A-1
B. ADA PROGRAM SPECIFICATIONS	B-1
C. C BINDING SPECIFICATIONS	C-1
D. EDDIC DATA BASES	D-1
E. EDDIC dBASE DATA BASES	E-1
F. EDDIC ENVIRONMENT VARIABLES	F-1

Page

LIST OF TABLES

Table 3-1. Echelon codes 3-7

3-2. Unit type codes 3-7

3-3. Battle function codes 3-8

3-4. Unit relationship codes 3-8

3-5. Unit activity codes 3-9

3-6. Unit mission codes 3-9

3-7. Control measure types 3-11

3-8. Product description files 3-22

3-9. Tactical map menu files 3-23

3-10. Tactical situation files 3-25

3-11. Task organization tool files 3-26

4-1. Data required by ARREPORT 4-134

4-2. Data required by RPTWGAM 4-136

4-3. Data required by RPTCCOA 4-138

4-4. EDDIC application file usage 4-139

4-5. EDDIC export application file usage 4-146

4-6. EDDIC scenario application file usage 4-147

Page

Table E-1. EDDIC Sun-based data bases E-1

E-2. EDDIC PC-based data bases E-13

LIST OF FIGURES

Figure 1-1. Directory tree structure 1-1

2-1. Software distribution 2-1

2-2. File server 2-2

2-3. Workstation 2-4

4-1. Tree display options 4-25

4-2. Form field editors 4-28

4-3. Internet communications 4-32

4-4. Tactical map packages 4-36

TACTICAL PLANNING WORKSTATION SOFTWARE DESCRIPTION

1. INTRODUCTION

This document describes the Tactical Planning Workstation software, the dBASE scenario development system, and the dBASE data recording and analysis system. These systems are integrated into the Experimental Development Demonstration and Integration Center (EDDIC) facility. Reference the Tactical Planning Workstation functional description document for a description of the system and for definitions of terms. This document is for programmers who require detailed knowledge of the software, data files, and/or command files.

The overall system architecture of the Tactical Planning Workstation is in Section 2. Section 3 describes the operations and maintenance procedures. It includes a description of the command files, dBASE operating instructions, installation procedures, and procedures to adapt the system for specific experiments. Section 4 describes the software and contains Ada Utilities, Ada programs, C utilities, and dBASE programs.

Appendices A, B and C contain the Ada package specifications for the Ada utilities, the Ada programs and the Ada to C bindings. Appendices D and E describe the data base formats for the EDDIC workstation data bases and for the PC-based dBASE data bases. Appendix F describes the Unix environment variables.

Figure 1-1 shows a high level tree diagram of the Tactical Planning Workstation (EDDIC) directory structure. The Ada directory contains the Ada libraries and Ada source code. The data files are in the data directory and the gen directory contains the program executables and the C object libraries. The command files are in the shell directory and the C source code is in the source directory.

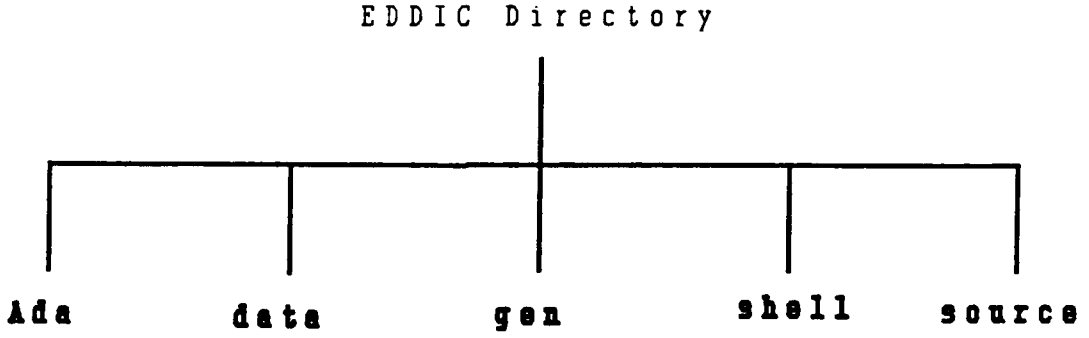


Figure 1-1. Directory Tree Structure

2. SYSTEM ARCHITECTURE

The Tactical Planning Workstation is a distributed network system consisting of a server and numerous workstations as shown in Figure 2-1. The server contains the data base managers and data routers and is a repository for Network File System (NFS) shared files. The high-resolution color workstations contain the software to control the display of and interaction with the windows. The PCs use the PC-NFS package to load scenario data into the server and to transfer data recorded during an experiment to dBASE for data organization and analysis.

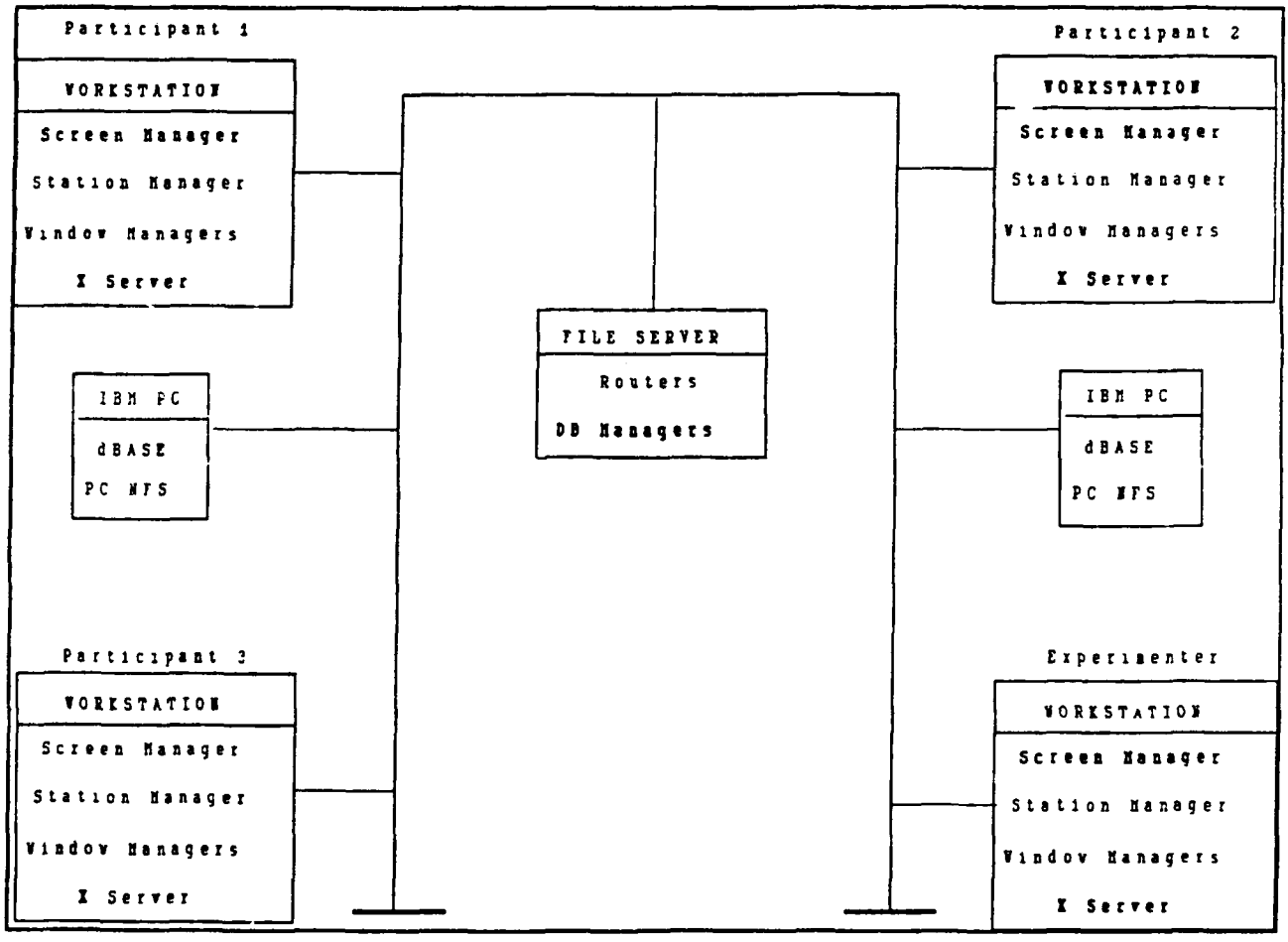


Figure 2-1. Software Distribution

2.1 FILE SERVER

Figure 2-2 shows the server processes. The routers are message routing processes that handle all interaction between the window display manager processes in the workstations and the data base managers in the server. They also perform all the data recording in the system. The following routers are part of the server:

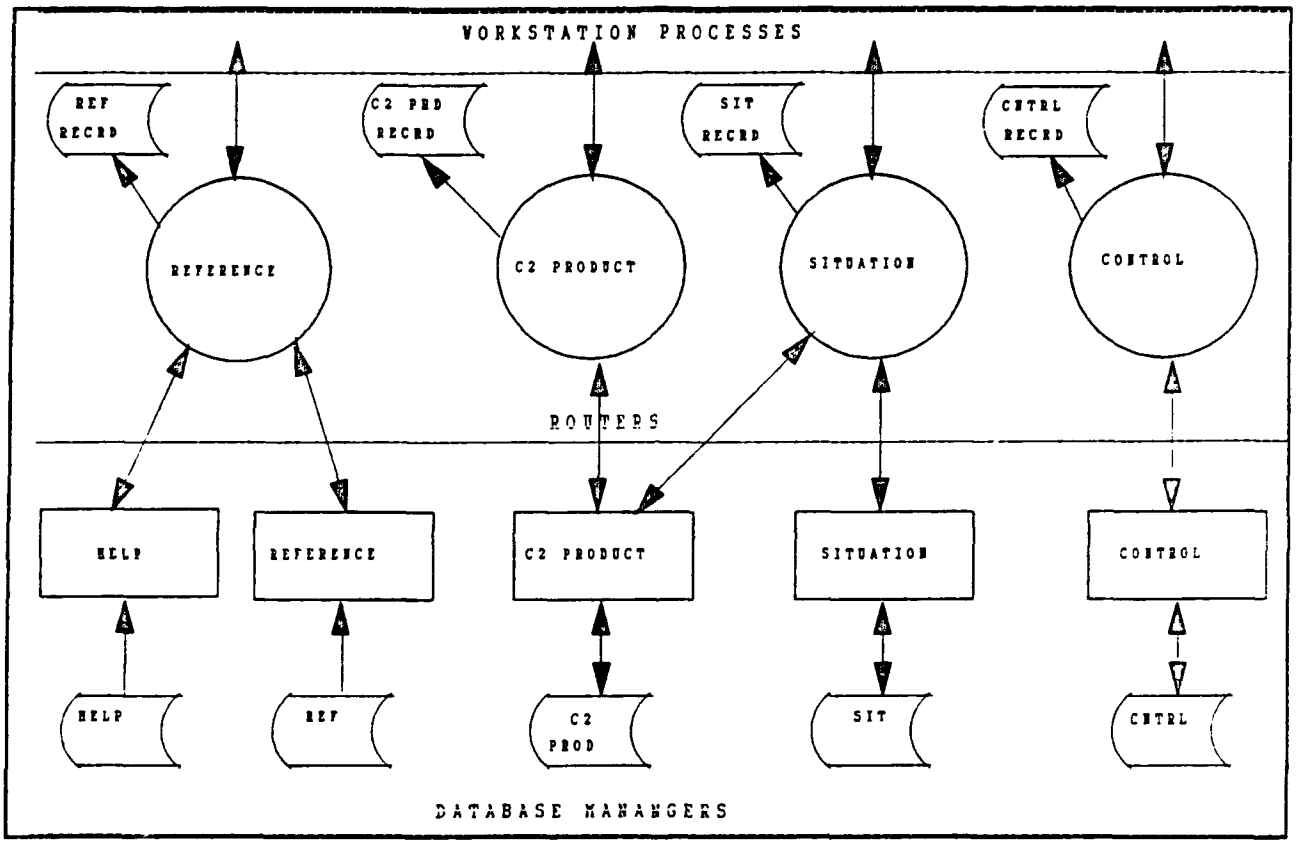


Figure 2-2. File Server

REFERENCE - Handles the message passing duties for all Reference and Help data. The View Reference Window requests the reference data through the router and the Reference Data Base Manager provides the data. The Help Button process requests the help data through the router and the Help Data Base Manager provides the data.

C2 PRODUCT - Handles the message passing duties for all Command and Control (C2) Products. The View Situation window views the products, the Build window builds and transmits the products, the View Message window receives and processes the messages, and the C2 Product Data Base Manager maintains the products.

SITUATION - Handles the message passing duties for all tactical situation data such as unit strengths and locations. The View Situation, Build, and View Messages windows require the situation data for the display of the tactical map overlays and the Task Organization Tool requires the situation data for display of the task organization. The Situation Data Base Manager maintains the situation data.

CONTROL - Handles the message passing duties for all experiment control data and color lookup table updates. The Experiment Display window generates the experiment control products and the Control Display window displays them. The Control Data Base Manager maintains the Experiment Control Products. The Map Control software passes the color lookup table updates through the router to the Station Control Manager on the workstation.

The Data Base Managers maintain the data bases and provide access to the data in the data bases through the routers. The following data base managers are part of the EDDIC server:

HELP - Maintains the help data base. The help data base contains the help messages displayed in the Help window on the workstations.

REFERENCE - Maintains the technical reference data base. The reference data base contains the products displayed in the View Reference window on the workstations.

C2 PRODUCT - Maintains the command and control data base. The C2 product data base contains textual products, computer-generated report layouts, and tactical overlay descriptions. In addition to sending products to requesting processes, this process is responsible for generating the computer-generated products using the product layout along with data from the Situation Data Base Manager. This process also maintains all messages generated in the Build window and passes the messages to the View Message window for display.

SITUATION - Maintains the situation data base. The situation data base contains the current and past tactical scenario data. The scenario data includes unit strengths, unit locations, task organization, control measures and obstacles. This process passes situation data to requesting processes through the Situation Data Router and updates the data base with data received from the Build window and the Task Organization Tool in the Tool window.

CONTROL - Maintains the experiment control data base. The experiment control data base contains the predefined control messages and any new messages defined in the Experimenter Display window during an experiment. The Control Display window displays the experiment control messages.

2.2 WORKSTATION

Figure 2-3 shows the workstation processes. It also shows the connection of the processes to the window creation buttons and the use of server routers by each process. The workstation processes consist of Station Control processes and Window Display Manager processes.

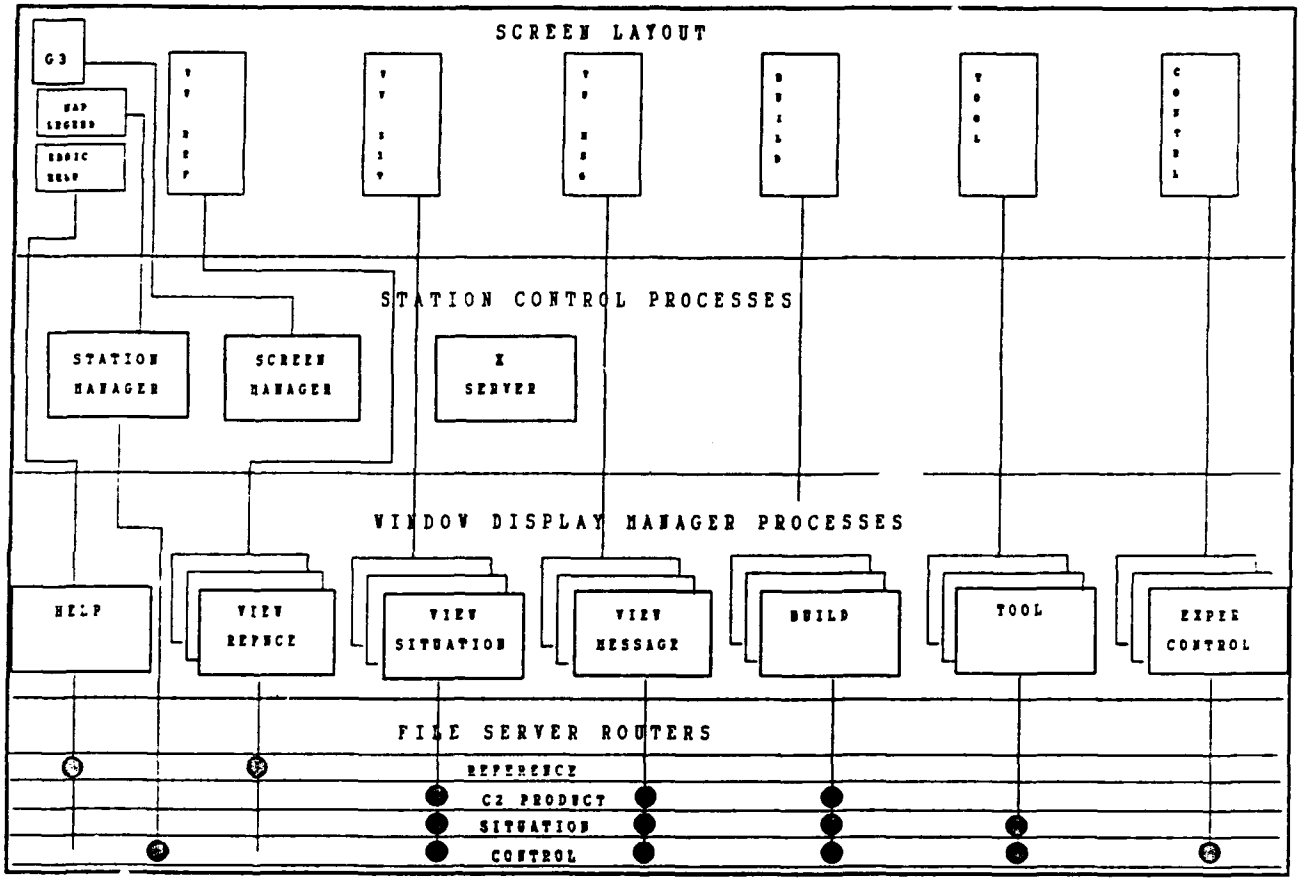


Figure 2-3. Workstation

The Station Control processes are responsible for initializing the screen, interfacing with the window creation buttons, controlling the color lookup table and the map legend. The following processes are the Station Control processes:

STATION MANAGER - Controls the color lookup table for the workstation and controls the interface for the map legend button. It also starts up the View Message Window process and the Control Window process when it receives a message and a window

of the appropriate type does not exist. The Station Manager uses the Control Router to receive color lookup table updates and to transmit color lookup table status to requesting processes. Currently, the only processes that update the lookup table are those which contain a tactical map. Only one Station Manager process exists in a workstation.

SCREEN MANAGER - The Screen Manager is the process running in the Console Icon (upper left corner of the screen). It displays the window creation buttons along the top of the screen and waits for a select (left) mouse button click on one of the buttons. When a click occurs, the Screen Manager starts up the appropriate Window Display Manager process. Only one Screen Manager exists in a workstation. The Screen Manager is the only process that is written in C.

X SERVER - The X-Window server is in complete control of the workstation screen and the Console Icon. All processes that interact with the screen are clients to the X-Window server. The Tactical Planning Workstation is currently using Version 10, Release 4 of X-Windows.

The Window Display Manager processes are responsible for the contents of the Tactical Planning Workstation specific windows. The Screen Manager starts the Window Display Manager processes, except the Help window, when the user clicks the mouse select button on one of the window creation buttons. The Help Window Manager starts during workstation initialization. The following processes are the Window Display Managers:

HELP - Controls the interaction with the Help Icon and the Help Windows. The Help Display Manager uses Ada tasking to allow the display of multiple Help Windows at the same time. A separate Ada task controls each Help Window. This process uses the Reference Router to obtain a list of the help products for display of the walking menu on the Help Icon and to request help products from the Help Data Base Manager. Only one Help Display Manager exists in a workstation.

VIEW REFERENCE - Controls the display of and interaction with the View Technical Reference Windows. The Screen Manager initiates this process when the user clicks the select mouse button on the VW REF button. It uses the Reference Router to obtain a list of the reference products and to request products from the Reference Data Base Manager. Up to seven View Reference processes can exist at one time.

VIEW SITUATION - Controls the display of and interaction with the View Situation Windows. The Screen Manager initiates this process when the user clicks the select mouse button on the VW SIT button. It uses the C2 Product Router to obtain a list of the C2 products and to request products from the C2 Product Data Base Manager. It uses the Situation Router to obtain tactical overlay data when the product is a tactical map and uses the Control Router for color lookup table updates. Up to seven View Situation processes can exist at one time.

VIEW MESSAGE - Controls the display of and interaction with the View Message Windows. The Screen Manager initiates this process when the user clicks the select mouse button on the VW MSG button or the Station Manager initiates it when it receives a message but no View Message Window is open. It uses the C2 Product Router to obtain the message log and to request products from the C2 Product Data Base Manager. It uses the Situation Router to obtain tactical overlay data when the product is a tactical map and uses the Control Router for color lookup table updates. Up to seven View Message processes can exist at one time.

BUILD - Controls the display of and interaction with the Build Windows. The Screen Manager initiates this process when the user clicks the select mouse button on the BUILD button. It uses the C2 Product Router to obtain a list of the build products, to request a product from the C2 Product Data Base Manager, and to send a message to other participants. When the product is a tactical overlay, it uses the Situation Router to obtain the tactical overlay data and to update the Situation Data Base with changes made to the tactical overlay. It uses the Control Router for color lookup table updates. Up to seven Build processes can exist at one time.

TOOL - Controls the display of and interaction with the Tool Windows. The Screen Manager initiates this process when the user clicks the select mouse button on the TOOL button. It uses the Situation router as part of the Task Organization Tool to obtain the current task organization and to send task organization updates to the Situation Data Base Manager. It sends all window control interactions to the Control Router for data recording. Up to seven Tool processes can exist at one time.

EXPERIMENT CONTROL - The Experiment Control process is one of two different processes depending upon if the workstation is a participant or experimenter workstation.

If the workstation is a participant workstation, this process controls the display of and interaction with the Control Display Windows. The Station Manager initiates this process when it receives an Experiment Control Message and no Control Window exists. It uses the Control Router to request the control message to display and to route the message back to the experimenter if the message requires an answer. Up to seven Control Display processes can exist at one time.

If the workstation is an experimenter workstation, this process controls the display of and interaction with the Experiment Display Windows. The Screen Manager initiates this process when the experimenter clicks the select mouse button on the CONTROL button. It uses the Control Router to obtain a list of Experiment Control products, to request a product from the Control Data Base Manager, and to route an Experiment Control Message to the appropriate participants. Up to seven Experiment Display processes can exist at one time.

3. OPERATIONS AND MAINTENANCE

This section describes the procedures for running and maintaining the Tactical Planning Workstation and the PC based dBASE systems, instructions for installing the workstation software on another compatible system, and procedures to customize the system for an experiment.

3.1 RUNNING AND MAINTAINING THE SYSTEM

The system consists of programs that run on the Sun system and on Personal Computers (PC). The programs that run on the Sun system are executed by command files and are described in section 3.1.1 The PC based programs are dBASE programs and are used for maintaining the scenario and performing post-experiment data analysis. The procedures for running the programs are contained in the follow sections.

3.1.1 Tactical Planning Workstation Command Files

Workstation command files provide an easy method to run the Tactical Planning Workstation system and to maintain the data bases. The command files are in the shell directory and have a suffix of ".csh".

The following command files perform different aspects of running the system:

alone.csh - Starts both the server and workstation processes in one computer. For optimum performance, the server and workstation should be different computers. Terminate the 'xterm' window to stop this command file.

demo.csh - Starts the interface on a high-resolution color workstation. The file server must be running on another computer and the environment variable 'server' must be set to the server computer name before executing this command file. Terminate the 'xterm' window to stop this command file.

record_convert.csh - Converts the data recorded during an experiment to ASCII files for import into dBASE. Stop the file server (stop_eddic.csh) before executing this command file.

server.csh - Starts the file server processes in a computer. A high-resolution screen is not necessary for the server processes. Terminate the server processes by executing the 'stop_eddic' command file.

stop_eddic.csh - Stops the file server. To insure all data bases are properly closed, always use this command file to terminate the Tactical Planning Workstation system. This command will only work on the computer where the server is running.

The following command files perform the data base maintenance functions:

build.csh - Builds all the data bases from the ASCII source files. This command file initializes the following data bases: C2 Product, Experiment Control, Help, Reference, and Situation Data. Execute this command file on the file server if possible. It takes approximately twenty minutes to execute.

help_build.csh - Builds the Help Product data base and Help walking menu description file. Because the Help data base is a read-only data base, the only time you need to execute this command file is when the help source file has changed. See section 3.3.1 for help source file update instructions.

offgerm_c2_product_build.csh - Builds the C2 Product data base and the View Situation and Build walking menu description files for the Central Germany offensive scenario. Execute this command file when the C2 Product source file is updated or when the C2 Product data base requires reinitialization. Reinitialization is required to empty the message queues or when the allocated file space for the C2 Product data base is exceeded (causes a 'CONSTRAINT ERROR' in CDB_C2_PRODUCT_DB_MANAGER). See section 3.3.1 for C2 product source file update instructions.

offgerm_control_build.csh - Builds the Experiment Control data base and the Control Product walking menu description file for the Central Germany offensive scenario. Execute this command file when the Control source file is updated or when the Control data base requires reinitialization. Reinitialization is required when the allocated file space for the Control data base is exceeded (causes a 'CONSTRAINT ERROR' in HDB_HELP_DB_MANAGER). See section 3.3.1 for Experiment control source file update instructions.

offgerm_load_higher_ech.csh - Loads the unit asset portions of the Situation Data Base for the parent units. The assets of a parent unit are equal to the sum of the assets of the children units. Execute this command file whenever the Situation Data Base is rebuilt (offgerm_situation_build.csh). Execute this command file on the file server if possible. It takes approximately ten minutes to execute.

offgerm_reference_build.csh - Builds the Reference Product data base and the Reference walking menu description file. Because the Reference data base is a read-only data base, the only time you need to execute this command file is when the reference source file has changed. See section 3.3.1 for reference source file update instructions.

offgerm_situation_build.csh - Builds the Situation Data base for the Central Germany offensive scenario. Execute this command file when any of the situation data source files are updated or when the Situation Data base requires reinitialization. Reinitialization is required to eliminate changes to the tactical overlay or task organization or when the allocated file space for the Situation Data base is exceeded (causes a 'CONSTRAINT ERROR' in SDB_SITUATION_DB_MANAGER). See section 3.3.3 for Situation Data source file update instructions.

reference_hardcopy.csh - Produces a listing file of all the products contained in the Reference data base. The Reference walking menu description file is used to define the list of reference products.

report_hardcopy.csh - Produces a listing file of all the products contained in the C2 Product data base. The View Situation walking menu description file is used to define the list of C2 products.

3.1.2 Scenario Maintainer

This section describes how to run the scenario maintainer program. It is a dBASE program that runs on an IBM PC or compatible clone.

3.1.2.1 Getting Started

A. Power up the PC. (should come up with default drive = D:)

B. Change the default directory to the scenario directory by entering:
CD D:\SCENARIO
(this MUST be D:\SCENARIO)

C. Start DBASE SCENARIO PROGRAM by entering:
DBASE SCENARIO

3.1.2.2 dBASE General Rules

The following rules apply to the interacting with the screen layouts and menus used in the scenario program.

A. Selecting Options from a Menu

Options are selected from a menu by typing in the number that is directly left of the desired option followed by a RETURN.

B. Selecting Items from a List

Items are selected from a list by using the arrow, PGUP, or PGDN keys to backlight the desired item followed by an ESC.

C. Entering Data in Screen Layouts

Data is entered in the fields by typing the input data followed by a RETURN. The RETURN will automatically advance to the next field. The arrow keys may be used to move to other fields on the screen.

D. Terminating the Entry of Data in Screen Layouts

To Write the changes to the data base enter a CTRL W.
To Quit the changes without saving enter a CTRL Q.

E. Overhitting Keys

All keyboard inputs are buffered up until the computer can process them. Because of the time required to update the data base, it may appear at times that the computer is not working. Wait for a prompt before hitting multiple RETURNs. In many situations multiple returns will cause exiting from a menu and possibly out of dBASE.

3.1.2.3 Scenario Program Operation

The program is designed to use data from either an offensive or defensive scenario. The two scenarios were developed by SAIC as part of the C2LAB prototype development. The choice of scenario must be made immediately after starting the program and can only be changed by exiting and restarting the program. The same screen provides for a selection of side, Blue or Red.

This selection will be followed by a prompt for selection of the desired day of the scenario, if the offensive or defensive scenario was chosen. Days are numbered in chronological order, 1 through 3, and then Current. The selection of the day will be followed by the program main menu.

The following options are available from the scenario main menu:

1. Base Unit Update
2. Copy One Day to Another
3. Adjust Unit Strength by %
4. Adjust Unit Strength by #

5. Adjust Personnel Strength by Loss/Gain Factors
6. Task Organize Units
7. Report Unit Strength
8. OPLAN Update
9. DAY Update
10. Control Measure Update
11. Change Scenario Day

A. Base Unit Update

This option will provide the following menu.

1. Add Base Unit
2. Change Base Unit
3. Remove a Base Unit
4. Display a Base Unit
5. Print Base Unit Data Base
0. EXIT

This option provides the capability to Add, Change, Delete, Display or Print the Base Units. The Base Units are generic units that define the authorized assets for different unit types. They are used as part of the company definition in the Task Organization Option.

B. Copy One Day to Another

This option provides the capability to copy the Task Organization and current strengths from Day 1 to Day 2, Day 2 to Day 3, or Day 3 to Current. This option should be executed only when data for the day to be copied from is complete. It will destroy any data which exists in the Copy To data bases.

C. Adjust Unit Strength by %

This option provides the capability to adjust the assets for each company assigned to a battalion, by a percent. When selected, a list of all battalion size units will appear. The desired unit is selected by positioning the cursor (backlight) and pressing ESC(ape). The officers, enlisted personnel, and total equipment can each be adjusted for the desired percentage strength.

D. Adjust Unit Strength by #

This option provides the capability to adjust the number of officers, enlisted personnel, and each equipment line item assigned to a company. When selected, a

list of all company size units will appear. The desired company is selected by positioning the cursor (backlight) and pressing ESC(ape).

E. Adjust Personnel Strength by Loss/Gain Factors

This option will provide the following menu:

1. Update Loss Rate Data for DAY X
2. Adjust Strength for DAY X
3. Print Loss Rate Data for DAY X
0. EXIT

This option provides the capability to adjust the personnel strength of each company sized unit using a loss rate factor for each category of personnel, officer and enlisted. A single gain rate factor is used for both officers and enlisted personnel. These factors must be input for each company sized unit. The loss rate data base is required for each day, except Day 1. Losses are computed by applying the loss rate factor to the unit strength of the previous day. Gains are computed by applying the gain factor to the unit shortage after the losses have been computed and subtracted. A loss and gain report is available showing the losses and gains and net for officers, enlisted, and total. The routine may be run without actually changing the strengths in the company data base, this allows trial runs to be made and factors to be adjusted prior to updating the company data base.

F. Task Organize Units

This option provides the following menu:

1. Update Company Data Base
2. Battalion Task Organize
3. Brigade Task Organize
4. Division Task Organize
5. Verify Task Organization
6. Print Task Organization
7. Print Unit Status Report
0. EXIT

This option provides the capability to define company level units in terms of the base units, task organize companies into battalions, battalions into brigades, and brigades into divisions, verify the task organization, print the task organization, and print unit status reports. Tables 3-1 through 3-6 shows the codes to use for echelon, unit type, battle function, unit relationship, activity, and mission and their corresponding enumeration value in the SDB_SITUATION_DB package.

Table 3-1. Echelon Codes

Echelon	SDB_FORCE_ECHELON
ARMGRP	ARMY_GROUP
FRONT	FRONT
ARMY	ARMY
CORPS	CORPS
DIV	DIVISION
BDE	BRIGADE
RGMT	REGIMENT
GROUP	GROUP
BN	BATTALION
SQDRN	SQUADRON
CO	COMPANY
BTRY	BATTERY
TROOP	TROOP
PLTN	PLATOON
SECT	SECTION
SQUAD	SQUAD
TEAM	TEAM

Table 3-2. Unit Type Codes

Type	SDB_UNIT_TYPE
AIRBRN	AIRBORNE
AIRASL	AIR_ASSAULT
AIRDEF	AIR_DEFENSE
AIRDFM	AIR_DEFENSE_MISSLE
ANTARM	ANTI_ARMOR
ARMCAV	ARMOR_CAV
ARMTNK	ARMOR_TANK
ARTYTW	ARTY_TOWED
ARTYSP	ARTY_SP
ATKHEL	ATTACK_HELICOPTER
AVATON	AVIATION
AVATFW	AVIATION_FW
AVATRW	AVIATION_RW
BAND	BAND
CAVRCN	CAV_RECON
CHEM	CHEMICAL
CIVAFR	CIVIL_AFFAIRS
CAA	COMBINED_ARMS_ARMY
ENGR	ENGINEER
FNANCE	FINANCE

Table 3-2. Unit Type Codes (Continued)

Type	SDB_UNIT_TYPE
INFMCH	INF_MECHANIZED
INFMTR	INF_MOTORIZED
MAINT	MAINTENANCE
MEDICL	MEDICAL
MILINT	MILITARY_INTEL
MILPOL	MILITARY_POLICE
ORDNCE	ORDNANCE
PERSVC	PERS_SVC
PSYOPS	PSYCH_OPNS
QMASTR	QUATERMASTER
RCKART	ROCKET_ARTILLERY
SIGNAL	SIGNAL
SPFORC	SPECIAL_FORCES
SPTCOM	SPT_COM
SUPSRV	SUPPLY_SERVICES
S2SMSL	SURF_TO_SURF_MISSLE
TRANSP	TRANSPORTATION

Table 3-3. Battle Function Codes

Battle Function	SDB_BATTLE_FUNCTION
COMBAT	COMBAT_MANEUVER
CS	COMBAT_SUPPORT
CSS	COMBAT_SERVICE_SUPPORT
COMMIT	COMMITTED
REINF	REINFORCE
ARTIL	ARTILLERY

Table 3-4. Unit Relationship Codes

Relate	SDB_BLUEFOR_TO_RELATE
ORGNIC	ORGANIC_ASSIGNED
ATTACH	ATTACHED
DS	DS
GS	GS
GSR	GSR
OPCON	OPCON

Table 3-5. Unit Activity Codes

Activity	SDB_FORCE_ACTIVITY
ADVGRD	ADVANCE_GUARD
ADVANC	ADVANCING
AIRAST	AIR_ASSAULT
ARBAST	AIRBORNE_ASSAULT
ARMAST	AIRMOBILE_ASSAULT
AMPLND	AMPHIBIOUS_LANDING
CLOSNG	CLOSING
COMNCT	COMMUNICATION
CNTATK	COUNTER_ATTACK
COVFRC	COVERING_FORCE
XPOLIT	EXPLOITATION
FLNKGD	FLANK_GUARD
INFILT	INFILTRATION
MAINTN	MAINTAINING
MANAGE	MANAGING
OCCUPY	OCCUPY
PENETR	PENETRATION
PURSUT	PURSUIT
PREPAR	PREPARING
RAOPS	REAR_AREA_OPERATIONS
REARGD	REAR_GUARD
REAMFL	REARM_REFUEL
RECON	RECONNAISSANCE
REINF	REINFORCING
REORGN	REORGANIZATION
RIVCRS	RIVER_CROSSING
SEARCH	SEARCH
SCREEN	SCREEN
SERVIC	SERVICE
SUPPLY	SUPPLY
TRNSPT	TRANSPORT

Table 3-6. Unit Mission Codes

Misson	SDB_FORCE_MISSION
ATTACK	ATTACK
DEFEND	DEFEND
DELAYD	DELAYED
RESERV	RESERVE
SUPPRT	SUPPORT
WTHDR	WITHDRAW

The task organization verification goes through the task organization verifying the unit names. An error report is generated on the printer if any errors are found. (The error tells you that the name listed in the higher level organization cannot be found at the lower level.) Invalid companies must be corrected in either the battalion task organization or in the company data base. Invalid battalions must be corrected in either the brigade task organization or the battalion task organization. Invalid brigades must be corrected in either the division task organization or the brigade task organization.

G. Report Unit Strength

This option provides the following menu:

1. Company Strength
2. Battalion Strength
3. Brigade Strength
4. Division Strength
5. Print Percent Strength, Units of a Brigade
6. Page or Line Feed
0. EXIT

This option provides the capability to get reports of actual unit strength for each organization from division (all brigades in the brigade data base) to company level. The report provides the authorized, on-hand, and the percentage on-hand for officers, enlisted, and each line item of equipment. An option is also provided to print out the percentage strength, officer, enlisted, and total equipment, for each unit with recaps for each battalion of a brigade.

H. OPLAN Update

This option provides the following menu:

1. Add OPLAN
2. Change OPLAN
3. Delete OPLAN
4. OPLAN Report
0. Exit

This option provides the capability to Add, Change, Delete, Display or Print the Operation Plans (OPLAN). The OPLAN system provides a convenient way to segregate planning situation data. There should always be a current situation OPLAN along with a working OPLAN for each participant.

I. DAY Update

This option provides the following menu:

1. Add DAY
2. Change DAY
3. Delete DAY
4. DAY Report
0. Exit

This option provides the capability to Add, Change, Delete, Display, or Print the Days contained in the scenario. Each day has a list of data bases that contain the scenario data for that day. If data do not change from one day to the next, the same file name can be used for both days.

J. Control Measure Update

This option provides the following menu:

1. Add Control Measure
2. Change Control Measure
3. Delete Control Measure
4. Control Measure Report
0. Exit

This option provides the capability to Add, Change, Delete, Display or Print the control measures contained in the scenario. See Table 3-7 for the control measure type codes.

Table 3-7. Control Measure Types

CODE	TYPE
AROPS	AREA OF OPERATIONS
ASMAR	ASSEMBLY AREA
ATKPOS	ATTACK POSITION
BTLPOS	BATTLE POSITION
BGSPAR	BRIGADE SUPPORT AREA
BNSPAR	BATTALION SUPPORT AREA
DVSPAR	DIVISION SUPPORT AREA
DROPZN	DROP ZONE
FREFIR	FREE FIRE AREA

Table 3-7. Control Measure Types (Continued)

CODE	TYPE
LANDZN	LANDING ZONE
NFIRAR	NO FIRE AREA
OBJCTV	OBJECTIVE
RSFRAR	RESTRICTIVE FIRE AREA
ZONACT	ZONE OF ACTION
ASLTCS	ASSAULT CROSSING
RFTSIT	RAFT SITE
GRPTGT	GROUP OF TARGETS
BONDRY	BOUNDARY
BRDGLN	BRIDGEHEAD LINE
CRDFLN	COORDINATED FIRE LINE
FEBA	FORWARD EDGE OF BATTLE AREA
FSPCDL	FIRE SUPPORT COORDINATION LINE
FLOT	FOWARD LINE OF TROOPS
HOLDLN	HOLDING LINE
LITELN	LIGHT LINE
LMTADV	LIMIT OF ADANCE
LNCNCT	LINE OF CONTACT
LNDEPT	LINE OF DEPARTURE
PHASLN	PHASE LINE
COALN	COURSE OF ACTION LINE
RSTFLN	RESTRICTIVE FIRE LINE
AIRFLD	AIRFIELD
BRIDGE	BRIDGE
BLDING	BUILDING
CITY	CITY
LAKE	LAKE
MPRFPT	MAP REFERENCE POINT
MTNPK	MOUNTAIN PEAK/HILL TOP
RDXING	ROAD INTERSECTION
TOWN	TOWN
VILLAG	VILLAGE
CHKPNT	CHECKPOINT
CLCTPT	COLLECTION POINT
CNCTPT	CONTACT POINT
CORDPT	COORDINATING POINT
CRTEVT	CRITICAL EVENT
LKUPPT	LINK UP POINT
PASGPT	PASSAGE POINT

Table 3-7. Control Measure Types (Continued)

CODE	TYPE
PTDEPT	POINT OF DEPARTURE
RELSPT	RELEASE POINT
STRTPT	START POINT
STNGPT	STRONG POINT
TRFCTL	TRAFFIC CONTROL POINT
AAXADV	AIR AXIS OF ADVANCE
ARCORR	AIR CORRIDOR
GAXAVM	GROUND AXIS OF ADVANCE MAIN ATTACK
GAXAVS	GROUND AXIS OF ADVANCE SUPPORT
DIRATK	DIRECTION OF ATTACK
FEINT	FEINT
MNSPRT	MAIN SUPPLY ROUTE
ROUTE	ROUTE

J. Change Scenario Day

The default day is Day 1, this must be verified or changed at startup of the system. This option provides the capability to change that selection (after system startup) to Day 1, Day 2, Day 3, or the Current day of the scenario. Make sure the day exists before selecting it. Days 2 through Current are created by the Copy One Day to Another option. No loss rates are required or used for Day 1, and strengths cannot be adjusted by these factors for Day 1.

3.1.2.4 Ending the Scenario Session

Select EXIT on the top level scenario menu. This will exit dBASE and set the default drive to D:.

3.1.3 Exporting the Scenario to the Sun System

This section describes the procedures to transfer the dBASE Scenario data to the Sun system. The following scenario data CAN be exported to the Sun system:

- BLUEFOR Equipment
- BLUEFOR Personnel
- BLUEFOR Fuel

- BLUEFOR Ammunition
- BLUEFOR Task Organization
- BLUEFOR Unit Locations
- BLUEFOR Unit Status
- BLUEFOR Asset Unit List
- OPFOR Equipment
- OPFOR Asset Unit List
- Control Measures

The following scenario data currently CANNOT be exported and must be updated manually on the Sun system:

- OPFOR Task Organization
- OPFOR Unit Locations
- OPFOR Unit Status
- Obstacles

Perform the follow steps to export the scenario data:

1. Copy the scenario data bases and index files into the Export directory (D:\SCENARIO\EXPORT).

2. Set the default directory to the Export Directory.

3. Start dBASE IV by entering:
   dBASE

4. Start the EXPORT by selecting the EDDIC_EX Application.

5. Choose the UPDATE DAY option and make updates so the DATE/TIMEs and File Suffixes are correct for the scenario data that is to be exported.

6. The displayed menu system can be used to create the export files; however, it will take forever and I suggest exiting back to the Control Center and execute individual programs to create the desired files. The following lists the scenario data and the corresponding dBASE program:

Scenario Data	Program
All BLUEFOR Items	EX_BALL
BLUEFOR Equipment	EX_BEQP
BLUEFOR Personnel	EX_BPRS
BLUEFOR Fuel	EX_BFUL
BLUEFOR Ammunition	EX_BAMM
BLUEFOR Task Organization	EX_BTSK

BLUEFOR Unit Locations	EX_BLOC
BLUEFOR Unit Status	EX_BSTAT
BLUEFOR Asset Units	EX_BUNIT
All OPFOR Items	EX_RALL
OPFOR Equipment	EX_REQP
OPFOR Asset Units	EX_RUNIT
Control Measures	EX_CM

7. Each of the above programs create an ASCII file that must be transferred to the Sun system. The following table shows the Data type, MS-DOS file name, and the Sun system file name.

Data Type	MS-DOS File	Sun File
BLUEFOR Equipment	BEQUIP.LIS	beqload.dat
BLUEFOR Personnel	BPERS.LIS	bprload.dat
BLUEFOR Fuel	BFUEL.LIS	bfuel.dat
BLUEFOR Ammunition	BAMMO.LIS	bamload.dat
BLUEFOR Task Org	BTSKHST.LIS	btskhst.dat
BLUEFOR Unit Locations	BUNLOC.LIS	bunloc.dat
BLUEFOR Unit Status	BUNSTAT.LIS	bunstat.dat
BLUEFOR Asset Units	BUNIT.LIS	bunit.dat
OPFOR Equipment	REQUIP.LIS	reqload.dat
OPFOR Asset Units	RUNIT.LIS	runit.dat
Control Measures	CNTLMSR.LIS	cntrlmsr.dat

3.1.4 Data Analysis

The data analysis program was developed using the dBASE IV application developer and therefore must be executed in dBASE IV. To start the program, perform the following steps:

1. CD \TASK3\dBASE
2. dBASE/T EDDIC

The data analysis program starts by presenting a welcome screen and then displays the following top level menu:

The options in the menus can be selected either by typing the first letter of the option or by moving the cursor to the desired option and hitting the ENTER key. Each option in the top-level menu (except the Exit key), displays a pull-down menu with subsequent options. The following section gives a brief description of the options available within each option on the main menu.

A. ADD

The following types of data can be added to the data analysis data base:

- Automated data from Sun and Symbolics
- COA analysis data:
  - Critical Event Identification
  - War Gaming Summary
  - COA Comparison Measure Weights
  - COA Comparison Measure Scales
- Experiment Questionnaires:
  - COA Task Evaluation
  - Personal Demographics
  - Human Machine Interface
  - Human Machine Interface (with COAAT)
  - Personal Style
  - Situation Awareness
  - Workload Assessment
- Experimenter Observations:
  - Team Profile
  - Experiment Time Line
- Experiment Scoring:
  - Gathering Pertinent Facts
  - Arraying Forces
  - Identifying Critical Events
  - COA Justification
  - Concept of Operations

B. CHANGE

The following types of data can be changed in the data analysis data base:

- COA analysis data:
  - Critical Event Identification
  - War Gaming Summary
  - COA Comparison Measure Weights
  - COA Comparison Measure Scales
- Experiment Questionnaires:

COA Task Evaluation
Personal Demographics
Human Machine Interface
Human Machine Interface (with COAAT)
Personal Style
Situation Awareness
Workload Assessment

- Experimenter Observations:
  - Team Profile
  - Experiment Time Line
- Experiment Scoring:
  - Gathering Pertinent Facts
  - Arraying Forces
  - Identifying Critical Events
  - COA Justification
  - Concept of Operations

C. DELETE

The following types of data can be deleted from the data analysis data base:

- Automated data from Sun and Symbolics
- COA analysis data:
  - Critical Event Identification
  - War Gaming Summary
  - COA Comparison Measure Weights
  - COA Comparison Measure Scales
- Experiment Questionnaires:
  - COA Task Evaluation
  - Personal Demographics
  - Human Machine Interface
  - Human Machine Interface (with COAAT)
  - Personal Style
  - Situation Awareness
  - Workload Assessment
- Experimenter Observations:
  - Team Profile
  - Experiment Time Line
- Experiment Scoring:
  - Gathering Pertinent Facts
  - Arraying Forces
  - Identifying Critical Events
  - COA Justification
  - Concept of Operations

D. REPORT

The following reports can be printed from the data analysis data base:

- Automated Data from the Sun System
  - View Situation Requests
  - View Reference Requests
  - Map Control Interaction
  - Workstation Window Operations
  - New Control Measures
  - BLUEFOR Task Organization Updates
  - Unit Location Updates
- COA Analysis
  - Critical Event Identification
  - War Gaming Summary
  - COA Comparison Objective Measures
  - COA Comparison Subjective Measures
- Questionnaires:
  - COA Task Evaluation
  - Personal Demographics
  - Human Machine Interface
  - Human Machine Interface (with COAAT)
  - Personal Style
  - Situation Awareness
  - Workload Assessment
- Experimenter Observations:
  - Team Profile
  - Experiment Time Line
- Experiment Scoring:
  - Gathering Pertinent Facts
  - Arraying Forces
  - Identifying Critical Events
  - War Gaming
  - COA Comparison
  - COA Justification
  - Concept of Operations

E. SPECIAL

The following special functions are contained in the data analysis program:

Process the ASCII files received from the Sun system. This option must be completed before adding the automated data to

the data base. If this step is skipped, old recorded data will be added to the data base with the new experiment code.

Export the data bases to ASCII files for import into SAS or other statistical, analytical, or database management packages.

Remove all deleted records from the data base.

F. EXIT

This option exits the data analysis program and returns to dBASE.

3.2 INSTALLATION INSTRUCTIONS

This section describes the typical steps to install the Tactical Planning Workstation executable and data files. EDDIC should run on any Sun system with the following attributes:

- 120 MB of disk space
  (note: less disk space is required, if some of the digital map files are not installed)
- 32 MB swap partition
- Sun 3/160C compatible display
  (note: The system will run on a 3/110 display with a different version of X-Windows)
- Maxusers set to at least 12
- Ethernet enabled system

Use the following steps to load the system. Some of the steps require 'Superuser' privileges.

1. Add an EDDIC account to the passwd file.

2. Make a Directory for EDDIC:

```
su
cd /(disk path)
mkdir eddic
chown eddic "EDDIC account"
<CTRL>d
```

3. Login to the new "EDDIC account".

4. Copy System from tape:

   tar xvf /dev/rmt0

5. Set-up Links

   cd shell
   (Make path changes to links.csh)

   1. Replace /usr/cherokee/eddic_demo with disk path from step 2.
   2. Replace /usr2/demo with location of map files.
   3. Replace /usr/cherokee/Xrun with /edemo/Xrun.

   su
   links.csh
   <CTRL>d

6. If network system, change server name to the
   new servers name in the following file:

   demo.csh

7. Append services.add to /etc/services (you must be SU).

8. Load the Digital Map Tapes

   cd /emaps
   cd ..
   tar xvf /dev/rmt8
   (Repeat the above command for all 3 map tapes)

9. To start the File Server:

   Login to EDDIC
   cd shell
   server.csh

10. To start a Workstation:

    Login to EDDIC
    cd shell
    demo.csh

11. To start the Standalone System:

    Login to EDDIC
    cd shell
    alone.csh

12. To reinitialize data bases

    build.csh

3.3 CUSTOMIZATION PROCEDURES

This section describes the procedures to customize the Tactical Planning Workstation interface for a specific experiment. ASCII files contain the descriptions of the system attributes and can be modified using an editor on the Sun fileserver. The following classes of system attributes can be modified:

- Products Available in the Windows
- Tactical Map Menu Options
- Tactical Situation Data
- Task Organization Tool Options

Be sure to make backup copies of the files before making modifications to them. See Appendix D for the format of the description files.

3.3.1 Products Available in Windows

Directory "data/offgerm" contains the ASCII files describing the products available in the workstation windows. The list of product description files is shown in Table 3-8. See Section 3.1.1 for the description of the command files used to integrate the updates into the system.

Table 3-8. Product Description Files

Filename	Description
c2ref.dat	Products available in the View Reference window. See REFERENCE_SOURCE in Appendix D for the format of this file. Updates to this file are integrated into the system with the offgerm_reference_build.csh command file.
control.src	Products available in the Experimenter's Experiment Control window. See EXP_CONTROL_SOURCE in Appendix D for the format of this file. Updates to this file are integrated into the system with the offgerm_control_build.csh command file.
edhist.dat	Products available in the *View Situation and Build* windows. See C2_PRODUCT_SOURCE in Appendix D for the format of this file. Updates to this file are integrated into the system with the offgerm_c2_product_build.csh command file.
help_source.dat	Products available in the Help window. See HELP_SOURCE in Appendix D for the format of this file. Updates to this file are integrated into the system with the help_build.csh command file.
menu/tools	Tools available in the Tool window. See TOOL_MENU in Appendix D for the format of this file. No command file is required to integrate updates to this file into the system.

3.3.2 Tactical Map Menu Options

Directory "data/maps/menu" contains the ASCII files describing the map menu options available in the workstation windows containing the tactical map. The list of menu files is shown in Table 3-9. No command files are required to integrate updates to these files into the system. The updates are integrated as soon as the file has been saved by the editor.

Table 3-9. Tactical Map Menu Files

Filename	Description
blue_cm.edit	Options for BLUEFOR control measures with edit capability. See BLUEFOR_CM_EDIT_MENU in Appendix D for the format of this file.
blue_cm.view	Options for BLUEFOR control measures without edit capability. See BLUEFOR_CM_VIEW_MENU in Appendix D for the format of this file.
blue_obs.edit	Options for BLUEFOR obstacles with edit capability. See BLUEFOR_OBS_EDIT_MENU in Appendix D for the format of this file.
blue_obs.view	Options for BLUEFOR obstacles without edit capability. See BLUEFOR_OBS_VIEW_MENU in Appendix D for the format of this file.
blue_unit.edit	Options for BLUEFOR units with edit capability. See BLUEFOR_UNIT_EDIT_MENU in Appendix D for the format of this file.
blue_unit.view	Options for BLUEFOR units without edit capability. See BLUEFOR_UNIT_VIEW_MENU in Appendix D for the format of this file.
map.edit	Options for the map with edit capability. See MAP_BUILD_MENU in Appendix D for the format of this file.
map.view	Options for the map without edit capability. See MAP_VIEW_C2_MENU in Appendix D for the format of this file.
opfor_cm.edit	Options for OPFOR control measures with edit capability. See OPFOR_CM_EDIT_MENU in Appendix D for the format of this file.
opfor_cm.view	Options for OPFOR control measures without edit capability. See OPFOR_CM_VIEW_MENU in Appendix D for the format of this file.

opfor_obs.edit	Options for OPFOR obstacles with edit capability. See OPFOR_OBS_EDIT_MENU in Appendix D for the format of this file.
opfor_obs.view	Options for OPFOR obstacles without edit capability. See OPFOR_OBS_VIEW_MENU in Appendix D for the format of this file.
opfor_unit.edit	Options for OPFOR units with edit capability. See OPFOR_UNIT_EDIT_MENU in Appendix D for the format of this file.
opfor_unit.view	Options for OPFOR units without edit capability. See OPFOR_UNIT_VIEW_MENU in Appendix D for the format of this file.

3.3.3 Tactical Situation Data

Directory "data/offgerm/situation" contains the ASCII files describing the tactical situation. The Tactical Situation data are also maintained in dBASE on a PC. The normal procedure for updating the scenario is to use the dBASE scenario program (see section 3.1.2) to change and export the files to the Sun fileserver for integration. However, emergency updates can be made directly to the ASCII files on the Sun. The list of tactical situation description files is shown in Table 3-10. See offgerm_situation_build.csh in Section 3.1.1 to integrate updates into the system.

Table 3-10. Tactical Situation Files

Filename	Description
bamload.dat	Ammunition strengths for BLUEFOR units. See BLUEFOR_AMMO_SOURCE in Appendix D for the format of this file.
beqload.dat	Equipment strengths for BLUEFOR units. See BLUEFOR_EQUIP_SOURCE in Appendix D for the format of this file.
bfuel.dat	Fuel strengths for BLUEFOR units. See BLUEFOR_FUEL_SOURCE in Appendix D for the format of this file.
blue_organic.dat	Organic BLUEFOR Task Organization. See BLUEFOR_ORGANIC_TASK_ORG in Appendix D for the format of this file.
bprload.dat	Personnel strengths for BLUEFOR units. See BLUEFOR_PERSONNEL_SOURCE in Appendix D for the format of this file.
btskhst.dat	BLUEFOR task organization. See BLUEFOR_TASK_ORG_SOURCE in Appendix D for the format of this file.
bunloc.dat	Unit locations for BLUEFOR units. See BLUEFOR_UNIT_LOC_SOURCE in Appendix D for the format of this file.
cntrlmsr.dat	Control measures. See CONTROL_MEASURE_SOURCE in Appendix D for the format of this file.
c2obst.dat	Obstacles. See OBSTACLE_SOURCE in Appendix D for the format of this file.
opplan.asc	Predefined Operational Plans. See OPLAN_LIST_SOURCE in Appendix D for the format of this file.

reqload.dat	Equipment strengths for OPFOR units. See OPFOR_EQUIP_SOURCE in Appendix D for the format of this file.
rreinf.dat	Reinforcing Times for OPFOR units. See OPFOR_REINFORCE_TIME in Appendix D for the format of this file.
rtskhst.dat	OPFOR Task Organization. See OPFOR_TASK_ORG_SOURCE in Appendix D for the format of this file.
runitptch.dat	Unit status for OPFOR units. See OPFOR_UNIT_STATUS_SOURCE in Appendix D for the format of this file.
runloc.dat	Unit locations for OPFOR units. See OPFOR_UNIT_LOC_SOURCE in Appendix D for the format of this file.

3.3.4 Task Organization Tool Options

Directory "data" contains the ASCII files describing the menu options available for the Task Organization Tool in the workstation Tool window. The list of menu description files is show in Table 3-11. No command files are required to integrate updates to these files into the system. The updates are integrated as soon as the file has been saved by the editor.

Table 3-11. Task Organization Tool Files

Filename	Description
tot_options	Task Organization Tool options. See TASK_ORG_TOOL_MENU in Appendix D for the format of this file.
tou_options	Options for units in the Task Organization Tool. See TASK_ORG_TOP_UNIT_MENU in Appendix D for the format of this file.
utb_options	Options for the unit type button. See TASK_ORG_UNIT_TYPE_MENU in Appendix D for the format of this file.

4. SOFTWARE

This section describes the software architecture of the Tactical Planning Workstation. Since the Tactical Planning Workstation is part of an experimental design and development system, the software design uses programming standards that allow easy porting to other system architectures. The major standards include Ada as the programming language and X-Windows as the network windowing system. The software consists of the following modules:

Ada Utilities - Ada packages that provide commonly used capabilities to the Ada programs. All utility packages start with the letter "U".

Ada Programs - Each Ada Program is an individual process in the system. The data base managers, routers, and window managers are all examples of Ada programs. Ada programs start with a three-letter prefix that abbreviates the function of the program. The letter "R" is reserved for network message routers.

C Utilities - C procedures were written to allow access to the low-level system, communication, and X-Window capabilities from Ada. The C procedures are functionally separated into libraries and each library has an Ada binding specification for interfacing with Ada. All C utilities start with the letter "C".

dBASE Programs - The EDDIC system requires a relational data base system to easily maintain and update the scenario data and to process data recorded during an experiment. Because dBASE is readily available and it is easy to transfer data from the Sun fileserver to a PC, dBASE is the EDDIC relational data base system. dBASE programs provide a user-friendly interface for working with the scenario and experiment data.

4.1 ADA UTILITIES

The Ada utilities consist of a large pool of powerful utility procedures that provide a consistent and modular approach to software development. Each utility package has a specification ("_s.a" suffix) and a body ("_b.a" suffix). The specification is the interface between the package and the calling application and the body is the Ada code to perform the function of the procedures. To use a utility package, a programmer should use this document along with the appropriate package specification. The only access required to a package body is to update specific procedures. The following major utility categories are in the Tactical Planning Workstation.

COMMON - Ada types that are available throughout the system.

UED - EDDIC utilities such as math functions, string functions, and list and queue managers.

UFM - Form Manager utilities

UIN - Internet communications utilities

UIW - Color image window utilities

UTM - Tactical map utilities

UUX - Unix command utilities

UWN - Window display and control utilities

4.1.1 Common Ada Specifications

The common Ada specifications provide global access to system types and objects. Many of the types defined in these specifications are required for the message passing utilities (UIN). The following Ada specifications are contained in the common library:

- CDB_C2_PRODUCT_DB
- CTL_CONTROL_DB
- FDB_REFERENCE_DB
- HDB_HELP_DB
- LUT_SYSTEM
- MSG_MESSAGE
- SDB_SITUATION_DB
- SYSTEM_PACKAGE

4.1.1.1 CDB_C2_PRODUCT_DB

Abstract. Global Command and Control (C2) product Ada specification.

Major Capabilities. This package contains Ada type specifications for the C2 product data base records and for all the C2 product messages. All types in this package start with "CDB".

Special Instructions. The CDB messages must be passed through the C2 product router (CDB).

Data Bases. None

Environment Variables. None

4.1.1.2 CTL_CONTROL_DB

Abstract. Global experiment control product Ada specification.

Major Capabilities. This package contains Ada type specifications for the experiment control product data base records and for all the experiment control product messages. All types in this package start with "CTL". The CTL messages must be passed through the experiment control router (RCN).

Special Instructions. None

Data Bases. None

Environment Variables. None

4.1.1.3 FDB_REFERENCE_DB

Abstract. Global reference product Ada specification.

Major Capabilities. This package contains Ada type specifications for the reference product data base records and for all the reference product messages. All types in this package start with "FDB".

Special Instructions. The FDB messages must be passed through the reference router (RRF).

Data Bases. None

Environment Variables. None

4.1.1.4 HDB_HELP_DB

Abstract. Global help product Ada specification.

Major Capabilities. This package contains Ada type specifications for the help product data base records and for all the help product messages. All types in this package start with "HDB".

Special Instructions. The HDB messages must be passed through the reference router (RRF).

Data Bases. None

Environment Variables. None

4.1.1.5 LUT_SYSTEM

Abstract. Global color lookup table Ada specification.

Major Capabilities. This package contains Ada type specifications to define the color lookup table and messages to update and determine the status of the lookup table. It also contains objects that define specific colors in the lookup table. All types in this package start with "LUT".

Special Instructions. The majority of the types in this packages are reserved for use by the lookup table manager (LUT) in the station control manager (SCL). Normal applications will use this package to define messages to communicate with the station control manager and to use predefined colors in the lookup table. The LUT messages must be passed through the experiment control router (RCN).

4.1.1.6 SDB_SITUATION_DB

Abstract. Global situation data Ada specification.

Major Capabilities. This package contains Ada type specifications for the situation data base records and for all the situation data messages. All types in this package start with "SDB".

Special Instructions. This package is the primary source to use for all tactical situation data. The messages in this package must be routed through the situation data router (RSD).

Data Bases. None

Environment Variables. None

Data Bases. None

Environment Variables. None

4.1.1.7 MSG_MESSAGE

Abstract. Global Ada Internet communications message specification.

Major Capabilities. This package contains Ada type specifications for all the messages that are passed through the Internet communications utilities (UIN). All types in this package start with "MSG".

Special Instructions. The normal procedure for adding a message to this package is to define the message in the appropriate package specification in the common library and then using that type to define a new message variant.

This package controls which messages are recorded by each router.

Data Bases. None

Environment Variables. None

4.1.1.8 SYSTEM_PACKAGE

Abstract. Global system Ada specification.

Major Capabilities. This package contains Ada type specifications for the system. These types include data base limitations, menu limitations, window system limitations and types, color lookup table limits, system error codes, and the processes. All types in this package start with "SYS".

Special Instructions. All types and objects that are defined in other packages and programs should use the base types defined in this package rather than Ada types such as INTEGER and FLOAT. There should be a type in this package that is appropriate for all objects in the system. If there is not, add one.

Data Bases. None

Environment Variables. None

4.1.2 UWN Window System

The window system contains utilities for displaying and interacting with windows. The utilities are divided into the following separate Ada specifications:

- Button Menu Manager
- Walking menu Utilities
- Walking Menu Layout Display
- Window Utilities

4.1.2.1 UWN_WINDOW_SYSTEM

Abstract. UWN_Window_System is the window utilities system using the X-window protocol. This package is linked to the cwn.lib library via the Ada binding CWN_Window_System.

Major Capabilities. The Soldier Machine Interface (SMI) software is based on a window and icon protocol with user selection and input via the mouse or keyboard. Application programmers can create many user interface tools within process windows and popup windows, using any combination of subwindows, panels, and subpanels. The interfaces provided by the window utilities include system messages, message boxes, buttons, menus, and a variety of field editors.

As previously stated, the devices for user input are the mouse and the keyboard. Typically, any input within a tool is processed by the tool. The application is notified only of the resultant input, not the means by which the input was performed. At times, user input or software will cause parts of the display tools to be partially hidden, and later exposed. Tools defined with the UWN utilities will redisplay automatically.

The only outside input (i.e. not by the user) which an application may receive, is input from any open network socket. The application calls UWN_ADD_INPUT_SOCKET with the socket id to notify the window system to watch for input. When input is detected, UWN notifies the application which is responsible for getting the input from the socket. The socket may be removed when it is no longer needed.

Input is sent to an application from UWN_INPUT in the form of an event as outlined below:

Input Event	Description
Exit	The user wishes to terminate a process window.
Menu	A menu option has been selected.
Checkbox	A checkbox has been selected.
Scrollbar	A scrollbar has been scrolled.
XrFILE	A network message is waiting to be read.
Button	A button has been toggled.
Mouse Button Pressed	The user has pressed a mouse button within a window which selected this event notification.
Mouse Button Released	The user has released a mouse button within a window which selected this event notification.
Field Traversal	A traversal key was entered from the keyboard within an activated numeric or string field.
Exposure	Some portion of a window which selected this event notification may need to be redrawn.
Open Window	A process window was opened from its icon state.

Window Resized	A process window was resized.
Close Window	A process window was closed into an icon.
Pushbutton	A pushbutton was selected.
Radiobutton	A radiobutton was selected.

Most communications with the interface tools are for the following operations and are indicated in the names of the functions:

activate	-	Activates a specific tool, making it visible and ready for input.
change	-	Change a specific aspect of a defined tool.
create/ define	-	Usually creates a tool with a specified location and size.
delete	-	Deletes the tool. Once deleted the tool should not be referenced again.
move	-	Provides the capability of changing the tool's location within the window or panel.
query	-	Provides the application the ability to query the size of the already defined tool in screen pixels.
resize	-	Provides the capability of resizing and changing the location of the tool on the screen.

.Windows

Every window application relies on at least one of the two basic types of windows, process windows and popup windows. These windows can be created and destroyed as required by the program. All windows, when defined, may be created to be visible on the screen, "mapped", or invisible, "unmapped". The mapping and unmapping operations of the windows are provided in UWN.

All windows are also capable of displaying bit images, pixmaps, or raster images; e.g., a map, and performing other graphics operations using other utilities in the system. (See UIW and UTM.) Applications using these utilities will need to know when an exposure event has occurred. The application can notify the window system it needs this information to redraw the graphics by calling UWN_SELECT_INPUT. This routine also provides for the notification of mouse button input that takes place within the window but outside of any of the window's tools.

A process window is one associated with one of the Tactical Planning Workstation system processes and associated icon stacks as follows:

View Reference
View Situation
Process Messages
Build Situation
Tools
Experiment Control

An application creates a process window via UWN_CREATE_WINDOW, specifying the window's label and process type. It will return the icon stack position assigned to the window. The window label will appear in a popup identification window in the upper left corner of the display. This will signal the user which window is waiting to be manually sized and/or placed. If the label input is NULL then there will be no window title or prompt displayed.

A flashing dotted line display will show the window's default size when the user clicks the left mouse button at the location desired for the window. The user may increase the window's size by pressing the middle mouse button to define one corner and, keeping the mouse button pressed, moving the cursor to the desired opposite corner before releasing the button.

The displayed process window consists of a border containing a title, darkened corners, and a popup menu option. The border provides the capability of moving the window by pressing the left button anywhere except the darkened corner areas, and releasing the button at the new desired location. The darkened corner areas provide window resizing when the left mouse button is clicked within that area. The popup menu is activated by pressing the right mouse button in the border area. The popup menu, entitled Frame Menu, offers options for the window operations of moving, resizing, hiding, exposing, redisplaying, closing the window into an icon on the corresponding icon stack, zooming the window to full screen display, and terminating the process.

Since a process window is associated with a process, UWN provides some routines specifically for use only with a process window. For instance, if an application wishes to change all the tools within the window, the application can call UWN_CLEAR_WINDOW to delete all buttons, editors, and panels. The entire window can be terminated via UWN_TERMINATE_WINDOW and a new window created. A popup window is quite limited in its dynamic operations compared to the process window. It is located as specified in the call to UWN_DEFINE_POPUP_ WINDOW and does not have the border or the inherent operations of the frame menu. However, the programmer may give some of these features to the popup window using menus and other operations provided in UWN; e.g., UWN_HANDLE_WINDOW_MOVE.

Subwindows are windows that are considered to be children of process windows, popup windows, or of other subwindows. A subwindow's size may exceed the size of its

parent window, but the visible portion of the window is limited to, or clipped by, the parent window. A subwindow, like the popup window must be given operational aspects.

.Panels and Subpanels

A panel is a collection of field editors displayed in a window and controlled by the panel manager. The panel manager is responsible for the display of the editors, passing control to a field editor selected by the user, and routing the editor activity back to the application. A panel must have at least one field editor defined and the application must determine the size of the panel needed, using UWN_QUERY_PANEL_SIZE, so that all of the editors are visible.

Panels may contain subpanels when the application requires different editors according to the state of the panel or application. A panel may contain an unlimited number of subpanels, but caution should be taken not to overlap the editors of panels and subpanels. By default, panels and subpanels are displayed when completely defined, but may be hidden and displayed again as required.

.System Messages

UWN provides the capability for displaying messages about what is going on in the system. Many times it takes a while for an application to respond to user input because of initialization or processing time. When this happens, usually nothing is visibly taking place on the screen, thus leaving the user to wonder if the system received the input. System messages e.g. "BUSY", are provided to give a visual status of something taking place in the software. They are displayed at the top middle of the display screen when created. The messages are limited to one line.

.Message Boxes

Many times an application needs to relay a message to the user in the form of a warning or instruction on how to do something or as a simple question asking or confirming the user's intentions. This may be done through a message box via UWN_MESSAGE_BOX, which displays a message in a rectangular region centered in the middle of the screen and grabs all mouse input until one that is acceptable causes the message box to be taken away. The application must inform the routine of the message and the mouse button events which are acceptable for having the message removed. The capability of multiple mouse events is provided for simple multiple choice questions which the application will process according to the input returned.

.Buttons

Buttons can be defined anywhere within a window as long as it is not within a panel. Each button may include a text label which will be displayed in the center of the rectangle

representing the button. This label may be single or multiple lines of text; however, care should be taken to insure the button is sized correctly to have the full label visible. There is no routine to determine the minimum size required, so the application should calculate the size required based on font size, number of characters, and number of lines in the label.

When the button is defined as being "enabled," it will toggle its background when selected and UWN_INPUT will notify the application. In some cases, the application may want the toggle feature only as an indication to the user that performance of some action is taking place; i.e., the button is not truly representing an "on" state. UWN_TOGGLE_BUTTON has been provided to toggle the button back off when the processing is finished. This routine may also be used to show default "on" states of buttons as they are first created and displayed. The routine toggles the button from the current state, off to on, to the other state. A button may activate a popup or walking menu through UWN_ACTIVATE_MENU.

.Menus

Menus in UWN are presented in a popup window either as simple list of options or as a tree of primary options with branches to submenus which are presented after selection of a primary option. One menu may be activated per window, button, or panel. If the application wishes to activate another menu for the same display unit, it must first deactivate the present menu and then activate the defined menu.

An activated menu is always brought up by the user by pressing the right mouse button within the unit where the menu was activated. A deactivated menu can be displayed by capturing the right mouse button with UWN_SELECT_INPUT and displaying the menu with UWN_POST_MENU.

A tree menu, called a walking menu, is displayed similar to the simple list menu but will have arrows on the right side of those options which have submenus. The user "walks" down the menu by placing the cursor over one of the options and proceeding over to the section of the box containing the right arrow. When the cursor comes in close proximity to the arrow, the submenu will be presented. A walking menu may have an infinite number of submenus.

Another menu system, the button menu manager, is also provided in the UWN utilities and is discussed in Section 4.1.2.2.

.Field Editors

Field editors are the collection of interface tools that aid application programmers by providing them with a common user interface across a wide variety of programs. Field editors are defined in size by the number of display pixels. All input within field editors is processed by the editors and the application is notified of the final selection through UWN_INPUT. The field editors provided by UWN are described in the following sections.

Checkbox Editor

A checkbox editor is used to create and process a group of related checkboxes, where the user is allowed to select a number of options. The application creating the editor has flexibility in the number of rows and columns into which the boxes will be displayed. Each checkbox may have an optional label which will be displayed to the right of the box. The editor determines from these factors the layout of the display. UWN provides the capability to query the created editor for the actual coordinates of the checkboxes. The application may specify which checkboxes should be selected upon creation and may change states as required.

Number Field Editor

The number field editor allows applications to create and display single lines of editable numeric text within a rectangular region. The numeric text entered is limited to integers. The application may limit the range of integers that can be entered as well as the length of the string. The field may be created with an optional label and an initial value. The activated field accepts insertions, deletions, and traversal keys. Traversal keys consist of the up and down arrow keys, the tab, back tab (shift tab), and return. Input of any of the traversal keys causes the field to be exited and the application notified of the direction of traversal as follows:

Traversal Key	Direction of Traversal
Tab	Next field
Back Tab	Previous field
Return	Next field
Up Arrow	Up field
Down Arrow	Down field

Use of the traversal keys is applicable if the application is a form manager where the user is allowed to move about numerous fields with keystrokes instead of constantly mousing new fields to input. The application uses UWN_ACTIVATE_NUMBER_FIELD to activate the appropriate field for the user whenever a traversal event is received.

Pushbuttons

Pushbuttons are provided for immediate action selections. They are drawn as an oval with an optional label displayed inside the oval. The editor is defined in the number of rows and columns into which the button is to be displayed. The actual coordinates of the individual buttons may be queried from UWN after the editor has been created.

Radiobutton

The radiobutton editor is similar to the checkbox editor except that the editor operates in a fashion very similar to the channel select buttons on a radio. At no time will it allow the situation to occur where no button is active. Each time a new button is selected, the previously active button is made inactive. Like the checkbox and pushbutton editors, the application has great flexibility in the editor's layout in terms of the number of rows, columns, and labels and can query the editor for the actual coordinates of each individual button after creation.

Scrollbar

An application can create either a vertical or horizontal scrollbar wherever the need arises. The scrollbar provides the capability to position the display within a file, document, or display that would be impractical or impossible to display in its entirety. When a scrollbar is drawn, it is drawn as a rectangular box, with a scroll arrow at each end. The area between the two scroll arrows is known as the scroll region and contains the scroll box. The scroll region represents the information in its entirety whereas the size of the scroll box represents the portion of information which is currently being displayed. The position of the scroll box in the scroll region portrays the position of the displayed information with respect to the entire informational unit.

All selections within the scrollbar cause a slide position to be returned to the application via UWN_INPUT. It is up to the application to do the actual scrolling. If the application wishes to change the scrollbar size or notes a change in size of the informational unit, it can use the routine UWN_CHANGE_SCROLLBAR to make adjustments.

Static Text

The static text editor provides the application with the means for placing uneditable blocks of text anywhere within the bounds of a panel or window. The application specifies the rectangular region, text to be drawn into it, and the alignment of the text within the region. If the text will not fit completely within the rectangle, then only that portion which fits will be displayed. The static text may be displayed as multiple lines by including a carriage return character at the end of each line.

The four forms of text alignment provided by the editor are:

Centered	-	The center of each line of text is positioned at the center of the rectangle. All leading and trailing spaces in the line will be stripped.
Left	-	The first character of the line is positioned at the leftmost edge of the rectangle, with all leading and trailing spaces in the line stripped.
Right	-	The last character in each line is placed at the rightmost edge of the rectangle, with all leading and trailing spaces in the line stripped.
None	-	The first character of the line is positioned at the leftmost edge of the specified rectangle. Leading and trailing spaces in the text are not stripped.

The static text editor also has a popup menu option for copying text in to a text editor.

String Field Editor

The string field editor is similar to the number field editor except that it creates and displays single lines of editable alphanumeric text. It too accepts traversal keys as described in the above number field section.

Text Editor

The text editor provides an application the means for displaying multiple lines of text with the option of allowing the user to edit the text. It consists of a scrollbar on the left side of the text buffer. When activated, a cursor is visible to the user in the buffer to indicate the place of operation within the text buffer. The operations available to the user are displayed in a popup menu activated by a right mouse button click within the editor. When created as a read-only buffer, the only operations available are find and copy. The find function locates the next occurrence of user selected text. The copy function saves user selected text for later retrieval into another text editor that is defined to be editable. A text editor created without the read only option has these two functions along with cut and paste and insert and delete text functions.

An application may use the same editor for numerous documents by simply using UWN_CHANGE_EDITOR_TEXT to change from one document to another. All text contained within the buffer at the time of creation will be displayed. A text editor may be created without any text to allow users to create their own text buffer.

User Input Field

The user input field is a special interface tool using either the number field or string field editor in a popup window. The intention of this tool is for the occasion when the application needs to have immediate specific input before proceeding. The user input field is displayed and ignores all input of the system until the information is complete.

Special Instructions. Use of the window system first requires its initialization by calling UWN_INITIALIZE_WINDOW_SYSTEM.

All UWN utilities using text use a default font whose size may be queried by the function UWN_QUERY_FONT_SIZE.

All text passed to a routine is expected to be a null terminated string.

Every object of the window system is located with respect to its destination's origin which is defined to be 0,0 in the upper left hand corner. The coordinate system increases positively for X going to the right, and for Y going down.

All object operations require the id that was given to the object at the time of it's creation. The object creation procedures return the object id for all objects except menus, which are defined by the calling application.

.Windows

A process window's default size is that required for an 80 character wide by 24 character high text editor and one row of buttons. A process window cannot be initially defined with a size smaller than this default. After activation it may be resized as desired.

If a process window's label input is NULL then there will be no window title or prompt displayed.

An application is limited to one process window but is unlimited in the number of popup windows, subwindows, panels, and subpanels.

UWN_CLEAR_WINDOW clears every object; i.e., buttons, panels, and menus. Subwindows must be deleted by the application.

UWN_TERMINATE_WINDOW performs the UWN_CLEAR_WINDOW operation so the application need not call both.

An application may not select exposure events for windows containing field editors.

Exposure events are sent for a window and all of its subwindows when the window has been moved. This is the only way of detecting whether a window has been moved.

All input selected for a window and any menu activated for a window will also be effective for its subwindows if input is not individually selected or menus activated for them.

.Panels

Before creation of any field editors in a panel, UWN_DEFINE_PANEL must be called. The user then creates the desired field editors, which will not be displayed and operational until UWN_END_PANEL has been called.

If a panel is to be deleted, all of its field editors must be deleted first.

The query size routines for panels and subpanels are unlike the other UWN query size routines in that they do not return the size of the panel as specified by the user when the panel's definition was ended via UWN_END_PANEL. The size returned is the size needed if all the defined field editors were to fit completely within it and not be clipped. The size also includes white space padding on the bottom and right sides of the panel.

If many changes are being made within a panel, the application can reduce the number of window redraws and "screen flashing" by first calling UWN_HIDE_PANEL. After the changes are made UWN_SHOW_PANEL can be called to show the final output.

UWN_UPDATE_PANEL must be called to show any editors added to or clear any deleted from a panel after the panel has been completely defined; i.e., calls to both UWN_DEFINE_PANEL and UWN_END_PANEL have been made.

.Message Boxes

UWN_MESSAGE_BOX is the one exception to the rule that all input will be returned via UWN_INPUT. This tool will ignore all input other than that specified when the call was made. It "grabs" the server and the mouse input and will return them to the application only after its processing is complete. Any other event; e.g., exposure events will be ignored and lost.

A message may consist of multiple lines. The carriage return character must be included in the message string to indicate new lines.

.Menus

All arrays used in the definition of a menu are indexed starting at zero.

.Field Editors

Checkbox Editor

The checkbox labels are not included in the coordinates returned by a query operation.

Pushbutton Editor

The pushbutton labels are not included in the coordinates returned by a query operation.

Radiobutton Editor

The radiobutton labels are not included in the coordinates returned by a query operation.

Text Editor

The programmer must be aware of the width and height parameters in UWN_DEFINE_EDITOR and UWN_RESIZE_EDITOR, in that they take exception to the rule of defining these dimensions in screen pixels. Rather, they are defined in the number of character rows and columns.

If the text is in a special format requiring specific new lines, the buffer must contain the line feed at the end of each line.

To compile this package, the following paths must be established using the "a.path" command:

/eddic/Ada/common
/usr.MC68020/cherokee/VADS55/verdixlib
/usr.MC68020/cherokee/VADS55/standard
/eddic/Ada/ued

To load a program that uses this package, links must be established with the following libraries using the "a.info" command:

/egen/cwn_util/cwn.lib
/usr/lib/libXr.a
/usr/lib/libX_p.a

/usr/lib/libX.a
/usr/lib/libm.a

Data Bases

ICON_STACK_DB	INPUT/OUTPUT

Environment Variables

ICON_PATH

4.1.2.2 UWN_BUTTON_MENU_MANAGER

Abstract. The UWN_BUTTON_MENU_MANAGER is contained in the UWN_WINDOW_SYSTEM package as a manager of a popup menu system capable of allowing multiple selections.

Major Capabilities. The button menu manager is a unique menu tool consisting of a combination of a title, a scrollbar, either a radiobutton or checkbox editor, and a pushbutton editor. It supports either a single selection or multiple selection menu as defined by the application with flexibility in the number of columns in which the options are to be displayed. The application may also specify the number of rows of options which may visible to the user at one time. If the number of rows visible is less than the total number of rows required to display all the options, a scrollbar will be incorporated in the creation of the tool, thus giving a means for the user to scroll through all the options.

Pushbuttons are an option in the tool for specifying four specific functions:

DONE	An indication to the application that the user is done selecting options.
CANCEL	An option which allows the user to cancel the menu selection process.
SET ALL	An option which applies only to multiple selection menus and causes all the options to be selected.
CLEAR ALL	An option which applies only to multiple selection menus and causes all option selections to be cleared.

Special Instructions. A button menu may be defined as unmapped and be mapped later via UWN_MAP_WINDOW passing the window id of the popup window in which the menu was defined. After selections have been made the menu may be unmapped, available for future use, or deleted. The application is responsible for detecting input to the button menu using

UWN_INPUT. Any input received must then be passed to UWN_BUTTON_MENU_INPUT for processing. This input will consist of one of three possibilities: DONE, CANCEL, or NO_ACTION_REQUIRED. If the DONE or CANCEL was selected, the application can detect what was selected through the button menu's description buffer. This buffer is owned by the application and is only updated from the button menu manager. A NO_ACTION_REQUIRED output indicates that the input was handled internally by the button menu manager; i.e., scrolling of the options or resetting all the options on or off.

The scrollbar will not be displayed when the number of visible rows requested is less than three, the minimum space required for displaying a scrollbar.

To compile this package, the following paths must be established using the "a.path" command:

/eddic/Ada/common
/usr.MC68020/cherokee/VADS55/verdixlib
/usr.MC68020/cherokee/VADS55/standard
/eddic/Ada/ued

To load a program that uses this package, links must be established with the following libraries using the "a.info" command:

/egen/cwn_util/cwn.lib
/usr/lib/libXr.a
/usr/lib/libX_p.a
/usr/lib/libX.a
/usr/lib/libm.a

Data Bases. None

Environment Variables. None

4.1.2.3 UWN_WALKING_MENU

Abstract. Walking and multiple selection menu utilities.

Major Capabilities. The walking menu utilities provide procedures for reading a walking menu file into an ASCII buffer and its associated array and loading the ASCII buffer into the walking menu structures required by the UWN window utilities.

Special Instructions. This is a generic package and must be instantiated with the associated data type, the associated array type, and a pointer to the associated array type.

When the menu description file contains multiple selection menus, UWN_READ_WALKING_MENU builds a list of the multiple selection menu records. The application is responsible for looping through the list and calling UWN_BUILD_MULTIPLE for each multiple selection menu in the list. See the UTM map menu software for an example.

See Appendix D for the format of some sample walking menu files.

To compile this package, the following paths must be established using the "a.path" command:

/eddic/Ada/common
/usr.MC68020/cherokee/VADS55/verdixlib
/usr.MC68020/cherokee/VADS55/standard
/eddic/Ada/ued

To load a program that uses this package, links must be established with the following libraries using the "a.info" command:

/egen/cwn_util/cwn.lib
/usr/lib/libXr.a
/usr/lib/libX_p.a
/usr/lib/libX.a

Data Bases. None

Environment Variables. None

4.1.2.4 DML_DSPL_MENU_LAYOUT

Abstract. DML_DSPL_MENU_LAYOUT is an acronym for the Display Menu Layout package, which draws, graphically, the walking and/or multiple selection menu hierarchy.

Major Capabilities. DML_DSPL_MENU_LAYOUT displays a hierarchical picture of any walking menu or multiple selection menu in the Tactical Planning Workstation, using the tree structure builder (TSB) package. DML_DSPL_MENU_LAYOUT will not edit a menu; it merely displays it. If a display will not fit in the window, then scroll bars will be added automatically in the direction(s) needed.

Special Instructions. DML_DSPL_MENU_LAYOUT will only display one menu at a time, and that menu's file name is passed in by argument. It is the applications responsibility to create the file name of the menu to be displayed, and provide the user a means of selecting the desired menu.

The display menu layout uses Ada tasking. There are three tasks within DML: First, is a one time, per execution, initialization (DSPL_INIT_MENU); second, is a one time, per execution, termination (TERMINATE_TASK); third, is all other event processing (PROCESS_INPUT). PROCESS_INPUT does not receive events directly from the system, via UWN_INPUT; the calling process passes input events to it through the procedures arguments. Because PROCESS_INPUT does not have its own call to UWN_INPUT, it must tell the application if the event received was a window termination event. PROCESS_INPUT is called once for each event.

To compile this package, the following paths must be established using the "a.path" command:

/eddic/Ada/common
/usr.MC68020/cherokee/VADS55/verdixlib
/usr.MC68020/cherokee/VADS55/standard
/eddic/Ada/uiw

To load a program that uses this package, links must be established with the following libraries using the "a.info" command:

/egen/cwn_util/cwn.lib
/egen/ciw_util/ciw_util.lib
/usr/lib/libXr.a
/usr/lib/libX_p.a
/usr/lib/libX.a

Data Bases. Any menu file, determined by the input arguments.

Environment Variables. None

4.1.3 UED EDDIC Utilities

The EDDIC utilities consist of general purpose utilities that are used throughout the system. The utilities provide the following major functions:

Math Functions
String Manipulation Utilities
List Manager
Queue Manager
Tree Layout Manager

Each major function has its own Ada specification and body and will be described separately in the following section.

4.1.3.1 UED_EDDIC_MATH_UTIL

Abstract. All-purpose math utilities.

Major Capabilities. The math utility package provides the following capabilities:

- Ordering units in a task organization
- Distance between two points
- Distance between a point and a line segment
- Intersection of two lines
- Sine and cosine of a line
- Intersection of two line segments
- Offset a point a distance from a line
- Intersection of a point and a line

Special Instructions. To compile this package, the following paths must be established using the "a.path" command:

/eddic/Ada/common
/usr.MC68020/cherokee/VADS55/verdixlib
/usr.MC68020/cherokee/VADS55/standard

To load a program that uses this package, links must be established with the following libraries using the "a.info" command:

/usr/lib/libm.a

Data Bases. None

Environment Variables. None

4.1.3.2 UED_LIST

Abstract. UED_LIST is a generic utility package for creating and maintaining an ordered list of data items.

Major Capabilities. The list utility enables the user to create a list by providing the capability to insert before or after another data item. Functions are provided to go to the beginning of the list, check for the end of a list, or obtain the count of the number of elements in the list. Other capabilities include retrieving an item from the list without deleting the item from the list, deleting an item from the list, setting the current item in the list to a specific item, or querying the contents of the list via receiving an array of the list's data.

Special Instructions. The list always maintains a pointer to the current item in a list and operates with respect to the current position. An insertion operation always causes the newly inserted item to be the current item. Therefore, if item A is inserted before item B and the next operation performed is a retrieval of the next item, item B would be retrieved, whereas, if the next operation was delete, item A would be deleted from the list. A query of the list's contents always sets the current position to the beginning of the list.

To compile this package, the following paths must be established using the "a.path" command:

/eddic/Ada/common
/usr.MC68020/cherokee/VADS55/verdixlib
/usr.MC68020/cherokee/VADS55/standard

Data Bases. None

Environment Variables. None

4.1.3.3 UED_QUEUE

Abstract. UED_QUEUE is a generic utility package for creating and maintaining a queue of data items where the data items are added to the end of the list and retrieved from the beginning of the list. In other words, this is a first-in-first-out list utility.

Major Capabilities. The user has the capability of peeking at the next item on the queue or deleting the item from the queue. Functions are provided for determining if the queue is empty or how many items are left in the queue. One may also query the contents of the queue via receiving an array of the queue's data.

Special Instructions. To compile this package, the following paths must be established using the "a.path" command:

/eddic/Ada/common
/usr.MC68020/cherokee/VADS55/verdixlib
/usr.MC68020/cherokee/VADS55/standard

Data Bases. None

Environment Variables. None

4.1.3.4 UED_STRING_UTILITIES

Abstract. All-purpose string utilities.

Major Capabilities. The string utility packages provides the following general purpose string utilities:

- Count the number of lines in a string buffer
- Convert an integer to a string
- String search

Special Instructions. To compile this package, the following paths must be established using the "a.path" command:

/eddic/Ada/common
/usr.MC68020/cherokee/VADS55/verdixlib
/usr.MC68020/cherokee/VADS55/standard

Data Bases. None

Environment Variables. None

4.1.3.5 TSB_LOCATION

Abstract. TSB_LOCATION is an acronym for Tree Structure Builder Location package, which determines (builds) the placement (location) of all the elements in a hierarchical tree structure, as well as drawing all of the connecting lines to each element.

Major Capabilities. Before addressing the capabilities, some terms need to be defined. All elements in the tree will be equated to a family. The first element is called a parent. If that element has any elements hierarchically below it (inferior), then these are his children. If children have children then the former also become parents, and so on. If any element has any elements hierarchically equivalent to it, then these are his siblings. This is true for parents as well as children. The very first element in the list is the 'oldest', and each subsequent set of children is a generation.

TSB_LOCATION builds a hierarchical tree structure where a parent is above, in the Y-direction, and centered, in the X-direction, on it's children. The calculation of locations is done in inverse order, lowest to highest. The application passes in the oldest (highest) element and the algorithm steps down until an element is found with no children, and it is placed. Then its siblings are placed, again stepping down to a level of no children before placing. Internally there is a position availability tracker for each generation. Entire

generations may be shifted to the right in order to accommodate elements from a previous lower generation. This is accomplished with recursive programming.

In a lot of tree structures the display X to Y ratio is very lop-sided in the X-direction. In an attempt to make this ratio more even; i.e., the display more square, children may be displayed by one of three methods, see Figure 4-1:

1. This method is the normal way trees are displayed with each child displayed one next to the other forming a horizontal line under the parent. This method does not save any horizontal space. This method is accomplished by setting both the VRT_CHLDRN_R_LEGAL and VRT_SIBLNG_R_LEGAL arguments to 'False' on the call to TSB_FIND_XY_LOC.

2. This method will save space by stacking the children under the parent forming a vertical line down. This can only be done to a generation of children who have no children of their own. If any child in that generation has a child, then the whole generation must be displayed horizontally. There are also checks in the program to make sure that vertical placement will result in actual X-direction space savings. If a previous siblings lower generation would block the families vertical display then the entire mini-family would be shifted over to accommodate. If the shift is so great that displaying horizontally would have been less costly, then the whole generation is displayed horizontally. This method is accomplished by setting the VRT_CHLDRN_R_LEGAL argument to 'True' on the call to TSB_FIND_XY_LOC.

3. This method will also save space and is a spin off of method two. This method is best explained with an example: There are six children in the family, the first two (oldest) siblings have children, the last (youngest) four siblings do not; The last four children are then displayed vertically after the last sibling with children, child number two. The resultant display would have the first three children in a horizontal line with the last three children in a vertical line underneath the third child. The parent is then centered in the X direction above the first three children. This method is accomplished by setting the VRT_SIBLNG_R_LEGAL argument to 'True' on the call to TSB_FIND_XY_LOC.

The greatest space savings occurs with a combination of methods two and three. This is accomplished by setting both the VRT_CHLDRN_R_LEGAL and VRT_SIBLNG_R_LEGAL arguments to 'True' on the call to TSB_FIND_XY_LOC.

Special Instructions. The TSB_LOCATION package is called by the outside world for one of two reasons: (a) to determine the x-y location of each element in the tree, (b) to draw the connecting lines between each element of the tree.

Upon return the upper left corner x,y and center x,y points will be set for each element.

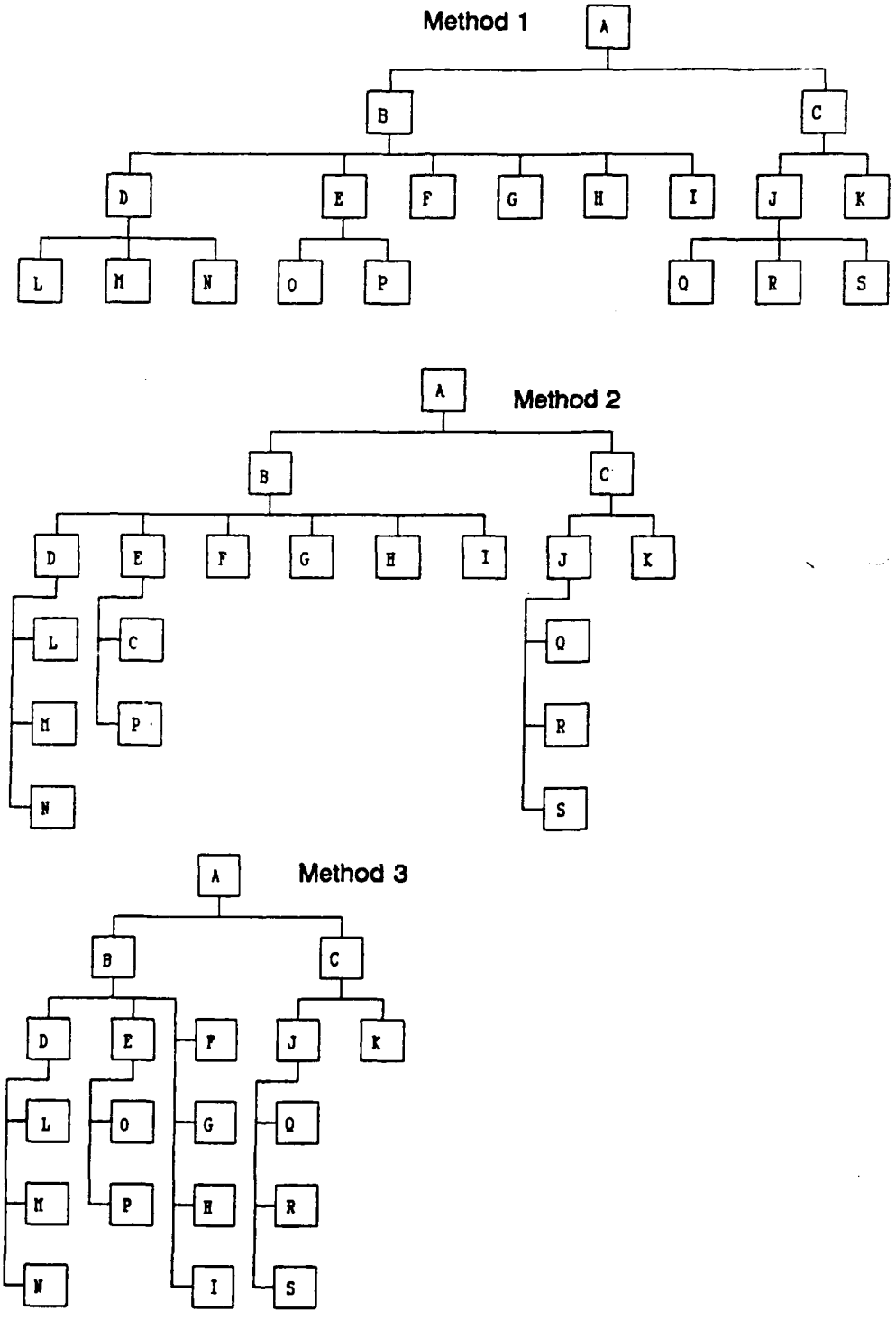


Figure 4-1. Tree Display Options

Within this package there is a record structure which contains the attributes of each tree element. One of the attributes in this structure is a pointer to the elements first child; another attribute points to the elements next sibling. Using these two attributes of the structure a forward pointing link list can be built. A parent with multiple children points to its first child and each child then points to its next sibling. This record structure also contains an attribute reserved for application dependent data. This data structure is determined by the application. This is accomplished by having the application define the data structure type, then instantiating this package with that structure type. If the application does not require any special data associated with each element, then it must create a dummy structure type.

Before this package is called the entire link list must be established and the width and height attributes must be set for each element.

To compile this package, the following paths must be established using the "a.path" command:

/eddic/Ada/common
/usr.MC68020/cherokee/VADS55/verdixlib
/usr.MC68020/cherokee/VADS55/standard
/eddic/Ada/uiw

To load a program that uses this package, links must be established with the following libraries using the "a.info" command:

/egen/cwn_util/ciw.lib
/usr/lib/libX_p.a
/usr/lib/libX.a

Data Bases. None

Environment Variables. None

4.1.4 UFM Form Manager

The form manager consists of a high-level package that controls form display and interaction, and a low-level package for interacting with individual editors within a form. The high-level package (UFM_FORM_MANAGER) uses an ASCII buffer to describe the layout and contents of a form. Once the ASCII buffer is passed to the form manager, the form manager takes control of the form. The low-level package (UFM_FORM_FIELDS) provides procedures to dynamically add or delete individual editors in a form. The calling application is responsible for the location of all editors and processing all input except field traversal.

4.1.4.1 UFM_FORM_MANAGER

Abstract. High-level form manager package that provides capabilities to read a form description from a file, validate a form description, and display and maintain a form.

Major Capabilities. The Form Manager provides a easy and flexible tool for designing and maintaining a form based user interface. A form consists of static text, form field editors, geometric symbols. Figure 4-2 shows the editors that are available in a form. The form manager provides procedures to read a form description buffer from a file, validate a form buffer, and display the form buffer.

The ASCII buffer that describes the form is divided into three sections. The first section contains the static text and the layout or position of the editors. Space must be reserved in both the X and Y direction for the size of the editors. The second section describes the geometric symbols to be drawn in the form. The third section describes the attributes of each editor. See "FORM_DESCRIPTION" in appendix D for a complete description of the form ASCII buffer.

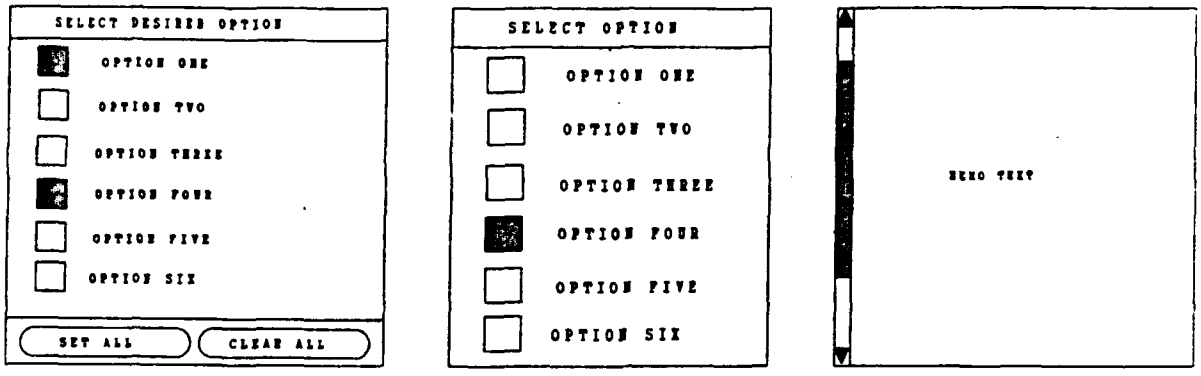
Special Instructions. The validator procedure checks a form description for accuracy and completeness. The following lists some of the warnings or errors which are detected:

Invalid form size
Invalid static text size in terms of absence of carriage returns
Absence of a static text section terminator
Invalid editor parameter values
Insufficient parameters to define an editor
Detection of an editor located but not described
Detection of an editor described but not located

If insufficient parameters are used to define an editor, the validator will display warnings stating that default parameters will be used. The default parameters for the editors are shown as part of the "FORM_DESCRIPTION" in appendix D.

The form manager was implemented as an Ada task. The typical order of processing from the application's standpoint, is to define the form, pass detected input to the task for processing, query the results of the input processing, and lastly, terminate the form task.

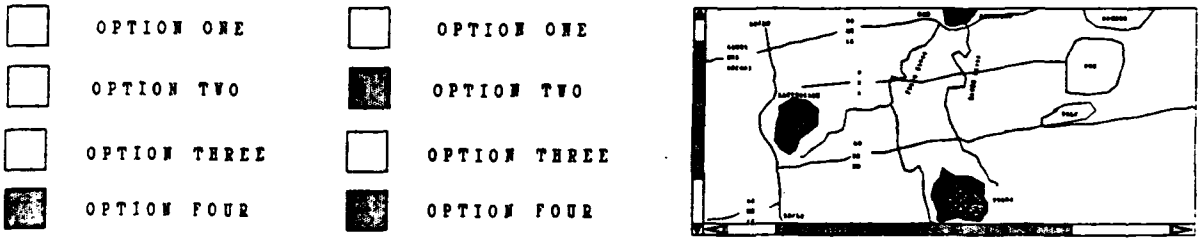
The form's visible size is specified by the application. If the form exceeds the size in either width or height, a scrollbar is inserted into the form. The form manager utilizes three windows for displaying a form (form, parent, and clipping). The form window contains the whole form including the portions that are outside the visible window. The parent window contains the optional scrollbars and the clipping window. The clipping window defines the visible portion of the form. If the whole form fits into the window, the form window and the clipping window are the same size.



Multiple Selection Menu

Single Selection Menu

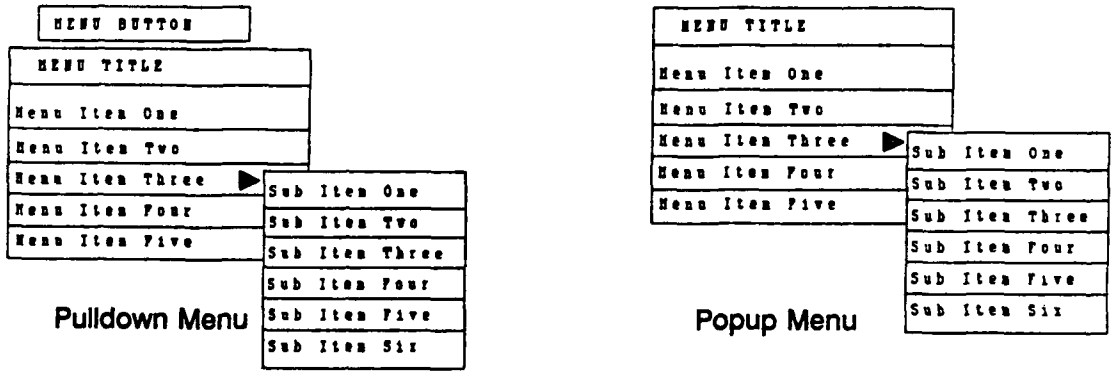
Memo Text Editor



Radio Button

Checkbox

Digital Map



Pulldown Menu

Popup Menu

Push Button

String Editor

Numeric Editor

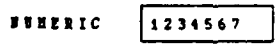


Figure 4-2 Form Field Editors

An additional feature was incorporated into the form manager to handle the memo-text scrollbars. If the user scrolls the form so that both of the memo-text scroll arrows are not visible, the form manager resizes the memo-text field to fit in the visible portion of the form.

Once a static text indicator is found in the static text section, all lines of the text will align with the column where the indicator was first detected, until another static text indicator is found.

Entry point "TERMINATE_FORM_TASK" deletes the form editors, therefore, "DELETE_FORM" does not need to be called before "TERMINATE_FORM_TASK".

To compile this package, the following paths must be established using the "a.path" command:

/eddic/Ada/common
/usr/cherokee/VADS55/verdixlib
/usr/cherokee/VADS55/standard
/eddic/Ada/ued
/eddic/Ada/utm
/eddic/Ada/uux
/eddic/Ada/uwn
/eddic/Ada/uin

To load a program that uses this package, links must be established with the following libraries using the "a.lnfo" command:

/egen/cin_util/cin_util.lib
/egen/ciw_util/ciw_util.lib
/egen/cux_util/cux_util.lib
/egen/cwn_util/cwn.lib
/usr/lib/libXr.a
/usr/lib/libX_p.a
/usr/lib/libX.a
/usr/lib/libm.a

Data Bases. FORM_DESCRIPTION INPUT/OUTPUT

Environment Variables. None

4.1.4.2 UFM_FORM_FIELDS

Abstract. UFM_FORM_FIELDS is the base utility package for defining and managing fields within a form defined via UFM_FORM_MANAGER or manually from software.

Major Capabilities. The form field manager is similar to that of the form manager but has the distinct advantage of performing dynamic interaction within a form; i.e., defining, moving, deleting, and resizing individual editors. It also performs the traversal from one traversal editor to another. The traversal editors include the number field editor, the string field editor, and the full page text editor.

Special Instructions. Procedure UFM_INITIALIZE_FORM_FIELDS must be called before any editors are added to a form.

The form field manager was developed to be used in conjunction with the UWN window utilities system. It does not keep track of individual panels, windows, or menus an application wishes to use. It is only concerned with the location of the physically displayed objects within a form.

The valid form fields consist of the following:

Field	Description
Button_Walk	A pulldown walking menu activated by a button
Checkbox_Menu	A checkbox editor
Memo	A full page text editor
Number_Field	A numeric field editor
Pushbutton	A push button field editor
Radiobutton	A radio button field editor
Scrollbar	A scrollbar field editor
Static_Text	A static text field editor
String_Field	A string field editor

A button_walk editor can be defined, but cannot be changed or moved. The incorporation of this editor was for the development of the form manager only. It should also be noted, that the tracking of a digital map's location and size is not included in this package.

To compile this package, the following paths must be established using the "a.path" command:

/eddic/Ada/common
/usr/cherokee/VADS55/verdixlib
/usr/cherokee/VADS55/standard
/eddic/Ada/ued

To load a program that uses this package, links must be established with the following libraries using the "a.info" command:

/egen/cwn_util/cwn.lib
/usr/lib/libXr.a
/usr/lib/libX_p.a
/usr/lib/libX.a

Data Bases. None

Environment Variables. None

4.1.5 UIN INTERNET COMMUNICATIONS

Abstract. UIN_INTERNET_COMMUNICATIONS is an acronym for a set of utility communications primitives that allows processes to communicate with each other using an InterNet protocol. Programs may communicate with each other both within one processor and over an ethernet network.

Major Capabilities. UIN_INTERNET_COMMUNICATIONS is a stand alone utility package (not a process) that does not require the fileserver routers and/or data base managers to operate. This utility package is founded on a server client relationship, which is defined below in the special instructions.

Special Instructions. The server client relationship is best defined with an example (see Figure 4-3): Process A needs to communicate with process B, so process A becomes the server via UIN_ESTABLISH_SERVER. Once established as the server, process A may go about other business, but must ultimately come to the place where it sits, via UIN_SERVER_WAIT, and listens for others to call. Process B then connects, via UIN_CLIENT_CONNECT_SERVER, to the server, via UIN_SERVER_CONNECT_CLIENT, becoming a client. This is a one time connection and the connection should not be broken or closed via UIN_CLOSE_SOCKET, until the process is terminated or the client is sure there is no longer a need to communicate with the server. Once these connections are established, process B may go about other business. When the need arises for process B to communicate with process A, a message is sent via UIN_SEND_MSG, and the server, who is waiting via UIN_SERVER_WAIT receives it via UIN_RECV_MSG. There is a current UNIX system limit of thirty-one processes (clients) connected to any given server.

Within the Tactical Planning Workstation there are special processes set up, called routers, whose soul purpose is to facilitate communications among other processes. These routers are the servers, and all communications get routed (hence the clever name) through them en route to their final destination.

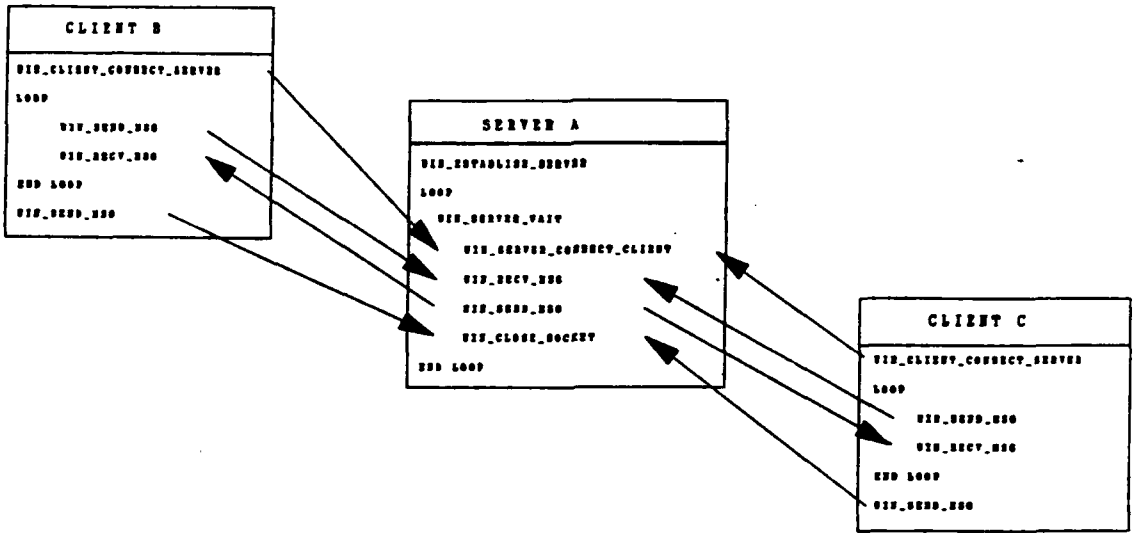


Figure 4-3 Internet Communications

The structure and layout of the message is almost entirely process dependent; i.e., the process may create an Ada structure of any size, shape, and type. The only stipulation is the beginning four bytes must contain the length (size in bytes) of the message. It then becomes the applications responsibility to make sure the clients and server have matching Ada structures, as UIN_SEND_MSG and UIN_RECV_MSG will merely pass a bit stream.

To compile this package, the following paths must be established using the "a.path" command:

/eddic/Ada/common
/usr.MC68020/cherokee/VADS55/verdixlib
/usr.MC68020/cherokee/VADS55/standard

To load a program that uses this package, links must be established with the following libraries using the "a.info" command:

/egen/cin_util/cin_util.lib
/egen/cux_util/cux_util.lib

Data Bases. None

Environment Variables. None explicitly, two implicitly passed in by argument.

host_id - name of the server machine.
service_id - name of the service id (INET port number).

4.1.6 UIW Image Window System

The Image Window system contains the utilities to display color images, fonts, and text. The image processing utilities are separated into a generic package (UIW_GENERIC) to allow the display of different image types. The other color utilities are contained in UIW_IMAGE_WINDOW.

4.1.6.1 UIW_GENERIC

Abstract. UIW_GENERIC is an acronym for a set of Utility Image Windowing primitives, which allows programmers to perform certain color graphics imaging functions within the X Windows System environment.

Major Capabilities. UIW_GENERIC is a stand alone utility package (not a process) which does not require the fileserver routers and/or data base managers to operate. This utility package allows programs to access X Windows color graphics imaging commands from high level languages, without having an intimate knowledge of the X Windows system. However, the programmer must have some knowledge or concept of X Windows or graphics processing. There is not a one to one pairing of modules to X Windows commands; only those commands required by the Tactical Planning Workstation have been developed.

Special Instructions. The application calling this package must define a data structure type for the image data, then instantiate this package with that structure type.

An image, laid straight into a bit pattern will be the inverse of what X Windows is looking for. So adjust it with UIWuux_16BIT_SWAP. Then a pixmap of the swapped image must be created with UIW_CREATE_PIXMAP, and finally it can be displayed with UIW_DISPLAY_BIT_IMAGE (uiw_image_window_s.a). UIW_DISPLAY_IMAGE displays images that are eight bits deep.

To compile this package, the following paths must be established using the "a.path" command:

/eddic/Ada/common
/usr.MC68020/cherokee/VADS55/verdixlib
/usr.MC68020/cherokee/VADS55/standard

To load a program that uses this package, links must be established with the following libraries using the "a.info" command:

/egen/ciw_util/ciw_util.lib
/usr/lib/libX.a
/usr/lib/libX_p.a

Data Bases. None

Environment Variables. None

4.1.6.2 UIW_IMAGE_WINDOW

Abstract. UIW_IMAGE_WINDOW is an acronym for a set of Utility Image Windowing primitives, which allows programmers to perform certain color graphics imaging functions within the X Windows System environment.

Major Capabilities. UIW_IMAGE_WINDOW is a stand alone utility package (not a process) which does not require the fileserver routers and/or data base managers to operate. This utility package allows programs to access X Windows color graphics imaging commands from high level languages, without having an intimate knowledge of the X Windows system. However, the programmer must have some knowledge or concept of X Windows or graphics processing. There is not a one to one pairing of modules to X Windows commands; only those commands required by the Tactical Planning Workstation have been developed.

Special Instructions. UIW_INIT_FONT must be called one time, up front, for each font type being used, before UIW_DISPLAY_SYMBOL or UIW_DISPLAY_TEXT may be used.

UIW_INIT_LOOKUP_TABLE must be called one time, up front. UIW_LOAD_LOOKUP_TABLE is the means for loading colors into the lookup table, but the addition or alteration of color entries will not appear until a UIW_STORE_LOOKUP_TABLE is performed.

A plane mask returned by UIW_PLANE_MASK is required by UIW_DISPLAY_BIT_IMAGE, UIW_DISPLAY_CIRCLE, UIW_DISPLAY_IMAGE (uiw_generic_s.a), UIW_DISPLAY_LINE, UIW_DISPLAY_LINES, UIW_DISPLAY_SYMBOL, UIW_DISPLAY_TEXT, UIW_ERASE_PLANES, and UIW_RUBBERBAND_LINE.

A bit image must be converted to a pixmap by UIW_CREATE_PIXMAP (uiw_generic_s.a) before UIW_DISPLAY_BIT_IMAGE can be called. Provided the application does not need the pixmap again the memory can be freed by calling UIW_FREE_PIXMAP.

UIW_FLUSH_BUFFER forces a flushing of the graphics command buffer. This module is called automatically by any X Windows module which returns a value or calls to XPending, XNextEvent, XWindowEvent, or XSync. Three of these, XPending, XNextEvent, and XSync, are called in UWN_INPUT. Which means that UWN_INPUT flushes the graphics command buffer.

To compile this package, the following paths must be established using the "a.path" command:

/eddic/Ada/common
/usr.MC68020/cherokee/VADS55/verdixlib
/usr.MC68020/cherokee/VADS55/standard

To load a program that uses this package, links must be established with the following libraries using the "a.info" command:

/egen/ciw_util/ciw_util.lib
/usr/lib/libX.a
/usr/lib/libX_p.a

Data Bases. None

Environment Variables. None

4.1.7 UTM TACTICAL MAP

The Tactical Map Ada package provides the procedures to display and interact with a digital map and the tactical overlays on top of the map. The Tactical Map software consists of several layered packages with "UTM" being on top. Figure 4-4 shows the Tactical Map package hierarchy.

The four system packages (MAP_SYSTEM, UNIT_SYSTEM, CM_SYSTEM, and OBS_SYSTEM) contain data types and objects that are common for the whole map system. Descriptions of the individual Tactical Map packages follow.

4.1.7.1 CM_SYSTEM

Abstract. Data types and objects for the control measures displayed on the digital map.

Major Capabilities. Type definition and object storage.

4.1.7.1.1 Special Instructions. This package is meant for internal use to the UTM package. Although the objects in this package are visible to the calling application, it is unwise to change the contents of any of the objects.

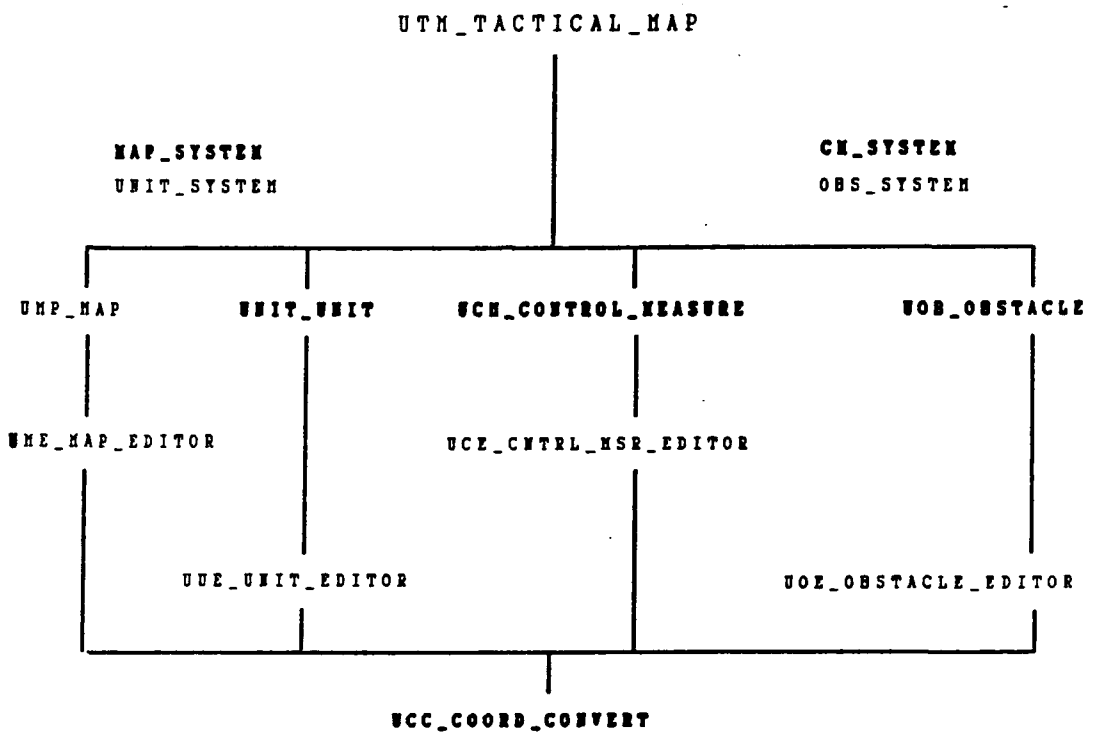


Figure 4-4. Tactical Map Packages

To compile this package, the following paths must be established using the "a.path" command:

/eddic/Ada/common
/usr.MC68020/cherokee/VADS55/verdixlib
/usr.MC68020/cherokee/VADS55/standard

Data Bases. None

Environment Variables. None

4.1.7.2 MAP_SYSTEM

Abstract. Data types and objects for the map objects of the digital map system and object orientated graphics utilities.

Major Capabilities. Type definition and object storage and object control utilities to determine which object was selected on the map and to delete objects displayed on the map.

Special Instructions. This package is meant for internal use to the UTM package. Although the objects in this package are visible to the calling application, it is unwise to change the contents of any of the objects.

The overlays displayed on the map consist of the following types: lines, points, polygons, rectangles, and circles. A list of the displayed objects and a list of popup menus associated with the objects are maintained by this package. Working with the lists is very simple and uses the UED_LIST utilities. The only time a programmer must work with these lists is to implement a new control measure or obstacle, or add a new overlay category to the map system.

To implement a new control measure or obstacle, the new object must be added to the object list when it is displayed. This package contains a package to determine if any part of an object will be displayed in the digital map window.

To add a new map overlay category, such as a new operational planning tool, the following steps must be completed:

1. A new popup menu must be defined for the new object. Normally this would be accomplished by adding a new menu definition procedure to UTM specification.

2. The popup menu description must be added to the menu list in the package.

3. The new object must be added to the object list when it is displayed.

4. Procedures must be added to UTM_PROCESS_INPUT to process the selections on the new popup menu.

To compile this package, the following paths must be established using the "a.path" command:

/eddic/Ada/common
/usr.MC68020/cherokee/VADS55/verdixlib
/usr.MC68020/cherokee/VADS55/standard
/eddic/Ada/ued

Data Bases. None

Environment Variables. None

4.1.7.3 OBS_SYSTEM

Abstract. Data types and objects for the object displayed on the digital map.

Major Capabilities. Type definition and object storage.

Special Instructions. This package is meant for internal use to the UTM package. Although the objects in this package are visible to the calling application, it is unwise to change the contents of any of the objects.

To compile this package, the following paths must be established using the "a.path" command:

/eddic/Ada/common
/usr.MC68020/cherokee/VADS55/verdixlib
/usr.MC68020/cherokee/VADS55/standard

Data Bases. None

Environment Variables. None

4.1.7.4 UNIT_SYSTEM

Abstract. Data types and objects for the BLUEFOR and OPFOR units displayed on the digital map.

Major Capabilities. Type definition and object storage.

Special Instructions. This package is meant for internal use to the UTM package. Although the objects in this package are visible to the calling application, it is unwise to change the contents of any of the objects.

To compile this package, the following paths must be established using the "a.path" command:

/eddic/Ada/common
/usr.MC68020/cherokee/VADS55/verdixlib
/usr.MC68020/cherokee/VADS55/standard

Data Bases. None

Environment Variables. None

4.1.7.5 UCE_CNTRL_MSR_EDITOR

Abstract. Low level control measure utilities for displaying and erasing specific control measure types.

Major Capabilities. This package contains an individual procedure for each control measure type in the system. Each procedure is responsible for the display and erasure of a specific control measure and is the only software that knows exactly what the control measure looks like in the digital map window.

Special Instructions. These procedures are responsible for adding control measure objects to the object list in MAP_SYSTEM.

To compile this package, the following paths must be established using the "a.path" command:

/eddic/Ada/common
/usr.MC68020/cherokee/VADS55/verdixlib
/usr.MC68020/cherokee/VADS55/standard
/eddic/Ada/ued
/eddic/Ada/uwn
/eddic/Ada/uiw

To load a program that uses this package, links must be established with the following libraries using the "a.info" command:

/egen/ciw_util/ciw_util.lib
/egen/cwn_util/cwn.lib
/usr/lib/libXr.a
/usr/lib/libX_p.a
/usr/lib/libX.a
/usr/lib/libm.a

Data Bases. None

Environment Variables. None

4.1.7.6 UCM_CONTROL_MEASURE

Abstract. Intermediate level control measure display package responsible for the defining, displaying, and interacting with control measures on the digital map.

Major Capabilities. This package provides procedures to:

1. Define control measure areas, crossings, fire plans, lines, map features, points, and routes.

2. Selective display and erase of control measures by echelon, type, and color.

3. Delete a control measure from the digital map window.

4. Move a control measure in the digital map window.

5. Interact with the control measure definition and display menus.

6. Redisplay all control measures in the digital map window.

Special Instructions. The control measure initialization procedure UCM_INITIALIZE_CNTRL_MSR must be called as part of the map initialization steps and before other procedures in this package are used.

The normal procedures for defining control measures with multiple points is for the "DEFINE" procedure to display the control measure define menu and accept the first point. The other points are processed by UCE_DEFINE_NEXT_POINT.

To compile this package, the following paths must be established using the "a.path" command:

/eddic/Ada/common
/usr.MC68020/cherokee/VADS55/verdixlib
/usr.MC68020/cherokee/VADS55/standard
/eddic/Ada/uwn
/eddic/Ada/uiw

To load a program that uses this package, links must be established with the following libraries using the "a.info" command:

/egen/ciw_util/ciw_util.lib
/egen/cwn_util/cwn.lib
/usr/lib/libXr.a
/usr/lib/libX_p.a

/usr/lib/libX.a

Data Bases. None

Environment Variables. None

4.1.7.7 UME_MAP_EDITOR

Abstract. Low level digital map utilities for reading and displaying digital map images and displaying grids.

Major Capabilities. The map editor utilities determine the file name to use for a digital map image, open the map image file, determine the map blocks to display, display the map blocks, and close the map image file. They also determine the grid interval and display the grid lines and labels.

Special Instructions. Procedure UME_INIT_MAP_SYSTEM must be called before the other procedures in this package. UME_DEFINE_MAP_COORD should be called whenever the size of the map panel changes or the map scale changes.

To compile this package, the following paths must be established using the "a.path" command:

/eddic/Ada/common
/usr.MC68020/cherokee/VADS55/verdixlib
/usr.MC68020/cherokee/VADS55/standard
/eddic/Ada/uux
/eddic/Ada/ued
/eddic/Ada/uiw

To load a program that uses this package, links must be established with the following libraries using the "a.info" command:

/egen/ciw_util/ciw_util.lib
/egen/cux_util/cux_util.lib
/usr/lib/libXr.a
/usr/lib/libX_p.a
/usr/lib/libX.a
/usr/lib/libm.a

Data Bases

CONTOUR_1TO160	INPUT
CONTOUR_1TO400	INPUT
CONTOUR_1TO80	INPUT
CONTOUR_1TO800	INPUT
CONTOUR_DESC	INPUT
CONTOUR_DESC_1TO160	INPUT
CONTOUR_DESC_1TO400	INPUT
CONTOUR_DESC_1TO80	INPUT
CONTOUR_DESC_1TO800	INPUT
ELEVATION_1TO400	INPUT
ELEVATION_DESC_1TO400	INPUT
ELEV_BAND_1TO160	INPUT
ELEV_BAND_1TO400	INPUT
ELEV_BAND_1TO80	INPUT
ELEV_BAND_1TO800	INPUT
ELEV_BAND_DESC_1TO160	INPUT
ELEV_BAND_DESC_1TO400	INPUT
ELEV_BAND_DESC_1TO80	INPUT
ELEV_BAND_DESC_1TO800	INPUT
MAP_DESC	INPUT
SHAD_RELF_1TO160	INPUT
SHAD_RELF_1TO400	INPUT
SHAD_RELF_1TO80	INPUT
SHAD_RELF_1TO800	INPUT
SHAD_RELF_DESC_1TO160	INPUT
SHAD_RELF_DESC_1TO400	INPUT
SHAD_RELF_DESC_1TO80	INPUT
SHAD_RELF_DESC_1TO800	INPUT
VEGETATION_1TO160	INPUT
VEGETATION_1TO400	INPUT
VEGETATION_1TO80	INPUT
VEGETATION_1TO800	INPUT
VEGETATION_DESC_1TO160	INPUT
VEGETATION_DESC_1TO400	INPUT
VEGETATION_DESC_1TO80	INPUT
VEGETATION_DESC_1TO800	INPUT

Environment Variables. None

4.1.7.8 UMP_MAP

Abstract. Intermediate level digital map utilities for displaying and erasing the digital map and digital map features.

Major Capabilities. The map utilities provide the procedures to display and erase the digital map, contours, and grids. Also included are procedures for highlighting and unhighlighting hydrography, roads, urban areas, and miscellaneous features on the digital map.

Special Instructions. Procedure UMP_INITIALIZE_MAP must be called before the other procedures in this package.

To compile this package, the following paths must be established using the "a.path" command:

/eddic/Ada/common
/usr.MC68020/cherokee/VADS55/verdixlib
/usr.MC68020/cherokee/VADS55/standard
/eddic/Ada/uin
/eddic/Ada/uux
/eddic/Ada/ued
/eddic/Ada/uwn
/eddic/Ada/uiw

To load a program that uses this package, links must be established with the following libraries using the "a.info" command:

/egen/cin_util/cin_util.lib
/egen/ciw_util/ciw_util.lib
/egen/cux_util/cux_util.lib
/egen/cwn_util/cwn.lib
/usr/lib/libXr.a
/usr/lib/libX_p.a
/usr/lib/libX.a
/usr/lib/libm.a

Data Bases. None

Environment Variables.

CHARACTER_FONT_FILE
CONTROL_ROUTER_HOST
CONTROL_ROUTER_SERV
CONTOUR_DESCRIPTION_FILE

ELEV_DESCRIPTION_FILE
MAP_DESCRIPTION_FILE
SYMBOL_FONT_FILE

4.1.7.9 UNT_UNIT

Abstract. Intermediate unit display utilities for displaying and erasing units on the digital map.

Major Capabilities. The unit utilities provide the procedures for displaying, moving, and erasing the BLUEFOR and OPFOR units and displaying the OPFOR unit status report.

Special Instructions. Procedure UNT_INITIALIZE_UNITS must be called before the other procedures in this package.

To compile this package, the following paths must be established using the "a.path" command:

/eddic/Ada/common
/usr.MC68020/cherokee/VADS55/verdixlib
/usr.MC68020/cherokee/VADS55/standard
/eddic/Ada/uin

To load a program that uses this package, links must be established with the following libraries using the "a.info" command:

/egen/cin_util/cin_util.lib
/egen/ciw_util/ciw_util.lib
/egen/cux_util/cux_util.lib
/egen/cwn_util/cwn.lib
/usr/lib/libXr.a
/usr/lib/libX_p.a
/usr/lib/libX.a
/usr/lib/libm.a

Data Bases. None

Environment Variables. None

4.1.7.10 UOB_OBSTACLE

Abstract. Intermediate obstacle display utilities for displaying and erasing obstacles on the digital map.

Major Capabilities. The obstacle utilities provide the procedures for displaying, moving, and erasing obstacles on the digital map.

Special Instructions. Procedure UOB_INITIALIZE_OBSTACLE must be called before the other procedures in this package.

To compile this package, the following paths must be established using the "a.path" command:

/eddic/Ada/common
/usr.MC68020/cherokee/VADS55/verdixlib
/usr.MC68020/cherokee/VADS55/standard

To load a program that uses this package, links must be established with the following libraries using the "a.info" command:

/egen/ciw_util/ciw_util.lib
/egen/cux_util/cux_util.lib
/egen/cwn_util/cwn.lib
/usr/lib/libXr.a
/usr/lib/libX_p.a
/usr/lib/libX.a
/usr/lib/libm.a

Data Bases. None

Environment Variables. None

4.1.7.11 UOE_OBSTACLE_EDITOR

Abstract. Low level obstacle utilities for displaying and erasing specific obstacle types.

Major Capabilities. This package contains an individual procedure for each obstacle type in the system. Each procedure is responsible for the display and erasure of a specific obstacle and is the only software that knows exactly what the obstacle looks like in the digital map window.

Special Instructions. These procedures are responsible for adding obstacle objects to the object list in MAP_SYSTEM.

To compile this package, the following paths must be established using the "a.path" command:

/eddic/Ada/common
/usr.MC68020/cherokee/VADS55/verdixlib
/usr.MC68020/cherokee/VADS55/standard
/eddic/Ada/uwn
/eddic/Ada/uiw

To load a program that uses this package, links must be established with the following libraries using the "a.info" command:

/egen/ciw_util/ciw_util.lib
/egen/cwn_util/cwn.lib
/usr/lib/libXr.a
/usr/lib/libX_p.a
/usr/lib/libX.a
/usr/lib/libm.a

Data Bases. None

Environment Variables. None

4.1.7.12 UTM_TACTICAL_MAP

Abstract. Top level digital map utilities for displaying the digital map and overlays. Overlays consist of units, control measures, and obstacles.

Major Capabilities. The tactical map utilities provide the interface between the digital map system and the application program. This is the only package an application should require to use the complete capabilities of the tactical map system. The tactical map system consists of map, unit, control measure, and obstacle utilities.

The map utilities include procedures to display the digital map background, resize the map panel, erase the overlays, and delete the map panel. The utilities for units, control measures, and obstacles include procedures to display the overlay, change the overlay, and attach a popup menu to the overlay.

Special Instructions. Procedure UTM_DEFINE_MAP_PANEL must be called before the other procedures in this package.

Procedure UTM_DEFINE_OPLAN must be called whenever the overlay data's date-time group or OPLAN id changes.

To allow interaction with the tactical map, procedure UTM_INPUT must be used for all input instead of UWN_INPUT. UTM_INPUT passes all input, that is not part of the map panel,

to the calling process. It also passes the following map updates back to the calling process: unit location change, control measure location change, new control measure, obstacle location change, and new obstacle.

After deleting the map panel, procedure UTM_DELETE_MAP_MENUS should be called to deallocate the memory used for the map walking menus.

To compile this package, the following paths must be established using the "a.path" command:

/eddic/Ada/common
/usr.MC68020/cherokee/VADS55/verdixlib
/usr.MC68020/cherokee/VADS55/standard
/eddic/Ada/uin
/eddic/Ada/uux
/eddic/Ada/ued
/eddic/Ada/uwn
/eddic/Ada/uiw

To load a program that uses this package, links must be established with the following libraries using the "a.info" command:

/egen/cin_util/cin_util.lib
/egen/ciw_util/ciw_util.lib
/egen/cux_util/cux_util.lib
/egen/cwn_util/cwn.lib
/usr/lib/libXr.a
/usr/lib/libX_p.a
/usr/lib/libX.a
/usr/lib/libm.a

Data Bases. None

Environment Variables. RECORD_MAP_INTERACTION

4.1.7.13 UUE_STATUS_REPORT

Abstract. Displays the graphical unit summary and detail status reports.

Major Capabilities. The status report package provides procedures to display detail and summary reports. The detail report presents unit strength with a "Mercedes-type" chart surrounded by the actual counts and percentages corresponding to each section of the

"Mercedes-type" chart. The summary report presents the percentage strength for a unit and its subordinates.

Special Instructions. Procedure UUE_DEFINE_STATUS_PIXMAP must be called before using the other procedures in this package.

This package uses Ada tasking to allow the display of multiple status windows at the same time. There are two tasks in this package, the detail status task and the summary status task. Both tasks have the same entry points, as follows:

INITIALIZE - Creates the popup window and displays the report.

PROCESS_INPUT - Processes all input for the status popup window. The application is responsible for determining if the input belongs to the status window and for calling this procedure to process it.

TERMINATE_TASK - Deletes the status report window and deallocates memory allocated for status structures. This procedure should be called when the application program wants to terminate the status report rather than having the user terminate it from the popup menu.

To compile this package, the following paths must be established using the "a.path" command:

/eddic/Ada/common
/usr.MC68020/cherokee/VADS55/verdixlib
/usr.MC68020/cherokee/VADS55/standard
/eddic/Ada/uin
/eddic/Ada/uux
/eddic/Ada/ued
/eddic/Ada/uwn
/eddic/Ada/uiw

To load a program that uses this package, links must be established with the following libraries using the "a.info" command:

/egen/cin_util/cin_util.lib
/egen/ciw_util/ciw_util.lib
/egen/cux_util/cux_util.lib
/egen/cwn_util/cwn.lib
/usr/lib/libXr.a
/usr/lib/libX_p.a
/usr/lib/libX.a
/usr/lib/libm.a

Data Bases. None

Environment Variables. CHARACTER_FONT_FILE

4.1.7.14 UUE_UNIT_EDITOR

Abstract. Low-level unit display utilities to display the unit symbol, echelon, name and status.

Major Capabilities. The unit editor package contains procedures to display and erase unit symbols, echelon symbols, names, and unit status report. The unit symbol procedure is the only software that knows exactly what the unit looks like in the digital map window.

Special Instructions. These procedures are responsible for adding unit objects to the object list in MAP_SYSTEM.

To compile this package, the following paths must be established using the "a.path" command:

/eddic/Ada/common
/usr.MC68020/cherokee/VADS55/verdixlib
/usr.MC68020/cherokee/VADS55/standard
/eddic/Ada/uux
/eddic/Ada/ued
/eddic/Ada/uwn
/eddic/Ada/uiw

To load a program that uses this package, links must be established with the following libraries using the "a.info" command:

/egen/ciw_util/ciw_util.lib
/egen/cux_util/cux_util.lib
/egen/cwn_util/cwn.lib
/usr/lib/libXr.a
/usr/lib/libX_p.a
/usr/lib/libX.a
/usr/lib/libm.a

Data Bases. None

Environment Variables. None

4.1.7.15 UCC_COORD_CONVERT

Abstract. General purpose coordinate conversion utilities.

Major Capabilities. The coordinate conversion package contains procedures to perform the following conversions.

World Coordinate	to	Military Grid
World Coordinate	to	Pixel
Military Grid	to	World Coordinate
Pixel	to	World Coordinate

Special Instructions. Procedure UCC_DEFINE_MAP_AREA must be called whenever the size of the digitized map area changes and UCC_DEFINE_MAP_DISPLAY must be called whenever the map scale or map window size changes.

Currently, this package will only work with the Central Germany digitized area. Future enhancements should be made to allow this package to work for any digitized area in the world.

To compile this package, the following paths must be established using the "a.path" command:

/eddic/Ada/common
/usr.MC68020/cherokee/VADS55/verdixlib
/usr.MC68020/cherokee/VADS55/standard

Data Bases. None

Environment Variables. None

4.1.8 UUX Unix Utilities

The Unix Utilities consist of procedures to communicate with the Unix operating system. The input and output (I/O) utilities are separated into a generic package (UUX_IO) to allow binary I/O of all types of buffers. The other Unix utilities are located in the UUX_UTIL package.

4.1.8.1 UUX_IO

Abstract. UUX_IO is an acronym for a set of utility input and output primitives, which allow programs access to low level input and output.

Major Capabilities. UUX_IO is a stand alone utility package (not a process) which does not require the fileserver routers and/or data base managers to operate. This utility package provides a means for programmers to perform very rudimentary input and output functions, which some high level languages do not permit.

Special Instructions. The application calling this package must define a data structure type for the input/output data, then instantiate this package with that structure type.

All files must be opened with UUX_OPEN_FILE before they can be read (UUX_BINARY_READ) or written to (UUX_BINARY_WRITE). Once a file is opened it must be explicitly closed with UUX_CLOSE_FILE; do not expect the system to close the file at process termination, because it won't.

To compile this package, the following paths must be established using the "a.path" command:

/eddic/Ada/common
/usr.MC68020/cherokee/VADS55/verdixlib
/usr.MC68020/cherokee/VADS55/standard

To load a program that uses this package, links must be established with the following libraries using the "a.info" command:

/egen/cux_util/cux_util.lib

Data Bases. None explicitly, all implicitly. Procedures uux_open_file, uux_close_file, uux_binary_read, and uux_binary_write will work with any data base.

Environment Variables. None

4.1.8.2 UUX_UTIL

Abstract. UUX_UTIL is an acronym for a set of utility primitives, which allow programs to access Unix operating commands.

Major Capabilities. UUX_UTIL is a stand alone utility package (not a process) which does not require the fileserver routers and/or data base managers to operate. This utility package provides a means for programmers to perform certain Unix operating system commands which many high level languages do not permit. There is not a one to one pairing of modules to Unix commands; only those commands required by the Tactical Planning Workstation have been developed.

Special Instructions. UUX_SYSTEM will not return any data, so query type commands must redirect their output to a file.

To compile this package, the following paths must be established using the "a.path" command:

/eddic/Ada/common
/usr.MC68020/cherokee/VADS55/verdixlib
/usr.MC68020/cherokee/VADS55/standard

To load a program that uses this package, links must be established with the following libraries using the "a.info" command:

/egen/cux_util/cux_util.lib

Data Bases. None

Environment Variables. None explicitly, all implicitly. Procedure uux_getenv will decipher any environment variable.

4.2 ADA PROGRAMS

The Ada programs are the processes in the Tactical Planning Workstation. The data base managers, routers, and window managers are all examples of Ada programs. Each program source file has "_p.a" suffix. The following major programs are in the system:

CDB - Command and Control (C2) product data base manager

CTL - Experiment control data base manager

FDB - Reference data base manager

HDB - Help data base manager

HLP - Help window display manager

RCN - Experiment control message route

RCP - Command and Control (C2) product message router

RRF - Reference and help message router

RSD - Situation data message router

SCL - Station control manager

SDB - Situation data base manager

WBD - Build window display manager

WCD - Participant experiment control window display manager

WED - Experimenter's experiment control window display manager

WMS - View message window display manager

WTD - Tool window display manager

WVC - View situation window display manager

WVR - View reference window display manager

4.2.1 C2 Product Data Base Manager (CDB)

The Command and Control (C2) data base manager consists of a program to build the C2 data base, a program to control access to the C2 data base, a program to print hardcopy reports of the command and control products, and a specification ("_s.a" suffix) and body ("_b.a" suffix) to generate the reports that require tactical situation data.

4.2.1.1 CDB_C2_PRODUCT_DB_MANAGER

Abstract. Command and Control (C2) product data base manager.

Major Capabilities. The C2 product data base manager maintains the C2 product data base and allows network access to it. C2 products include the products in the view situation, view message, and build windows. A product can be a textual report, a computer generated

report, or a digital map with tactical overlay. This process also maintains the message log and controls the routing of messages from a build window to a view situation window.

This program maintains a list of the number of view message windows that are active on each workstation. When a summary message is received, the summary message is routed to all active view message windows. If a view message window does not exist for a participant that is a recipient of a message, a window creation message is sent to the station control manager to create a view situation window.

Special Instructions. The following processes must be executing before the C2 product data base manager is started:

RCP_C2_PRODUCT_ROUTER
RSD_SITUATION_DATA_ROUTER
SDB_SITUATION_DB_MANAGER

All C2 product requests must be routed through the C2 product router (RCP). The process name that must be used is C2_DB_MANAGER. The following messages are processed by the C2 product data base manager:

Message Requests
- MSG_MENU_TREE
- MSG_TEXT_BUFFER
- MSG_HEADER_BUFFER
- MSG_C2_PRODUCTS
- MSG_C2_PART_LIST
- MSG_MESSAGE_LOG

Other Messages
- MSG_TEXT_BUFFER
- MSG_C2_MESSAGE
- MSG_TERM_WINDOW
- MSG_STOP

The C2 product data base and the message log are initialized by CDB_PRODUCT_BUILD.

To compile this program, the following paths must be established using the "a.path" command:

/eddic/Ada/common
/usr/cherokee/VADS55/verdixlib
/usr/cherokee/VADS55/standard
/eddic/Ada/uux
/eddic/Ada/uin

To load this program, links must be established with the following libraries using the "a.info" command:

/egen/cin_util/cin_util.lib
/egen/cux_util/cux_util.lib

Data Bases

C2_PRODUCT	INPUT/OUTPUT
C2_PRODUCT_DESC	INPUT/OUTPUT
C2_PRODUCT_HEADER	INPUT/OUTPUT
G2_BUILD_MENU	INPUT
G2_VIEW_C2_MENU	INPUT
G3_BUILD_MENU	INPUT
G3_VIEW_C2_MENU	INPUT
G4_BUILD_MENU	INPUT
G4_VIEW_C2_MENU	INPUT
MESSAGE_LOG	INPUT/OUTPUT
SEND_PARTICIPANT_SOURCE	INPUT

Environment Variables

BUILD_EX
BUILD_G2
BUILD_G3
BUILD_G4
C2_PRODUCT_ROUTER_HOST
C2_PRODUCT_ROUTER_SERV
CDB_HEADER_DB
CDB_PARTICIPANT_DB
CDB_PROD_DESC_DB
CDB_PRODUCT_DB
CONTROL_ROUTER_HOST
CONTROL_ROUTER_SERV
MESSAGE_DISPLAY_MANAGER
MESSAGE_LOG_DB
SITUATION_ROUTER_SERV
SITUATION_ROUTER_HOST

START_DATE
VIEW_EX
VIEW_G2
VIEW_G3
VIEW_G4

4.2.1.2 CDB_GENERATE_PRODUCT

Abstract. Generates the Command and Control (C2) reports that contain tactical situation data.

Major Capabilities. The generate product package formats the following reports into a ASCII buffer:

BLUEFOR Ammunition
BLUEFOR Equipment
BLUEFOR Fuel
BLUEFOR Personnel
BLUEFOR Task Organization
OPFOR Committed Forces
OPFOR Equipment
OPFOR Reinforcing Units
OPFOR Task Organization

Special Instructions. The calling application must connect to the situation data router before using the procedures in this package.

To compile this package, the following paths must be established using the "a.path" command:

/eddic/Ada/common
/usr/cherokee/VADS55/verdixlib
/usr/cherokee/VADS55/standard
/eddic/Ada/uin
/eddic/Ada/ued

To load a program that uses this package, links must be established with the following libraries using the "a.info" command:

/egen/cin_util/cin_util.lib
/egen/cux_util/cux_util.lib
/usr/lib/libm.a

Data Bases. None

Environment Variables. None

4.2.1.3 CDB_HARDCOPY

Abstract. Creates an ASCII print file of C2 or reference products.

Major Capabilities. This program provides the capability to print selective reports from the C2 product or reference data base.

Special Instructions. To print C2 products, the following processes must be executing before starting this process:

RCP_C2_PRODUCT_ROUTER
RSD_SITUATION_DATA_ROUTER
SDB_SITUATION_DB_MANAGER
CDB_C2_PRODUCT_DB_MANAGER

Environment variable "db_manager" must be set to C2_DB_MANAGER and "view_G2" must be set to the file that contains the list of products to print. The file is the same format as G2_VIEW_C2_MENU.

To print reference products, the following processes must be executing before starting this process:

RCP_REFERENCE_ROUTER
FDB_REFERENCE_DB_MANAGER

Environment variable "db_manager" must be set to REFERENCE_DB_MANAGER and "ref_view_one" must be set to the file that contains the list of products to print. The file is the same format as G2_REFERENCE_MENU.

To compile this program, the following paths must be established using the "a.path" command:

/eddic/Ada/common
/usr/cherokee/VADS55/verdixlib
/usr/cherokee/VADS55/standard
/eddic/Ada/uux
/eddic/Ada/uin
/eddic/Ada/uwn

To load this program, links must be established with the following libraries using the "a.info" command:

/egen/cin_util/cin_util.lib
/egen/cux_util/cux_util.lib
/usr/lib/libm.a

Data Bases

PRODUCT_HARDCOPY

Environment Variables

DB_MANAGER
REPORT_OUTPUT
ROUTER_HOST
ROUTER_SERV

4.2.1.4 CDB_PRODUCT_BUILD

Abstract. Builds the C2 product data base and initializes the message log.

Major Capabilities. Creates the C2 product data base from the C2 product source data base. It also creates the menu description files for the product selection walking menus in the view situation and build windows.

Special Instructions. This program must be run after changes have been made to the C2 product source file.

To compile this program, the following paths must be established using the "a.path" command:

/eddic/Ada/common
/usr/cherokee/VADS55/verdixlib
/usr/cherokee/VADS55/standard
/eddic/Ada/uux
/eddic/Ada/ued

To load this program, links must be established with the following libraries using the "a.info" command:

/egen/cux_util/cux_util.lib
/usr/lib/libm.a

Data Bases

BLUEFOR_UNIT_CONVERT	INPUT
C2_PRODUCT	OUTPUT
C2_PRODUCT_DESC	OUTPUT
C2_PRODUCT_HEADER	OUTPUT
C2_PRODUCT_NAME	OUTPUT
C2_PRODUCT_SOURCE	INPUT
G2_BUILD_MENU	OUTPUT
G2_VIEW_C2_MENU	OUTPUT
G3_BUILD_MENU	OUTPUT
G3_VIEW_C2_MENU	OUTPUT
G4_BUILD_MENU	OUTPUT
G4_VIEW_C2_MENU	OUTPUT
MESSAGE_LOG	OUTPUT
OPFOR_UNIT_CONVERT	INPUT

Environment Variables

BLUEFOR_UNIT_CONVERSION
BUILD_ONE
BUILD_THREE
BUILD_TWO
C2LAB_DB
HEADER_DB
MESSAGE_LOG_DB
OPFOR_UNIT_CONVERSION
PRODUCT_DB
PRODUCT_DESC_DB
PRODUCT_XREF
VIEW_ONE
VIEW_THREE
VIEW_TWO

4.2.2 Experiment Control Product Data Base Manager (CTL)

The Experiment Control data base manager consists of a program to build the experiment control data base, a program to control access to it, and a program to stop the system.

4.2.2.1 CTL_EXPERIMENT_CONTROL

Abstract. Experiment control product data base manager.

Major Capabilities. The experiment control product data base manager maintains the experiment control product data base and allows network access to it. The experiment control products appear in the experimenter's experiment control window, and in the participants experiment control window when the experimenter sends a control message to a participant. A product can be a textual report, a computer generated report, or a digital map with tactical overlay. This program also controls the creation of the participant experiment control window. The following rules are used for the creation of experiment control windows:

If the experiment control message requires a response, a window creation message is sent to the station control manager to create a new experiment control window. If the message is informative and doesn't require a response, the message is sent to an existing informative experiment control window if one exists. If one does not exist, a window creation message is sent the station control manager to create it.

Special Instructions. The following processes must be executing before the experiment control product data base manager is started:

RCN_CONTROL_ROUTER
RSD_SITUATION_DATA_ROUTER
SDB_SITUATION_DB_MANAGER

All experiment control product requests must be routed through the experiment control router (RCN). The process name that must be used is CONTROL_MANAGER. The following messages are processed by the experiment control product data base manager:

Message Requests

- MSG_MENU_TREE
- MSG_TEXT_BUFFER
- MSG_HEADER_BUFFER
- MSG_CONTROL_PRODUCTS
- MSG_CONTROL_PART_LIST

Other Messages

- MSG_TEXT_BUFFER
- MSG_CONTROL_ROUTING
- MSG_STATION_UP
- MSG_TERM_WINDOW
- MSG_STOP

The experiment control product data base is initialized by CONTROL_PRODUCT_BUILD.

To compile this program, the following paths must be established using the "a.path" command:

/eddic/Ada/common
/usr/cherokee/VADS55/verdixlib
/usr/cherokee/VADS55/standard
/eddic/Ada/uux
/eddic/Ada/uin

To load this program, links must be established with the following libraries using the "a.info" command:

/egen/cin_util/cin_util.lib
/egen/cux_util/cux_util.lib
/usr/lib/libm.a

Data Bases

EXP_CONTROL_MENU	INPUT
EXP_CONTROL_PARTICIPANT	INPUT
EXP_CONTROL_PRODUCT	INPUT/OUTPUT
EXP_CONTROL_PROD_DESC	INPUT/OUTPUT

Environment Variables

C2_PRODUCT_ROUTER_HOST
C2_PRODUCT_ROUTER_SERV
CONTROL_DISPLAY_MANAGER
CONTROL_MENU
CONTROL_ROUTER_HOST
CONTROL_ROUTER_SERV
CTL_PARTICIPANT_DB
CTL_PROD_DESC_DB
CTL_PRODUCT_DB
START_DATE

4.2.2.2 CTL_PRODUCT_BUILD

Abstract. Builds the experiment control product data base and the experiment control product selection walking menu file.

Major Capabilities. Creates the experiment control product data base from the experiment control source data base. It also creates the menu description file for the product selection walking menu in the experimenter's experiment control window.

Special Instructions. This program must be executed after changes have been made to the experiment control product source file.

To compile this program, the following paths must be established using the "a.path" command:

/eddic/Ada/common
/usr/cherokee/VADS55/verdixlib
/usr/cherokee/VADS55/standard
/eddic/Ada/uux
/eddic/Ada/ued

To load this program, links must be established with the following libraries using the "a.info" command:

/egen/cux_util/cux_util.lib
/usr/lib/libm.a

Data Bases

EXP_CONTROL_MENU	OUTPUT
EXP_CONTROL_NAME	OUTPUT
EXP_CONTROL_PRODUCT	OUTPUT
EXP_CONTROL_PROD_DESC	OUTPUT
EXP_CONTROL_SOURCE	INPUT

Environment Variables

CONTROL_DB
CONTROL_MENU
PRODUCT_DB
PRODUCT_DESC_DB
PRODUCT_XREF

4.2.2.3 STOP_EDDIC

Abstract. Stops the Tactical Planning Workstation system.

Major Capabilities. Sends a stop (MSG_STOP) message to all the routers. In turn, the routers forward the message to all processes connected to them.

Special Instructions. This program must be executed in the file server computer. It connects to the following routers:

C2_PRODUCT_ROUTER
CONTROL_ROUTER
REFERENCE_ROUTER
SITUATION_ROUTER

To compile this program, the following paths must be established using the "a.path" command:

/eddic/Ada/common
/usr/cherokee/VADS55/verdixlib
/usr/cherokee/VADS55/standard
/eddic/Ada/uin

To load this program, links must be established with the following libraries using the "a.info" command:

/egen/cin_util/cin_util.lib

Data Bases. None

Environment Variables

C2_PRODUCT_ROUTER_HOST
C2_PRODUCT_ROUTER_SERV
CONTROL_ROUTER_HOST
CONTROL_ROUTER_SERV
REFERENCE_ROUTER_SERV
REFERENCE_ROUTER_HOST
SITUATION_ROUTER_SERV
SITUATION_ROUTER_HOST

4.2.3 Reference Product Data Base Manager (FDB)

The reference data base manager consists of a program to build the reference data base, and a program to control access to it.

4.2.3.1 FDB_REFERENCE_DB_BUILD

Abstract. Builds the reference product data base and reference product selection walking menu file.

Major Capabilities. Creates the reference product data base from the reference source data base. It also creates the menu description file for the product selection walking menu in the view reference window.

Special Instructions. This program must be run after changes have been made to the reference product source file.

To compile this program, the following paths must be established using the "a.path" command:

/eddic/Ada/common
/usr/cherokee/VADS55/verdixlib
/usr/cherokee/VADS55/standard
/eddic/Ada/ued
/eddic/Ada/uux

To load this program, links must be established with the following libraries using the "a.info" command:

/egen/cux_util/cux_util.lib
/usr/lib/libm.a

Data Bases

G2_REFERENCE_MENU	OUTPUT
G3_REFERENCE_MENU	OUTPUT
G4_REFERENCE_MENU	OUTPUT
REFERENCE_DB	OUTPUT
REFERENCE_HEADER	OUTPUT
REFERENCE_NAME	OUTPUT
REFERENCE_PROD_DESC	OUTPUT
REFERENCE_SOURCE	INPUT

Environment Variables

C2LAB_DB
HEADER_DB
PRODUCT_DB
PRODUCT_DESC_DB

PRODUCT_XREF
VIEW_ONE
VIEW_THREE
VIEW_TWO

4.2.3.2 FDB_REFERENCE_DB_MANAGER

Abstract. Reference product data base manager.

Major Capabilities. The reference product data base manager maintains the reference data base and allows network access to it.

Special Instructions. The following process must be executing before the reference product data base manager is started:

RRF_REFERENCE_ROUTER

All reference product requests must be routed through the reference router (RRF). The process name that must be used is REFERENCE_DB_MANAGER. The following messages are processed by the reference product data base manager:

Message Requests
- MSG_MENU_TREE
- MSG_TEXT_BUFFER
- MSG_HEADER_BUFFER
- MSG_REFERENCE_PRODUCTS

Other Messages
- MSG_STOP

The reference product data base is initialized by FDB_REFERENCE_DB_BUILD.

To compile this program, the following paths must be established using the "a.path" command:

/eddic/Ada/common
/usr/cherokee/VADS55/verdixlib
/usr/cherokee/VADS55/standard
/eddic/Ada/uux
/eddic/Ada/uin

To load this program, links must be established with the following libraries using the "a.info" command:

/egen/cin_util/cin_util.lib
/egen/cux_util/cux_util.lib

Data Bases

G2_REFERENCE_MENU	INPUT
G3_REFERENCE_MENU	INPUT
G4_REFERENCE_MENU	INPUT
REFERENCE_DB	INPUT
REFERENCE_HEADER	INPUT
REFERENCE_PROD_DESC	INPUT

Environment Variables

FDB_HEADER_DB
FDB_PROD_DESC_DB
FDB_PRODUCT_DB
REF_VIEW_EX
REF_VIEW_ONE
REF_VIEW_THREE
REF_VIEW_TWO
REFERENCE_ROUTER_SERV
REFERENCE_ROUTER_HOST

4.2.4 Help Product Data Base Manager (HDB)

The help data base manager consists of a program to build the help data base, and a program to control access to it.

4.2.4.1 HDB_HELP_DB_BUILD

Abstract. Builds the help product data base and help product selection walking menu file.

Major Capabilities. Creates the help product data base from the help source data base. It also creates the menu description file for the product selection walking menu on the help button.

Special Instructions. This program must be run after changes have been made to the help product source file.

To compile this program, the following paths must be established using the "a.path" command:

/eddic/Ada/common
/usr/cherokee/VADS55/verdixlib
/usr/cherokee/VADS55/standard
/eddic/Ada/ued
/eddic/Ada/uux

To load this program, links must be established with the following libraries using the "a.info" command:

/egen/cux_util/cux_util.lib
/usr/lib/libm.a

Data Bases

HELP_MENU	OUTPUT
HELP_NAME	OUTPUT
HELP_PROD_DESC	OUTPUT
HELP_PRODUCT	OUTPUT
HELP_SOURCE	INPUT

Environment Variables

HELP_MENU_FILE
HELP_SOURCE
PRODUCT_DB
PRODUCT_DESC_DB
PRODUCT_XREF

4.2.4.2 HDB_HELP_DB_MANAGER

Abstract. Help product data base manager.

Major Capabilities. The help product data base manager maintains the help data base and allows network access to it. Help products include textual reports and menu tree layouts.

Special Instructions. The following process must be executing before the help product data base manager is started:

RRF_REFERENCE_ROUTER

All help product requests must be routed through the reference router (RRF). The process name that must be used is HELP_DB_MANAGER. The following messages are processed by the reference product data base manager:

Message Requests

MSG_MENU_TREE
MSG_TEXT_BUFFER
MSG_HEADER_BUFFER
MSG_HELP_PRODUCTS

Other Messages

MSG_STOP

The help product data base is initialized by HDB_HELP_DB_BUILD.

To compile this program, the following paths must be established using the "a.path" command:

/eddic/Ada/common
/usr/cherokee/VADS55/verdixlib
/usr/cherokee/VADS55/standard
/eddic/Ada/uux
/eddic/Ada/uin

To load this program, links must be established with the following libraries using the "a.info" command:

/egen/cin_util/cin_util.lib
/egen/cux_util/cux_util.lib

Data Bases

HELP_MENU	INPUT
HELP_PROD_DESC	INPUT
HELP_PRODUCT	INPUT

Environment Variables

HDB_HELP_DESC_DB
HDB_HELP_TEXT_DB
HELP_MENU
REFERENCE_ROUTER_HOST
REFERENCE_ROUTER_SERV

4.2.5 Help Display Manager (HLP)

The help display manager consists of a program to interact with the help button and an Ada task to display the textual help windows. The walking menu layout help windows are displayed by the DML task (Part of the UWN library).

4.2.5.1 HLP_HELP_DISPLAY_MANAGER

Abstract. Help button and window display manager.

Major Capabilities. The help display manager controls the interaction with the help button and uses either the help report or menu layout Ada task to display the appropriate help window. Because Ada tasks are used to display the help windows, an unlimited (within reason) number of help windows can be displayed at the same time.

Special Instructions. The help display manager maintains a link-list of Ada help tasks and their associated window id. When input is received from the UWN system, the window id is used to determine which task to pass the input to. When a help window is terminated, the task is deleted from the link-list.

The following programs must be executing before this program is started:

RRF_REFERENCE_ROUTER
HDB_HELP_DB_MANAGER

To compile this program, the following paths must be established using the "a.path" command:

/eddic/Ada/common
/usr/cherokee/VADS55/verdixlib
/usr/cherokee/VADS55/standard
/eddic/Ada/uin
/eddic/Ada/uwn
/eddic/Ada/ued
/eddic/Ada/uux
/eddic/Ada/uiw

To load this program, links must be established with the following libraries using the "a.info" command:

/egen/cin_util/cin_util.lib
/egen/cux_util/cux_util.lib
/egen/cwn_util/cwn.lib

/egen/ciw_util/ciw_util.lib
/usr/lib/libXr.a
/usr/lib/libX_p.a
/usr/lib/libX.a
/usr/lib/libm.a

Data Bases. None

Environment Variables

CHARACTER_FONT_FILE
EDDIC_STATION_USER
REFERENCE_ROUTER_SERV
REFERENCE_ROUTER_HOST

4.2.5.2 HLP_HELP_REPORT

Abstract. Textual help window control task.

Major Capabilities. Displays a textual help report window and processes all input for the window. The calling process is responsible for passing all user input to the task via the PROCESS_INPUT entry point.

Special Instructions. To compile this package, the following paths must be established using the "a.path" command:

/eddic/Ada/common
/usr/cherokee/VADS55/verdixlib
/usr/cherokee/VADS55/standard
/eddic/Ada/uwn
/eddic/Ada/ued
/eddic/Ada/uiw

To load a program that uses this package, links must be established with the following libraries using the "a.info" command:

/egen/cux_util/cux_util.lib
/egen/cwn_util/cwn.lib
/egen/ciw_util/ciw_util.lib
/usr/lib/libXr.a
/usr/lib/libX_p.a
/usr/lib/libX.a
/usr/lib/libm.a

Data Bases. None

Environment Variables. None

4.2.6 Experiment Control Message Router (RCN)

The experiment control router consists of a program to route experiment control messages among the experiment control processes and a program to convert the recorded messages to ASCII format.

4.2.6.1 RCN_CONTROL_ROUTER

Abstract. Experiment control message router.

Major Capabilities. Routes and records experiment control messages among processes that are connected to the router. Experiment control messages include experiment control products, product requests, lookup table updates, and window operations.

Special Instructions. The normal sequence of operations to use a router is to connect to it and then send and receive messages through it. The connect is accomplished by calling UIN_CLIENT_CONNECT_SERVER and then sending the process id to the router via the connect message (MSG_CONNECT). Messages are sent through the router by UIN_SEND_MSG and received by UIN_RECV_MSG. There are two ways to determine if a message is being sent to a process. The first way is to use UIN_RECV_MSG with the no-peek flag. The process will suspend operation at this statement until a message is received. The other way is to use UWN_ADD_INPUT_SOCKET to tell the UWN system to watch for input from a socket number. When a message is received, UWN_INPUT returns with a data type of SYS_INPUT_MESSAGE. This method allows a process to handle both window and message inputs.

Before terminating a process that is connected to a router, a close socket message (MSG_CLOSE_SOCKET) must be sent to the router.

Environment variable RECORD_SESSION must be set to true to record routed messages. Only messages identified in MSG_EC_RECORD_LIST will be recorded. The recorded message data base is initialized when the router is started.

To compile this program, the following paths must be established using the "a.path" command:

/eddic/Ada/common
/usr/cherokee/VADS55/verdixlib

/usr/cherokee/VADS55/standard
/eddic/Ada/uux
/eddic/Ada/uin

To load this program, links must be established with the following libraries using the "a.info" command:

/egen/cin_util/cin_util.lib
/egen/cux_util/cux_util.lib

Data Bases

EXP_CONTROL_RECORD	OUTPUT

Environment Variables

CONTROL_RECORD_DB
CONTROL_ROUTER_HOST
CONTROL_ROUTER_SERV
RECORD_SESSION

4.2.6.2 RCN_RECORD_TO_ASCII

Abstract. Converts the recorded experiment control data to ASCII.

Major Capabilities. Reads the experiment control recorded data base and creates an individual ASCII file for each record type in the recorded data base.

Special Instructions. The system must be stopped (see STOP_EDDIC in CTL) before running this program. The recording data base is initialized whenever the router is started.

To compile this program, the following paths must be established using the "a.path" command:

/eddic/Ada/common
/usr/cherokee/VADS55/verdixlib
/usr/cherokee/VADS55/standard
/eddic/Ada/uux
/eddic/Ada/ued

To load this program, links must be established with the following libraries using the "a.info" command:

/egen/cux_util/cux_util.lib
/usr/lib/libm.a

Data Bases

EXP_CONTROL_RECORD	INPUT
TRAN_CONTROL_REQUEST	OUTPUT
TRAN_CONTROL_WINDOW	OUTPUT
TRAN_LOOKUP_TABLE	OUTPUT
TRAN_MAP	OUTPUT

Environment Variables

CONTROL_RECORD
CONTROL_REQUEST
CONTROL_WINDOW
LUT_UPDATE
MAP_STATUS

4.2.7 C2 Product Message Router (RCP)

The Command and Control (C2) router consists of a program to route C2 product messages between the processes that require C2 product data, and a program to convert the recorded messages to ASCII format.

4.2.7.1 RCP_C2_PRODUCT_ROUTER

Abstract. Command and Control (C2) product message router.

Major Capabilities. Routes and records C2 product messages between processes that are connected to the router. C2 product messages include C2 products, product requests, summary messages, message logs, and window operations.

Special Instructions. The normal sequence of operations to use a router is to connect to it and then send and receive messages through it. The connection is accomplished by calling UIN_CLIENT_CONNECT_SERVER and then sending the process id to the router via the connect message (MSG_CONNECT). Messages are sent through the router by UIN_SEND_MSG and received by UIN_RECV_MSG. There are two ways to determine if a message is being sent to a process. The first way is to use UIN_RECV_MSG with the no-peek flag. The process will suspend operation at this statement until a message is received. The other way is to use UWN_ADD_INPUT_SOCKET to tell the UWN system to watch for input from a socket number. When a message is received, UWN_INPUT returns with a data type of

SYS_INPUT_MESSAGE. This method allows a process to process both window and message inputs.

Before terminating a process that is connected to a router, a close socket message (MSG_CLOSE_SOCKET) must be sent to the router.

Environment variable RECORD_SESSION must be set to true to record routed messages. Only messages identified in MSG_C2_RECORD_LIST will be recorded. The recorded message data base is initialized when the router is started.

To compile this program, the following paths must be established using the "a.path" command:

/eddic/Ada/common
/usr/cherokee/VADS55/verdixlib
/usr/cherokee/VADS55/standard
/eddic/Ada/uux
/eddic/Ada/uin

To load this program, links must be established with the following libraries using the "a.info" command:

/egen/cin_util/cin_util.lib
/egen/cux_util/cux_util.lib

Data Bases

C2_PRODUCT_RECORD	OUTPUT

Environment Variables

C2_PRODUCT_RECORD_DB
C2_PRODUCT_ROUTER_HOST
C2_PRODUCT_ROUTER_SERV
RECORD_SESSION

4.2.7.2 RCP_RECORD_TO_ASCII

Abstract. Converts the recorded C2 product data to ASCII.

Major Capabilities. Reads the C2 product recorded data base and creates an individual ASCII file for each record type in the recorded data base.

Special Instructions. The system must be stopped (see STOP_EDDIC in CTL) before running this program. The recording data base is initialized whenever the router is started.

To compile this program, the following paths must be established using the "a.path" command:

/eddic/Ada/common
/usr/cherokee/VADS55/verdixlib
/usr/cherokee/VADS55/standard
/eddic/Ada/uux
/eddic/Ada/ued

To load this program, links must be established with the following libraries using the "a.info" command:

/egen/cux_util/cux_util.lib
/usr/lib/libm.a

Data Bases

C2_PRODUCT_RECORD	INPUT
TRAN_C2_REQUEST	OUTPUT
TRAN_C2_WINDOW	OUTPUT
TRAN_NEW_C2	OUTPUT

Environment Variables

C2_NEW_PROD
C2_RECORD
C2_REQUEST
C2_WINDOW

4.2.8 Reference Message Router (RRF)

The reference router consists of a program to route reference and help product messages between the processes that require reference or help data, and a program to convert the recorded messages to ASCII format.

4.2.8.1 RRF_REFERENCE_ROUTER

Abstract. Reference and help product message router.

Major Capabilities. Routes and records reference and help messages between processes that are connected to the router. Reference and help messages include reference and help products, product requests, and window operations.

Special Instructions. The normal sequence of operations to use a router is to connect to it and then send and receive messages through it. The connection is accomplished by calling UIN_CLIENT_CONNECT_SERVER and then sending the process id to the router via the connect message (MSG_CONNECT). Messages are sent through the router by UIN_SEND_MSG and received by UIN_RECV_MSG. There are two ways to determine if a message is being sent to a process. The first way is to use UIN_RECV_MSG with the no-peek flag. The process will suspend operation at this statement until a message is received. The other way is to use UWN_ADD_INPUT_SOCKET to tell the UWN system to watch for input from a socket number. When a message is received, UWN_INPUT returns with a data type of SYS_INPUT_MESSAGE. This method allows a process to process both window and message inputs.

Before terminating a process that is connected to a router, a close socket message (MSG_CLOSE_SOCKET) must be sent to the router.

Environment variable RECORD_SESSION must be set to true to record routed messages. Only messages identified in MSG_RF_RECORD_LIST will be recorded. The recorded message data base is initialized when the router is started.

To compile this package, the following paths must be established using the "a.path" command:

/eddic/Ada/common
/usr/cherokee/VADS55/verdixlib
/usr/cherokee/VADS55/standard
/eddic/Ada/uux
/eddic/Ada/uin

To load a program that uses this package, links must be established with the following libraries using the "a.info" command:

/egen/cin_util/cin_util.lib
/egen/cux_util/cux_util.lib

Data Bases

REFERENCE_RECORD	OUTPUT

Environment Variables

RECORD_SESSION
REFERENCE_RECORD_DB
REFERENCE_ROUTER_SERV
REFERENCE_ROUTER_HOST

4.2.8.2 RRF_RECORD_TO_ASCII

Abstract. Converts the recorded reference data to ASCII.

Major Capabilities. Reads the reference recorded data base and creates an individual ASCII file for each record type in the recorded data base.

Special Instructions. The system must be stopped (see STOP_EDDIC in CTL) before running this program. The recording data base is initialized whenever the router is started.

To compile this package, the following paths must be established using the "a.path" command:

/eddic/Ada/common
/usr/cherokee/VADS55/verdixlib
/usr/cherokee/VADS55/standard
/eddic/Ada/uux
/eddic/Ada/ued

To load a program that uses this package, links must be established with the following libraries using the "a.info" command:

/egen/cux_util/cux_util.lib
/usr/lib/libm.a

Data Bases

REFERENCE_RECORD	INPUT
TRAN_REF_REQUEST	OUTPUT
TRAN_REF_WINDOW	OUTPUT

Environment Variables

REF_RECORD
REF_REQUEST
REF_WINDOW

4.2.9 Situation Data Message Router (RSD)

The situation data router consists of a program to route situation data messages between the processes that require situation data, and a program to convert the recorded messages to ASCII format.

4.2.9.1 RSD_SITUATION_DATA_ROUTER

Abstract. Tactical situation data message router.

Major Capabilities. Routes and records situation data messages between processes that are connected to the router. Situation data messages include unit status, control measures, obstacles, and situation data updates.

Special Instructions. The normal sequence of operations to use a router is to connect to it and then send and receive messages through it. The connection is accomplished by calling UIN_CLIENT_CONNECT_SERVER and then sending the process id to the router via the connect message (MSG_CONNECT). Messages are sent through the router by UIN_SEND_MSG and received by UIN_RECV_MSG. There are two ways to determine if a message is being sent to a process. The first way is to use UIN_RECV_MSG with the no-peek flag. The process will suspend operation at this statement until a message is received. The other way is to use UWN_ADD_INPUT_SOCKET to tell the UWN system to watch for input from a socket number. When a message is received, UWN_INPUT returns with a data type of SYS_INPUT_MESSAGE. This method allows a process to process both window and message inputs.

Before terminating a process that is connected to a router, a close socket message (MSG_CLOSE_SOCKET) must be sent to the router.

Environment variable RECORD_SESSION must be set to true to record routed messages. Only messages identified in MSG_SD_RECORD_LIST will be recorded. The recorded message data base is initialized when the router is started.

To compile this program, the following paths must be established using the "a.path" command:

/eddic/Ada/common
/usr/cherokee/VADS55/verdixlib
/usr/cherokee/VADS55/standard
/eddic/Ada/uux
/eddic/Ada/uin

To load this program, links must be established with the following libraries using the "a.info" command:

/egen/cin_util/cin_util.lib
/egen/cux_util/cux_util.lib

Data Bases

SITUATION_RECORD	OUTPUT

Environment Variables

RECORD_SESSION
SITUATION_RECORD_DB
SITUATION_ROUTER_SERV
SITUATION_ROUTER_HOST

4.2.9.2 RSD_RECORD_TO_ASCII

Abstract. Converts the recorded situation data to ASCII.

Major Capabilities. Reads the recorded situation data base and creates and individual ASCII file for each record type in the recorded data base.

Special Instructions. The system must be stopped (see STOP_EDDIC in CTL) before running this program. The recording data base is initialized whenever the router is started.

To compile this program, the following paths must be established using the "a.path" command:

/eddic/Ada/common
/usr/cherokee/VADS55/verdixlib
/usr/cherokee/VADS55/standard
/eddic/Ada/uux
/eddic/Ada/ued

To load this program, links must be established with the following libraries using the "a.info" command:

/egen/cux_util/cux_util.lib
/usr/lib/libm.a

Data Bases

SITUATION_RECORD	INPUT
TRAN_ACTIVITY	OUTPUT
TRAN_AMMUNITION	OUTPUT
TRAN_BLUEFOR_TASK_ORG	OUTPUT
TRAN_CNTRL_MSR_DEL	OUTPUT
TRAN_CNTRL_MSR_EFF_TIME	OUTPUT
TRAN_CNTRL_MSR_LOC	OUTPUT
TRAN_CNTRL_MSR_STAT	OUTPUT
TRAN_EQUIPMENT	OUTPUT
TRAN_FUEL	OUTPUT
TRAN_NEW_CNTRL_MSR	OUTPUT
TRAN_NEW_OBSTACLE	OUTPUT
TRAN_OBSTACLE_DEL	OUTPUT
TRAN_OBSTACLE_EFF_TIME	OUTPUT
TRAN_OBSTACLE_LOC	OUTPUT
TRAN_OBSTACLE_STAT	OUTPUT
TRAN_OPFOR_REINFORCE	OUTPUT
TRAN_OPFOR_STRENGTH	OUTPUT
TRAN_OPFOR_TASK_ORG	OUTPUT
TRAN_PERSONNEL	OUTPUT
TRAN_SITUATION_REQUEST	OUTPUT
TRAN_SITUATION_WINDOW	OUTPUT
TRAN_UNIT_LOCATION	OUTPUT
TRAN_UNIT_MISSION	OUTPUT

Environment Variables

SIT_ACTIVITY
SIT_AMMO
SIT_BLUE_TASK_ORG
SIT_CNTRL_MSR_DELETE
SIT_CNTRL_MSR_EFFECT
SIT_CNTRL_MSR_LOCATE
SIT_CNTRL_MSR_STATUS
SIT_EQUIP
SIT_FUEL
SIT_MISSION
SIT_NEW_CNTRL_MSR
SIT_NEW_OBSTACLE
SIT_OBSTACLE_DELETE
SIT_OBSTACLE_EFFECT
SIT_OBSTACLE_LOCATE

SIT_OBSTACLE_STATUS
SIT_OPFOR_TASK_ORG
SIT_PERS
SIT_RECORD
SIT_REINF
SIT_REQUEST
SIT_STRENGTH
SIT_UNIT_LOC
SIT_WINDOW

4.2.10 Station Control Manager (SCL)

The station control manager consists of the station control program and a set of lookup table utilities.

4.2.10.1 SCL_STATION_CONTROL_MANAGER

Abstract. Station control manager for a workstation.

Major Capabilities. The station control manager controls the color lookup table for the workstation, controls the screen (root) popup menu, handles the interaction with the map legend, and creates new view message and experiment control windows when a create window message is received.

Special Instructions. The following processes must be executing before the station control manager is started:

RCN_CONTROL_ROUTER

All lookup table updates must be routed through the experiment control router to this process using the MSG_LUT_UPDATE message. The process name that must be used is G2_STATION_MANAGER, G3_STATION_MANAGER, G4_STATION_MANAGER, or EX_STATION_MANAGER depending upon which station is being used. It has total control of all 255 colors available in the system. This was required to display the map with two overlay planes (Blue and Red).

To compile this program, the following paths must be established using the "a.path" command:

/eddic/Ada/common
/usr.MC68020/cherokee/VADS55/verdixlib
/usr.MC68020/cherokee/VADS55/standard

/eddic/Ada/uin
/eddic/Ada/uux
/eddic/Ada/uiw
/eddic/Ada/uwn

To load this program, links must be established with the following libraries using the "a.info" command:

/egen/cin_util/cin_util.lib
/egen/ciw_util/ciw_util.lib
/egen/cwn_util/cwn.lib
/egen/cux_util/cux_util.lib
/usr/lib/libXr.a
/usr/lib/libX_p.a
/usr/lib/libX.a

Data Bases

MAP_LEGEND	INPUT
ROOT_WINDOW_MENU	INPUT

Environment Variables

CHARACTER_FONT_FILE
CONTROL_ROUTER_HOST
CONTROL_ROUTER_SERV
EDDIC_STATION_USER
HILITE_DESCRIPTION_FILE
LASER_SERVER
MAP_LEGEND
OVERLAY_LOOKUP_TABLE
ROOT_MENU
SPOOL_PATH
UNHILITE_DESCRIPTION_FILE

4.2.10.2 LUT_MANAGER

Abstract. Low-level color lookup table utilities.

Major Capabilities. Procedures to initialize the color lookup table, read the lookup description files, and to load the colors into the lookup table.

Special Instructions. In the current configuration, these utilities should only be used by the station control manager (SCL).

To compile this package, the following paths must be established using the "a.path" command:

/eddic/Ada/common
/usr.MC68020/cherokee/VADS55/verdixlib
/usr.MC68020/cherokee/VADS55/standard

To load a program that uses this package, links must be established with the following libraries using the "a.info" command:

/egen/ciw_util/ciw_util.lib
/usr/lib/libX_p.a
/usr/lib/libX.a

Data Bases. None

Environment Variables. None

4.2.11 SCREEN MANAGER

Abstract. The screen manager is the top level foreground process which starts the Soldier Machine Interface system. It creates the base window creation icons and spawns the appropriate background processes on icon selection.

Major Capabilities. The screen manager displays a window creation icon for every base process in the workstation system. It determines if it should also put up a "Control" process icon via the environment variable eddic_station_user being set to "experimenter". It opens or creates the icon data base file specified by the environment variable, Icon_Path, stores the upper X coordinate of each icon for retrieval by other processes, and initializes the file for storage of the process ids associated with the maximum number of process windows that can be stacked as icons on the base icon. When an icon is selected, the appropriate background process is executed.

Special Instructions. The processes associated with each base icon are programmed into the file /esource/cwn_util/Create_Functions.c. If the application programmer changes this file, the screen manager can be rebuilt by the following command:

```
make -f smc.make
```

The programmer should also be aware that if Create_Functions.c is modified, then the cwn library should also have this module replaced after the above command has been performed. To do this, perform the commands:

```
cd /egen/cwn_util
ar r cwn.lib Create_Functions.o
ranlib cwn.lib
```

Data Bases

ICON_STACK_DB	OUTPUT

Environment Variables

EDDIC_STATION_USER
ICON_PATH

4.2.12 Situation Data Base Manager (SDB)

The situation data base manager consists of a program to build the situation data base, a program to build the situation data index files, a program to control access to it, a program to load asset levels into the higher echelon units, a specification ("_s.a" suffix) and body ("_b.a" suffix) to handle all input and output to the situation data base, a specification and body to control updates to the situation data base, and a specification and body to send data to requesting processes.

4.2.12.1 SDB_INPUT_OUTPUT

Abstract. Situation data input and output utilities.

Major Capabilities. The input and output utilities handle all the interaction with the Ada situation data bases. This includes, finding, reading, writing, updating, and maintaining the appropriate index files.

Special Instructions. Procedure SDB_OPEN_SITUATION_DB must be called to open the situation data bases and SDB_READ_INDEX_FILES must be called to read the index files before the other procedures in the package are used. Before the calling process terminates, procedures SDB_WRITE_INDEX_FILES and SDB_CLOSE_SITUATION_DB must be called to save any changes made to the situation data base.

To compile this package, the following paths must be established using the "a.path" command:

/eddic/Ada/common
/usr.MC68020/cherokee/VADS55/verdixlib
/usr.MC68020/cherokee/VADS55/standard
/eddic/Ada/uux

To load a program that uses this package, links must be established with the following libraries using the "a.info" command:

/egen/cux_util/cux_util.lib

Data Bases

BLUEFOR_AUTH_AMMO	INPUT/OUTPUT
BLUEFOR_AUTH_AMMO_INDEX	INPUT/OUTPUT
BLUEFOR_AUTH_EQUIP_INDEX	INPUT/OUTPUT
BLUEFOR_AUTH_EQUIP	INPUT/OUTPUT
BLUEFOR_CURR_AMMO	INPUT/OUTPUT
BLUEFOR_CURR_AMMO_INDEX	INPUT/OUTPUT
BLUEFOR_CURR_EQUIP_INDEX	INPUT/OUTPUT
BLUEFOR_CURR_EQUIP	INPUT/OUTPUT
BLUEFOR_FUEL	INPUT/OUTPUT
BLUEFOR_FUEL_INDEX	INPUT/OUTPUT
BLUEFOR_PERSONNEL	INPUT/OUTPUT
BLUEFOR_PERSONNEL_INDEX	INPUT/OUTPUT
BLUEFOR_UNIT_LOC	INPUT/OUTPUT
BLUEFOR_UNIT_LOC_INDEX	INPUT/OUTPUT
BLUEFOR_UNIT_STATUS	INPUT/OUTPUT
BLUEFOR_UNIT_STATUS_INDEX	INPUT/OUTPUT
CNTRL_MSR_POINT	INPUT/OUTPUT
CNTRL_MSR_POINT_INDEX	INPUT/OUTPUT
CONTROL_MEASURE	INPUT/OUTPUT
CONTROL_MEASURE_INDEX	INPUT/OUTPUT
OBSTACLE	INPUT/OUTPUT
OBSTACLE_INDEX	INPUT/OUTPUT
OPFOR_AUTH_EQUIP	INPUT/OUTPUT
OPFOR_AUTH_EQUIP_INDEX	INPUT/OUTPUT
OPFOR_CURR_EQUIP_INDEX	INPUT/OUTPUT
OPFOR_CURR_EQUIP	INPUT/OUTPUT
OPFOR_UNIT_LOC	INPUT/OUTPUT
OPFOR_UNIT_LOC_INDEX	INPUT/OUTPUT
OPFOR_UNIT_STATUS_INDEX	INPUT/OUTPUT

OPFOR_UNIT_STATUS	INPUT/OUTPUT

Environment Variables

BLUEFOR_AMMO_AUTH
BLUEFOR_AMMO_AUTH_NDX
BLUEFOR_AMMO_CURR
BLUEFOR_AMMO_CURR_NDX
BLUEFOR_EQUIP_AUTH_NDX
BLUEFOR_EQUIP_AUTH
BLUEFOR_EQUIP_CURR_NDX
BLUEFOR_EQUIP_CURR
BLUEFOR_FUEL
BLUEFOR_FUEL_NDX
BLUEFOR_LOCATION_NDX
BLUEFOR_LOCATION
BLUEFOR_PERS
BLUEFOR_PERS_NDX
BLUEFOR_UNIT_STATUS_NDX
BLUEFOR_UNIT_STATUS
CNTRL_MSR_POINT_NDX
CONTROL_MEASURE
CONTROL_MEASURE_NDX
CONTROL_MEASURE_POINT
OBSTACLE
OBSTACLE_NDX
OPFOR_EQUIP_AUTH
OPFOR_EQUIP_AUTH_NDX
OPFOR_EQUIP_CURR_NDX
OPFOR_EQUIP_CURR
OPFOR_LOCATION
OPFOR_LOCATION_NDX
OPFOR_UNIT_STATUS_NDX
OPFOR_UNIT_STATUS

4.2.12.2 SDB_LOAD_HIGHER_ECH

Abstract. Load assets into higher echelon units.

Major Capabilities. In the Tactical Planning Workstation system, the situation data base contains asset levels for units of all echelons. The initial scenario data contain assets for only the low-level units, so this program is necessary to roll-up the assets into the higher echelon units.

Special Instructions. Currently this program operates on only BLUEFOR units. The OPFOR higher echelon units were assigned assets in the initial scenario data.

The situation data base build program (SDB_SITUATION_DB_BUILD) must be run before this program. If it is not run, the higher echelon units will have twice as many assets as authorized.

To compile this program, the following paths must be established using the "a.path" command:

/eddic/Ada/common
/usr.MC68020/cherokee/VADS55/verdixlib
/usr.MC68020/cherokee/VADS55/standard
/eddic/Ada/uux

To load this program, links must be established with the following libraries using the "a.info" command:

/egen/cux_util/cux_util.lib

Data Bases

BLUEFOR_ASSET_UNIT	INPUT
BLUEFOR_TASK_ORG_SOURCE	INPUT
BLUEFOR_UNIT_CONVERT	INPUT
OPFOR_ASSET_UNIT	INPUT
OPFOR_TASK_ORG_SOURCE	INPUT
OPFOR_UNIT_CONVERT	

Environment Variables

BLUE_ASSET_UNIT
BLUEFOR_UNIT_CONVERSION
C2LAB_BLUE_TASK_ORG
C2LAB_OPFOR_TASK_ORG
OPFOR_ASSET_UNIT
OPFOR_UNIT_CONVERSION

4.2.12.3 SDB_PACKAGE

Abstract. Situation data manager object storage area.

Major Capabilities. This package is used as a common storage area for objects that must be visible to the whole situation data base system. The objects include situation data messages, Operational Planning (OPLAN) lists, and other objects required by the situation data system.

Special Instructions. Although the objects in this package are visible to the world, it is highly recommended that the SDB utility packages be used for accessing data, rather than accessing the objects in this package directly.

To compile this package, the following paths must be established using the "a.path" command:

/eddic/Ada/common
/usr.MC68020/cherokee/VADS55/verdixlib
/usr.MC68020/cherokee/VADS55/standard

Data Bases. None

Environment Variables. None

4.2.12.4 SDB_SEND_DATA

Abstract. Situation data utilities to send data to requesting processes through the situation data router.

Major Capabilities. When situation data are requested of the situation data base manager through the situation data router, the situation data base manager uses the procedures in this package to read the requested data, format it into a message, and send it to the requesting process.

Special Instructions. This package exists mainly for use by the situation data base manager. It may be used by other programs but it is not recommended.

To compile this package, the following paths must be established using the "a.path" command:

/eddic/Ada/common
/usr.MC68020/cherokee/VADS55/verdixlib
/usr.MC68020/cherokee/VADS55/standard
/eddic/Ada/uin

To load a program that uses this package, links must be established with the following libraries using the "a.info" command:

/egen/cin_util/cin_util.lib

Data Bases. None

Environment Variables. None

4.2.12.5 SDB_SITUATION_DB_BUILD

Abstract. Builds the tactical situation data bases.

Major Capabilities. Creates the situation data bases from the situation scenario source data bases. It also creates the OPLAN data base.

Special Instructions. This program must be run after changes have been made to the situation source files. The following separates are included in this program:

SDB_BLUEFOR_DBASE_STAT_BUILD
SDB_BLUEFOR_UNIT_FUEL_BUILD
SDB_BLUEFOR_UNIT_AMMO_BUILD
SDB_BLUEFOR_UNIT_PERS_BUILD
SDB_BLUEFOR_UNIT_STAT_BUILD
SDB_BLUEFOR_UNIT_EQUIP_BUILD
SDB_BLUEFOR_UNIT_ULOC_BUILD
SDB_CONTROL_MEASURE_BUILD
SDB_DBASE_CNTRL_MSR_BUILD
SDB_OBSTACLE_BUILD
SDB_OPFOR_UNIT_EQUIP_BUILD
SDB_OPFOR_UNIT_REINF_BUILD
SDB_OPFOR_UNIT_STAT_BUILD
SDB_OPFOR_UNIT_ULOC_BUILD
SDB_OPPLAN_DB_BUILD

To compile this program, the following paths must be established using the "a.path" command:

/eddic/Ada/common
/usr.MC68020/cherokee/VADS55/verdixlib
/usr.MC68020/cherokee/VADS55/standard
/eddic/Ada/uux
/eddic/Ada/ued

To load this program, links must be established with the following libraries using the "a.info" command:

/egen/cux_util/cux_util.lib
/usr/lib/libm.a

Data Bases

BLUEFOR_AMMO_SOURCE	OUTPUT
BLUEFOR_AMMO_TRACK	OUTPUT
BLUEFOR_AUTH_AMMO	OUTPUT
BLUEFOR_AUTH_EQUIP	OUTPUT
BLUEFOR_CURR_AMMO	OUTPUT
BLUEFOR_CURR_EQUIP	OUTPUT
BLUEFOR_EQUIP_SOURCE	INPUT
BLUEFOR_EQUIP_TRACK	INPUT
BLUEFOR_FUEL	OUTPUT
BLUEFOR_FUEL_SOURCE	INPUT
BLUEFOR_ORGANIC_TASK_ORG	INPUT
BLUEFOR_PERSONNEL	OUTPUT
BLUEFOR_PERSONNEL_SOURCE	OUTPUT
BLUEFOR_UNIT_CONVERT	INPUT/OUTPUT
BLUEFOR_UNIT_LOC_SOURCE	INPUT
BLUEFOR_UNIT_LOC	OUTPUT
BLUEFOR_UNIT_NAME	OUTPUT
BLUEFOR_UNIT_STATUS	OUTPUT
CNTRL_MSR_POINT	OUTPUT
CNTRL_MSR_POINT_NAME	OUTPUT
CONTROL_MEASURE	OUTPUT
CONTROL_MEASURE_SOURCE	INPUT
CONTROL_MEASURE_NAME	OUTPUT
OBSTACLE	OUTPUT
OBSTACLE_NAME	OUTPUT
OBSTACLE_SOURCE	INPUT
OPFOR_AUTH_EQUIP	OUTPUT
OPFOR_CURR_EQUIP	OUTPUT
OPFOR_EQUIP_NAME	OUTPUT
OPFOR_EQUIP_SOURCE	INPUT
OPFOR_ORGANIC_TASK_ORG	INPUT
OPFOR_REINFORCE_TIME	INPUT
OPFOR_TASK_ORG_SOURCE	INPUT
OPFOR_UNIT_CONVERT	INPUT/OUTPUT
OPFOR_UNIT_LOC	OUTPUT
OPFOR_UNIT_LOC_SOURCE	INPUT

OPFOR_UNIT_STATUS	OUTPUT
OPLAN_LIST	OUTPUT
OPLAN_LIST_SOURCE	INPUT

Environment Variables

BLUEFOR_AMMO_AUTH
BLUEFOR_AMMO_CURR
BLUEFOR_AMMO_TRACK
BLUEFOR_EQUIP_AUTH
BLUEFOR_EQUIP_CURR
BLUEFOR_EQUIP_TRACK
BLUEFOR_FUEL
BLUEFOR_LOCATION
BLUEFOR_ORGANIC_UNIT
BLUEFOR_TOP_UNIT
BLUEFOR_UNIT_CONVERSION
BLUEFOR_UNIT_STATUS
BLUEFOR_UNIT_XREF
BUILD_BLUE_AMMO
BUILD_BLUE_EQUIP
BUILD_BLUE_FUEL
BUILD_BLUE_PERS
BUILD_BLUE_STATUS
BUILD_BLUE_ULOC
BUILD_CNTRL_MSR
BUILD_OBSTACLE
BUILD_OPFOR_EQUIP
BUILD_OPFOR_REINF
BUILD_OPFOR_STATUS
BUILD_OPFOR_ULOC
BUILD_OPPLAN
C2LAB_BLUE_TASK_ORG
C2LAB_BLUEFOR_AMMO
C2LAB_BLUEFOR_EQUIP
C2LAB_BLUEFOR_FUEL
C2LAB_BLUEFOR_LOCATION
C2LAB_CONTROL_MEASURE
C2LAB_OBSTACLE
C2LAB_OPFOR_EQUIP
C2LAB_OPFOR_LOCATION
C2LAB_OPFOR_REINFORCE
C2LAB_OPFOR_TASK_ORG

CNTRL_MSR_XREF
CNTRL_POINT_XREF
CONTROL_MEASURE
CONTROL_MEASURE_POINT
OBSTACLE
OBSTACLE_XREF
OPFOR_EQUIP_AUTH
OPFOR_EQUIP_CURR
OPFOR_EQUIP_LIST
OPFOR_LOCATION
OPFOR_ORGANIC_UNIT
OPFOR_UNIT_CONVERSION
OPFOR_UNIT_STATUS
OPPLAN_DB
OPPLAN_SOURCE
USE_DBASE_BLUE_STATUS
USE_DBASE_CNTRL_MSR

4.2.12.6 SDB_SITUATION_DB_MANAGER

Abstract. Tactical situation data base manager.

Major Capabilities. The situation data base manager maintains the situation data base and allows network access to it. Access includes both retrieving and updating situation data.

Special Instructions. The following process must be executing before the situation data base manager is started:

RSD_SITUATION_DATA_ROUTER

All situation data requests must be routed through the situation data router (RSD). The process name that must be used is SITUATION_DB_MANAGER. The following messages are processed by the situation data base manager:

Message Requests
- MSG_CONTROL_MEASURE
- MSG_CNTRL_MSR_POINT
- MSG_OBSTACLE
- MSG_AMMO_AUTH
- MSG_AMMO_ON_HAND
- MSG_EQUIP_AUTH
- MSG_EQUIP_OPER
- MSG_PERSONNEL

MSG_FUEL
MSG_BLUEFOR_STATUS
MSG_LOCATION
MSG_BLUEFOR_TASK_ORG
MSG_ALL_LOCATIONS
MSG_OPFOR_STATUS
MSG_OPFOR_TASK_ORG
MSG_OPPLAN_LIST

Other Messages

MSG_AMMO_UPDATE
MSG_EQUIP_UPDATE
MSG_PERS_UPDATE
MSG_FUEL_UPDATE
MSG_LOC_UPDATE
MSG_BLUE_TASK_ORG_UPDATE
MSG_ACTIVITY_UPDATE
MSG_MISSION_UPDATE
MSG_OPFOR_TASK_ORG_UPDATE
MSG_REINFORCE_UPDATE
MSG_STRENGTH_UPDATE
MSG_CNTRL_MSR_ADD
MSG_CNTRL_POINT_ADD
MSG_CNTRL_MSR_CHG_EFF
MSG_CNTRL_MSR_CHG_LOC
MSG_CNTRL_MSR_CHG_STAT
MSG_CNTRL_MSR_DEL
MSG_OBSTACLE_ADD
MSG_OBSTACLE_CHG_EFF
MSG_OBSTACLE_CHG_LOG
MSG_OBSTACLE_CHG_STAT
MSG_OBSTACLE_DEL
MSG_NEW_OPPLAN
MSG_STOP

To compile this program, the following paths must be established using the "a.path" command:

/eddic/Ada/common
/usr.MC68020/cherokee/VADS55/verdixlib
/usr.MC68020/cherokee/VADS55/standard
/eddic/Ada/uux
/eddic/Ada/uin

To load this program, links must be established with the following libraries using the "a.info" command:

/egen/cin_util/cin_util.lib
/egen/cux_util/cux_util.lib

Data Bases

OPLAN_LIST	INPUT/OUTPUT

Environment Variables

OPPLAN_DB
SITUATION_ROUTER_HOST
SITUATION_ROUTER_SERV
START_DATE

4.2.12.7 SDB_SITUATION_INDX_BUILD

Abstract. Situation data base index file build.

Major Capabilities. Reads the situation data base files and creates and index file for each situation data base file. The files are indexed in OPLAN, date/time order.

Special Instructions. This program must be run after SDB_SITUATION_DB_BUILD is executed.

To compile this program, the following paths must be established using the "a.path" command:

/eddic/Ada/common
/usr.MC68020/cherokee/VADS55/verdixlib
/usr.MC68020/cherokee/VADS55/standard
/eddic/Ada/uux

To load this program, links must be established with the following libraries using the "a.info" command:

/egen/cux_util/cux_util.lib

Data Bases

BLUEFOR_AUTH_AMMO	INPUT
BLUEFOR_AUTH_AMMO_INDEX	OUTPUT

BLUEFOR_AUTH_EQUIP_INDEX	OUTPUT
BLUEFOR_AUTH_EQUIP	INPUT
BLUEFOR_CURR_AMMO	INPUT
BLUEFOR_CURR_AMMO_INDEX	OUTPUT
BLUEFOR_CURR_EQUIP_INDEX	OUTPUT
BLUEFOR_CURR_EQUIP	INPUT
BLUEFOR_FUEL	INPUT
BLUEFOR_FUEL_INDEX	OUTPUT
BLUEFOR_PERSONNEL	INPUT
BLUEFOR_PERSONNEL_INDEX	OUTPUT
BLUEFOR_UNIT_LOC	INPUT
BLUEFOR_UNIT_LOC_INDEX	OUTPUT
BLUEFOR_UNIT_STATUS	INPUT
BLUEFOR_UNIT_STATUS_INDEX	OUTPUT
CNTRL_MSR_POINT	INPUT
CNTRL_MSR_POINT_INDEX	OUTPUT
CONTROL_MEASURE	INPUT
CONTROL_MEASURE_INDEX	OUTPUT
OBSTACLE	INPUT
OBSTACLE_INDEX	OUTPUT
OPFOR_AUTH_EQUIP	INPUT
OPFOR_AUTH_EQUIP_INDEX	OUTPUT
OPFOR_CURR_EQUIP_INDEX	OUTPUT
OPFOR_CURR_EQUIP	INPUT
OPFOR_UNIT_LOC	INPUT
OPFOR_UNIT_LOC_INDEX	OUTPUT
OPFOR_UNIT_STATUS_INDEX	OUTPUT
OPFOR_UNIT_STATUS	INPUT

Environment Variables

BLUEFOR_AMMO_AUTH
BLUEFOR_AMMO_AUTH_NDX
BLUEFOR_AMMO_CURR
BLUEFOR_AMMO_CURR_NDX
BLUEFOR_EQUIP_AUTH_NDX
BLUEFOR_EQUIP_AUTH
BLUEFOR_EQUIP_CURR_NDX
BLUEFOR_EQUIP_CURR
BLUEFOR_FUEL
BLUEFOR_FUEL_NDX
BLUEFOR_LOCATION_NDX
BLUEFOR_LOCATION
BLUEFOR_PERS

BLUEFOR_PERS_NDX
BLUEFOR_UNIT_STATUS_NDX
BLUEFOR_UNIT_STATUS
CNTRL_MSR_POINT_NDX
CONTROL_MEASURE
CONTROL_MEASURE_NDX
CONTROL_MEASURE_POINT
OBSTACLE
OBSTACLE_NDX
OPFOR_EQUIP_AUTH
OPFOR_EQUIP_AUTH_NDX
OPFOR_EQUIP_CURR_NDX
OPFOR_EQUIP_CURR
OPFOR_LOCATION
OPFOR_LOCATION_NDX
OPFOR_UNIT_STATUS_NDX
OPFOR_UNIT_STATUS

4.2.12.8 SDB_UPDATE_DB

Abstract. Situation data base update utilities.

Major Capabilities. The procedures in this package extract update information from the situation data update messages and uses the procedures in SDB_INPUT_OUTPUT to update the data base.

Special Instructions. This package is currently used by SDB_SITUATION_DB_MANAGER and SDB_LOAD_HIGHER_ECH.

To compile this package, the following paths must be established using the "a.path" command:

/eddic/Ada/common
/usr.MC68020/cherokee/VADS55/verdixlib
/usr.MC68020/cherokee/VADS55/standard
/eddic/Ada/uin
/eddic/Ada/ued

To load a program that uses this package, links must be established with the following libraries using the "a.info" command:

/egen/cin_util/cin_util.lib
/usr/lib/libm.a

Data Bases. None

Environment Variables. None

4.2.13 WBD BUILD DISPLAY MANAGER

Abstract. Build window display manager.

Major Capabilities. This process controls the interaction with the build window. It is responsible for allowing the user to select a build product, displaying the product, and sending the product to another participant. Build products can be textual reports or digital maps with tactical overlays.

Special Instructions. The build display manager is started by the screen manager when the left mouse button is clicked on the build window creation button.

The following processes must be executing before the build window display manager is started:

RCP_C2_PRODUCT_ROUTER
RSD_SITUATION_DATA_ROUTER
CDB_C2_PRODUCT_DB_MANAGER
SDB_SITUATION_DB_MANAGER

To compile this program, the following paths must be established using the "a.path" command:

/eddic/Ada/common
/usr.MC68020/cherokee/VADS55/verdixlib
/usr.MC68020/cherokee/VADS55/standard
/eddic/Ada/uin
/eddic/Ada/uux
/eddic/Ada/ued
/eddic/Ada/uwn
/eddic/Ada/utm

To load this program, links must be established with the following libraries using the "a.info" command:

/egen/cin_util/cin_util.lib
/egen/ciw_util/ciw_util.lib
/egen/cux_util/cux_util.lib

/egen/cwn_util/cwn.lib
/usr/lib/libXr.a
/usr/lib/libX_p.a
/usr/lib/libX.a
/usr/lib/libm.a

Data Bases

MAP_BUILD_MENU	INPUT

Environment Variables

BUILD_C2_MAP_MENU
C2_PRODUCT_ROUTER_SERV
C2_PRODUCT_ROUTER_HOST
EDDIC_STATION_USER
SITUATION_ROUTER_SERV
SITUATION_ROUTER_HOST

4.2.14 WCD CONTROL DISPLAY MANAGER

Abstract. Participant experiment control window display manager.

Major Capabilities. This process controls the interaction with the participants experiment control window. It is responsible for displaying the experiment control product and sending the product to the experimenter. Experiment control products can be informative or require a response. Informative messages are displayed in the experiment control window until another experiment control product is received or until the window is terminated. If the window is closed into an icon when a new message is received, a blue bar is displayed in the icon. Experiment control messages that require a response are displayed with a send button in the upper left corner of the screen. The message is displayed in the window until the send button is hit by the participant.

Special Instructions. The experiment control display manager is started by the station control manager (SCL) when it receives a window creation message from the experiment control manager (CTL).

The following processes must be executing before the experiment control window display manager is started:

RCN_CONTROL_ROUTER
CTL_EXPERIMENT_CONTROL

To compile this program, the following paths must be established using the "a.path" command:

/eddic/Ada/common
/usr.MC68020/cherokee/VADS55/verdixlib
/usr.MC68020/cherokee/VADS55/standard
/eddic/Ada/uin
/eddic/Ada/uux
/eddic/Ada/uwn
/eddic/Ada/uiw

To load this program, links must be established with the following libraries using the "a.info" command:

/egen/cin_util/cin_util.lib
/egen/ciw_util/ciw_util.lib
/egen/cux_util/cux_util.lib
/egen/cwn_util/cwn.lib
/usr/lib/libXr.a
/usr/lib/libX_p.a
/usr/lib/libX.a
/usr/lib/libm.a

Data Bases. None

Environment Variables

CONTROL_ROUTER_HOST
CONTROL_ROUTER_SERV
EDDIC_STATION_USER

4.2.15 WED EXPERIMENT DISPLAY MANAGER

Abstract. Experimenter's experiment control window display manager.

Major Capabilities. This process controls the interaction with the experimenter's experiment control window. It is responsible for allowing the experimenter to select an experiment control product, for displaying the experiment control product, and for sending the product to the participant. Experiment control products can be informative or require a response.

Special Instructions. The experimenter experiment control display manager is started by the screen manager when the left mouse button is clicked on the experiment control window creation button.

The following processes must be executing before the experiment control window display manager is started:

RCN_CONTROL_ROUTER
CTL_EXPERIMENT_CONTROL

To compile this program, the following paths must be established using the "a.path" command:

/eddic/Ada/common
/usr.MC68020/cherokee/VADS55/verdixlib
/usr.MC68020/cherokee/VADS55/standard
/eddic/Ada/uin
/eddic/Ada/uux
/eddic/Ada/ued
/eddic/Ada/uwn
/eddic/Ada/utm

To load this program, links must be established with the following libraries using the "a.info" command:

/egen/cin_util/cin_util.lib
/egen/cux_util/cux_util.lib
/egen/cwn_util/cwn.lib
/usr/lib/libXr.a
/usr/lib/libX_p.a
/usr/lib/libX.a
/usr/lib/libm.a

Data Bases

MAP_BUILD_MENU	INPUT

Environment Variables

BUILD_C2_MAP_MENU
CONTROL_ROUTER_HOST
CONTROL_ROUTER_SERV
SITUATION_ROUTER_HOST
SITUATION_ROUTER_SERV

4.2.16 WMS MESSAGE DISPLAY MANAGER

Abstract. View message window display manager.

Major Capabilities. This process controls the interaction with the view message window. It is responsible for displaying the incoming message, maintaining the message queue, and maintaining the save and message log.

If a message is currently displayed when a new message is received, the message is added to the message queue and the number of messages in the queue is updated in the drop button. If the window is closed into an icon when the message is received, a blue bar is displayed in the icon.

Special Instructions. The view message display manager is started by the screen manager when the left mouse button is clicked on the view message window creation button or by the station control manager (SCL) when it receives a window creation message from the C2 product data base manager (CDB).

The following processes must be executing before the view message window display manager is started:

RCP_C2_PRODUCT_ROUTER
RSD_SITUATION_DATA_ROUTER
CDB_C2_PRODUCT_DB_MANAGER
SDB_SITUATION_DB_MANAGER

To compile this program, the following paths must be established using the "a.path" command:

/eddic/Ada/common
/usr.MC68020/cherokee/VADS55/verdixlib
/usr.MC68020/cherokee/VADS55/standard
/eddic/Ada/uin
/eddic/Ada/uux
/eddic/Ada/ued
/eddic/Ada/uwn
/eddic/Ada/uiw
/eddic/Ada/utm

To load this program, links must be established with the following libraries using the "a.info" command:

/egen/cin_util/cin_util.lib
/egen/ciw_util/ciw_util.lib

/egen/cux_util/cux_util.lib
/egen/cwn_util/cwn.lib
/usr/lib/libXr.a
/usr/lib/libX_p.a
/usr/lib/libX.a
/usr/lib/libm.a

Data Bases

MAP_MESSAGE_MENU	INPUT

Environment Variables

C2_PRODUCT_ROUTER_HOST
C2_PRODUCT_ROUTER_SERV
EDDIC_STATION_USER
MESSAGE_CREATED_BY_USER
MESSAGE_MAP_MENU
SITUATION_ROUTER_HOST
SITUATION_ROUTER_SERV

4.2.17 WTD Tool Display Manager

The tool display manager consists of the tool window manager program, the calculator specification, and the task organization tool.

4.2.17.1 CALC_CALCULATOR

Abstract. Graphical calculator tool.

Major Capabilities. The calculator is a mouse-based tool that functionally duplicates the Microsoft Windows calculator.

Special Instructions. To compile this package, the following paths must be established using the "a.path" command:

/eddic/Ada/common
/usr/cherokee/VADS55/verdixlib
/usr/cherokee/VADS55/standard
/eddic/Ada/uux
/eddic/Ada/uwn

To load a program that uses this package, links must be established with the following libraries using the "a.info" command:

/egen/cux_util/cux_util.lib
/egen/cwn_util/cwn.lib
/usr/lib/libXr.a
/usr/lib/libX_p.a
/usr/lib/libX.a
/usr/lib/libm.a

Data Bases. None

Environment Variables. None

4.2.17.2 WTD_TOOL_DISPLAY_MANAGER

Abstract. Tool window display manager.

Major Capabilities. This process controls the interaction with the tool window. It is responsible for allowing the user to select a tool, calling the procedure to display the tool, and passing input to the appropriate input processing procedure for the selected tool.

Special Instructions. The tool display manager is started by the screen manager when the left mouse button is clicked on the tool window creation button.

The following processes must be executing before the tool window display manager is started:

RCN_CONTROL_ROUTER

To compile this program, the following paths must be established using the "a.path" command:

/eddic/Ada/common
/usr/cherokee/VADS55/verdixlib
/usr/cherokee/VADS55/standard
/eddic/Ada/uin
/eddic/Ada/uux
/eddic/Ada/uwn
/eddic/Ada/ufm
/eddic/Ada/ued
/eddic/Ada/utm

To load this program, links must be established with the following libraries using the "a.info" command:

/egen/cin_util/cin_util.lib
/egen/cux_util/cux_util.lib
/egen/cwn_util/cwn.lib
/egen/ciw_util/ciw_util.lib
/usr/lib/libXr.a
/usr/lib/libX_p.a
/usr/lib/libX.a
/usr/lib/libm.a

Data Bases

TOOL_MENU	INPUT

Environment Variables

CONTROL_ROUTER_HOST
CONTROL_ROUTER_SERV
EDDIC_STATION_USER
TOOLS

4.2.17.3 TOT_EDITOR

Abstract. TOT_EDITOR is an acronym for the Task Organization Tool Editor package, which edits a task organization unit structure.

Major Capabilities. TOT_EDITOR displays and edits task organizations using the tree structure builder (TSB) package, for any OPLAN currently in the system. Some of the task organization displays can be quite large. These displays can be decluttered by displaying only certain unit types (combat, combat support, combat service support) or altering the top unit (a particular division, or a particular brigade...), thus displaying a smaller subset of the original. If a task organization display will not fit in a single window, and the user can not or does not wish to declutter, the user may optionally split the view. Splitting the view means the user divides the existing window either horizontally or vertically in what-ever proportions are desired. The view may be split an infinite number of times. When the split view is no longer required it may be destroyed, as long as there is always at least one view.

Once a task organization is displayed the user may request detail and summary reports on any given unit. The user can also alter the task organization by attaching or direct supporting a unit(s) from one place to another.

Special Instructions Use of the task organization tool requires the situation router to be up. This is needed so that OPLANs can be retrieved and updated.

There are three main functions within TOT: First, is a one time, per execution, initialization (TOT_INITIALIZE); second, is a one time, per execution, termination (TOT_TERMINATE); third, is all other event processing (TOT_PROCESS_INPUT). TOT_PROCESS_INPUT does not receive events directly from the system, via UWN_INPUT, so the calling process, WTD, passes input events to it through the procedure arguments. TOT_PROCESS_INPUT does not call UWN_INPUT. It must handle events that have nothing to do with task organization and report back when it is finished processing a given event. TOT_PROCESS_INPUT is called once for each event.

To compile this package, the following paths must be established using the "a.path" command:

/eddic/Ada/common
/usr/cherokee/VADS55/verdixlib
/usr/cherokee/VADS55/standard
/eddic/Ada/uin
/eddic/Ada/uux
/eddic/Ada/uwn
/eddic/Ada/ued
/eddic/Ada/uiw
/eddic/Ada/utm

To load a program that uses this package, links must be established with the following libraries using the "a.info" command:

/egen/cin_util/cin_util.lib
/egen/cux_util/cux_util.lib
/egen/cwn_util/cwn.lib
/egen/ciw_util/ciw_util.lib
/usr/lib/libXr.a
/usr/lib/libX_p.a
/usr/lib/libX.a
/usr/lib/libm.a

Data Bases

TASK_ORG_TOP_UNIT_MENU
TASK_ORG_UNIT_MENU
TASK_ORG_UNIT_TYPE_MENU
TASK_ORG_TOOL_MENU

Environment Variables

CHARACTER_FONT_FILE
SITUATION_ROUTER_HOST
SITUATION_ROUTER_SERV
SYMBOL_FONT_FILE
TOP_UNIT_MENU
UNIT_MENU
UNIT_TYPE_BTN_MENU
VIEW_MENU

4.2.18 WVC VIEW C2 DISPLAY MANAGER

Abstract. View situation window display manager.

Major Capabilities. This process controls the interaction with the view situation window. It is responsible for allowing the user to select a C2 product and displaying the selected product. The product can be either a textual report or a digital map with tactical overlay.

Special Instructions. The view situation display manager is started by the screen manager when the left mouse button is clicked on the view situation window creation button.

The following processes must be executing before the view situation window display manager is started:

RCP_C2_PRODUCT_ROUTER
RSD_SITUATION_DATA_ROUTER
CDB_C2_PRODUCT_DB_MANAGER
SDB_SITUATION_DB_MANAGER

To compile this program, the following paths must be established using the "a.path" command:

/eddic/Ada/common
/usr.MC68020/cherokee/VADS55/verdixlib
/usr.MC68020/cherokee/VADS55/standard
/eddic/Ada/uin
/eddic/Ada/uux
/eddic/Ada/ued
/eddic/Ada/uwn
/eddic/Ada/utm

To load this program, links must be established with the following libraries using the "a.info" command:

/egen/cin_util/cin_util.lib
/egen/ciw_util/ciw_util.lib
/egen/cux_util/cux_util.lib
/egen/cwn_util/cwn.lib
/usr/lib/libXr.a
/usr/lib/libX_p.a
/usr/lib/libX.a
/usr/lib/libm.a

Data Bases

MAP_VIEW_C2_MENU	INPUT

Environment Variables

C2_PRODUCT_ROUTER_HOST
C2_PRODUCT_ROUTER_SERV
EDDIC_STATION_USER
SITUATION_ROUTER_HOST
SITUATION_ROUTER_SERV
VIEW_C2_MAP_MENU

4.2.19 WVR VIEW REFERENCE DISPLAY MANAGER

Abstract. View reference window display manager.

Major Capabilities. This process controls the interaction with the view reference window. It is responsible for allowing the user to select a reference product and displaying the selected product. Currently reference products can only be textual reports.

Special Instructions. The view reference display manager is started by the screen manager when the left mouse button is clicked on the view reference window creation button.

The following processes must be executing before the view reference window display manager is started:

RRF_REFERENCE_ROUTER
FDB_REFERENCE_DB_MANAGER

To compile this program, the following paths must be established using the "a.path" command:

/eddic/Ada/common
/usr.MC68020/cherokee/VADS55/verdixlib
/usr.MC68020/cherokee/VADS55/standard
/eddic/Ada/uin
/eddic/Ada/uux
/eddic/Ada/ued
/eddic/Ada/uwn

To load this program, links must be established with the following libraries using the "a.info" command:

/egen/cin_util/cin_util.lib
/egen/cux_util/cux_util.lib
/egen/cwn_util/cwn.lib
/usr/lib/libXr.a
/usr/lib/libX_p.a
/usr/lib/libX.a
/usr/lib/libm.a

Data Bases. None

Environment Variables

EDDIC_STATION_USER
REFERENCE_ROUTER_HOST
REFERENCE_ROUTER_SERV

4.3 C UTILITIES

The C utilities are low level utilities required to access existing system capabilities. The C utilities are functionally organized with each function being contained on a separate library. The following C libraries exist in the Tactical Planning Workstation:

CIN - Internet communications utilities

CIW - Color image window utilities

CUX - Unix command utilities

CWN - Window display and control utilities

XR - Hewlett-Packard X-window utilities

Many of the C procedures have Ada bindings with the same name except the procedure name starts with a "U" instead of a "C". Those procedures are described in section 4.1. The following sections describe only C utilities that are not currently accessible from Ada.

4.3.1 CIN UTIL

Abstract. CIN_UTIL is an acronym for a set of utility communications primitives which allows processes to communicate with each other using an InterNet protocol. Programs may communicate with each other both within one processor and over an ethernet network.

Ada Binding. The routines in this library are used by the UIN_INTERNET_COMMUNICATIONS package and have Ada bindings to it through the CIN_INTERNET_COMMUNICATIONS specification package. Both of these packages can be found in the /eddic/Ada/uin directory.

Major Capabilities. CIN_UTIL is a stand alone utility library (not a process) which does not require the fileserver routers and/or data base managers to operate. This utility library is premised on a server-client relationship. That relationship is defined in the UIN_INTERNET_COMMUNICATIONS special instructions.

Special Instructions. All of the routines which bind with an application must pass their arguments as pointers (starting address in memory) rather than passing actual data in the arguments.

The complete instructions for CIN_UTIL use can be found in the UIN_INTERNET_COMMUNICATIONS section.

There are a couple of routines which are not used by the outside world which will be described here. The first routine (CIN_MSTR_SOCK_INFO) will load the master socket (server) address information structure. That information is needed when creating the server and creating a client. The other routine (CIN_WHICH_BIT_ON) not used by the outside world, determines which bit, in a 32 bit word, is on. That information is used by the server waiting routine to determine who just called him.

CIN_RECV_MSG needs a special mention. The first item in the message structure being passed is a four byte word which holds the length of the message. The remainder of the structure is application dependent, and the exact format and layout are of no concern to this routine because it will not try to interpret the message, it will merely be passed on as a bit stream. CIN_RECV_MSG has been modified to accommodate Ada programs, in that the

first two bytes of the message are some sort of Ada overhead, so they must be ignored by "C", and the next four bytes are the length of the message. Therefore "C" language routines need to use structures which begin with an appropriate two byte buffer.

There are no include files used in CIN_UTIL; everything is passed by argument. To compile this library, use the "make" command and associated Makefile found in the directory.

To load a program with this library, links must be established with the following libraries using the "a.info" command:

/egen/cin_util/cin_util.lib

Data Bases. None

Environment Variables. None explicitly, two implicitly passed in by argument.

host_id	- name of the server machine
service_id	- name of the service id (INET port number).

4.3.2 CIW UTIL

Abstract. CIW_UTIL is an acronym for a set of Image Windowing primitives which provides a means for programmers to perform certain color graphics imaging functions within the X Windows System environment.

Ada Binding. The routines in this library are used by the UIW_IMAGE_WINDOW and UIW_GENERIC packages and have Ada bindings to it through the CIW_IMAGE_WINDOW specification package. Both of these packages can be found in the /eddic/Ada/uiw directory.

Major Capabilities. CIW_UTIL is a stand alone utility library (not a process) which does not require the fileserver routers and/or data base managers to operate. This utility library allows programs to access X Windows color graphics imaging commands from high level languages without having an intimate knowledge of the X Windows system. However, the programmer must have some knowledge or concept of X Windows or graphics processing. There is not a one to one pairing of routines to X Windows commands; only those commands required by the Tactical Planning Workstation have been developed.

Special Instructions. All of the routines which bind with an application must pass their arguments as pointers (starting address in memory) rather than passing actual data in the arguments.

The complete instructions for CIW use can be found in the UIW_GENERIC and UIW_IMAGE_WINDOW sections.

The C include files used in CIW are:

ciw_parm.h - Some parameter constants used by the C routines.
ciw_color.h - Variables that hold interim values used in initializing, loading, and storing the color lookup table.

To compile this library, use the "make" command and associated Makefile found in the directory.

To load a program with this library, links must be established with the following libraries using the "a.info" command:

/egen/ciw_util/ciw_util.lib
/usr/lib/libX.a
/usr/lib/libX_p.a

Data Bases. Whatever font file is passed in by the application.

Environment Variables. None

4.3.3 CUX UTIL

Abstract. CUX_UTIL is an acronym for a set of utility primitives, which allow programs to access UNIX operating system commands.

Ada Binding. The routines in this library are used by the UUX_UTIL and UUX_IO packages and have Ada bindings to it through the CUX_UTIL specification package. Both of these packages can be found in the /eddic/Ada/uux directory.

Major Capabilities. CUX_UTIL is a stand alone utility library (not a process), which does not require the fileserver routers and/or data base managers to operate. This utility library provides a means for programmers to perform certain UNIX operating system commands, or very rudimentary functions, that high-level languages do not permit. There is not a one-to-one pairing of routines to UNIX commands; only those commands required by the Tactical Planning Workstation have been developed.

Special Instructions. All of the routines that bind with an application must pass their arguments as pointers (starting address in memory) rather than passing actual data in the arguments.

The complete instructions for CUX use can be found in the UUX_UTIL and UUX_IO sections.

There are no include files used in CUX_UTIL; everything is passed by argument.

To compile this library, use the "make" command and associated Makefile found in the directory.

To load a program with this library, links must be established with the following libraries using the "a.info" command:

/egen/cux_util/cux_util.lib

Data Bases. No explicit data base, all are implicit. cux_open_file, cux_close_file, cux_binary_read, and cux_binary_write will open, close, read, or write to any data base.

Environment Variables. No explicit environment variables, all are implicit.

cux_getenv will decipher any environment variable.

4.3.4 CWN UTIL

Abstract. The CWN library consists of the window utilities written in the C language for the Tactical Planning Workstation. It uses the X Window System protocol designed at MIT and a modified version of the Xrlib user interface library developed by Hewlett-Packard. Some of the Hewlett-Packard Xrlib routines of version 10 Release 4, were corrected for errors or enhanced under the present effort and were therefore placed in this library also. The library was also designed to be used in conjunction with the system's start-up routine screen_manager.

Ada Binding. The routines in this library are used by the UWN_WINDOW_SYSTEM package and have Ada bindings to it through the CWN_WINDOW_SYSTEM specification package. Both of these packages can be found in the /eddic/Ada/uwn directory.

Major Capabilities. CWN contains the C equivalence of all UWN window system utilities, including the internal routines called to support the capabilities and automatic functions of some of the UWN utilities using Xrlib and X utilities.

Special Instructions

.A Brief History

The implementation of the window utilities uses the Xrlib programming tools described in the manual "Programming with the X Window System", November 1986. The programmer not familiar with the capabilities of Xrlib should read this manual.

The CWN system has undergone extensive changes since its initial conception and implementation. This is important to keep in mind, as some remnants of the earlier developments may be disconcerting to the programmer looking at it for the first time. A brief history is therefore given to give insight into the change of the usage of terminology, structures, and routines evident throughout the system.

Initially, the system presumed all utilities would be displayed in what is now referred to as a process window. The window was created by placing a subwindow inside another window to give the visual effects of the subwindow having a wide border. The border window was considered the primary window and the subwindow was considered the working window, the area in which applications could define and work within. Most utilities were named and documented in the comments with respect to this type of window referencing. Also, the coordinate system used in defining was in terms of character rows and columns instead of pixels.

All field editors were required to be defined within a panel or subpanel, as the panel manager eliminated the handling of a number of events an application would otherwise have to deal with. However, the application was responsible for assigning unique IDs to each object, despite the fact Xr used a pointer to the editor instance as a unique ID for all operations on an object. To keep track of both IDs and be able to search for one or the other, link lists were the primary data structures utilized.

As the need arose for the capability of defining within windows, the automatic functions of the panel manager had to be incorporated; e.g., redrawing when an exposure event was received. At the same time, development of the system required the notification of special events initially considered of no import to the application. These and the increase of other capabilities, along with the rise of problems associated with using them, has caused the CWN library to become increasingly complex and under constant development.

. Overview of the Library Design Structure

The main C include files of the CWN window system are cwn_window_system_types.h and cwn_window.h. The rest of the include files are used only by specific routines, although they may contain the external declarations of some variables declared in the former two include files. The specific include files are as follows:

Include File	Used By:
cwn_window_system_init.h	cwn_initialize_window_system.c
cwn_create_window.h	cwn_create_window.c
cwn_icon.h	Any routine requiring icon information
screen_manager.h	screen_manager.c

The CWN system uses link list structures for buttons, menus, panels, and windows as found in the include file cwn_window_system_types.h. These link structures were designed with record components to store data needed for window system management of all that has been defined and requested by an application. The most notable link lists are those for determining where an editor was defined; in a panel, a subpanel, or a window. Two data components are used for determining this; a pointer to the panel list and a pointer to the subpanel list. A panel defined editor will have a NULL value for its subpanel designation, whereas a window will have a NULL value for its panel designation. A subpanel will have a non-null value in both the panel and subpanel designators. The following table summarizes the algorithm for the process of determining an editor's destination when defined:

Destination	Panel Pointer	Subpanel Pointer
Panel	X	NULL
Subpanel	X	X
Window	NULL	X

Menus in Xr, especially walking menus, are not defined using one simple data structure, thus requiring a link list structure and recursive routines for definition. Note that such a menu is defined by linking submenus to menus or other submenus and requiring separate menu structures and routine calls for definition. Rather than forcing the application to be responsible for the incorporation of code handling these details, the menu system simply receives from the application arrays of labels and indices indicating the links to be made and handles the definition process.

Xr panels were difficult to implement for the dynamic requirements of the system, thus they have a more complex link list structure. The panel structure used for defining a panel in Xr requires the application to know up front the number of editors to be created within the panel. This implied that if any editors were to be added or deleted after the initial creation the application had to delete the original panel and define a panel with additional or fewer editors. To expedite this process the design approach taken had the user define a panel up front, call the appropriate define editor routines, and notify CWN when the definition was complete by calling cwn_end_panel.

When defined, a panel first allocates a field list of five fields. Each time an editor is specified to be within the panel, the panel's field allocation is checked to see if more fields need to be allocated. In the event of adding or deleting editors it is mandatory that the

application calls cwn_update_panel. This routine performs the actual removal of the original panel and creates a new panel with the new set of editors.

. Event Processing

The processing of events for multiple purposes has always been a nemesis for the window system. This is mostly due to the methodology used in Xr for processing event lists. One way event lists are established is by setting up xrPFI functions which process automatically upon a particular input or inputs to a window. If a registered window has more than one function declared for the same input event, only the function declared first will be invoked. In other words Xr, XrInput in particular, loops through the event list until it finds a process flag set for the selected input. When one is found, the function is called, returns, and XrInput exits without continuing its search for any further processing of the same event. Even if a copy of the event was pushed back onto the input queue, only the first function would be performed as XrInput would have started its search at the beginning of the event list.

Another factor in the event processing problem is that for XrInput to receive a particular event for a window the event has to be selected for the window via XSelectInput. The problem is that any subsequent calls to this routine nullify the previous calls. This problem became evident when field editors were permitted to be defined within a window instead of a panel, and applications were allowed to select exposure events of a window for notification.

To perform automatic redraw capability for the editors defined within a window an XSelectInput was performed to select exposure event notification for the window. The xrPFI function cwn_redraw_editor was then established to handle the redraw. However, to prevent multiple redraws of the screen, screen flashing, the routine cwn_purge_window_exposure was defined to purge other exposure events caused by the redraw itself.

If an application also wished to be notified of an exposure event it called cwn_select_input with the exposure flag set true. This routine would also perform XSelectInput and set up the process function cwn_input_selected for pushing a notification onto the input queue for cwn_input to return to the application. Again, depending on which was performed first, only one of these functions would be performed.

The final solution implemented is the routine cwn_XSelectInput. This routine is used throughout the CWN system and is the only routine with a direct call to XSelectInput. The reasoning is that different editors require different event selections and to insure all events needed are selected for a particular window, cwn_XSelectInput searches the window editor field list "or-ing" all the necessary event masks. Once this is complete the final XSelectInput call is made.

. Notes

All routines passing information back to the application require the application to pass the starting address of memory for storage of the information.

A null event must be pushed onto the event queue after processing has been completed in an xrPFI function call. Failure to do this may cause some user interaction to go undetected.

The cwn_delete_subwindow routine does not actually attempt to delete a subwindow by removing it from Xr's registered window list. Instead, it simply unmaps the window. This approach was taken after it was discovered that too many levels of subwindows seem to cause the Xrlib to lose track of the window's information. The problem is probably in the QueryTree routine used in Xr.

. Linking to cwn.lib

To load an Ada application to cwn.lib, the following libraries must be visible:

/usr/lib/libX.a
/usr/lib/libXr.a
/usr/lib/libX_p.a
/egen/cwn_util/cwn.lib (where egen is a symbolic link to */usr2/eddic/gen*)

To link a C application, the link command line would have the following appended to it:

-lX -lXr -lX_p /egen/cwn_util/cwn.lib

Data Bases

ICON_STACK_DB	INPUT/OUTPUT

Environment Variables

ICON_PATH

4.3.5 Xrlib Corrections and Enhancements

Abstract. X-Window high-level utilities developed by Hewlett-Packard.

Ada Binding. None

Major Capabilities. X-Windows version 10 release 4 contained the X-Window system and some high-level graphic utilities developed by various companies. To save software development time and cost, it was decided to use the Hewlett-Packard utilities as a starting point. The utilities provide such things as scrollbars, text editors, button utilities and walking menus. This section does not describe the whole Xr system, only the C procedures that were changed for integration into the system. A complete description of the Xr system is contained in the X-Window documentation.

Special Instructions. The following changes were made to the Xr system for integration into the system:

ExpPageEdit.c

ExpPageEdit is a text field editor developed and expanded from the Xrlib field editor PageEdit. The editor was enhanced with a scrollbar on the left side of the editor for scrolling within the editor buffer, an option for read-only, and a popup menu was incorporated for editor operations. These operations include the capabilities to: (1) copy text to and from editors or static text, (2) cut and paste, (3) find other instances of a text string within the editor or another editor. The editor defined as read-only has only the operations of copy and find. Each of these operations may be invoked using the keyboard rather than the popup menu. Usually, entering the first letter of the function name as a control character invokes the operation. For example, entering a Control C, invokes the Copy function whereas entering a Control T would invoke the Cut function.

Other menu options coded in ExpPageEdit, but not currently in use, include a reset function, a save function, and a split screen function.

NumericEdit.c

This special editor is a modified version of the Xrlib TextEdit routine, the string field editor in UWN. The modification required adding new event types to the defs.h include file, a new editor information structure (XrNumericEditInfo, to types.h), and an internal editor data structure (XrNumericEditData, in the file in_types.h). The editor was modified to accept the traversal keyboard keys to send appropriate events back to the application. Numeric input is limited to integers, but the application was given the capability to constrain the input range by specifying the minimum and maximum values accepted.

MenuEdit.c and MenuMgr.c

These two modules of Xr were modified to decrease the sensitivity of menu selections within the system. Often times, users were found to have clicked the right mouse button rapidly in an area where a menu was activated. The menu would popup and detect an item selection because the menu was displayed with the mouse within a menu selection area. To place the mouse outside any active selection fields, MenuEdit.c required a check for the cursor being in a null area of the menu's x coordinates. MenuMgr.c was changed to calculate the menu's origins with respect to the cursor position.

RadioButton.c

This module was changed to display radiobuttons as squares like checkboxes. The procedure is a modified copy of the Xrlib routine RButton.c which displays round radiobuttons.

Scrollbar.c

The scrollbar displayed on the Sun workstation using the original Xr utility did not display the scroll arrows at the end. To correct this problem calls to XrFillPoly were replaced with calls to XrPoly.

StaticText.c

This editor was enhanced by adding a copy function invoked by popup menu selection. The primary purpose for this function is to provide the capability for copying static text into an editor. The internal data structure for static text (stData) was modified by adding components to keep track of text selection endpoints. The selected text is stored in the X buffer zero, using XStoreBytes, for later retrieval via the paste function option in the page editor.

panelmgr.c

The panel manager routine panelmgr.c, was modified as follows:

1. The xrPanelInfo data structure had a record component specified as "relativeTo" in which the application was to specify the window the panel window was to be created relative to. This field was ignored in the panel manager, making the option useless. The code was changed to create the window as a child of the relativeTo window under the case statement of MSG_NEW, instead of creating the panel's window with the RootWindow as the parent.

2. Another problem was that the panel's group instance was returned despite input taking place within a subpanel of the panel. The variable "valuePtr" was corrected to return a subpanel editor group instance if editor input was to a subpanel. This correction is under the case statement of MSG_EDIT.

3. Every panel was given its own panelContext instead of pointing to the default panel context which caused every panel to have the same current editor despite the fact the editor was not in every panel. The code would loop forever reactivating the same editor when the user selected an editor in another panel. The event would never be processed as the proper panel would always activate an editor it did not have. This correction may be found in the case MSG_NEW statement.

4. A similar situation existed when no action was taken in an activated field and the user opted to select into another field. The process had already set the current editor to the activated one and would loop forever activating the same editor. The event would be passed to the editor, the editor it would recognize the event was not within its rectangle of definition and push the event back onto the input queue. Since the current Editor had not been changed, the same process would then repeat. The condition was corrected by setting the current Editor field to null whenever panel input was a known editor event type which was pushed back onto the event queue. This correction may be found under the MSG_EDIT case statement.

Data Bases. None

Environment Variables. None

4.4 dBASE PROGRAMS

dBASE is used in EDDIC for maintaining the scenario data, exporting the scenario data to the Sun system, and maintaining the experiment analysis data. These programs are described in the following section.

4.4.1 EDDIC

Abstract. Maintains the experiment analysis data.

Major Capability. This program accepts three categories of experiment data:

(1) - Computer recorded
(2) - Experiment observations
(3) - Participant Questionnaires

The program maintains the data by experiment id and allows complete editing and printing capabilities. It also provides the capability to export the data in ASCII format for the purpose of importing into SAS or other statistical packages.

Special Instructions. This program was developed using the dBASE IV application developer and will not run in dBASE III Plus. Use the application developer to modify this program.

The following section describes the EDDIC application. dBASE applications consist of a group of menus chained together. The top menu is a bar menu that appears at the top of the screen and all other menus are pull-down menus from the bar menu. The format of this section shows the menu layout as it would appear on the screen, the menu attributes, and the menu item attributes. The menu attributes include the following items:

Menu Name - Name assigned to the menu.

Data Base - The database assigned to this menu.

Embedded Code Before - Logical indicator if dBASE code has been embedded in the application to be executed before the menu is displayed.

Embedded Code After - Logical indicator if dBASE code has been embedded in the application to be executed after the menu is exited.

After the menu attributes are described, the attributes for each item in the menu is described. The menu item attributes include the following items:

Item No. - Sequential number of the menu items. This number corresponds with the items in the menu layout.

Action - The action to be taken when this menu item is selected. The following actions are used in this program:

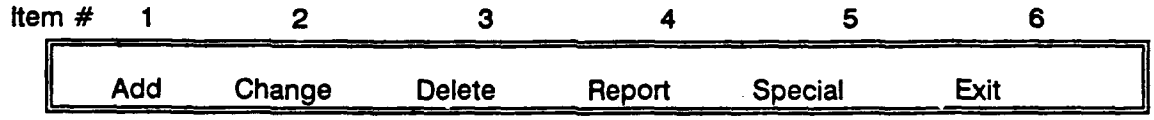
Open Menu	-	Opens the named menu
Quit	-	Exit the program
Run Program	-	Run a dBASE program
Append	-	Add a record to the data base
Edit	-	Modify a record in the database
Delete Records	-	Delete a record from the data base

Run Report	-	Generate the named report
dBASE Code	-	Execute the embedded dBASE code
No Action	-	Ignores user selection

Data Base - The database assigned to this menu item.

Embedded Code Before - Logical indicator if dBASE code has been embedded in the application to be executed before the menu item is executed.

Embedded Code After - Logical indicator if dBASE code has been embedded in the application to be executed after the menu item is executed.



MENU NAME: MAIN DATA BASE: DUMMY
Embedded Code Before: Yes Embedded Code After: No

ITEM NO.	ACTION	DATA BASE	EMBEDDED CODE BEFORE	AFTER	
1	Open Menu: ADD		No	No	
2	Open Menu: CHANGE		No	No	
3	Open Menu: DELETE			No	No
4	Open Menu: REPORT		No	No	
5	Open Menu: SPECIAL		No	No	
6	Quit		No	No	

ITEM#

1	Automated Data
2	COA Analysis
3	Questionnaires
4	Observations
5	CCAB
6	Scores

Menu Name: ADD
Embedded Code Before: No

Data Base:
Embedded Code After: No

ITEM NO.	ACTION	DATA BASE	EMBEDDED CODE BEFORE	AFTER
1	Run Program: DBASE_LD		No	No
2	Open Menu: ADDCOA		No	No
3	Open Menu: ADDQST		No	No
4	Open Menu: ADDOBS		No	No
5	Append: CCAB	CCAB	No	No
6	Open Menu: ADDSCOR		No	No

ITEM#

1	COA Analysis
2	Questionnaires
3	Observations
4	CCAB
5	Scores

Menu Name: CHANGE
Embedded Code Before: No

Data Base:
Embedded Code After: No

ITEM NO.	ACTION	DATA BASE	EMBEDDED CODE BEFORE	AFTER
1	Open Menu: CHGCOA		No	No
2	Open Menu: CHGQST		No	No
3	Open Menu: CHGOBS		No	No
4	Edit: CCAB	CCAB	Yes	No
5	Open Menu: CHGSCOR		No	No

ITEM#

1	Automated Data
2	COA Analysis
3	Questionnaires
4	Observations
5	CCAB
6	Whole Session
7	Scores

Menu Name: DELETE
Embedded Code Before: No

Data Base:
Embedded Code After: No

ITEM NO.	ACTION	DATA BASE	EMBEDDED CODE BEFORE	EMBEDDED CODE AFTER
1	Run Program: DELAUTO		No	No
2	Open Menu: DELCOA		No	No
3	Open Menu: DELQST		No	No
4	Open Menu: DELOBS		No	No
5	Delete Records	CCAB	Yes	No
6	Run Program: DELEXPR		No	No
7	Open Menu: DELSCOR		No	No

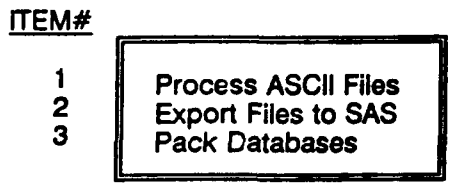
ITEM#

1	Automated Data
2	COA Analysis
3	Questionnaires
4	Observations
5	CCAB
6	Scores
7	All

Menu Name: REPORT
Embedded Code Before: No

Data Base:
Embedded Code After: No

ITEM NO.	ACTION	DATA BASE	EMBEDDED CODE BEFORE	EMBEDDED CODE AFTER
1	Open Menu: RPTAUTO		No	No
2	Open Menu: RPTCOA		No	No
3	Open Menu: RPTQST		No	No
4	Open Menu: RPTOBS		No	No
5	Run Report: CCAB	CCAB	Yes	No
6	Open Menu: RPTSCOR		No	No
7	dBASE Code		No	No



Menu Name: SPECIAL
Embedded Code Before: No

Data Base:
Embedded Code After: No

ITEM NO.	ACTION	DATA BASE	EMBEDDED CODE BEFORE	EMBEDDED CODE AFTER
1	Run Program: EDDIC_LD		No	No
2	dBASE Code		No	No
3	Run Program: PACK		No	No

Menu Name: ADDCOA
Embedded Code Before: No

Data Base:
Embedded Code After: No

ITEM NO.	ACTION	DATA BASE	EMBEDDED CODE BEFORE	EMBEDDED CODE AFTER
1	Append: COAATM1	COAATM1	No	No
2	Append: COAATM2	COAATM2	No	Yes
3	Append: COAATWT	COAATWT	No	Yes
4	Append: COAATSC	COAATSC	No	No

ITEM#	
1	COA Task Evaluation
2	Demographics
3	HMI (EDDIC)
4	HMI (EDDIC/COAAT)
5	Personal Style
6	Situation Awareness
7	Workload Assessment

Menu Name: ADDQST
Embedded Code Before: No

Data Base:
Embedded Code After: No

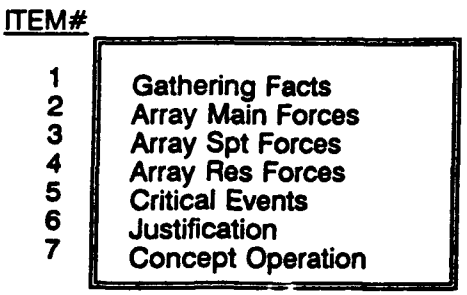
ITEM NO.	ACTION	DATA BASE	EMBEDDED CODE BEFORE	AFTER	
1	Append: TASKEVAL	TASKEVAL	No	No	
2	Append: PERSON	PERSON	No	No	
3	Append: HMIED	HMIED		No	No
4	Append: HMIEDCT	HMIEDCT	No	No	
5	Append: PERSTYLE	PERSTYLE	No	No	
6	Append: SITAWARE	SITAWARE	No	No	
7	Append: WORKASMT	WORKASMT		No	No

ITEM#	
1	Team Profile
2	Time Line

Menu Name: ADDOBS
Embedded Code Before: No

Data Base:
Embedded Code After: No

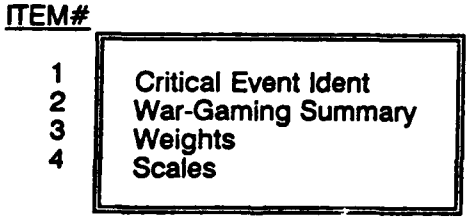
ITEM NO.	ACTION	DATA BASE	EMBEDDED CODE BEFORE	AFTER
1	Append: TEAMPRF	TEAMPRF	No	No
2	Append: TIMELINE	TIMELINE	No	No



Menu Name: ADDSCOR
Embedded Code Before: No

Data Base:
Embedded Code After: No

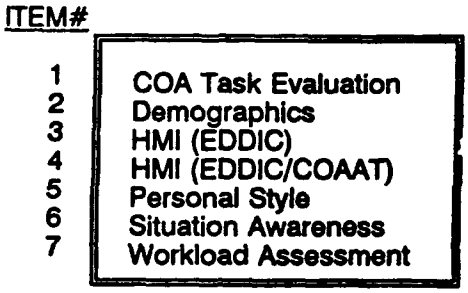
ITEM NO.	ACTION	DATA BASE	EMBEDDED CODE BEFORE	EMBEDDED CODE AFTER
1	Append: SCFACTS	SCFACTS	No	No
2	Run Program: AR_FORCE		Yes	No
3	Run Program: AR_FORCE		Yes	No
4	Run Program: AR_FORCE		Yes	No
5	Run Program: IDENTCE	SCCRTEVT	No	No
6	Append: SCJUST	SCJUST	No	No
7	Append: SCCNOP	SCCNOP	No	No



Menu Name: CHGCOA
Embedded Code Before: No

Data Base:
Embedded Code After: No

ITEM NO.	ACTION	DATA BASE	EMBEDDED CODE BEFORE	EMBEDDED CODE AFTER
1	Edit: COAATM1	COAATM1	Yes	No
2	Edit: COAATM2	COAATM2	Yes	No
3	Edit: COAATWT	COAATWT	Yes	No
4	Edit: COAATSC	COAATSC	Yes	No



Menu Name: CHGQST
Embedded Code Before: No

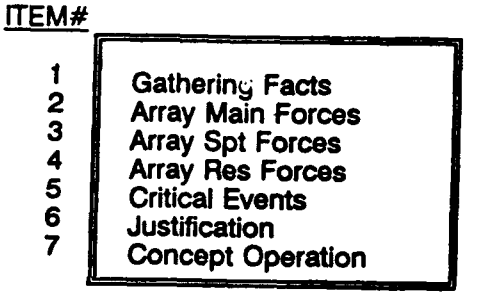
Data Base:
Embedded Code After: No

ITEM NO.	ACTION	DATA BASE	EMBEDDED CODE BEFORE	EMBEDDED CODE AFTER
1	Edit: TASKEVAL	TASKEVAL	Yes	No
2	Edit: PERSON	PERSON	Yes	No
3	Edit: HMIED	HMIED	Yes	No
4	Edit: HMIEDCT	HMIEDCT	Yes	No
5	Edit: PERSTYLE	PERSTYLE	Yes	No
6	Edit: SITAWARE	SITAWARE	Yes	No
7	Edit: WORKASMT	WORKASMT	Yes	No

Menu Name: CHGOBS
Embedded Code Before: No

Data Base:
Embedded Code After: No

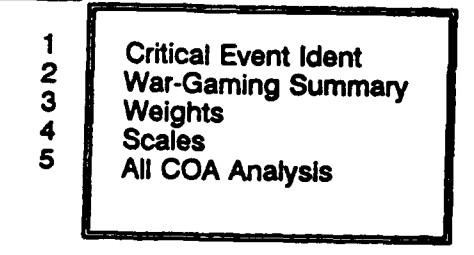
ITEM NO.	ACTION	DATA BASE	EMBEDDED CODE BEFORE	EMBEDDED CODE AFTER
1	Edit: TEAMPRF	TEAMPRF	Yes	No
2	Edit: TIMELINE	TIMELINE	Yes	No



Menu Name: CHGSCOR
Embedded Code Before: No

Data Base:
Embedded Code After: No

ITEM NO.	ACTION	DATA BASE	EMBEDDED CODE BEFORE	EMBEDDED CODE AFTER
1	Edit: SCFACTS	SCFACTS	Yes	No
2	Run Program: AR_FORCE		Yes	No
3	Run Program: AR_FORCE		Yes	No
4	Run Program: AR_FORCE		Yes	No
5	Run Program: IDENTCE		No	No
6	Edit: SCJUST	SCJUST	Yes	No
7	Edit: SCCNOP	SCCNOP	Yes	No



Menu Name: DELCOA
Embedded Code Before: No

Data Base:
Embedded Code After: No

ITEM NO.	ACTION	DATA BASE	EMBEDDED CODE BEFORE	EMBEDDED CODE AFTER
1	Delete Records	COAATM1	Yes	No
2	Delete Records	COAATM2	Yes	No
3	Delete Records	COAATWT	Yes	No
4	Delete Records	COAATSC	Yes	No
5	Run Program: DELACOA		No	No

ITEM#

1	COA Task Evaluation
2	Demographics
3	HMI (EDDIC)
4	HMI (EDDIC/COAAT)
5	Personal Style
6	Situation Awareness
7	Workload Assessment
8	All Questionnaires

Menu Name: DELQST
Embedded Code Before: No

Data Base:
Embedded Code After: No

ITEM NO.	ACTION	DATA BASE	EMBEDDED CODE BEFORE	EMBEDDED CODE AFTER
1	Delete Records	TASKEVAL	Yes	No
2	Delete Records	PERSON	Yes	No
3	Delete Records	HMIED	Yes	No
4	Delete Records	HMIEDCT	Yes	No
5	Delete Records	PERSTYLE	Yes	No
6	Delete Records	SITAWARE	Yes	No
7	Delete Records	WORKASMT	Yes	No
8	Run Program: DELAQST		No	No

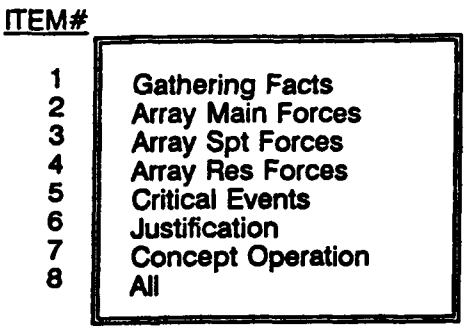
ITEM#

1	Team Profile
2	Time Line
3	All Observe

Menu Name: DELOBS
Embedded Code Before: No

Data Base:
Embedded Code After: No

ITEM NO.	ACTION	DATA BASE	EMBEDDED CODE BEFORE	EMBEDDED CODE AFTER
1	Delete Records	TEAMPRF	Yes	No
2	Delete Records	TIMELINE	Yes	No
3	Run Program: DELAOBS		No	No



Menu Name: DELSCOR
Embedded Code Before: No

Data Base:
Embedded Code After: No

ITEM NO.	ACTION	DATA BASE	EMBEDDED CODE BEFORE	EMBEDDED CODE AFTER
1	Delete Records	SCFACTS	Yes	No
2	Delete Records	SCFORCE	Yes	No
3	Delete Records	SCFORCE	Yes	No
4	Delete Records	SCFORCE	Yes	No
5	Delete Records	SCCRTEVT	Yes	No
6	Delete Records	SCJUST	Yes	No
7	Delete Records	SCCNOP	Yes	No
8	Run Program: DELASCR		Yes	No

ITEM#

1	View Situation
2	View Reference
3	Map Control
4	Window Operations
5	New Cntrl Measures
6	BLUEFOR Task Org
7	Unit Location Update
8	All

Menu Name: RPTAUTO
Embedded Code Before: No

Data Base:
Embedded Code After: No

ITEM NO.	ACTION	DATA BASE	EMBEDDED CODE BEFORE	EMBEDDED CODE AFTER
1	Run Report: C2_RQST	C2_RQST	Yes	No
2	Run Report: REF_RQST	REF_RQST	Yes	No
3	Run Report: MAP_CTRL	MAP_CTRL	Yes	No
4	Run Report: WINDOW	WINDOW	Yes	No
5	No Action			
6	No Action			
7	No Action			
8	Run Program: ALLAUTO		No	No

ITEM#

1	Critical Event Ident
2	War-Gaming Summary
3	Objective Measures
4	Subjectives Measures
5	All

Menu Name: RPTCOA
Embedded Code Before: No

Data Base:
Embedded Code After: No

ITEM NO.	ACTION	DATA BASE	EMBEDDED CODE BEFORE	EMBEDDED CODE AFTER
1	Run Report: COAATM1	COAATM1	Yes	No
2	Run Report: COAATM2	COAATM2.QBE	Yes	No
3	Run Report: COAOBJ	COAOBJ.QBE	Yes	No
4	Run Report: COASUB	COASUB.QBE	Yes	No
5	Run Program: COAATPRT		No	No

ITEM#

1	COA Task Evaluation
2	Demographics
3	HMI (EDDIC)
4	HMI (EDDIC/COAAT)
5	Personal Style Data
6	Personal Style Profi
7	Situation Awareness
8	Workload Assessment
9	All Questionnaires

Menu Name: RPTQST Data Base:
Embedded Code Before: No Embedded Code After: No

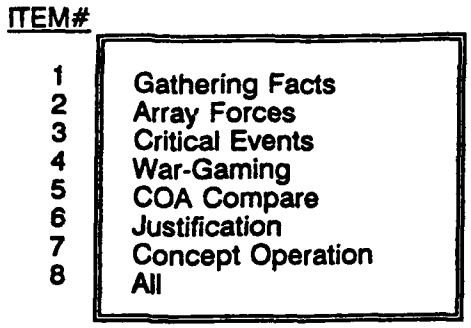
ITEM NO.	ACTION	DATA BASE	EMBEDDED CODE BEFORE	AFTER
1	Run Report: TASKEVAL	TASKEVAL	Yes	No
2	Run Report: PERSON	PERSON	Yes	No
3	Run Report: HMIED	HMIED		Yes No
4	Run Report: HMIEDCT	HMIEDCT	Yes	No
5	Run Report: STYLDATA	PERSTYLE	Yes	No
6	Run Report: PERSTYLE	PERSTYLE.QBE	Yes	No
7	Run Program: RPTSITA	Yes	No	
8	Run Report: WORKASMT	WORKASMT	Yes	No
9	Run Program: ALLQUEST		No	No

ITEM#

1	Team Profile
2	Time Line
3	All

Menu Name: RPTOBS Data Base:
Embedded Code Before: No Embedded Code After: No

ITEM NO.	ACTION	DATA BASE	EMBEDDED CODE BEFORE	AFTER
1	Run Report: TEAMPRF	TEAMPRF	Yes	No
2	Run Report: TIMELINE	TIMELINE	Yes	No
3	Run Program: ALLOBS		No	No



Menu Name: RPTSCOR Data Base:
Embedded Code Before: No Embedded Code After: No

ITEM NO.	ACTION	DATA BASE	EMBEDDED CODE BEFORE	AFTER
1	Run Program: RPTFACT		Yes	No
2	Run Program: ARREPORT		Yes	No
3	Run Program: RPTSCCE		Yes	No
4	Run Program: RPTWGAM		Yes	No
5	Run Program: RPTCCOA		Yes	No
6	Run Program: RPTJUST		Yes	No
7	Run Program: RPTCNOP		Yes	No
8	Run Program: RPTSCALL		Yes	No

Some of the experiment scoring programs require certain data to exist in certain data bases. The data defines thresholds and expert solutions required for generation of scoring reports. By having this data in a data base, a scoring report can be changed without changing the dBASE source code. Tables 4-1 through 4-3 show the programs that require the data and which data is required.

Table 4-1. Data Required by ARREPORT

Data Base	FIELD	VALUE	DESCRIPTION
SCFORCE	SEQ_NO	AEXPT	Expert score for the main attack for COA 1 in balance condition A.
	COA	1	
	POWER	26.30	
	MISSION	M	
SCFORCE	SEQ_NO	AEXPT	Expert score for the supporting attack for COA 1 in balance condition A.
	COA	1	
	POWER	11.10	
	MISSION	S	
SCFORCE	SEQ_NO	AEXPT	Expert score for the reserve forces for COA 1 in balance condition A.
	COA	1	
	POWER	7.50	
	MISSION	V	
SCFORCE	SEQ_NO	AEXPT	Expert score for the main attack for COA 2 in balance condition A.
	COA	2	
	POWER	26.60	
	MISSION	M	
SCFORCE	SEQ_NO	AEXPT	Expert score for the supporting attack for COA 2 in balance condition A.
	COA	2	
	POWER	11.20	
	MISSION	S	
SCFORCE	SEQ_NO	AEXPT	Expert score for the reserve forces for COA 2 in balance condition A.
	COA	2	
	POWER	7.10	
	MISSION	V	
SCFORCE	SEQ_NO	BEXPT	Expert score for the main attack for COA 1 in balance condition B.
	COA	1	
	POWER	26.60	
	MISSION	M	
SCFORCE	SEQ_NO	BEXPT	Expert score for the supporting attack for COA 1 in balance condition B.
	COA	1	
	POWER	11.20	
	MISSION	S	
SCFORCE	SEQ_NO	BEXPT	Expert score for the reserve forces for COA 1 in balance condition B.
	COA	1	
	POWER	7.10	
	MISSION	V	

Table 4-1. Data Required by ARREPORT (*Continued*)

DATA BASE	FIELD	VALUE	DESCRIPTION
SCFORCE	SEQ_NO	BEXPT	Expert score for the main attack for COA 2 in balance condition B.
	COA	2	
	POWER	26.30	
	MISSION	M	
SCFORCE	SEQ_NO	BEXPT	Expert score for the supporting attack for COA 2 in balance condition B.
	COA	2	
	POWER	11.10	
	MISSION	S	
SCFORCE	SEQ_NO	BEXPT	Expert score for the reserve forces for COA 2 in balance condition B.
	COA	2	
	POWER	7.50	
	MISSION	V	
SCFORCE	SEQ_NO	PRCNT	Percentage threshold around the experts score to count the participants score as a match.
	COA	0	
	POWER	10.00	
SCFORCE	SEQ_NO	WEGHT	Weight to assign to the participants score for a match on the main attack.
	POWER	50.00	
	MISSION	M	
SCFORCE	SEQ_NO	WEGHT	Weight to assign to the participants score for a match on the supporting attack.
	POWER	20.00	
	MISSION	S	
SCFORCE	SEQ_NO	WEGHT	Weight to assign to the participants score for a match on the reserves.
	POWER	30.00	
	MISSION	V	

Table 4-2. Data Required by RPTWGAM

DATA BASE	FIELD	VALUE	DESCRIPTION
COAATM2	SEQ_NO	AEXPT	Expert war-gaming results for COA 1 in balance condition A. The main attack avenue (A) is defined in AVENUE. It is used to sum the FEBA and TIME for only the main attack.
	COA	1	
	AVENUE	A	
	FR_PERS	0	
	FR_EQUIP	341	
	EN_PERS	0	
	EN_EQUIP	219	
	POL	0	
	AMMO	0	
	FEBA	0	
	TIME	46.4	
COAATM2	SEQ_NO	AEXPT	Expert war-gaming results for COA 2 in balance condition A. The main attack avenue (B) is defined in AVENUE. It is used to sum the FEBA and TIME for only the main attack.
	COA	2	
	AVENUE	B	
	FR_PERS	0	
	FR_EQUIP	327	
	EN_PERS	0	
	EN_EQUIP	224	
	POL	0	
	AMMO	0	
	FEBA	0	
	TIME	33.2	
COAATM2	SEQ_NO	BEXPT	Expert war-gaming results for COA 1 in balance condition B. The main attack avenue (B) is defined in AVENUE. It is used to sum the FEBA and TIME for only the main attack.
	COA	1	
	AVENUE	B	
	FR_PERS	0	
	FR_EQUIP	327	
	EN_PERS	0	
	EN_EQUIP	224	
	POL	0	
	AMMO	0	
	FEBA	0	
	TIME	33.2	
COAATM2	SEQ_NO	BEXPT	Expert war-gaming results for COA 2 in balance condition B. The main attack avenue (A) is defined in AVENUE. It is used to sum the FEBA and TIME for only the main attack.
	COA	2	
	AVENUE	A	
	FR_PERS	0	
	FR_EQUIP	341	
	EN_PERS	0	
	EN_EQUIP	219	
	POL	0	
	AMMO	0	
	FEBA	0	
	TIME	46.4	

Table 4-2. Data Required by RPTWGAM (Continued)

DATA BASE	FIELD	VALUE	DESCRIPTION
COAATM2	SEQ_NO	APRCT	Acceptable percentage threshold of the expert war-gaming results for each COA divided by the base COA results for balance condition A. The base COA is contained in COA and the percentage is in FR_PERS.
	COA	2	
	FR_PERS	10	
COAATM2	SEQ_NO	BPRCT	Acceptable percentage threshold of the expert war-gaming results for each COA divided by the base COA results for balance condition B. The base COA is contained in COA and the percentage is in FR_PERS.
	COA	1	
	FR_PERS	10	
COAATM2	SEQ_NO	WEGHT	The weight assigned to each category for a participant value that falls within the acceptable range.
	FR_PERS	0	
	FR_EQUIP	33	
	EN_PERS	0	
	EN_EQUIP	33	
	POL	0	
	AMMO	0	
	FEBA	0	
	TIME	33	

Table 4-3. Data Required by RPTCCOA

DATA BASE	FIELD	VALUE	DESCRIPTION
COAATWT	SEQ_NO	PRCNT	Percentage threshold of the experts weights for the participants weights to be acceptable.
	FR_PERS	20	
COAATWT	SEQ_NO	EXPRT	Expert weights assigned to objective and subjective measures.
	FR_PERS	0	
	FR_EQUIP	100	
	EN_PERS	0	
	EN_EQUIP	60	
	POL	0	
	AMMO	0	
	FEBA	0	
	TIME	80	
	SUB_A	100	
	SUB_B	60	
	SUB_C	80	
	SUB_D	40	
	SUB_E	80	
	SUB_F	0	
	SUB_G	0	
	SUB_H	0	
COAATSC	SEQ_NO	AEXPT	Expert subjective scales for COA 1 in balance condition A.
	COA	1	
	SSUB_A	5	
	SSUB_B	6	
	SSUB_C	4	
	SSUB_D	7	
	SSUB_E	5	
COAATSC	SEQ_NO	AEXPT	Expert subjective scales for COA 2 in balance condition A.
	COA	2	
	SSUB_A	8	
	SSUB_B	8	
	SSUB_C	7	
	SSUB_D	6	
	SSUB_E	8	
COAATSC	SEQ_NO	BEXPT	Expert subjective scales for COA 1 in balance condition B.
	COA	1	
	SSUB_A	8	
	SSUB_B	8	
	SSUB_C	7	
	SSUB_D	6	
	SSUB_E	8	

Table 4-3. Data Required by RPTCCOA (Continued)

DATA BASE	FIELD	VALUE	DESCRIPTION
COAATSC	SEQ_NO	BEXPT	Expert subjective scales for COA 2 in balance condition B.
	COA	2	
	SSUB_A	5	
	SSUB_B	6	
	SSUB_C	4	
	SSUB_D	7	
	SSUB_E	5	
COAATSC	SEQ_NO	WEGHT	Weight to assign the absolute and relative subjective scaling scores. SFR_PERS is absolute and SFR_EQUIP is relative. With the current numbers, each score in the relative scaling counts twice as much as a score in the absolute scale.
	SFR_PERS	1	
	SFR_EQUIP	2	

Data Bases. The data bases for the dBASE programs include the dBASE data bases, report layouts, form layouts, program files that are called by a program, and queries. Table 4-4 shows which files are used by each program within the EDDIC Application.

Table 4-4. EDDIC Application File Usage

PROGRAM	DATA BASES	REPORTS	FORMS	PROGRAMS	QUERIES
ACDATA			PERSTYLE	CHKDATA	
ALLAUTO	C2_RQST	C2_RQST		GETEXPR	
	MAP_CTRL	MAP_CTRL		PRINTSET	
	REF_RQST	REF_RQST			
	WINDOW	WINDOW			
ALLOBS	TEAMPRF	TEAMPRF		GETEXPR	
	TIMELINE	TIMELINE		PRINTSET	
ALLQUEST	HMIED	HMIED		GETEXPT	PERSTYLE
	HMIEDCT	HMIEDCT		PRINTSET	
	PERSON	PERSON			
	PERSTYLE	PERSTYLE			
	SITAWARE	SITAWARE			
	TASKEVAL	STLYDATA			
		TASKEVAL			
ARREPORT	COAATSC	ARREPORT			FRCERPT
AR_FORCE	SCFORCE			DSP_LIST	
	SCPOWER				

Table 4-4. EDDIC Application File Usage (Continued)

PROGRAM	DATA BASES	REPORTS	FORMS	PROGRAMS	QUERIES
COAATPRT	COAATM1	COAATM1 COAOBJ COASUB		GETEXPR PRINTSET	COAATM2 COAOBJ COASUB
DBASE_LD	C2_RQST COAATM1 COAATM2 COAATSC COAATWT CTL_RQST ED_LUT ED_MAP ED_WIND EDC2RQ EDCOTM1 EDCOTM2 EDCOTSC EDCOTWT EDCTLRQ EDNEWC2 EDREFRQ EDSTBLTO EDSTCMDL EDSTCMLC EDSTNWCM EDSTOPTO EDSTRQST EDSTULOC LUT_CTRL MAP_CTRL NEW_C2 REF_RQST SITCMDEL SITCMLOC SITNEWCM SITRQST SITTASKO SITULOC WINDOW				
DELACOA	COAATM1 COAATM2 COAATSC COAATWT			DELWIND	
DELAOBS	TEAMPRG TIMELINE			DELWIND	

Table 4-4. EDDIC Application File Usage (Continued)

PROGRAM	DATA BASES	REPORTS	FORMS	PROGRAMS	QUERIES
DELAQST	HMIED HMIEDCT PERSON PERSTYLE SITAWARE TASKEVAL WORKASMT			DELWIND	
DELASCR	SCCNOP SCCRTEVT SCFACTS SCJUST			DELWIND	
DELAUTO	C2_RQST COAATM1 COAATM2 COAATSC COAATWT CTL_RQST LUT_CTRL MAP_CTRL NEW_C2 REF_RQST SITCMDEL SITCMLOC SITNEWCM SITRQST SITTASKO SITULOC WINDOW			DELWIND	

Table 4-4. EDDIC Application File Usage (Continued)

PROGRAM	DATA BASES	REPORTS	FORMS	PROGRAMS	QUERIES
DELEXPR	C2_RQST CCAB COAATM1 COAATM2 COAATSC COAATWT CTL_RQST HMIED HMIEDCT LUT_CTRL MAP_CTRL NEW_C2 PERSON PERSTYLE REF_RQST SITAWARE SITCMDEL SITCMLOC SITNEWCM SITRQST SITTASKO SITULOC TASKEVAL TEAMPRF TIMELINE WINDOW WORKASMT			DELWIND	
EDDIC_LD	BUNXREF CMXREF CTL_XREF HLP_XREF HST_XREF REF_XREF RUNXREF		SESSION	ED_TRANS	

Table 4-4. EDDIC Application File Usage (Continued)

PROGRAM	DATA BASES	REPORTS	FORMS	PROGRAMS	QUERIES
ED_TRANS	BUNXREF CMXREF CTL_XREF ED_LUT ED_MAP ED_STBLTO ED_WIND EDC2RQ EDCOTM1 EDCOTM2 EDCOTSC EDCOTWT EDCTLRQ EDNEWC2 EDREFRQ EDSTCMDL EDSTCMLC EDSTNWCM EDSTOPTO EDSTRQST EDSTULOC HLP_XREF HST_XREF REF_XREF RUNXREF				
IDENTCE	SCCRTEVT			GETEXPR	
PACK	C2_RQST COAATM1 COAATM2 COAATSC COAATWT CTL_RQST LUT_CTRL MAP_CTRL NEW_C2 REFRQST SITCMDEL SITCMLOC SITNEWCM SITRQST SITTASKO SITULOC WINDOW				

Table 4-4. EDDIC Application File Usage (Continued)

PROGRAM	DATA BASES	REPORTS	FORMS	PROGRAMS	QUERIES
RPTCCOA	COAATSC COAATWT				COAOBJ COASUB
RPTCNOP	SCCNOP				
RPTFACT	SCFACTS				
RPTJUST	SCJUST				
RPTSCALL				ARREPORT RPTCCOA RPTCNOP RPTFACT RPTJUST RPTSCCE RPTWGAM	
RPTSCCE	COAATSC SCCRTEVT	SCCRTEVT			
RPTWGAM	COAATM2 COAATSC				

4.4.2 EDDIC EX

Abstract. Exports the scenario data base to ASCII files.

Major Capability. This program exports the scenario data (maintained by the SCENARIO program) to ASCII files for the purpose of moving the files to the Sun fileserver.

Special Instructions. This program was developed in dBASE IV because dBASE III Plus does not allow printing to a file.

The scenario data base files must be copied into EXPORT directory before running this program.

The following section describes the EDDIC scenario export application. dBASE applications consist of a group of menus chained together. Each item of a menu is assigned an action and can have a data base assigned to it. The following menus exist in the EDDIC scenario export application.

Item #	1	2	3	4	5
	UPDATE DAY	EXPORT ITEM	EXPORT BLUEFOR	EXPORT OPFOR	EXIT

Menu Name: EDDIC
Embedded Code Before: No

Data Base: DAY
Embedded Code After: No

ITEM NO.	ACTION	DATA BASE	EMBEDDED CODE BEFORE	EMBEDDED CODE AFTER
1	Browse File		No	No
2	Open Menu: EX_ITEM		No	No
3	Run Program: EX_BALL		No	No
4	Run Program: EX_RALL		No	No
5	Quit		No	No

ITEM#	
1	BLUEFOR TASK ORG
2	BLUEFOR EQUIP
3	BLUEFOR AMMO
4	BLUEFOR PERS
5	BLUEFOR FUEL
6	OPFOR TASK ORG
7	OPFOR EQUIP

Menu Name: EX_ITEM
Embedded Code Before: No

Data Base:
Embedded Code After: No

ITEM NO.	ACTION	DATA BASE	EMBEDDED CODE BEFORE	EMBEDDED CODE AFTER
1	Run Program: EX_BTSK		No	No
2	Run Program: EX_BEQP		No	No
3	Run Program: EX_BAMM		No	No
4	Run Program: EX_BPERS		No	No
5	Run Program: EX_BFUL		No	No
6	Run Program: EX_RTSK		No	No
7	Run Program: EX_REQP		No	No

Data Bases. The data bases for the dBASE programs include the dBASE data bases, report layouts, form layouts, program files that are called by a program, and queries. Table 4-5 shows which files are used by each program within the EDDIC Export Application.

Table 4-5. EDDIC Export Application File Usage

PROGRAM	DATA BASES	REPORTS	FORMS	PROGRAMS	QUERIES
EX_BALL				EX_BAMM EX_BEQP EX_BFUL EX_BLOC EX_BPRS EX_BTSK	
EX_BAMM	BASEUNIT COMPANY1 DAY				
EX_BEQP	BASEUNIT COMPANY1 DAY				
EX_BFUL	BASEUNIT COMPANY1 DAY				
EX_BLOC	BATTAL1 BRIGADE1 COMPANY1 DAY DIVISN1			PRNTLOC	
EX_BPRS	BASEUNIT COMPANY1 DAY				
EX_BTSK	BATTAL1 BRIGADE1 COMPANY1 DAY DIVISN1			PRNTASK	
EX_CM	CNTLMSR1 DAY				
EX_OPLAN	OPLAN				
EX_RALL				EX_REQP EX_RTSK EX_RUNIT	
EX_REQP	DAY RBASEUNI RCOMPNY1				

Table 4-5. EDDIC Export Application File Usage (Continued)

PROGRAM	DATA BASES	REPORTS	FORMS	PROGRAMS	QUERIES
EX_RTSK	DAY				
EX_RUNIT	DAY RCOMPNY1				

4.4.3 Scenario

Abstract. Maintains the scenario data base.

Major Capability. All EDDIC scenario data except the following can be updated and printed with this program:

- OPFOR Task Organization
- OPFOR Unit Locations
- OPFOR Unit Status

Special Instructions. This program can be executed in either dBASE III Plus or dBASE IV. It executes much faster in dBASE IV.

Data Bases. The data bases for the dBASE programs include the dBASE data bases, report layouts, form layouts, program files that are called by a program, and queries. Table 4-6 shows which files are used by each program within the EDDIC Scenario Application.

Table 4-6. EDDIC Scenario Application File Usage

PROGRAM	DATA BASES	REPORTS	FORMS	PROGRAMS	QUERIES
ADJUST	DAY PERDISP			OPENLR OPEN PERSTRNG	
AMOUNT	READONLY				
BASEUPDT			BASELOOK BASEUNIT	BSREPORT	
BDETASK			BDEADD BDEFORM		
BNTASK			BNADD BNFORM		
CNTL_MSR	CNTLMSR1	CNTL_MSR		READONLY	

Table 4-6. EDDIC Scenario Application File Usage (Continued)

PROGRAM	DATA BASES	REPORTS	FORMS	PROGRAMS	QUERIES
COUPDATE			COFORM		
DAY	DAY	DAY	DAY		
COPYDAY				OPENDB	
DAYSEL	DAY				
DIVTASK			DIVADD DIVFORM		
OPEN	BASEUNIT BATTAL1 BRIGADE1 COMPANY1 DIVISN1 RBASEUNI RBATTL1 RBRIGAD1 RCOMPNY1 RDIVISN1				
OPENDB	BASEUNIT BATTAL1 BRIGADE1 COMPANY1 DIVISN1 RBASEUNI RBATTL1 RBRIGAD1 RCOMPNY1 RDIVISN1				
OPENLR	LOSSRATE				
OPLAN	OPLAN	OPLAN	OPLAN		
PERCENT	PERCENT			READONLY	
PERSTRNG	BDEDISP PERDISP	PERDISP		OPENLR READONLY	

Table 4-6. EDDIC Scenario Application File Usage (Continued)

PROGRAM	DATA BASES	REPORTS	FORMS	PROGRAMS	QUERIES
SCENARIO				ADJUST AMOUNT BASEUPDT CNTL_MSR DAY DAYCOPY DAYSEL INIT OPENDB OPLAN PERCENT STRNGRPT TASKORG	
STRNGRPT	EQDISPLA			BNSTRNG COREPORT FRNTPC STRENGTH STREPORT	
TASKORG				BDETASK BNTASK COUPDATE DIVTASK PRNSTAT PRNTASK READONLY	
VERTASK	VERTASK	VERTASK	VERTASK		

APPENDIX A - Ada UTILITY SPECIFICATIONS

This appendix contains the Ada package specifications for the EDDIC utility packages. The appendix is divided into the following major utility categories:

COMMON	-	Ada types that are available throughout the EDDIC system.
UED	-	General EDDIC utilities such as math functions, string functions, and list and queue managers.
UFM	-	Form Manager utilities
UIN	-	Internet communications utilities
UIW	-	Color image utilities
UTM	-	Tactical map utilities
UUX	-	Unix command utilities
UWN	-	Window display and control utilities

COMMON Utility Package Specifications

The following package specifications are contained in the COMMON function:

```
CDB_C2_PRODUCT_DB
CTL_CONTROL_DB
FDB_REFERENCE_DB
HDB_HELP_DB
LUT_SYSTEM
MSG_MESSAGE
SDB_SITUATION_DB
SYSTEM_PACKAGE
TSTM_DB
```

```
--cpc package specification name: CDB_C2_PRODUCT_DB
--
--cpc description: The CDB_C2_PRODUCT_DB cpc describes the objects that are used
--                 for interacting with the c2 product databases.
--
--cpc design notes:
--
--cpc package author: Bruce Packard
--                    Science Applications International Corporation
--                    424 Delaware, Suite C3
--                    Leavenworth, KS 66048
--

with SYSTEM_PACKAGE;        use SYSTEM_PACKAGE;

package CDB_C2_PRODUCT_DB is

   -- Number of records in header and product database
   subtype CDB_NUM_HEADER_REC    is  SYS_DB_SIZE  range 0..5000;
   subtype CDB_NUM_PRODUCT_REC   is  SYS_DB_SIZE  range  0..20000;

   -- Product Description record
   type CDB_PRODUCT_DESC_TYPE  is
      record
         CDB_PRODUCT_CAT         :  SYS_PRODUCT_CAT;
         CDB_PRODUCT_HDR_START   :  CDB_NUM_HEADER_REC;
         CDB_PRODUCT_HDR_END     :  CDB_NUM_HEADER_REC;
         CDB_PRODUCT_START       :  CDB_NUM_PRODUCT_REC;
         CDB_PRODUCT_END         :  CDB_NUM_PRODUCT_REC;
         CDB_PRODUCT_DATE        :  SYS_DATE_TIME;
         CDB_PRODUCT_OPPLAN      :  SYS_OPPLAN;
      end record;

   CDB_PRODUCT_DESC_REC          :  CDB_PRODUCT_DESC_TYPE;

   -- Report Header record
   CDB_HEADER_SIZE               :  SYS_HEADER_LENGTH   := 252;

   type CDB_HEADER_TYPE  is
      record
         CDB_HEAD_NUMBER_CHAR    :  SYS_HEADER_LENGTH  range 0..
                                    CDB_HEADER_SIZE;
         CDB_HEADER_TEXT         :  string (1..CDB_HEADER_SIZE);
      end record;
   type CDB_HEADER_POINT  is access CDB_HEADER_TYPE;
   CDB_HEADER_REC                :  CDB_HEADER_POINT := new CDB_HEADER_TYPE;

   -- Report record
   CDB_PRODUCT_SIZE              :  SYS_PRODUCT_LENGTH   := 252;

   type CDB_PRODUCT_TYPE  is
      record
         CDB_REPT_NUMBER_CHAR    :  SYS_PRODUCT_LENGTH  range 0..
                                    CDB_PRODUCT_SIZE;
         CDB_PRODUCT_TEXT        :  string (1..CDB_PRODUCT_SIZE);
      end record;
```

```
type CDB_PRODUCT_POINT  is access CDB_PRODUCT_TYPE;
CDB_PRODUCT_REC              : CDB_PRODUCT_POINT :=  new CDB_PRODUCT_TYPE;

-- C2 Product List
subtype CDB_NUM_PRODUCT       is  SYS_WALKING_CELL  range 0..800;
subtype CDB_NUM_BUILD         is  SYS_WALKING_CELL  range 0..100;
subtype CDB_NUM_VW_MENU       is  SYS_WALKING_MENU  range 0..250;
subtype CDB_NUM_BL_MENU       is  SYS_WALKING_MENU  range 0..50;


type CDB_PROD_LIST_TYPE is array (CDB_NUM_PRODUCT range <>) of
     CDB_NUM_PRODUCT;

type CDB_PROD_LIST_POINT  is access CDB_PROD_LIST_TYPE;

subtype CDB_MSG_NAME_LEN       is INTEGER     range 1..80;
subtype CDB_PART_NAME_LEN      is INTEGER     range 1..10;

-- Summary message record
subtype CDB_MSG_NAME_TEXT is string (CDB_MSG_NAME_LEN);
subtype CDB_PART_NAME_TEXT is string (CDB_PART_NAME_LEN);
type CDB_ROUTING_ARRAY    is   array (SYS_PARTICIPANTS) of BOOLEAN;

type CDB_SUM_MESSAGE_REC is
   record
      CDB_C2_FROM         :   CDB_PART_NAME_TEXT;
      CDB_C2_TO           :   CDB_ROUTING_ARRAY;
      CDB_C2_SUBJECT      :   CDB_MSG_NAME_TEXT;
      CDB_C2_DAY          :   SYS_DAY;
      CDB_C2_TIME         :   SYS_TIME;
      CDB_C2_PRODUCT      :   CDB_NUM_PRODUCT;
   end record;
type CDB_SUM_MESSAGE_POINT  is access CDB_SUM_MESSAGE_REC;

-- Message Log record
type CDB_LOG_LIMIT is  range 0..100;
type CDB_MESSAGE_LOG is  array (CDB_LOG_LIMIT) of
     CDB_SUM_MESSAGE_REC;
type CDB_MESSAGE_LOG_POINT  is access CDB_MESSAGE_LOG;

type CDB_MESSAGE_LOG_REC is
   record
      CDB_COUNT           :   CDB_LOG_LIMIT;
      CDB_LIST            :   CDB_MESSAGE_LOG;
   end record;

-- Participant Record
type CDB_PART_REC is
   record
      CDB_TEXT            :   CDB_PART_NAME_TEXT;
      CDB_PART            :   SYS_PARTICIPANTS;
end record;

-- List of participants that a message can be sent to
   subtype CDB_PART_LIMIT is SYS_MENU_BUTTON_INDEX range 0..4;
   type CDB_PART_ARRAY  is  array (CDB_PART_LIMIT) of CDB_PART_REC;
```

```
   type CDB_PARTICIPANT_POINT  is access CDB_PART_ARRAY;
end CDB_C2_PRODUCT_DB;
```

```
--cpc package specification name: CTL_CONTROL_DB
--
--cpc description: The CTL_CONTROL_DB cpc describes the objects that are used
--                 for interacting with the experiment control databases.
--
--cpc design notes:
--
--cpc package author: Bruce Packard
--                    Science Applications International Corporation
--                    424 Delaware, Suite C3
--                    Leavenworth, KS 66048
--

with SYSTEM_PACKAGE;        use SYSTEM_PACKAGE;

package CTL_CONTROL_DB is

   -- Number of records in control database
   subtype CTL_NUM_PRODUCT_REC    is  SYS_DB_SIZE  range  0..10000;

   -- Product Description record
   type CTL_PRODUCT_DESC_TYPE  is
      record
         CTL_PRODUCT_TYPE        :  SYS_PRODUCT;
         CTL_PRODUCT_START       :  CTL_NUM_PRODUCT_REC;
         CTL_PRODUCT_END         :  CTL_NUM_PRODUCT_REC;
         CTL_PRODUCT_DATE        :  SYS_DATE_TIME;
      end record;

   CTL_PRODUCT_DESC_REC          :  CTL_PRODUCT_DESC_TYPE;

   -- Report record
   CTL_PRODUCT_SIZE              :  SYS_PRODUCT_LENGTH   := 252;

   type CTL_PRODUCT_TYPE  is
      record
         CTL_REPT_NUMBER_CHAR    :  SYS_PRODUCT_LENGTH  range 0..
                                    CTL_PRODUCT_SIZE;
         CTL_PRODUCT_TEXT        :  string (1..CTL_PRODUCT_SIZE);
      end record;
   type CTL_PRODUCT_POINT  is access CTL_PRODUCT_TYPE;
   CTL_PRODUCT_REC               : CTL_PRODUCT_POINT :=  new CTL_PRODUCT_TYPE;

   -- Experiment control Product List
   subtype CTL_NUM_PRODUCT       is  SYS_WALKING_CELL  range 0..800;
   subtype CTL_NUM_VW_MENU       is  SYS_WALKING_MENU  range 0..50;


   type CTL_PROD_LIST_TYPE is array (CTL_NUM_PRODUCT range <>) of
        CTL_NUM_PRODUCT;

   type CTL_PROD_LIST_POINT  is access CTL_PROD_LIST_TYPE;

   -- Experiment control routing record
   type CTL_ROUTING_ARRAY    is   array (SYS_PARTICIPANTS) of BOOLEAN;
```

```
    type CTL_ROUTING_REC is
       record
          CTL_PRODUCT         :   CTL_NUM_PRODUCT;
          CTL_PRODUCT_TYPE    :   SYS_PRODUCT;
          CTL_LIST            :   CTL_ROUTING_ARRAY;
       end record;
    type CTL_ROUTING_POINT  is access CTL_ROUTING_REC;

    -- Participant Record
    subtype CTL_PART_NAME_LEN      is INTEGER     range 1..10;
    subtype CTL_PART_NAME_TEXT is string (CTL_PART_NAME_LEN);
    type CTL_PART_REC is
       record
          CTL_TEXT            :   CTL_PART_NAME_TEXT;
          CTL_PART            :   SYS_PARTICIPANTS;
    end record;

-- List of participants that a message can be sent to
    subtype CTL_PART_LIMIT is SYS_MENU_BUTTON_INDEX range 0..4;
    type CTL_PART_ARRAY  is  array (CTL_PART_LIMIT) of
            CTL_PART_REC;
    type CTL_PARTICIPANT_POINT  is access CTL_PART_ARRAY;

end CTL_CONTROL_DB;
```

```
--cpc package specification name: FDB_REFERENCE_DB
--
--cpc description: The FDB_REFERENCE_DB cpc describes the objects that are used
--                 for interacting with the reference databases.
--
--cpc design notes:
--
--cpc package author: Bruce Packard
--                    Science Applications International Corporation
--                    424 Delaware, Suite C3
--                    Leavenworth, KS 66048
--

with SYSTEM_PACKAGE;  use SYSTEM_PACKAGE;

package FDB_REFERENCE_DB is

   -- Number of records in header and reference product database
   subtype FDB_NUM_HEADER_REC    is  SYS_DB_SIZE  range 0..5000;
   subtype FDB_NUM_PRODUCT_REC   is  SYS_DB_SIZE  range  0..20000;

   -- Product Description record
   type FDB_PRODUCT_DESC_TYPE  is
      record
         FDB_PRODUCT_CAT         :  SYS_PRODUCT_CAT;
         FDB_PRODUCT_HDR_START   :  FDB_NUM_HEADER_REC;
         FDB_PRODUCT_HDR_END     :  FDB_NUM_HEADER_REC;
         FDB_PRODUCT_START       :  FDB_NUM_PRODUCT_REC;
         FDB_PRODUCT_END         :  FDB_NUM_PRODUCT_REC;
      end record;

   FDB_PRODUCT_DESC_REC          :  FDB_PRODUCT_DESC_TYPE;

   -- Report Header record
   FDB_HEADER_SIZE                  :  SYS_HEADER_LENGTH   := 252;

   type FDB_HEADER_TYPE  is
      record
         FDB_HEAD_NUMBER_CHAR       :  SYS_HEADER_LENGTH  range 0..
                                       FDB_HEADER_SIZE;
         FDB_HEADER_TEXT            :  string (1..FDB_HEADER_SIZE);
      end record;

   type FDB_HEADER_POINT  is access FDB_HEADER_TYPE;
   FDB_HEADER_REC                :  FDB_HEADER_POINT := new FDB_HEADER_TYPE;

   -- Report record
   FDB_PRODUCT_SIZE                 :  SYS_PRODUCT_LENGTH   := 252;

   type FDB_PRODUCT_TYPE  is
      record
         FDB_REPT_NUMBER_CHAR       :  SYS_PRODUCT_LENGTH  range 0..
                                       FDB_PRODUCT_SIZE;
         FDB_PRODUCT_TEXT           :  string (1..FDB_PRODUCT_SIZE);
      end record;
```

```
    type FDB_PRODUCT_POINT  is access FDB_PRODUCT_TYPE;
    FDB_PRODUCT_REC              : FDB_PRODUCT_POINT :=  new FDB_PRODUCT_TYPE;

    -- Reference Product List
    subtype FDB_NUM_PRODUCT       is  SYS_WALKING_CELL  range 0..300;
    subtype FDB_NUM_MENU          is  SYS_WALKING_MENU  range 0..75;

    type FDB_PROD_LIST_TYPE is array (SYS_WALKING_CELL range <>) of
         FDB_NUM_PRODUCT;

    type FDB_PROD_LIST_POINT  is access FDB_PROD_LIST_TYPE;

end FDB_REFERENCE_DB;
```

```
--cpc package specification name: HDB_HELP_DB
--
--cpc description: The HDB_HELP_DB cpc describes the objects that are used
--                 for interacting with the help databases.
--
--cpc design notes:
--
--cpc package author: Bruce Packard
--                    Science Applications International Corporation
--                    424 Delaware, Suite C3
--                    Leavenworth, KS 66048
--

with SYSTEM_PACKAGE;  use SYSTEM_PACKAGE;

package HDB_HELP_DB is

   -- Number of records in header and reference product database
   subtype HDB_NUM_PRODUCT_REC   is  SYS_DB_SIZE  range  0..20000;

   -- Product Description record
   type HDB_PRODUCT_DESC_TYPE  is
      record
         HDB_PRODUCT_CAT         :  SYS_PRODUCT_CAT;
         HDB_PRODUCT_START       :  HDB_NUM_PRODUCT_REC;
         HDB_PRODUCT_END         :  HDB_NUM_PRODUCT_REC;
      end record;

   HDB_PRODUCT_DESC_REC          :  HDB_PRODUCT_DESC_TYPE;

   -- Report record
   HDB_PRODUCT_SIZE                 :  SYS_PRODUCT_LENGTH   := 252;

   type HDB_PRODUCT_TYPE  is
      record
         HDB_REPT_NUMBER_CHAR      :  SYS_PRODUCT_LENGTH  range 0..
                                      HDB_PRODUCT_SIZE;
         HDB_PRODUCT_TEXT          :  string (1..HDB_PRODUCT_SIZE);
      end record;

   type HDB_PRODUCT_POINT  is access HDB_PRODUCT_TYPE;
   HDB_PRODUCT_REC              : HDB_PRODUCT_POINT :=  new HDB_PRODUCT_TYPE;

   -- Help Product List
   subtype HDB_NUM_PRODUCT       is  SYS_WALKING_CELL  range 0..300;
   subtype HDB_NUM_MENU          is  SYS_WALKING_MENU  range 0..75;

   type HDB_PROD_LIST_TYPE is array (SYS_WALKING_CELL range <>) of
        HDB_NUM_PRODUCT;

   type HDB_PROD_LIST_POINT  is access HDB_PROD_LIST_TYPE;

end HDB_HELP_DB;
```

```
--cpc package specification name: LUT_SYSTEM
--
--cpc description: Defines types and objects that are common to the color
--                 lookup table system.
--
--cpc design notes:
--
--cpc package author: Bruce Packard
--                    Science Applications International Corporation
--                    424 Delaware, Suite C3
--                    Leavenworth, KS 66048
--
with SYSTEM_PACKAGE;          use SYSTEM_PACKAGE;

package LUT_SYSTEM is

   -- Define the color tables
   type LUT_RGB is
      record
         LUT_RED                 :    SYS_COLOR;
         LUT_GREEN               :    SYS_COLOR;
         LUT_BLUE                :    SYS_COLOR;
   end record;

   -- Define the color look up table indexes
   subtype LUT_BACK_LUT_INDEX   is   SYS_COLOR_TABLE  range  2..30;
   subtype LUT_GEN_LUT_INDEX    is   SYS_COLOR_TABLE  range  31..38;
   subtype LUT_HYDRO_LUT_INDEX  is   SYS_COLOR_TABLE  range  39..41;
   subtype LUT_MISC_LUT_INDEX   is   SYS_COLOR_TABLE  range  52..61;
   subtype LUT_ROAD_LUT_INDEX   is   SYS_COLOR_TABLE  range  45..51;
   subtype LUT_URBAN_LUT_INDEX  is   SYS_COLOR_TABLE  range  42..44;
   subtype LUT_CONT_LUT_INDEX   is   SYS_COLOR_TABLE  range  63..63;
   subtype LUT_GRID_LUT_INDEX   is   SYS_COLOR_TABLE  range  62..62;
   subtype LUT_RED_LUT_INDEX    is   SYS_COLOR_TABLE  range  64..127;
   subtype LUT_BLUE_LUT_INDEX   is   SYS_COLOR_TABLE  range  128..255;

   -- Type and pointer for lookup table arrays
   type LUT_ARRAY is array (SYS_COLOR_TABLE range <>) of LUT_RGB;
   type LUT_POINT is access LUT_ARRAY;
   -- Color table for the background
   LUT_BACK_COLOR      :    LUT_POINT := new LUT_ARRAY (LUT_BACK_LUT_INDEX);
   LUT_CONT_COLOR      :    LUT_POINT := new LUT_ARRAY (LUT_CONT_LUT_INDEX);
   LUT_GRID_COLOR      :    LUT_POINT := new LUT_ARRAY (LUT_GRID_LUT_INDEX);
   LUT_GEN_COLOR       :    LUT_POINT := new LUT_ARRAY (LUT_GEN_LUT_INDEX);

   -- Color Tables with highlighted colors on
   LUT_HYDRO_COLOR_ON  :    LUT_POINT := new LUT_ARRAY (LUT_HYDRO_LUT_INDEX);
   LUT_MISC_COLOR_ON   :    LUT_POINT := new LUT_ARRAY (LUT_MISC_LUT_INDEX);
   LUT_ROAD_COLOR_ON   :    LUT_POINT := new LUT_ARRAY (LUT_ROAD_LUT_INDEX);
   LUT_URBAN_COLOR_ON  :    LUT_POINT := new LUT_ARRAY (LUT_URBAN_LUT_INDEX);

   -- Color Tables with highlighted colors off
   LUT_HYDRO_COLOR_OFF :    LUT_POINT := new LUT_ARRAY (LUT_HYDRO_LUT_INDEX);
   LUT_MISC_COLOR_OFF  :    LUT_POINT := new LUT_ARRAY (LUT_MISC_LUT_INDEX);
   LUT_ROAD_COLOR_OFF  :    LUT_POINT := new LUT_ARRAY (LUT_ROAD_LUT_INDEX);
```

```
LUT_URBAN_COLOR_OFF :    LUT_POINT := new LUT_ARRAY (LUT_URBAN_LUT_INDEX);

-- Color tables of the overlay planes
LUT_RED_OVERLAY_COLOR  : LUT_POINT := new LUT_ARRAY (LUT_RED_LUT_INDEX);
LUT_BLUE_OVERLAY_COLOR : LUT_POINT := new LUT_ARRAY (LUT_BLUE_LUT_INDEX);

-- Define the digital map planes
LUT_BACK_START_PLANE          :    SYS_COLOR_PLANE    :=  1;
LUT_BACK_END_PLANE            :    SYS_COLOR_PLANE    :=  6;
LUT_GRID_START_PLANE          :    SYS_COLOR_PLANE    :=  1;
LUT_GRID_END_PLANE            :    SYS_COLOR_PLANE    :=  6;
LUT_CONT_START_PLANE          :    SYS_COLOR_PLANE    :=  1;
LUT_CONT_END_PLANE            :    SYS_COLOR_PLANE    :=  6;

-- Define the overlay planes
LUT_BLUE_START_PLANE          :    SYS_COLOR_PLANE    :=  8;
LUT_BLUE_END_PLANE            :    SYS_COLOR_PLANE    :=  8;
LUT_RED_START_PLANE           :    SYS_COLOR_PLANE    :=  7;
LUT_RED_END_PLANE             :    SYS_COLOR_PLANE    :=  7;

-- Define the EDDIC specific colors
LUT_COLOR_BLUE                :    SYS_COLOR          := 128;
LUT_COLOR_RED                 :    SYS_COLOR          :=  64;
LUT_COLOR_CONTOUR             :    SYS_COLOR          :=  63;
LUT_COLOR_GRID                :    SYS_COLOR          :=  62;

-- Define the EDDIC General colors
LUT_COLOR_CYAN                :    SYS_COLOR          :=  31;
LUT_COLOR_GREEN               :    SYS_COLOR          :=  32;
LUT_COLOR_VIOLET              :    SYS_COLOR          :=  33;
LUT_COLOR_ORANGE              :    SYS_COLOR          :=  34;
LUT_COLOR_AMBER               :    SYS_COLOR          :=  35;
LUT_COLOR_BROWN               :    SYS_COLOR          :=  36;
LUT_COLOR_WHITE               :    SYS_COLOR          :=  37;
LUT_COLOR_YELLOW              :    SYS_COLOR          :=  38;
LUT_COLOR_BLACK               :    SYS_COLOR          :=  62;

-- Define the depth of the digital map image
LUT_LUT_DEPTH                 :    SYS_BITS_DEEP    :=  8;

-- Color lookup table file names
type  LUT_BACKGROUND is (LUT_NONE, LUT_SHADE_VEG);
type LUT_COUNT_LIMIT   is    range  0..1;

LUT_HILITE_LUT                :    array (LUT_COUNT_LIMIT) of
                                   string (SYS_NAME_SIZE);
LUT_UNHILITE_LUT              :    array (LUT_COUNT_LIMIT) of
                                   string (SYS_NAME_SIZE);

-- Definition of the color look-up table update
type LUT_UPDATE_RECORD is
   record
      LUT_BACK_TYPE           :     LUT_BACKGROUND;
      LUT_BACK                :     SYS_LUT_STATUS;
      LUT_ROAD                :     SYS_LUT_STATUS;
      LUT_WATER               :     SYS_LUT_STATUS;
```

```
      LUT_URBAN                 :     SYS_LUT_STATUS;
      LUT_MISC                  :     SYS_LUT_STATUS;
end record;

-- Current Status of the lookup table
LUT_CURR_STATUS                 :     LUT_UPDATE_RECORD;

-- Tactical Map Attribute record - Used to record changes in the map display
type LUT_MAP_ATTRIBUTES is
   record
      MAP_BACK_TYPE             :       SYS_MAP_BACKGROUND;
      MAP_MAP_SCALE             :       SYS_MAP_SCALES;
      MAP_GRID_STATUS           :       BOOLEAN;
      MAP_CONTOUR_STATUS        :       BOOLEAN;
      MAP_CENTER_X              :       SYS_COORDINATE;
      MAP_CENTER_Y              :       SYS_COORDINATE;
      BLUEFOR_UNIT_DIV          :       BOOLEAN;
      BLUEFOR_UNIT_BDE          :       BOOLEAN;
      BLUEFOR_UNIT_RGMT         :       BOOLEAN;
      BLUEFOR_UNIT_BN           :       BOOLEAN;
      BLUEFOR_UNIT_CO           :       BOOLEAN;
      BLUEFOR_UNIT_CBT_COMMIT   :       BOOLEAN;
      BLUEFOR_UNIT_CS_REINF     :       BOOLEAN;
      BLUEFOR_UNIT_CSS_ARTIL    :       BOOLEAN;
      BLUEFOR_UNIT_NAME         :       BOOLEAN;
      BLUEFOR_UNIT_SYMBOL       :       BOOLEAN;
      OPFOR_UNIT_DIV            :       BOOLEAN;
      OPFOR_UNIT_BDE            :       BOOLEAN;
      OPFOR_UNIT_RGMT           :       BOOLEAN;
      OPFOR_UNIT_BN             :       BOOLEAN;
      OPFOR_UNIT_CO             :       BOOLEAN;
      OPFOR_UNIT_CBT_COMMIT     :       BOOLEAN;
      OPFOR_UNIT_CS_REINF       :       BOOLEAN;
      OPFOR_UNIT_CSS_ARTIL      :       BOOLEAN;
      OPFOR_UNIT_NAME           :       BOOLEAN;
      OPFOR_UNIT_SYMBOL         :       BOOLEAN;
      CM_BLUE_EAC               :       BOOLEAN;
      CM_BLUE_CORP              :       BOOLEAN;
      CM_BLUE_DIV               :       BOOLEAN;
      CM_BLUE_BDE               :       BOOLEAN;
      CM_BLUE_BN                :       BOOLEAN;
      CM_BLUE_CO                :       BOOLEAN;
      CM_BLUE_POINT                     BOOLEAN;
      CM_BLUE_LINE              :       BOOLEAN;
      CM_BLUE_AREA              :       BOOLEAN;
      CM_BLUE_ROUTE             :       BOOLEAN;
      CM_BLUE_CROSSING          :       BOOLEAN;
      CM_BLUE_FIRE_PLAN         :       BOOLEAN;
      CM_BLUE_MAP_FEAT          :       BOOLEAN;
      CM_OPFOR_ARMY             :       BOOLEAN;
      CM_OPFOR_DIV              :       BOOLEAN;
      CM_OPFOR_RGMT             :       BOOLEAN;
      CM_OPFOR_BN               :       BOOLEAN;
      CM_OPFOR_CO               :       BOOLEAN;
      CM_OPFOR_POINT            :       BOOLEAN;
      CM_OPFOR_LINE             :       BOOLEAN;
```

```
        CM_OPFOR_AREA              :        BOOLEAN;
        CM_OPFOR_ROUTE             :        BOOLEAN;
        CM_OPFOR_CROSSING          :        BOOLEAN;
        CM_OPFOR_FIRE_PLAN         :        BOOLEAN;
        CM_OPFOR_MAP_FEAT          :        BOOLEAN;
    end record;

end LUT_SYSTEM;
```

```
--cpc package specification name: MSG_MESSAGE
--
--cpc description: The MSG_MESSAGE package contains the definitions of all
--                 messages that are transferred between process in EDDIC.
--
--cpc design notes:
--
--cpc package author: Bruce Packard
--                    Science Applications International Corporation
--                    424 Delaware, Suite C3
--                    Leavenworth, KS 66048
--

with SYSTEM_PACKAGE;     use SYSTEM_PACKAGE;
with SDB_SITUATION_DB;   use SDB_SITUATION_DB;
with CDB_C2_PRODUCT_DB;  use CDB_C2_PRODUCT_DB;
with FDB_REFERENCE_DB;   use FDB_REFERENCE_DB;
with HDB_HELP_DB;        use HDB_HELP_DB;
with CTL_CONTROL_DB;     use CTL_CONTROL_DB;
with TSTM_DB;            use TSTM_DB;
with LUT_SYSTEM;         use LUT_SYSTEM;
with CALENDAR;

package MSG_MESSAGE is

-- EDDIC message length

   type MSG_MESSAGE_LEN is range 0..32576;
   for MSG_MESSAGE_LEN'SIZE use 4*SYS_BITS_IN_BYTE;

   type MSG_MESSAGES is (MSG_IGNORE,
                         MSG_TEXT_BUFFER, MSG_HEADER_BUFFER, MSG_REQUEST,
                         MSG_LUT_UPDATE, MSG_CREATE_WINDOW, MSG_TERM_WINDOW,
                         MSG_STATION_UP, MSG_CONTROL_ROUTING, MSG_MENU_TREE,
                         MSG_CONTROL_PRODUCTS, MSG_CONTROL_PART_LIST,
                         MSG_REFERENCE_PRODUCTS, MSG_C2_PRODUCTS,MSG_C2_MESSAGE,
                         MSG_MESSAGE_LOG, MSG_NEW_OPPLAN, MSG_OPPLAN_LIST,
                         MSG_C2_PART_LIST, MSG_HELP_PRODUCTS, MSG_SD_REQUEST,
                         MSG_AMMO_AUTH, MSG_AMMO_ON_HAND, MSG_EQUIP_AUTH,
                         MSG_EQUIP_OPER, MSG_PERSONNEL, MSG_FUEL, MSG_LOCATION,
                         MSG_ALL_LOCATIONS, MSG_AMMO_UPDATE, MSG_EQUIP_UPDATE,
                         MSG_PERS_UPDATE, MSG_FUEL_UPDATE, MSG_LOC_UPDATE,
                         MSG_ACTIVITY_UPDATE, MSG_MISSION_UPDATE,
                         MSG_REINFORCE_UPDATE, MSG_STRENGTH_UPDATE,
                         MSG_BLUEFOR_STATUS, MSG_BLUEFOR_TASK_ORG,
                         MSG_BLUE_TASK_ORG_UPDATE,
                         MSG_OPFOR_STATUS, MSG_OPFOR_TASK_ORG,
                         MSG_OPFOR_TASK_ORG_UPDATE,
                         MSG_CONTROL_MEASURE, MSG_CNTRL_MSR_ADD,
                         MSG_CNTRL_MSR_POINT, MSG_CNTRL_POINT_ADD,
                         MSG_CNTRL_MSR_CHG_LOC, MSG_CNTRL_MSR_CHG_STAT,
                         MSG_CNTRL_MSR_CHG_EFF, MSG_CNTRL_MSR_DEL,
                         MSG_OBSTACLE, MSG_OBSTACLE_ADD,
                         MSG_OBSTACLE_CHG_LOC, MSG_OBSTACLE_CHG_STAT,
                         MSG_OBSTACLE_CHG_EFF, MSG_OBSTACLE_DEL,
```

```
                        MSG_TSTM_MATRIX, MSG_TSTM_COL_FBACK,
                        MSG_TSTM_ROW_FBACK, MSG_TSTM_OCOKA_FBACK,
                        MSG_TSTM_COA_FBACK, MSG_TSTM_MATRIX_SAVE,
                        MSG_TSTM_INIT_INSTR, MSG_TSTM_NEW_PHASE,
                        MSG_EC_ROUTER_BUFFER, MSG_RF_ROUTER_BUFF,
                        MSG_C2_ROUTER_BUFF, MSG_SD_ROUTER_BUFFER,
                        MSG_WINDOW_OPEN, MSG_WINDOW_CLOSE,
                        MSG_CONNECT, MSG_STOP, MSG_CLOSE_SOCKET,
                        MSG_MAP_STATUS, MSG_XFER_COMPLETE, MSG_TOOL);

   -- Buffer for Experiment Control Router (Max Var = MSG_TEXT_BUFFER);
   subtype MSG_EC_MAX_MSG_LEN   is SYS_DB_SIZE   range 1..1350;
   type MSG_EC_ROUTER_REC  is  array (MSG_EC_MAX_MSG_LEN) of INTEGER;

   -- Buffer for Reference Router (Max Var = MSG_TEXT_BUFFER);
   subtype MSG_RF_MAX_MSG_LEN   is SYS_DB_SIZE   range 1..2600;
   type MSG_RF_ROUTER_REC  is  array (MSG_RF_MAX_MSG_LEN) of INTEGER;

   -- Buffer for C2 Product Router (Max Var = MSG_MENU_TREE);
   subtype MSG_C2_MAX_MSG_LEN   is SYS_DB_SIZE   range 1..6100;
   type MSG_C2_ROUTER_REC  is  array (MSG_C2_MAX_MSG_LEN) of INTEGER;

   -- Buffer for Situation Data Router (Max Var = MSG_SD_RD_TASK_ORG_REC);
   subtype MSG_SD_MAX_MSG_LEN   is SYS_DB_SIZE   range 1..8000;
   type MSG_SD_ROUTER_REC  is  array (MSG_SD_MAX_MSG_LEN) of INTEGER;

   -- Length of the message header
   MSG_HEADER_LEN           :   MSG_MESSAGE_LEN   := 18;

-- EDDIC    M E S S A G E   V A R I A N C E   R E C O R D

   type MSG_VAR_MESSAGES (MSG_MESSAGE_TYPE:MSG_MESSAGES) is
      record

         MSG_BYTES_IN_MSG    :   MSG_MESSAGE_LEN;
         MSG_RECORD_TYPE     :   MSG_MESSAGES;
         MSG_DESTINATION     :   SYS_EDDIC_PROCESSES;
         MSG_STATION_ID      :   SYS_EDDIC_PROCESSES;
         MSG_ACKNOWLEDGE     :   BOOLEAN := false;
         MSG_DATE_TIME       :   CALENDAR.TIME;

         case MSG_MESSAGE_TYPE is

            -- Text Messages

            when MSG_TEXT_BUFFER =>

               MSG_TEXT_COUNT                :   SYS_PRODUCT_LENGTH;
               MSG_PROD_TYPE                 :   SYS_PRODUCT;
               MSG_OPPLAN_ID                 :   SYS_OPPLAN;
               MSG_TEXT_DATE_TIME            :   SYS_DATE_TIME;
               MSG_TEXT                      :   SYS_TEXT (1..
                                                 SYS_PRODUCT_LENGTH'LAST);

            -- Report Header Messages
```

```
when MSG_HEADER_BUFFER =>

   MSG_HEADER_COUNT              :   SYS_HEADER_LENGTH;
   MSG_HEADER                    :   SYS_TEXT (1..SYS_HEADER_LENGTH'
                                     LAST);

-- Request for message
when MSG_REQUEST =>

   MSG_MESSAGE_REQ               :  MSG_MESSAGES;
   MSG_RPT_REQUESTOR             :  SYS_EDDIC_PROCESSES;
   MSG_PRODUCT                   :  SYS_WALKING_CELL;

-- Color look-up table update
when MSG_LUT_UPDATE =>

   MSG_COLOR_UPDATE              :  LUT_UPDATE_RECORD;

-- Create a new window on a station
when MSG_CREATE_WINDOW =>

   MSG_WINDOW_EXEC               :  STRING (SYS_ENV_STRING);

-- Experiment Control window termination
when MSG_TERM_WINDOW =>

   MSG_TERM_TYPE                 :  SYS_PRODUCT;

-- Experiment Control routing message
when MSG_CONTROL_ROUTING =>

   MSG_CONTROL_ROUTE             :  CTL_ROUTING_REC;

-- Menu tree structure - Used by UED_WALKING_MENU
when MSG_MENU_TREE =>

   MSG_MENU_TREE_COUNT       :  SYS_MENU_TREE_LIMIT;
   MSG_TREE                  :  SYS_MENU_TREE (SYS_MENU_TREE_LIMIT);


-- Products available in the control database
when MSG_CONTROL_PRODUCTS =>

   MSG_NUMBER_CTL_PROD       :  CTL_NUM_PRODUCT;
   MSG_CTL_PROD              :  CTL_PROD_LIST_TYPE (CTL_NUM_PRODUCT);

when MSG_CONTROL_PART_LIST =>

   MSG_NUMBER_CTL_PART       :  CTL_PART_LIMIT;
   MSG_CTL_PART_LIST         :  CTL_PART_ARRAY;

-- Products available in the reference database
when MSG_REFERENCE_PRODUCTS =>

   MSG_NUMBER_REF_PROD       :  FDB_NUM_PRODUCT;
   MSG_REF_PROD              :  FDB_PROD_LIST_TYPE (FDB_NUM_PRODUCT);
```

```
-- Products available in the help database
when MSG_HELP_PRODUCTS =>

   MSG_NUMBER_HELP_PROD      :  HDB_NUM_PRODUCT;
   MSG_HELP_PROD             :  HDB_PROD_LIST_TYPE (HDB_NUM_PRODUCT);

-- C2 Products in the C2 Product database

when MSG_C2_PRODUCTS =>

   MSG_NUMBER_C2_PROD        :  CDB_NUM_PRODUCT;
   MSG_C2_PROD               :  CDB_PROD_LIST_TYPE (CDB_NUM_PRODUCT);

when MSG_C2_MESSAGE =>

   MSG_C2_SUM_MESSAGE        :  CDB_SUM_MESSAGE_REC;

when MSG_MESSAGE_LOG =>

   MSG_C2_MESSAGE_LOG        :  CDB_MESSAGE_LOG_REC;

when MSG_C2_PART_LIST =>

   MSG_NUMBER_PART           :  CDB_PART_LIMIT;
   MSG_PART_LIST             :  CDB_PART_ARRAY;

when MSG_OPPLAN_LIST =>

   MSG_SD_OPPLAN_LIST        :  SDB_OPPLAN_LIST_REC;

-- S I T U A T I O N    D A T A
when MSG_SD_REQUEST =>

   MSG_SD_MESSAGE_REQ        :  MSG_MESSAGES;
   MSG_SD_MSG_REQUESTOR      :  SYS_EDDIC_PROCESSES;
   MSG_SD_TIME               :  SYS_DATE_TIME;
   MSG_SD_OPPLAN             :  SYS_OPPLAN;
   MSG_SD_FORCE              :  SDB_SIDE_TYPE;
   MSG_SD_UNIT_ID            :  SDB_UNIT;

when MSG_AMMO_AUTH =>

   MSG_SD_AMMO_AUTH          :  SDB_AMMO_AUTH_LIST;

when MSG_AMMO_ON_HAND =>

   MSG_SD_AMMO_ON_HAND       :  SDB_AMMO_ON_HAND_REC;

when MSG_EQUIP_AUTH =>

   MSG_SD_EQUIP_AUTH         :  SDB_EQUIP_AUTH_LIST;

when MSG_EQUIP_OPER =>

   MSG_SD_EQUIP_OPER         :  SDB_EQUIP_OPER_REC;
```

```
when MSG_PERSONNEL =>
   MSG_SD_PERSONNEL          :  SDB_PERSONNEL;
when MSG_FUEL =>
   MSG_SD_FUEL               :  SDB_FUELS;
when MSG_LOCATION =>
   MSG_SD_LOCATION           :  SDB_UNIT_LOCATION;
when MSG_ALL_LOCATIONS =>
   MSG_SD_ALL_LOC            :  SDB_ALL_LOC_REC;
when MSG_AMMO_UPDATE =>
   MSG_SD_AMMO_UPD           :  SDB_AMMO_UPDATE_REC;
when MSG_EQUIP_UPDATE =>
   MSG_SD_EQUIP_UPD          :  SDB_EQUIP_UPDATE_REC;
when MSG_PERS_UPDATE =>
   MSG_SD_PERS_UPD           :  SDB_PERS_UPDATE_REC;
when MSG_FUEL_UPDATE =>
   MSG_SD_FUEL_UPD           :  SDB_FUEL_UPDATE_REC;
when MSG_LOC_UPDATE =>
   MSG_SD_LOCATION_UPD       :  SDB_LOCATION_UPDATE_REC;
when MSG_ACTIVITY_UPDATE =>
   MSG_SD_ACTIVITY_UPD       :  SDB_ACTIVITY_UPDATE_REC;
when MSG_MISSION_UPDATE =>
   MSG_SD_MISSION_UPD        :  SDB_MISSION_UPDATE_REC;
when MSG_REINFORCE_UPDATE =>
   MSG_SD_REINF_UPD          :  SDB_OPFOR_REINF_UPDATE_REC;
when MSG_STRENGTH_UPDATE =>
   MSG_SD_STR_UPD            :  SDB_OPFOR_STR_UPDATE_REC;
when MSG_BLUEFOR_STATUS =>
   MSG_SD_BL_STATUS          :  SDB_BLUE_UNIT_STATUS;
```

```
when MSG_BLUEFOR_TASK_ORG =>
   MSG_SD_BL_TASK_ORG        :  SDB_BLUE_TASK_ORG_REC;
when MSG_BLUE_TASK_ORG_UPDATE =>
   MSG_SD_BL_TASK_ORG_UPD    :  SDB_BLUE_TASK_ORG_UPDATE_REC;
when MSG_OPFOR_STATUS =>
   MSG_SD_RD_STATUS          :  SDB_OPFOR_UNIT_STATUS;
when MSG_OPFOR_TASK_ORG =>
   MSG_SD_RD_TASK_ORG        :  SDB_OPFOR_TASK_ORG_REC;
when MSG_OPFOR_TASK_ORG_UPDATE =>
   MSG_SD_RD_TASK_ORG_UPD    :  SDB_OPFOR_TASK_ORG_UPDATE_REC;
when MSG_CONTROL_MEASURE =>
   MSG_SD_CNTRL_MSR          :  SDB_ALL_CNTRL_MSR;
when MSG_CNTRL_MSR_ADD =>
   MSG_SD_CNTRL_MSR_NEW      :  SDB_CONTROL_MEASURE_REC;
when MSG_CNTRL_MSR_POINT =>
   MSG_SD_CNTRL_MSR_POINT    :  SDB_ALL_CNTRL_POINT;
when MSG_CNTRL_POINT_ADD =>
   MSG_SD_CNTRL_NEW_POINT    :  SDB_CNTRL_MSR_POINT_REC;
when MSG_CNTRL_MSR_CHG_LOC =>
   MSG_SD_CNTRL_MSR_LOC      :  SDB_CNTRL_MSR_LOC_REC;
when MSG_CNTRL_MSR_CHG_STAT =>
   MSG_SD_CNTRL_MSR_STAT     :  SDB_CNTRL_MSR_STAT_REC;
when MSG_CNTRL_MSR_CHG_EFF =>
   MSG_SD_CNTRL_MSR_EFF      :  SDB_CNTRL_MSR_EFF_REC;
when MSG_CNTRL_MSR_DEL =>
   MSG_SD_CNTRL_MSR_DEL_ID   :  SDB_CONTROL_MEASURE_ID;
   MSG_SD_DEL_TIME           :  SYS_DATE_TIME;
   MSG_SD_DEL_OPPLAN         :  SYS_OPPLAN;
   MSG_SD_LOCATION_TYPE      :  SDB_CONTROL_MEASURE_LOC_TYPE;
```

```
when MSG_OBSTACLE =>

   MSG_SD_OBSTACLES          :  SDB_ALL_OBSTACLE;

when MSG_OBSTACLE_ADD =>

   MSG_SD_OBSTACLE_NEW       :  SDB_OBSTACLE_REC;

when MSG_OBSTACLE_CHG_LOC =>

   MSG_SD_OBSTACLE_LOC       :  SDB_OBSTACLE_LOC_REC;

when MSG_OBSTACLE_CHG_STAT =>

   MSG_SD_OBSTACLE_STAT      :  SDB_OBSTACLE_STAT_REC;

when MSG_OBSTACLE_CHG_EFF =>

   MSG_SD_OBSTACLE_EFF       :  SDB_OBSTACLE_EFF_REC;

when MSG_OBSTACLE_DEL =>

   MSG_SD_OBSTACLE_DEL_ID    :  SDB_OBSTACLE_ID;
   MSG_SD_OB_DEL_TIME        :  SYS_DATE_TIME;
   MSG_SD_OBS_DEL_OPPLAN     :  SYS_OPPLAN;

when MSG_NEW_OPPLAN =>

   MSG_SD_NEW_OPPLAN         :  SDB_NEW_OPPLAN_REC;

-- Tactical Map Attributes
when MSG_MAP_STATUS =>

   MSG_MAP_ATTRIB            :  LUT_MAP_ATTRIBUTES;

-- TSTM Messages

 when MSG_TSTM_MATRIX =>

    MSG_MATRIX               :  TSTM_INITIAL_MATRIX;

 when MSG_TSTM_COL_FBACK =>

    MSG_COL_FBACK            :  TSTM_COLUMN_FEEDBACK;

 when MSG_TSTM_ROW_FBACK =>

    MSG_ROW_FBACK            :  TSTM_ROW_FEEDBACK;

 when MSG_TSTM_OCOKA_FBACK =>

    MSG_OCOKA_FBACK          :  TSTM_OCOKA_FEEDBACK;

 when MSG_TSTM_COA_FBACK =>

    MSG_COA_FBACK            :  TSTM_COA_FEEDBACK;
```

```
            when MSG_TSTM_MATRIX_SAVE =>

               MSG_MATRIX_SAVE          :  TSTM_MATRIX_SAVE;

            when MSG_TSTM_NEW_PHASE =>

               MSG_PHASE                :  TSTM_PHASE;

            -- Router Buffers

            when MSG_EC_ROUTER_BUFFER =>

               MSG_EC_STATION_ID        :  SYS_EDDIC_PROCESSES;
               MSG_EC_BUFFER            :  MSG_EC_ROUTER_REC;

            when MSG_RF_ROUTER_BUFF =>

               MSG_RF_STATION_ID        :  SYS_EDDIC_PROCESSES;
               MSG_RF_BUFFER            :  MSG_RF_ROUTER_REC;

            when MSG_C2_ROUTER_BUFF =>

               MSG_C2_STATION_ID        :  SYS_EDDIC_PROCESSES;
               MSG_C2_BUFFER            :  MSG_C2_ROUTER_REC;

            when MSG_SD_ROUTER_BUFFER =>

               MSG_SD_STATION_ID        :  SYS_EDDIC_PROCESSES;
               MSG_SD_BUFFER            :  MSG_SD_ROUTER_REC;

            when MSG_TOOL =>

               MSG_TOOL_TYPE            :  SYS_TOOLS;

            when others =>                    null;

         end case;

      end record;

type MSG_MESSAGE_POINT is access MSG_VAR_MESSAGES;

-- Message Recording Record Descriptions for the routers
-- Experiment control router
subtype  MSG_EC_REC_LIMIT is  SYS_DB_SIZE  range  1..12;
MSG_EC_RECORD_LIST   :   array (MSG_EC_REC_LIMIT) of  MSG_MESSAGES :=
     (MSG_REQUEST, MSG_LUT_UPDATE, MSG_CREATE_WINDOW, MSG_TERM_WINDOW,
      MSG_STATION_UP, MSG_WINDOW_OPEN, MSG_WINDOW_CLOSE, MSG_CONNECT,
      MSG_MAP_STATUS, MSG_CLOSE_SOCKET, MSG_STOP, MSG_TOOL);

-- Reference router
subtype  MSG_RF_REC_LIMIT is  SYS_DB_SIZE  range  1..6;
MSG_RF_RECORD_LIST   :   array (MSG_RF_REC_LIMIT) of  MSG_MESSAGES :=
     (MSG_REQUEST, MSG_WINDOW_OPEN, MSG_WINDOW_CLOSE, MSG_CONNECT,
      MSG_CLOSE_SOCKET, MSG_STOP);
```

```
    -- C2 product router
    subtype  MSG_C2_REC_LIMIT is  SYS_DB_SIZE  range  1..7;
    MSG_C2_RECORD_LIST   :   array (MSG_C2_REC_LIMIT) of  MSG_MESSAGES :=
         (MSG_REQUEST, MSG_C2_MESSAGE, MSG_WINDOW_OPEN, MSG_WINDOW_CLOSE,
          MSG_CONNECT, MSG_CLOSE_SOCKET, MSG_STOP);

    -- Situation DB router
    subtype  MSG_SD_REC_LIMIT is  SYS_DB_SIZE  range  1..28;
    MSG_SD_RECORD_LIST   :   array (MSG_SD_REC_LIMIT) of  MSG_MESSAGES :=
         (MSG_SD_REQUEST, MSG_AMMO_UPDATE, MSG_EQUIP_UPDATE, MSG_PERS_UPDATE,
          MSG_FUEL_UPDATE, MSG_LOC_UPDATE, MSG_BLUE_TASK_ORG_UPDATE,
          MSG_ACTIVITY_UPDATE, MSG_MISSION_UPDATE, MSG_OPFOR_TASK_ORG_UPDATE,
          MSG_REINFORCE_UPDATE, MSG_STRENGTH_UPDATE, MSG_CNTRL_MSR_ADD,
          MSG_CNTRL_POINT_ADD, MSG_CNTRL_MSR_CHG_STAT, MSG_CNTRL_MSR_CHG_EFF,
          MSG_CNTRL_MSR_CHG_LOC, MSG_CNTRL_MSR_DEL, MSG_OBSTACLE_ADD,
          MSG_OBSTACLE_CHG_LOC, MSG_OBSTACLE_CHG_STAT, MSG_OBSTACLE_CHG_EFF,
          MSG_OBSTACLE_DEL, MSG_WINDOW_OPEN, MSG_WINDOW_CLOSE, MSG_CONNECT,
          MSG_CLOSE_SOCKET, MSG_STOP);

end MSG_MESSAGE;
```

```
--cpc package specification name: SDB_SITUATION_DB
--
--cpc description: This package describes the records in the EDDIC situation
--                 database
--
--cpc design notes:
--
--cpc package author: Bruce Packard
--                    Science Applications International Corporation
--                    424 Delaware, Suite C3
--                    Leavenworth, KS 66048
--

with SYSTEM_PACKAGE;  use SYSTEM_PACKAGE;

package SDB_SITUATION_DB is

   subtype SDB_UNIT        is   SYS_DB_SIZE   range 0..500;
   subtype SDB_EQUIPMENT   is   SYS_DB_SIZE   range 0..100;
   subtype SDB_AMMUNITION  is   SYS_DB_SIZE   range 0..50;

   -- Situation Database Limitations
   subtype SDB_BLUEFOR_UNIT_ID     is       SDB_UNIT          range 0..200;
   subtype SDB_OPFOR_UNIT_ID       is       SDB_UNIT          range 0..400;
   subtype SDB_BLUEFOR_EQUIP_ID    is       SDB_EQUIPMENT     range 0..100;
   subtype SDB_OPFOR_EQUIP_ID      is       SDB_EQUIPMENT     range 0..100;
   subtype SDB_BLUEFOR_AMMO_ID     is       SDB_AMMUNITION    range 0..50;
   subtype SDB_BLUEFOR_EQUIP_OWN   is       SDB_EQUIPMENT     range 0..50;
   subtype SDB_OPFOR_EQUIP_OWN     is       SDB_EQUIPMENT     range 0..50;
   subtype SDB_BLUEFOR_AMMO_OWN    is       SDB_AMMUNITION    range 0..50;
   subtype SDB_CONTROL_MEASURE_ID  is       SYS_DB_SIZE       range 0..200;
   subtype SDB_CONTROL_MEASURE_PT  is       SYS_DB_SIZE       range 0..15;
   subtype SDB_OBSTACLE_ID         is       SYS_DB_SIZE       range 0..50;
   subtype SDB_BLUE_STAT_PTR       is       SYS_DB_SIZE       range 0..1024;
   subtype SDB_BLUE_EQ_AUTH_PTR    is       SYS_DB_SIZE       range 0..500;
   subtype SDB_BLUE_EQ_CURR_PTR    is       SYS_DB_SIZE       range 0..3000;
   subtype SDB_BLUE_AM_AUTH_PTR    is       SYS_DB_SIZE       range 0..500;
   subtype SDB_BLUE_AM_CURR_PTR    is       SYS_DB_SIZE       range 0..4000;
   subtype SDB_BLUE_FUEL_PTR       is       SYS_DB_SIZE       range 0..1024;
   subtype SDB_BLUE_PERS_PTR       is       SYS_DB_SIZE       range 0..1024;
   subtype SDB_BLUE_ULOC_PTR       is       SYS_DB_SIZE       range 0..1024;
   subtype SDB_OPFOR_STAT_PTR      is       SYS_DB_SIZE       range 0..1024;
   subtype SDB_OPFOR_EQ_AUTH_PTR   is       SYS_DB_SIZE       range 0..600;
   subtype SDB_OPFOR_EQ_CURR_PTR   is       SYS_DB_SIZE       range 0..3000;
   subtype SDB_OPFOR_ULOC_PTR      is       SYS_DB_SIZE       range 0..1024;
   subtype SDB_CNTRL_MSR_PTR       is       SYS_DB_SIZE       range 0..200;
   subtype SDB_CNTRL_POINT_PTR     is       SYS_DB_SIZE       range 0..200;
   subtype SDB_OBST_PTR            is       SYS_DB_SIZE       range 0..200;
   subtype SDB_UNIT_NAME_LEN       is       SYS_NAME_SIZE     range 1..15;
   subtype SDB_EQUIP_NAME_LEN      is       SYS_NAME_SIZE     range 1..12;
   subtype SDB_AMMO_NAME_LEN       is       SYS_NAME_SIZE     range 1..12;
   subtype SDB_CNTL_MSR_NAME_LEN   is       SYS_NAME_SIZE     range 1..12;

   -- EDDIC Coordinate location
   type SDB_LOCATION_REC is
```

```
    record
       SDB_X              :       SYS_COORDINATE;
       SDB_Y              :       SYS_COORDINATE;
    end record;

-- Side of the Confrontation (BLUEFOR, OPFOR)
type SDB_SIDE_TYPE is (BLUEFOR, OPFOR);

-- BLUEFOR Equipment Pointer Record
type SDB_BLUEFOR_EQUIP_PTR is
   record
      SDB_UNIT_ID         :       SDB_BLUEFOR_UNIT_ID;
      SDB_TIME            :       SYS_DATE_TIME;
      SDB_OPPLAN          :       SYS_OPPLAN;
      SDB_RECORD          :       SYS_DB_SIZE;
   end record;

-- BLUEFOR Unit Equipment Operational Quantity Database Record
type SDB_BLUEFOR_EQUIP_QTY is
   record
      SDB_UNIT_ID         :       SDB_BLUEFOR_UNIT_ID;
      SDB_EQUIP_ID        :       SDB_BLUEFOR_EQUIP_ID;
      SDB_TIME            :       SYS_DATE_TIME;
      SDB_OPPLAN          :       SYS_OPPLAN;
      SDB_OPERATIONAL     :       SYS_QUANTITY;
   end record;

-- BLUEFOR Equipment Quantity Pointer Record
type SDB_BLUEFOR_EQUIP_QTY_PTR is
   record
      SDB_UNIT_ID         :       SDB_BLUEFOR_UNIT_ID;
      SDB_EQUIP_ID        :       SDB_BLUEFOR_EQUIP_ID;
      SDB_TIME            :       SYS_DATE_TIME;
      SDB_OPPLAN          :       SYS_OPPLAN;
      SDB_RECORD          :       SYS_DB_SIZE;
   end record;

-- OPFOR Equipment Pointer Record
type SDB_OPFOR_EQUIP_PTR is
   record
      SDB_UNIT_ID         :       SDB_OPFOR_UNIT_ID;
      SDB_TIME            :       SYS_DATE_TIME;
      SDB_OPPLAN          :       SYS_OPPLAN;
      SDB_RECORD          :       SYS_DB_SIZE;
   end record;

-- OPFOR Unit Equipment Operational Quantity Database Record
type SDB_OPFOR_EQUIP_QTY is
   record
      SDB_UNIT_ID         :       SDB_OPFOR_UNIT_ID;
      SDB_EQUIP_ID        :       SDB_OPFOR_EQUIP_ID;
      SDB_TIME            :       SYS_DATE_TIME;
      SDB_OPPLAN          :       SYS_OPPLAN;
      SDB_OPERATIONAL     :       SYS_QUANTITY;
   end record;
```

```
-- OPFOR Equipment Quantity Pointer Record
type SDB_OPFOR_EQUIP_QTY_PTR is
   record
      SDB_UNIT_ID       :       SDB_OPFOR_UNIT_ID;
      SDB_EQUIP_ID      :       SDB_OPFOR_EQUIP_ID;
      SDB_TIME          :       SYS_DATE_TIME;
      SDB_OPPLAN        :       SYS_OPPLAN;
      SDB_RECORD        :       SYS_DB_SIZE;
   end record;

-- BLUEFOR Ammunition Pointer Record
type SDB_BLUEFOR_AMMO_PTR is
   record
      SDB_UNIT_ID       :       SDB_BLUEFOR_UNIT_ID;
      SDB_TIME          :       SYS_DATE_TIME;
      SDB_OPPLAN        :       SYS_OPPLAN;
      SDB_RECORD        :       SYS_DB_SIZE;
   end record;

-- BLUEFOR Unit Ammunition On-hand Quantity Database Record
type SDB_BLUEFOR_AMMO_QTY is
   record
      SDB_UNIT_ID       :       SDB_BLUEFOR_UNIT_ID;
      SDB_AMMO_ID       :       SDB_BLUEFOR_AMMO_ID;
      SDB_TIME          :       SYS_DATE_TIME;
      SDB_OPPLAN        :       SYS_OPPLAN;
      SDB_ON_HAND       :       SYS_QUANTITY;
   end record;

-- BLUEFOR Ammunition On-hand Pointer Record
type SDB_BLUEFOR_AMMO_QTY_PTR is
   record
      SDB_UNIT_ID       :       SDB_BLUEFOR_UNIT_ID;
      SDB_AMMO_ID       :       SDB_BLUEFOR_AMMO_ID;
      SDB_TIME          :       SYS_DATE_TIME;
      SDB_OPPLAN        :       SYS_OPPLAN;
      SDB_RECORD        :       SYS_DB_SIZE;
   end record;

-- BLUEFOR Fuel Description Database Record
type SDB_FUELS is
   record
      SDB_UNIT_ID       :       SDB_BLUEFOR_UNIT_ID;
      SDB_TIME          :       SYS_DATE_TIME;
      SDB_OPPLAN        :       SYS_OPPLAN;
      SDB_MOGAS_REQ     :       SYS_QUANTITY  range 0..999999;
      SDB_MOGAS_ON_HAND :       SYS_QUANTITY  range 0..999999;
      SDB_AVGAS_REQ     :       SYS_QUANTITY  range 0..999999;
      SDB_AVGAS_ON_HAND :       SYS_QUANTITY  range 0..999999;
      SDB_DIESEL_REQ    :       SYS_QUANTITY  range 0..999999;
      SDB_DIESEL_ON_HAND:       SYS_QUANTITY  range 0..999999;
   end record;

-- BLUEFOR Fuel Pointer Record
type SDB_BLUEFOR_FUEL_PTR is
   record
```

```
      SDB_UNIT_ID       :       SDB_BLUEFOR_UNIT_ID;
      SDB_TIME          :       SYS_DATE_TIME;
      SDB_OPPLAN        :       SYS_OPPLAN;
      SDB_RECORD        :       SYS_DB_SIZE;
   end record;

-- BLUEFOR Unit Personnel Database Record
type SDB_PERSONNEL is
   record
      SDB_UNIT_ID       :       SDB_BLUEFOR_UNIT_ID;
      SDB_TIME          :       SYS_DATE_TIME;
      SDB_OPPLAN        :       SYS_OPPLAN;
      SDB_OFFICERS_AUTH :       SYS_QUANTITY    range 0..9999;
      SDB_OFFICERS_CURR :       SYS_QUANTITY    range 0..9999;
      SDB_ENLISTED_AUTH :       SYS_QUANTITY    range 0..999999;
      SDB_ENLISTED_CURR :       SYS_QUANTITY    range 0..999999;
   end record;

-- BLUEFOR Personnel Pointer Record
type SDB_BLUEFOR_PERS_PTR is
   record
      SDB_UNIT_ID       :       SDB_BLUEFOR_UNIT_ID;
      SDB_TIME          :       SYS_DATE_TIME;
      SDB_OPPLAN        :       SYS_OPPLAN;
      SDB_RECORD        :       SYS_DB_SIZE;
   end record;

-- Force Echelons
-- The task organization tool needs these in descending order!
type SDB_FORCE_ECHELON is (
     ARMY_GROUP, FRONT, ARMY, CORPS, DIVISION, BRIGADE, REGIMENT, GROUP,
     BATTALION, SQUADRON, COMPANY, BATTERY, TROOP, PLATOON, SECTION, SQUAD,
     TEAM);

-- Unit Type (BLUEFOR and OPFOR)
type SDB_UNIT_TYPE is (AIRBORNE, AIR_ASSAULT, AIR_DEFENSE,
     AIR_DEFENSE_MISSLE, ANTI_ARMOR, ARMOR_CAV, ARMOR_TANK, ARTY_TOWED,
     ARTY_SP, ATTACK_HELICOPTER, AVIATION, AVIATION_FW, AVIATION_RW, BAND,
     CAV_RECON, CHEMICAL, CIVIL_AFFAIRS, COMBINED_ARMS_ARMY, ENGINEER,
     FINANCE, INF_MECHANIZED, INF_MOTORIZED,
     MAINTENANCE, MEDICAL, MILITARY_INTEL, MILITARY_POLICE, ORDNANCE,
     PERS_SVC, PSYCH_OPNS, QUARTERMASTER, ROCKET_ARTILLERY, SIGNAL,
     SPECIAL_FORCES, SPT_COM, SUPPLY_SERVICES, SURF_TO_SURF_MISSLE,
     TRANSPORTATION);

-- Battle Function
type SDB_BATTLE_FUNCTION is (
     COMBAT_MANEUVER, COMBAT_SUPPORT, COMBAT_SERVICE_SUPPORT,
     COMMITTED, REINFORCE, ARTILLERY);

-- BLUEFOR Task Organization Relationships
type SDB_BLUEFOR_TO_RELATE is (
     ORGANIC_ASSIGNED, ATTACHED, DS, GS, GSR, OPCON);

-- Force Activity
type SDB_FORCE_ACTIVITY is (
```

```
      ADVANCE_GUARD, ADVANCING, AIR_ASSAULT, AIRBORNE_ASSAULT,
      AIRMOBILE_ASSAULT, AMPHIBIOUS_LANDING, CLOSING, COMMUNICATION,
      COUNTER_ATTACK, COVERING_FORCE, EXPLOITATION, FLANK_GUARD,
      INFILTRATION, MAINTAINING, MANAGING, OCCUPY, PENETRATION, PURSUIT,
      PREPARING, REAR_AREA_OPERATIONS, REAR_GUARD, REARM_REFUEL,
      RECONNAISSANCE, REINFORCING, REORGANIZATION, RIVER_CROSSING, SEARCH,
      SCREEN, SERVICE, SUPPLY, TRANSPORT);

-- Force Mission
type SDB_FORCE_MISSION is (
     ATTACK, DEFEND, DELAYED, RESERVE, SUPPORT, WITHDRAW);

-- BLUEFOR Unit Status Database Record
type SDB_BLUE_UNIT_STATUS is
   record
      SDB_UNIT_ID        :       SDB_BLUEFOR_UNIT_ID;
      SDB_TIME           :       SYS_DATE_TIME;
      SDB_OPPLAN         :       SYS_OPPLAN;
      SDB_NAME           :       string (SDB_UNIT_NAME_LEN);
      SDB_ECHELON        :       SDB_FORCE_ECHELON;
      SDB_TYPE           :       SDB_UNIT_TYPE;
      SDB_BATTLE_FUNC    :       SDB_BATTLE_FUNCTION;
      SDB_TO_RELATE      :       SDB_BLUEFOR_TO_RELATE;
      SDB_PARENT         :       SDB_BLUEFOR_UNIT_ID;
      SDB_HIGHER_ECH     :       SDB_BLUEFOR_UNIT_ID;
      SDB_NEXT_SIBLING   :       SDB_BLUEFOR_UNIT_ID;
      SDB_ASSET_SIBLING  :       SDB_BLUEFOR_UNIT_ID;
      SDB_FIRST_CHILD    :       SDB_BLUEFOR_UNIT_ID;
      SDB_ACTIVITY       :       SDB_FORCE_ACTIVITY;
      SDB_MISSION        :       SDB_FORCE_MISSION;
   end record;

-- BLUEFOR Unit Status Pointer Record
type SDB_BLUEFOR_STATUS_PTR is
   record
      SDB_UNIT_ID        :       SDB_BLUEFOR_UNIT_ID;
      SDB_TIME           :       SYS_DATE_TIME;
      SDB_OPPLAN         :       SYS_OPPLAN;
      SDB_RECORD         :       SYS_DB_SIZE;
   end record;

-- BLUEFOR Unit Location Pointer Record
type SDB_BLUEFOR_LOCATION_PTR is
   record
      SDB_UNIT_ID        :       SDB_BLUEFOR_UNIT_ID;
      SDB_TIME           :       SYS_DATE_TIME;
      SDB_OPPLAN         :       SYS_OPPLAN;
      SDB_RECORD         :       SYS_DB_SIZE;
   end record;

-- OPFOR Unit Status Database Record
type SDB_OPFOR_UNIT_STATUS is
   record
      SDB_UNIT_ID        :       SDB_BLUEFOR_UNIT_ID;
      SDB_TIME           :       SYS_DATE_TIME;
      SDB_OPPLAN         :       SYS_OPPLAN;
```

```
      SDB_NAME           :       string (SDB_UNIT_NAME_LEN);
      SDB_ECHELON        :       SDB_FORCE_ECHELON;
      SDB_TYPE           :       SDB_UNIT_TYPE;
      SDB_PARENT         :       SDB_OPFOR_UNIT_ID;
      SDB_HIGHER_ECH     :       SDB_OPFOR_UNIT_ID;
      SDB_NEXT_SIBLING   :       SDB_OPFOR_UNIT_ID;
      SDB_FIRST_CHILD    :       SDB_OPFOR_UNIT_ID;
      SDB_MISSION        :       SDB_FORCE_MISSION;
      SDB_ACTIVITY       :       SDB_FORCE_ACTIVITY;
      SDB_REINFORCE_HR   :       SYS_HOUR;
      SDB_PERCENT_STR    :       SYS_PERCENT;
   end record;

-- OPFOR Unit Status Pointer Record
type SDB_OPFOR_STATUS_PTR is
   record
      SDB_UNIT_ID        :       SDB_OPFOR_UNIT_ID;
      SDB_TIME           :       SYS_DATE_TIME;
      SDB_OPPLAN         :       SYS_OPPLAN;
      SDB_RECORD         :       SYS_DB_SIZE;
   end record;

-- OPFOR Unit Location Pointer Record
type SDB_OPFOR_LOCATION_PTR is
   record
      SDB_UNIT_ID        :       SDB_OPFOR_UNIT_ID;
      SDB_TIME           :       SYS_DATE_TIME;
      SDB_OPPLAN         :       SYS_OPPLAN;
      SDB_RECORD         :       SYS_DB_SIZE;
   end record;

-- Control Measure Types in order by Area, Crossing, Fire Plan, Line,
-- Map Feature, Point, Route.
type SDB_CONTROL_MEASURE_TYPE is (
     AREA_OF_OPERATIONS, ASSEMBLY_AREA, ATTACK_POSITION, BATTLE_POSITION,
     BRIGADE_SUPPORT_AREA, BATTALION_SUPPORT_AREA, DIVISION_SUPPORT_AREA,
     DROP_ZONE, FREE_FIRE_AREA, LANDING_ZONE, NO_FIRE_AREA, OBJECTIVE,
     RESTRICTIVE_FIRE_AREA, ZONE_OF_ACTION,
     ASSAULT_CROSSING, RAFT_SITE,
     GROUP_OF_TARGETS,
     BOUNDARY, BRIDGEHEAD_LINE, COORDINATED_FIRE_LINE, FEBA,
     FIRE_SUP_COORD_LINE, FORWARD_LINE_OF_TROOPS, HOLDING_LINE, LIGHT_LINE,
     LIMIT_OF_ADVANCE, LINE_OF_CONTACT, LINE_OF_DEPARTURE, PHASE_LINE,
     COA_LINE, RESTRICTIVE_FIRE_LINE,
     AIR_FIELD, BRIDGE, BUILDING, CITY, LAKE, MAP_REFERENCE_POINT,
     MOUNTAIN_PEAK_HILL_TOP, ROAD_INTERSECTION, TOWN, VILLAGE,
     CHECKPOINT, COLLECTION_POINT, CONTACT_POINT, COORDINATING_POINT,
     CRITICAL_EVENT, LINK_UP_POINT, PASSAGE_POINT, POINT_OF_DEPARTURE,
     RELEASE_POINT, START_POINT, STRONG_POINT, TRAFFIC_CONTROL_POINT,
     AIR_AXIS_OF_ADVANCE, AIR_CORRIDOR, GRND_AXIS_OF_ADV_MAIN_ATK,
     GRND_AXIS_OF_ADV_SUPPORT, DIRECTION_OF_ATTACK, FEINT, MAIN_SUPPLY_ROUTE,
     ROUTE);

subtype SDB_AREA_CM_RANGE      is     SDB_CONTROL_MEASURE_TYPE range
                                      AREA_OF_OPERATIONS..ZONE_OF_ACTION;
subtype SDB_CROSSING_CM_RANGE  is     SDB_CONTROL_MEASURE_TYPE range
```

```
                                        ASSAULT_CROSSING..RAFT_SITE;
subtype SDB_FIRE_PLAN_CM_RANGE is       SDB_CONTROL_MEASURE_TYPE range
                                        GROUP_OF_TARGETS..GROUP_OF_TARGETS;
subtype SDB_LINE_CM_RANGE      is       SDB_CONTROL_MEASURE_TYPE range
                                        BOUNDARY..RESTRICTIVE_FIRE_LINE;
subtype SDB_MAP_FEAT_CM_RANGE  is       SDB_CONTROL_MEASURE_TYPE range
                                        AIR_FIELD..VILLAGE;
subtype SDB_POINT_CM_RANGE     is       SDB_CONTROL_MEASURE_TYPE range
                                        CHECKPOINT..TRAFFIC_CONTROL_POINT;
subtype SDB_ROUTE_CM_RANGE     is       SDB_CONTROL_MEASURE_TYPE range
                                        AIR_AXIS_OF_ADVANCE..ROUTE;

-- Control Measure Location Type
type SDB_CONTROL_MEASURE_LOC_TYPE is (
     AREA, CROSSING, FIRE_PLAN, LINE, MAP_FEATURE, POINT, ROUTE);

-- Control Measure Status
type SDB_CONTROL_MEASURE_STATUS is (
     PLANNED, ACTUAL);

-- Points defining the Control Measure
type SDB_CONTROL_MEASURE_POINTS is array (SDB_CONTROL_MEASURE_PT) of
                                SDB_LOCATION_REC;

-- Boolean array of map scales to be displayed in
type SDB_CONTROL_MEASURE_SCALES is array (SYS_MAP_SCALES) of
                                BOOLEAN;

-- Control Measure Database Record
type SDB_CONTROL_MEASURE_REC is
   record
      SDB_ID              :       SDB_CONTROL_MEASURE_ID;
      SDB_OPPLAN          :       SYS_OPPLAN;
      SDB_NAME            :       string (SDB_CNTL_MSR_NAME_LEN);
      SDB_SIDE            :       SDB_SIDE_TYPE;
      SDB_OWNER_BLUE      :       SDB_BLUEFOR_UNIT_ID;
      SDB_OWNER_OPFOR     :       SDB_OPFOR_UNIT_ID;
      SDB_TYPE            :       SDB_CONTROL_MEASURE_TYPE;
      SDB_LOCATION_TYPE   :       SDB_CONTROL_MEASURE_LOC_TYPE;
      SDB_SCALE           :       SDB_CONTROL_MEASURE_SCALES;
      SDB_STATUS          :       SDB_CONTROL_MEASURE_STATUS;
      SDB_EFF_FROM_DATE   :       SYS_DATE_TIME;
      SDB_EFF_TO_DATE     :       SYS_DATE_TIME;
      SDB_LABEL_ECHELON   :       SDB_FORCE_ECHELON;
      SDB_NUMBER_POINTS   :       SDB_CONTROL_MEASURE_PT;
      SDB_LOCATION        :       SDB_CONTROL_MEASURE_POINTS;
   end record;

-- Control Measure Pointer Record
type SDB_CONTROL_MEASURE_PTR is
   record
      SDB_CNTRL_MSR_ID    :       SDB_CONTROL_MEASURE_ID;
      SDB_OPPLAN          :       SYS_OPPLAN;
      SDB_EFF_FROM        :       SYS_DATE_TIME;
      SDB_EFF_TO          :       SYS_DATE_TIME;
      SDB_RECORD          :       SYS_DB_SIZE;
```

```
   end record;

-- Point Control Measure Database Record
type SDB_CNTRL_MSR_POINT_REC is
   record
      SDB_ID              :       SDB_CONTROL_MEASURE_ID;
      SDB_OPPLAN          :       SYS_OPPLAN;
      SDB_NAME            :       string (SDB_CNTL_MSR_NAME_LEN);
      SDB_SIDE            :       SDB_SIDE_TYPE;
      SDB_OWNER_BLUE      :       SDB_BLUEFOR_UNIT_ID;
      SDB_OWNER_OPFOR     :       SDB_OPFOR_UNIT_ID;
      SDB_TYPE            :       SDB_CONTROL_MEASURE_TYPE;
      SDB_LOCATION_TYPE   :       SDB_CONTROL_MEASURE_LOC_TYPE;
      SDB_SCALE           :       SDB_CONTROL_MEASURE_SCALES;
      SDB_STATUS          :       SDB_CONTROL_MEASURE_STATUS;
      SDB_EFF_FROM_DATE   :       SYS_DATE_TIME;
      SDB_EFF_TO_DATE     :       SYS_DATE_TIME;
      SDB_LABEL_ECHELON   :       SDB_FORCE_ECHELON;
      SDB_LOCATION        :       SDB_LOCATION_REC;
   end record;

-- Control Measure Pointer Record
type SDB_CNTRL_MSR_POINT_PTR is
   record
      SDB_CNTRL_MSR_ID    :       SDB_CONTROL_MEASURE_ID;
      SDB_OPPLAN          :       SYS_OPPLAN;
      SDB_EFF_FROM        :       SYS_DATE_TIME;
      SDB_EFF_TO          :       SYS_DATE_TIME;
      SDB_RECORD          :       SYS_DB_SIZE;
   end record;

-- Obstacle Types
type SDB_OBSTACLE_TYPE is (
     ABATIS, ANTI_TANK_DITCH, BRIDGE_DEMO, CHEMICAL, CRATER, DAM_DEMO,
     FLOODING, LOG_POSTS, MINEFIELD_AP, MINEFIELD_AT, MINEFIELD_AT_AP,
     NUCLEAR, SCAT_MINEFIELD_AP, SCAT_MINEFIELD_AT, SCAT_MINEFIELD_AT_AP,
     TUNNEL_DEMO, WIRE);

-- Obstacle Status
type SDB_OBSTACLE_STATUS is (
     PLANNED, PREPARED, EXECUTED, BREACHED);

--  Obstacle Database Record
type SDB_OBSTACLE_REC is
   record
      SDB_ID              :       SDB_OBSTACLE_ID;
      SDB_OPPLAN          :       SYS_OPPLAN;
      SDB_SIDE            :       SDB_SIDE_TYPE;
      SDB_TYPE            :       SDB_OBSTACLE_TYPE;
      SDB_STATUS          :       SDB_OBSTACLE_STATUS;
      SDB_EFF_FROM_DATE   :       SYS_DATE_TIME;
      SDB_EFF_TO_DATE     :       SYS_DATE_TIME;
      SDB_LOCATION        :       SDB_LOCATION_REC;
      SDB_FRONTAGE        :       SYS_WIDTH_DEPTH;
      SDB_DEPTH           :       SYS_WIDTH_DEPTH;
      SDB_ORIENTATION     :    p*p$ 85EGREE;
```

```
      SDB_LANES_OR_GAPS :        boolean;
      SDB_ECHELON       :        SDB_FORCE_ECHELON;
   end record;

-- Obstacle Pointer Record
type SDB_OBSTACLE_PTR is
   record
      SDB_ID            :        SDB_OBSTACLE_ID;
      SDB_OPPLAN        :        SYS_OPPLAN;
      SDB_EFF_FROM      :        SYS_DATE_TIME;
      SDB_EFF_TO        :        SYS_DATE_TIME;
      SDB_RECORD        :        SYS_DB_SIZE;
   end record;

-- Structures definitions for messages passed through routers

-- Ammunition Structures
-- Unit Authorized Ammunition Description
type SDB_AMMO_REC is
   record
      SDB_ID            :        SDB_AMMUNITION;
      SDB_NAME          :        string (SDB_AMMO_NAME_LEN);
      SDB_BASIC_LOAD    :        SYS_QUANTITY;
      SDB_KEY_ITEM      :        BOOLEAN;
   end record;

-- Authorized Ammunition Array
type SDB_AMMO_ARRAY is array (SDB_AMMUNITION) of
                              SDB_AMMO_REC;
type SDB_AMMO_POINT  is   access SDB_AMMO_ARRAY;

-- Unit Ammunition authorized list record
type SDB_AMMO_AUTH_LIST is
   record
      SDB_UNIT_ID       :        SDB_UNIT;
      SDB_TIME          :        SYS_DATE_TIME;
      SDB_OPPLAN        :        SYS_OPPLAN;
      SDB_COUNT         :        SDB_AMMUNITION;
      SDB_LIST          :        SDB_AMMO_ARRAY;
   end record;

-- Ammunition on-hand list
type SDB_AMMO_ON_HAND_LIST    is  array (SDB_BLUEFOR_AMMO_OWN)
                                      of SYS_QUANTITY;
type SDB_AMMO_ON_HAND_POINT is access SDB_AMMO_ON_HAND_LIST;

type SDB_AMMO_ON_HAND_REC is
   record
      SDB_UNIT_ID           :  SDB_UNIT;
      SDB_TIME              :  SYS_DATE_TIME;
      SDB_NUMBER_TYPES      :  SDB_AMMUNITION;
      SDB_LIST              :  SDB_AMMO_ON_HAND_LIST;
end record;

type SDB_AMMO_UPDATE_REC is
   record
```

```
      SDB_UNIT_ID           :  SDB_UNIT;
      SDB_TIME              :  SYS_DATE_TIME;
      SDB_OPPLAN            :  SYS_OPPLAN;
      SDB_SIDE              :  SDB_SIDE_TYPE;
      SDB_AMMO_ID           :  SDB_AMMUNITION;
      SDB_AMOUNT_CHG        :  SYS_QUANTITY;
      SDB_KEY_ITEM          :  BOOLEAN;
end record;

-- Equipment Structures
-- Unit Authorized Equipment Description
type SDB_EQUIP_CATEGORY is (PACING_ITEM, SUPPORT_SYSTEM, C3_SYSTEM,
             OTHER_ITEM);
type SDB_EQUIP_REC is
   record
      SDB_ID             :       SDB_EQUIPMENT;
      SDB_NAME           :       string (SDB_EQUIP_NAME_LEN);
      SDB_AUTHORIZED     :       SYS_QUANTITY;
      SDB_CATEGORY       :       SDB_EQUIP_CATEGORY;
   end record;

-- Authorized Equipment Array
type SDB_EQUIP_ARRAY is array (SDB_EQUIPMENT) of
                              SDB_EQUIP_REC;
type SDB_EQUIP_POINT  is   access SDB_EQUIP_ARRAY;

-- Unit Equipment authorized list record
type SDB_EQUIP_AUTH_LIST is
   record
      SDB_UNIT_ID        :       SDB_UNIT;
      SDB_TIME           :       SYS_DATE_TIME;
      SDB_OPPLAN         :       SYS_OPPLAN;
      SDB_COUNT          :       SDB_EQUIPMENT;
      SDB_LIST           :       SDB_EQUIP_ARRAY;
   end record;

-- Operational Equipment list
type SDB_EQUIP_OPER_LIST is  array (SDB_EQUIPMENT)
                                 of SYS_QUANTITY;
type SDB_EQUIP_OPER_POINT is access SDB_EQUIP_OPER_LIST;

type SDB_EQUIP_OPER_REC is
   record
      SDB_UNIT_ID           :  SDB_UNIT;
      SDB_TIME              :  SYS_DATE_TIME;
      SDB_SIDE              :  SDB_SIDE_TYPE;
      SDB_NUMBER_TYPES      :  SDB_EQUIPMENT;
      SDB_LIST              :  SDB_EQUIP_OPER_LIST;
end record;

type SDB_EQUIP_UPDATE_REC is
   record
      SDB_UNIT_ID           :  SDB_UNIT;
      SDB_TIME              :  SYS_DATE_TIME;
      SDB_OPPLAN            :  SYS_OPPLAN;
      SDB_SIDE              :  SDB_SIDE_TYPE;
```

```
      SDB_EQUIP_ID          :  SDB_EQUIPMENT;
      SDB_AMOUNT_CHG        :  SYS_QUANTITY;
      SDB_CATEGORY          :  SDB_EQUIP_CATEGORY;
end record;

-- Personnel Structures
type SDB_PERS_UPDATE_REC is
   record
      SDB_UNIT_ID           :  SDB_UNIT;
      SDB_TIME              :  SYS_DATE_TIME;
      SDB_OPPLAN            :  SYS_OPPLAN;
      SDB_SIDE              :  SDB_SIDE_TYPE;
      SDB_OFFICER_CHG       :  SYS_QUANTITY;
      SDB_ENLISTED_CHG      :  SYS_QUANTITY;
end record;

-- Fuel Structures
type SDB_FUEL_UPDATE_REC is
   record
      SDB_UNIT_ID           :  SDB_UNIT;
      SDB_TIME              :  SYS_DATE_TIME;
      SDB_OPPLAN            :  SYS_OPPLAN;
      SDB_SIDE              :  SDB_SIDE_TYPE;
      SDB_MOGAS_CHG         :  SYS_QUANTITY;
      SDB_AVGAS_CHG         :  SYS_QUANTITY;
      SDB_DIESEL_CHG        :  SYS_QUANTITY;
end record;

-- Unit Status Structures
type SDB_ACTIVITY_UPDATE_REC is
   record
      SDB_UNIT_ID           :  SDB_UNIT;
      SDB_TIME              :  SYS_DATE_TIME;
      SDB_OPPLAN            :  SYS_OPPLAN;
      SDB_SIDE              :  SDB_SIDE_TYPE;
      SDB_ACTIVITY          :  SDB_FORCE_ACTIVITY;
end record;

type SDB_MISSION_UPDATE_REC is
   record
      SDB_UNIT_ID           :  SDB_UNIT;
      SDB_TIME              :  SYS_DATE_TIME;
      SDB_OPPLAN            :  SYS_OPPLAN;
      SDB_SIDE              :  SDB_SIDE_TYPE;
      SDB_MISSION           :  SDB_FORCE_MISSION;
end record;

type SDB_OPFOR_REINF_UPDATE_REC is
   record
      SDB_UNIT_ID           :  SDB_OPFOR_UNIT_ID;
      SDB_TIME              :  SYS_DATE_TIME;
      SDB_OPPLAN            :  SYS_OPPLAN;
      SDB_HOUR_CHG          :  SYS_HOUR;
end record;

type SDB_OPFOR_STR_UPDATE_REC is
```

```
      record
         SDB_UNIT_ID            :  SDB_OPFOR_UNIT_ID;
         SDB_TIME               :  SYS_DATE_TIME;
         SDB_OPPLAN             :  SYS_OPPLAN;
         SDB_PERCENT_CHG        :  SYS_PERCENT;
   end record;

   -- Unit Location Structures
   type SDB_UNIT_LOCATION is
      record
         SDB_UNIT_ID        :        SDB_UNIT;
         SDB_TIME           :        SYS_DATE_TIME;
         SDB_OPPLAN         :        SYS_OPPLAN;
         SDB_LOCATION       :        SDB_LOCATION_REC;
      end record;

   type SDB_LOCATION_MSG_REC is
      record
         SDB_UNIT_ID            :  SDB_UNIT;
         SDB_NAME               :  string (SDB_UNIT_NAME_LEN);
         SDB_ECHELON            :  SDB_FORCE_ECHELON;
         SDB_TYPE               :  SDB_UNIT_TYPE;
         SDB_BATTLE_FUNC        :  SDB_BATTLE_FUNCTION;
         SDB_LOCATION           :  SDB_LOCATION_REC;
      end record;

   type SDB_LOCATION_LIST is array (SDB_UNIT range <>) of
                                    SDB_LOCATION_MSG_REC;
   type SDB_LOCATION_LIST_POINT is access SDB_LOCATION_LIST;

   type SDB_ALL_LOC_REC is
      record
         SDB_TIME               :  SYS_DATE_TIME;
         SDB_NUMBER_UNITS       :  SDB_UNIT;
         SDB_LIST               :  SDB_LOCATION_LIST (SDB_UNIT);
   end record;

   type SDB_LOCATION_UPDATE_REC is
      record
         SDB_UNIT_ID            :  SDB_UNIT;
         SDB_TIME               :  SYS_DATE_TIME;
         SDB_OPPLAN             :  SYS_OPPLAN;
         SDB_SIDE               :  SDB_SIDE_TYPE;
         SDB_LOCATION           :  SDB_LOCATION_REC;
   end record;

-- BLUEFOR Task Organization Structures
   type SDB_BLUE_TASK_RECORD is
      record
         SDB_UNIT_ID               :  SDB_BLUEFOR_UNIT_ID;
         SDB_NAME                  :  string (SDB_UNIT_NAME_LEN);
         SDB_ABBREV_NAME           :  string (SDB_UNIT_NAME_LEN);
         SDB_HIGHER_ECHELON        :  SDB_BLUEFOR_UNIT_ID;
         SDB_NEXT_SIBLING          :  SDB_BLUEFOR_UNIT_ID;
         SDB_FIRST_CHILD           :  SDB_BLUEFOR_UNIT_ID;
         SDB_RELATE                :  SDB_BLUEFOR_TO_RELATE;
```

```
        SDB_TYPE                    :  SDB_UNIT_TYPE;
        SDB_ECHELON                 :  SDB_FORCE_ECHELON;
        SDB_BATTLE_FUNC             :  SDB_BATTLE_FUNCTION;
    end record;

    type SDB_BLUEFOR_TASK_ORG_LIST is array (SDB_BLUEFOR_UNIT_ID range <>) of
                                        SDB_BLUE_TASK_RECORD;
    type SDB_BLUEFOR_TASK_POINT is access SDB_BLUEFOR_TASK_ORG_LIST;

    type SDB_BLUE_TASK_ORG_REC is
       record
          SDB_TIME                  :  SYS_DATE_TIME;
          SDB_NUMBER_UNITS          :  SDB_BLUEFOR_UNIT_ID;
          SDB_LIST                  :  SDB_BLUEFOR_TASK_ORG_LIST (
                                       SDB_BLUEFOR_UNIT_ID);
    end record;

    type SDB_BLUE_TASK_ORG_UPDATE_REC is
       record
          SDB_UNIT_ID               :  SDB_BLUEFOR_UNIT_ID;
          SDB_TIME                  :  SYS_DATE_TIME;
          SDB_OPPLAN                :  SYS_OPPLAN;
          SDB_HIGHER_ECHELON        :  SDB_BLUEFOR_UNIT_ID;
          SDB_RELATE                :  SDB_BLUEFOR_TO_RELATE;
    end record;

-- OPFOR Task Organization Structures
    type SDB_OPFOR_TASK_RECORD is
       record
          SDB_UNIT_ID               :  SDB_OPFOR_UNIT_ID;
          SDB_NAME                  :  string (SDB_UNIT_NAME_LEN);
          SDB_HIGHER_ECHELON        :  SDB_OPFOR_UNIT_ID;
          SDB_NEXT_SIBLING          :  SDB_OPFOR_UNIT_ID;
          SDB_FIRST_CHILD           :  SDB_OPFOR_UNIT_ID;
          SDB_TYPE                  :  SDB_UNIT_TYPE;
          SDB_ECHELON               :  SDB_FORCE_ECHELON;
          SDB_BATTLE_FUNC           :  SDB_BATTLE_FUNCTION;
    end record;

    type SDB_OPFOR_TASK_ORG_LIST   is array (SDB_OPFOR_UNIT_ID range <>) of
                                        SDB_OPFOR_TASK_RECORD;
    type SDB_OPFOR_TASK_POINT is access SDB_OPFOR_TASK_ORG_LIST;

    type SDB_OPFOR_TASK_ORG_REC is
       record
          SDB_TIME                  :  SYS_DATE_TIME;
          SDB_NUMBER_UNITS          :  SDB_OPFOR_UNIT_ID;
          SDB_LIST                  :  SDB_OPFOR_TASK_ORG_LIST (SDB_OPFOR_UNIT_ID);
    end record;

    type SDB_OPFOR_TASK_ORG_UPDATE_REC is
       record
          SDB_UNIT_ID               :  SDB_OPFOR_UNIT_ID;
          SDB_TIME                  :  SYS_DATE_TIME;
          SDB_OPPLAN                :  SYS_OPPLAN;
          SDB_HIGHER_ECHELON        :  SDB_OPFOR_UNIT_ID;
```

```
   end record;

-- Control Measure Structure
   type SDB_CONTROL_MSR_LIST      is array (SDB_CONTROL_MEASURE_ID range <>) of
                                     SDB_CONTROL_MEASURE_REC;
   type SDB_CONTROL_MSR_POINT is access SDB_CONTROL_MSR_LIST;

   type SDB_ALL_CNTRL_MSR is
      record
         SDB_NUMBER_CM          :  SDB_CONTROL_MEASURE_ID;
         SDB_LIST               :  SDB_CONTROL_MSR_LIST (SDB_CONTROL_MEASURE_ID);
   end record;

   type SDB_CNTRL_POINT_LIST      is array (SDB_CONTROL_MEASURE_ID range <>) of
                                     SDB_CNTRL_MSR_POINT_REC;
   type SDB_CNTRL_POINT_POINT is access SDB_CNTRL_POINT_LIST;

   type SDB_ALL_CNTRL_POINT is
      record
         SDB_NUMBER_CM          :  SDB_CONTROL_MEASURE_ID;
         SDB_LIST               :  SDB_CNTRL_POINT_LIST (SDB_CONTROL_MEASURE_ID);
   end record;

   type SDB_CNTRL_MSR_LOC_REC is
      record
         SDB_ID                 :  SDB_CONTROL_MEASURE_ID;
         SDB_TIME               :  SYS_DATE_TIME;
         SDB_OPPLAN             :  SYS_OPPLAN;
         SDB_LOCATION_TYPE      :  SDB_CONTROL_MEASURE_LOC_TYPE;
         SDB_LOCATION           :  SDB_CONTROL_MEASURE_POINTS;
   end record;

   type SDB_CNTRL_MSR_STAT_REC is
      record
         SDB_ID                 :  SDB_CONTROL_MEASURE_ID;
         SDB_TIME               :  SYS_DATE_TIME;
         SDB_OPPLAN             :  SYS_OPPLAN;
         SDB_LOCATION_TYPE      :  SDB_CONTROL_MEASURE_LOC_TYPE;
         SDB_STATUS             :  SDB_CONTROL_MEASURE_STATUS;
   end record;

   type SDB_CNTRL_MSR_EFF_REC is
      record
         SDB_ID                 :  SDB_CONTROL_MEASURE_ID;
         SDB_OPPLAN             :  SYS_OPPLAN;
         SDB_LOCATION_TYPE      :  SDB_CONTROL_MEASURE_LOC_TYPE;
         SDB_EFFECT_FROM        :  SYS_DATE_TIME;
         SDB_EFFECT_TO          :  SYS_DATE_TIME;
   end record;

-- Obstacle Structure
   type SDB_OBSTACLE_LIST         is array (SDB_OBSTACLE_ID range <>) of
                                     SDB_OBSTACLE_REC;
   type SDB_OBSTACLE_POINT is access SDB_OBSTACLE_LIST;

   type SDB_ALL_OBSTACLE is
```

```
      record
         SDB_NUMBER_OBS          :  SDB_OBSTACLE_ID;
         SDB_LIST                :  SDB_OBSTACLE_LIST (SDB_OBSTACLE_ID);
   end record;

   type SDB_OBSTACLE_LOC_REC is
      record
         SDB_ID                  :  SDB_OBSTACLE_ID;
         SDB_TIME                :  SYS_DATE_TIME;
         SDB_OPPLAN              :  SYS_OPPLAN;
         SDB_LOCATION            :  SDB_LOCATION_REC;
   end record;

   type SDB_OBSTACLE_STAT_REC is
      record
         SDB_ID                  :  SDB_OBSTACLE_ID;
         SDB_TIME                :  SYS_DATE_TIME;
         SDB_OPPLAN              :  SYS_OPPLAN;
         SDB_STATUS              :  SDB_OBSTACLE_STATUS;
   end record;

   type SDB_OBSTACLE_EFF_REC is
      record
         SDB_ID                  :  SDB_OBSTACLE_ID;
         SDB_OPPLAN              :  SYS_OPPLAN;
         SDB_EFFECT_FROM         :  SYS_DATE_TIME;
         SDB_EFFECT_TO           :  SYS_DATE_TIME;
   end record;

-- Operational Planning records

   type SDB_OPPLAN_TYPE is (G2_PERSONAL, G3_PERSONAL, G4_PERSONAL, EX_PERSONAL,
        SHARED, BASE_SCENARIO);

   -- List of Operational plans
   type SDB_OPPLAN_REC is
      record
         SDB_OPPLAN_ID           :  SYS_OPPLAN;
         SDB_TYPE                :  SDB_OPPLAN_TYPE;
         SDB_OPPLAN_NAME         :  STRING (SYS_POP_UP_TEXT);
         SDB_BASE                :  SYS_OPPLAN;
         SDB_DATE_TIME           :  SYS_DATE_TIME;
   end record;

   -- List of current Operational Plans
   type SDB_OPPLAN_LIST is array (SYS_OPPLAN) of SDB_OPPLAN_REC;

   type SDB_OPPLAN_LIST_REC is
      record
         SDB_COUNT               :  SYS_OPPLAN;
         SDB_LIST                :  SDB_OPPLAN_LIST;
   end record;

   -- New Operational Plan record
   type SDB_NEW_OPPLAN_REC is
      record
```

```
        SDB_OPPLAN_ID          :  SYS_OPPLAN;
        SDB_TYPE               :  SDB_OPPLAN_TYPE;
        SDB_OPPLAN_NAME        :  STRING (SYS_POP_UP_TEXT);
        SDB_BASE               :  SYS_OPPLAN;
        SDB_TIME               :  SYS_DATE_TIME;
    end record;

end SDB_SITUATION_DB;
```

```
--cpc package specification name: SYSTEM_PACKAGE
--
--cpc description: Defines types that are used throughout the EDDIC system.
--
--cpc design notes:
--
--cpc package author: Bruce Packard
--                    Science Applications International Corporation
--                    424 Delaware, Suite C3
--                    Leavenworth, KS 66048
--

package SYSTEM_PACKAGE is

-- Computer limitations

   -- Number of bits in a byte
   SYS_BITS_IN_BYTE          :  constant INTEGER  := 8;
   SYS_BITS_IN_NIBBLE        :  constant INTEGER  := 4;

-- EDDIC database limitations;

   -- Number of records in a database
   type SYS_DB_SIZE        is range 0..INTEGER'LAST;
   for SYS_DB_SIZE'SIZE use 4*SYS_BITS_IN_BYTE;

   -- Number of characters for names in a database
   subtype SYS_NAME_SIZE      is INTEGER range 1..40;

   -- File descriptor storage for using C based file I/O utilities
   type SYS_FILE_DESC      is range INTEGER'FIRST..INTEGER'LAST;
   for SYS_FILE_DESC'SIZE use 4*SYS_BITS_IN_BYTE;

   -- Asset Quantity range
   type SYS_QUANTITY       is range -999999..999999;

   -- Width and Depth of items
   type SYS_WIDTH_DEPTH    is range 0..9999;

   -- Orientation - as in a compass
   type SYS_DEGREE        is range 0..359;

   -- String Items
   subtype SYS_TEXT is string;
   type SYS_TEXT_PTR is access SYS_TEXT;

-- EDDIC product limits

   -- Number of characters in header
   subtype SYS_HEADER_LENGTH      is INTEGER range 0..600;

   -- Number of characters in a Product
   subtype SYS_PRODUCT_LENGTH     is INTEGER range 0..32000;

   -- Product Categories
```

```
    type SYS_PRODUCT_CAT is (DETAIL, AGGREGATE, SUMMARY);

    -- Blank Line for blank filling strings
    subtype SYS_BLANK_LEN  is INTEGER range 1..80;
    SYS_BLANK              :   string (SYS_BLANK_LEN) := (SYS_BLANK_LEN => ' ');

-- EDDIC pop-up and walking menu limitations

    -- Types used by UED_WALKING_MENU to create walking menus
    subtype SYS_MENU_NAME_LEN    is  INTEGER  range 1..30;
    type SYS_MENU_TREE_LIMIT     is  range 0..1000;
    subtype SYS_MENU_TREE_STRING is STRING (SYS_MENU_NAME_LEN);
    type SYS_MENU_TREE     is    array (SYS_MENU_TREE_LIMIT range <>) of
                                 SYS_MENU_TREE_STRING;
    type SYS_MENU_TREE_PTR  is   access SYS_MENU_TREE;

    -- Menu Tree limits for the types of menu trees in the system;
    subtype SYS_REF_TREE         is  SYS_MENU_TREE_LIMIT range 0..400;
    subtype SYS_HELP_TREE        is  SYS_MENU_TREE_LIMIT range 0..400;
    subtype SYS_VIEW_C2_TREE     is  SYS_MENU_TREE_LIMIT range 0..800;
    subtype SYS_BUILD_C2_TREE    is  SYS_MENU_TREE_LIMIT range 0..100;
    subtype SYS_CONTROL_TREE     is  SYS_MENU_TREE_LIMIT range 0..100;
    subtype SYS_MAP_TREE         is  SYS_MENU_TREE_LIMIT range 0..200;
    subtype SYS_UNIT_TREE        is  SYS_MENU_TREE_LIMIT range 0..30;
    subtype SYS_CM_TREE          is  SYS_MENU_TREE_LIMIT range 0..20;
    subtype SYS_OBS_TREE         is  SYS_MENU_TREE_LIMIT range 0..20;

    -- Number of pop-up menus in a walking menu
    type SYS_WALKING_MENU is range 0..250;
    for SYS_WALKING_MENU'SIZE use 2*SYS_BITS_IN_BYTE;

    -- Number of pop-up menu cells in a walking menu
    type SYS_WALKING_CELL is range 0..800;
    for SYS_WALKING_CELL'SIZE use 2*SYS_BITS_IN_BYTE;

    -- Values of pop-up menu cells in a walking menu
    type SYS_WALKING_CELL_VALUE is range -1..800;
    for SYS_WALKING_CELL_VALUE'SIZE use 2*SYS_BITS_IN_BYTE;

    -- Number of pop-up menu cells in a pop-up menu
    type SYS_POP_UP_CELL is range 0..20;
    for SYS_POP_UP_CELL'SIZE use 2*SYS_BITS_IN_BYTE;

    -- Length of the text in a pop-up menu element (Last char must be a null
    subtype SYS_POP_UP_TEXT is INTEGER range 1..21;

    -- Text for each cell of each pop-up menu in the walking menu
    subtype SYS_MENU_TEXT_STRING is STRING (SYS_POP_UP_TEXT);
    type SYS_MENU_TEXT is array (SYS_WALKING_CELL range <>) of
                          SYS_MENU_TEXT_STRING;
    type SYS_MENU_TEXT_PTR is access SYS_MENU_TEXT;

    -- Pop-up index of the pop-up menu that is the child of each pop-up menu cell
    -- index into UWN_POP_UP_START and UWN_POP_UP_LENGTH
    type SYS_POP_UP_CHILD is array (SYS_WALKING_CELL range <>) of
                         SYS_WALKING_MENU;
```

```
type SYS_POP_UP_CHILD_PTR  is   access SYS_POP_UP_CHILD;

-- Index into SYS_POP_UP_TEXT for the start of each pop-up menu in the
-- walking menu
type SYS_POP_UP_START is array (SYS_WALKING_MENU range <>) of
                      SYS_WALKING_CELL;
type SYS_POP_UP_START_PTR  is   access SYS_POP_UP_START;

-- Number of cells in each pop-up menu
type SYS_POP_UP_LENGTH is array (SYS_WALKING_MENU range <>) of
                      SYS_POP_UP_CELL;
type SYS_POP_UP_LENGTH_PTR  is   access SYS_POP_UP_LENGTH;

-- Types of Pop-up menus for the digital map
type SYS_MAP_MENUS is (MAP_CONTROL_MENU, BLUEFOR_UNIT_MENU,
     BLUEFOR_CNTRL_MSR_MENU, BLUEFOR_OBSTACLE_MENU, OPFOR_UNIT_MENU,
     OPFOR_CNTRL_MSR_MENU, OPFOR_OBSTACLE_MENU);

-- Types of products in the system
type SYS_PRODUCT  is  (TEXT_REPORT, TACTICAL_OVERLAY, FORM,
     INSTRUCTIONS, FEEDBACK, ACKNOWLEDGEMENT);

-- Map and map overlay menu options
type SYS_MAP_CONTROL is
   (GRID_ON, CONTOUR_ON, ROAD_ON, HYDRO_ON, URBAN_ON, MISC_ON,
    GRID_OFF, CONTOUR_OFF, ROAD_OFF, HYDRO_OFF, URBAN_OFF, MISC_OFF,
    FEATURE_MENU,
    BACK_CCM, BACK_ELEV, BACK_SHADE, BACK_3D, BACK_VEG, BACK_NONE,
    SCALE_40, SCALE_80, SCALE_160, SCALE_400, SCALE_800,
    BLUE_DIV_ON, BLUE_BDE_ON, BLUE_BN_ON, BLUE_CO_ON,
    BLUE_DIV_OFF, BLUE_BDE_OFF, BLUE_BN_OFF, BLUE_CO_OFF,
    BLUE_CBT_ON, BLUE_CS_ON, BLUE_CSS_ON,
    BLUE_CBT_OFF, BLUE_CS_OFF, BLUE_CSS_OFF,
    BLUE_NAME_ON, BLUE_NAME_OFF, BLUE_SYMBOL_ON, BLUE_SYMBOL_OFF,
    BLUE_UNIT_MENU,
    OPFOR_DIV_ON, OPFOR_REG_ON, OPFOR_BN_ON, OPFOR_CO_ON,
    OPFOR_DIV_OFF, OPFOR_REG_OFF, OPFOR_BN_OFF, OPFOR_CO_OFF,
    OPFOR_COMMIT_ON, OPFOR_REINF_ON, OPFOR_ARTIL_ON,
    OPFOR_COMMIT_OFF, OPFOR_REINF_OFF, OPFOR_ARTIL_OFF,
    OPFOR_NAME_ON, OPFOR_NAME_OFF, OPFOR_SYMBOL_ON, OPFOR_SYMBOL_OFF,
    OPFOR_UNIT_MENU,
    BLUE_CM_EAC_ON, BLUE_CM_CORP_ON, BLUE_CM_DIV_ON, BLUE_CM_BDE_ON,
    BLUE_CM_BN_ON, BLUE_CM_POINT_ON, BLUE_CM_LINE_ON, BLUE_CM_AREA_ON,
    BLUE_CM_ROUTE_ON, BLUE_CM_OBST_ON, BLUE_CM_CROSS_ON, BLUE_CM_FIRE_ON,
    BLUE_CM_MAPF_ON,
    BLUE_CM_EAC_OFF, BLUE_CM_CORP_OFF, BLUE_CM_DIV_OFF, BLUE_CM_BDE_OFF,
    BLUE_CM_BN_OFF, BLUE_CM_POINT_OFF, BLUE_CM_LINE_OFF, BLUE_CM_AREA_OFF,
    BLUE_CM_ROUTE_OFF, BLUE_CM_OBST_OFF, BLUE_CM_CROSS_OFF, BLUE_CM_FIRE_OFF,
    BLUE_CM_MAPF_OFF,
    BLUE_CM_ECHELON_MENU, BLUE_CM_TYPE_MENU,
    OPFOR_CM_ARMY_ON, OPFOR_CM_DIV_ON, OPFOR_CM_REG_ON, OPFOR_CM_BN_ON,
    OPFOR_CM_POINT_ON, OPFOR_CM_LINE_ON, OPFOR_CM_AREA_ON, OPFOR_CM_ROUTE_ON,
    OPFOR_CM_OBST_ON, OPFOR_CM_CROSS_ON, OPFOR_CM_FIRE_ON, OPFOR_CM_MAPF_ON,
    OPFOR_CM_ARMY_OFF, OPFOR_CM_DIV_OFF, OPFOR_CM_REG_OFF, OPFOR_CM_BN_OFF,
    OPFOR_CM_POINT_OFF, OPFOR_CM_LINE_OFF, OPFOR_CM_AREA_OFF,
    OPFOR_CM_ROUTE_OFF, OPFOR_CM_OBST_OFF, OPFOR_CM_CROSS_OFF,
```

```
OPFOR_CM_FIRE_OFF, OPFOR_CM_MAPF_OFF,
OPFOR_CM_ECHELON_MENU, OPFOR_CM_TYPE_MENU,
BLUE_NEW_AREA_OPER, BLUE_NEW_ASSMBLY_AREA, BLUE_NEW_ATTCK_POS,
BLUE_NEW_BATTLE_POS, BLUE_NEW_BDE_SPT_AREA, BLUE_NEW_BN_SPT_AREA,
BLUE_NEW_DIV_SPT_AREA, BLUE_NEW_DROP_ZONE, BLUE_NEW_FREE_FIRE_AREA,
BLUE_NEW_LAND_ZONE, BLUE_NEW_NO_FIRE_AREA, BLUE_NEW_OBJECTIVE,
BLUE_NEW_RSTRCT_FIRE_AREA, BLUE_NEW_ZONE_ACTION, BLUE_NEW_ASSLT_CROSS,
BLUE_NEW_RAFT_SITE, BLUE_NEW_GRP_TRGTS, BLUE_NEW_BOUNDARY,
BLUE_NEW_BRIDGE_LINE, BLUE_NEW_COORD_FIR_LN, BLUE_NEW_FEBA,
BLUE_NEW_FIRE_SPT_COORD_LN, BLUE_NEW_FLOT, BLUE_NEW_HOLD_LINE,
BLUE_NEW_LIGHT_LINE, BLUE_NEW_LIMIT_ADV, BLUE_NEW_LINE_CONTACT,
BLUE_NEW_LINE_DEPART, BLUE_NEW_PHASE_LINE, BLUE_NEW_COA_LINE,
BLUE_NEW_RSTRCT_FIRE_LINE,
BLUE_NEW_AIR_FIELD, BLUE_NEW_BRIDGE, BLUE_NEW_BUILDING, BLUE_NEW_CITY,
BLUE_NEW_LAKE, BLUE_NEW_MAP_REF, BLUE_NEW_MOUNT_PEAK,
BLUE_NEW_ROAD_INTRCT, BLUE_NEW_TOWN, BLUE_NEW_VILLAGE, BLUE_NEW_CHECKPNT,
BLUE_NEW_COLLECT_PNT, BLUE_NEW_CONTACT_PNT, BLUE_NEW_COORD_PNT,
BLUE_NEW_CRIT_EVENT, BLUE_NEW_LINK_UP_PNT, BLUE_NEW_PASS_PNT,
BLUE_NEW_PNT_DEPART, BLUE_NEW_RELEASE_PNT, BLUE_NEW_START_PNT,
BLUE_NEW_STRONG_PNT, BLUE_NEW_TRAFF_CNTRL_PNT, BLUE_NEW_AIR_AXIS_ADV,
BLUE_NEW_AIR_CORR, BLUE_NEW_GRND_AXIS_ATK, BLUE_NEW_GRND_AXIS_SUP,
BLUE_NEW_DIR_ATTACK, BLUE_NEW_FEINT, BLUE_NEW_MAIN_SPLY_RTE,
BLUE_NEW_ROUTE,
OPFOR_NEW_AREA_OPER, OPFOR_NEW_ASSMBLY_AREA, OPFOR_NEW_ATTCK_POS,
OPFOR_NEW_BATTLE_POS, OPFOR_NEW_BDE_SPT_AREA, OPFOR_NEW_BN_SPT_AREA,
OPFOR_NEW_DIV_SPT_AREA, OPFOR_NEW_DROP_ZONE, OPFOR_NEW_FREE_FIRE_AREA,
OPFOR_NEW_LAND_ZONE, OPFOR_NEW_NO_FIRE_AREA, OPFOR_NEW_OBJECTIVE,
OPFOR_NEW_RSTRCT_FIRE_AREA, OPFOR_NEW_ZONE_ACTION, OPFOR_NEW_ASSLT_CROSS,
OPFOR_NEW_RAFT_SITE, OPFOR_NEW_GRP_TRGTS, OPFOR_NEW_BOUNDARY,
OPFOR_NEW_BRIDGE_LINE, OPFOR_NEW_COORD_FIR_LN, OPFOR_NEW_FEBA,
OPFOR_NEW_FIRE_SPT_COORD_LN, OPFOR_NEW_FLOT, OPFOR_NEW_HOLD_LINE,
OPFOR_NEW_LIGHT_LINE, OPFOR_NEW_LIMIT_ADV, OPFOR_NEW_LINE_CONTACT,
OPFOR_NEW_LINE_DEPART, OPFOR_NEW_PHASE_LINE, OPFOR_NEW_COA_LINE,
OPFOR_NEW_RSTRCT_FIRE_LINE,
OPFOR_NEW_AIR_FIELD, OPFOR_NEW_BRIDGE, OPFOR_NEW_BUILDING,
OPFOR_NEW_CITY, OPFOR_NEW_LAKE, OPFOR_NEW_MAP_REF, OPFOR_NEW_MOUNT_PEAK,
OPFOR_NEW_ROAD_INTRCT, OPFOR_NEW_TOWN, OPFOR_NEW_VILLAGE,
OPFOR_NEW_CHECKPNT, OPFOR_NEW_COLLECT_PNT, OPFOR_NEW_CONTACT_PNT,
OPFOR_NEW_COORD_PNT, OPFOR_NEW_CRIT_EVENT, OPFOR_NEW_LINK_UP_PNT,
OPFOR_NEW_PASS_PNT, OPFOR_NEW_PNT_DEPART, OPFOR_NEW_RELEASE_PNT,
OPFOR_NEW_START_PNT, OPFOR_NEW_STRONG_PNT, OPFOR_NEW_TRAFF_CNTRL_PNT,
OPFOR_NEW_AIR_AXIS_ADV, OPFOR_NEW_AIR_CORR, OPFOR_NEW_GRND_AXIS_ATK,
OPFOR_NEW_GRND_AXIS_SUP, OPFOR_NEW_DIR_ATTACK, OPFOR_NEW_FEINT,
OPFOR_NEW_MAIN_SPLY_RTE, OPFOR_NEW_ROUTE,
BLUE_NEW_ABATIS, BLUE_NEW_AT_DITCH, BLUE_NEW_BRIDGE_DEMO,
BLUE_NEW_CHEMICAL, BLUE_NEW_CRATER, BLUE_NEW_DAM_DEMO,
BLUE_NEW_FLOODING, BLUE_NEW_LOG_POSTS, BLUE_NEW_MINE_AP,
BLUE_NEW_MINE_AT, BLUE_NEW_MINE_AP_AT, BLUE_NEW_NUCLEAR,
BLUE_NEW_SCT_MINE_AP, BLUE_NEW_SCT_MINE_AT, BLUE_NEW_SCT_MINE_AP_AT,
BLUE_NEW_TUNNEL_DEMO, BLUE_NEW_WIRE,
OPFOR_NEW_ABATIS, OPFOR_NEW_AT_DITCH, OPFOR_NEW_BRIDGE_DEMO,
OPFOR_NEW_CHEMICAL, OPFOR_NEW_CRATER, OPFOR_NEW_DAM_DEMO,
OPFOR_NEW_FLOODING, OPFOR_NEW_LOG_POSTS, OPFOR_NEW_MINE_AP,
OPFOR_NEW_MINE_AT, OPFOR_NEW_MINE_AP_AT, OPFOR_NEW_NUCLEAR,
OPFOR_NEW_SCT_MINE_AP, OPFOR_NEW_SCT_MINE_AT, OPFOR_NEW_SCT_MINE_AP_AT,
OPFOR_NEW_TUNNEL_DEMO, OPFOR_NEW_WIRE,
```

```
      NEW_OPPLAN, NEW_WORKING_OPPLAN, ELEVATION_QUERY, NO_ACTION);

   type SYS_UNIT_OPTION is
      (MOVE_UNIT, UNIT_STATUS, DEACTIVATE_UNIT);

   type SYS_CM_OPTION is
      (MOVE_CNTRL_MSR, CNTRL_MSR_STATUS, MOVE_POINT, INSERT_POINT,
       INSERT_POINT_AFTER, DELETE_CNTRL_MSR, DELETE_POINT);

   type SYS_OBS_OPTION is
      (MOVE_OBSTACLE, OBSTACLE_STATUS, DELETE_OBSTACLE);

   -- Map and map overlay menu limits
   subtype SYS_MAP_CELL       is  SYS_WALKING_CELL   range  0..200;
   subtype SYS_UNIT_CELL      is  SYS_WALKING_CELL   range  0..4;
   subtype SYS_CM_CELL        is  SYS_WALKING_CELL   range  0..7;
   subtype SYS_OBS_CELL       is  SYS_WALKING_CELL   range  0..4;
   subtype SYS_MAP_MENU       is  SYS_WALKING_MENU   range  0..50;
   subtype SYS_UNIT_MENU      is  SYS_WALKING_MENU   range  0..1;
   subtype SYS_CM_MENU        is  SYS_WALKING_MENU   range  0..1;
   subtype SYS_OBS_MENU       is  SYS_WALKING_MENU   range  0..1;

   -- Map and map overlay option array
   type SYS_MAP_CONTROL_ARRAY is  array (SYS_WALKING_CELL range <>) of
                                  SYS_MAP_CONTROL;
   type SYS_MAP_CONTROL_PTR  is   access SYS_MAP_CONTROL_ARRAY;


   -- Unit overlay menu option array
   type SYS_UNIT_OPTION_ARRAY is  array (SYS_WALKING_CELL range <>) of
                                  SYS_UNIT_OPTION;
   type SYS_UNIT_OPTION_PTR  is   access SYS_UNIT_OPTION_ARRAY;

   -- Control Measure overlay menu option array
   type SYS_CM_OPTION_ARRAY    is  array (SYS_WALKING_CELL range <>) of
                                   SYS_CM_OPTION;
   type SYS_CM_OPTION_PTR  is   access SYS_CM_OPTION_ARRAY;

   -- Obstacle overlay menu option array
   type SYS_OBS_OPTION_ARRAY   is  array (SYS_WALKING_CELL range <>) of
                                  SYS_OBS_OPTION;
   type SYS_OBS_OPTION_PTR  is   access SYS_OBS_OPTION_ARRAY;

-- EDDIC Textual message limitations

   -- Number of message routing options
   subtype SYS_ROUTE_OPTION is SYS_WALKING_CELL range 0..10;

   -- Number of Operational Plans and limits on the OPPLAN IDs
   type SYS_OPPLAN is range 0..50;

-- EDDIC Window system limitations

   -- Window Name
   subtype SYS_WINDOW_NAME is SYS_TEXT (1..30);  -- This allows name to have max
                                                 -- of 29 chars + terminating
```

```
                                                    -- NULL

-- Types for button_menus

type SYS_MENU_BUTTON_VALUES is range -1..32760;
for SYS_MENU_BUTTON_VALUES'SIZE use 2*SYS_BITS_IN_BYTE;
subtype SYS_MENU_BUTTON_INDEX is SYS_MENU_BUTTON_VALUES
                                    range 0..SYS_MENU_BUTTON_VALUES'LAST;

   -- Value for indicating no default pushbutton
SYS_NO_DEFAULT_PUSHBUTTON : constant SYS_MENU_BUTTON_VALUES := -1;

type SYS_MENU_BUTTON_LABEL is array (SYS_MENU_BUTTON_INDEX range <>) of
                        SYS_MENU_TEXT_STRING;
type SYS_MENU_BUTTON_LABEL_PTR is access SYS_MENU_BUTTON_LABEL;
type SYS_MENU_BUTTON_STATUS is array (SYS_MENU_BUTTON_INDEX range <>) of
             Boolean;
type SYS_MENU_BUTTON_STATUS_PTR is access SYS_MENU_BUTTON_STATUS;

-- scrollbar's orientation
type SYS_SB_DIRECTION  is   range -7..-6;
for SYS_SB_DIRECTION'SIZE use SYS_BITS_IN_BYTE;
SYS_SB_DIR_HORZ        :     SYS_SB_DIRECTION    :=     -7;
SYS_SB_DIR_VERT        :     SYS_SB_DIRECTION    :=     -6;

-- text alignment
type SYS_TEXT_ALIGNMENT  is   range   1..4;
for SYS_TEXT_ALIGNMENT'SIZE use 4*SYS_BITS_IN_BYTE;
SYS_TEXT_ALIGN_CENT    :     SYS_TEXT_ALIGNMENT      :=      1;
SYS_TEXT_ALIGN_LEFT    :     SYS_TEXT_ALIGNMENT      :=      2;
SYS_TEXT_ALIGN_RIGHT   :     SYS_TEXT_ALIGNMENT      :=      3;
SYS_TEXT_ALIGN_NONE    :     SYS_TEXT_ALIGNMENT      :=      4;

-- Window Types
type SYS_WINDOW_TYPE     is   range   0..2;
for SYS_WINDOW_TYPE'SIZE use 2*SYS_BITS_IN_BYTE;
SYS_WINDOW             :     SYS_WINDOW_TYPE         :=      0;
SYS_DISPLAY_PANEL      :     SYS_WINDOW_TYPE         :=      1;
SYS_DEFINED_BUTTON     :     SYS_WINDOW_TYPE         :=      2;
subtype SYS_DESTINATION_TYPE is SYS_WINDOW_TYPE range 0..1;
SYS_WINDOW_DEST        :     SYS_DESTINATION_TYPE    := 0;
SYS_PANEL_DEST         :     SYS_DESTINATION_TYPE    := 1;

-- Label position for field editors
type SYS_LABEL_POSITION  is   range   0..1;
for SYS_LABEL_POSITION'SIZE use SYS_BITS_IN_BYTE;
SYS_LABEL_LEFT         :     SYS_LABEL_POSITION      :=      0;
SYS_LABEL_RIGHT        :     SYS_LABEL_POSITION      :=      1;

-- Number of icon stacks on EDDIC screen and size of the stack
type SYS_ICON       is range 0..5;
for SYS_ICON'SIZE use 2*SYS_BITS_IN_BYTE;
type SYS_ICON_STACK is range 0..6;
for SYS_ICON_STACK'SIZE use 2*SYS_BITS_IN_BYTE;
SYS_ICON_REFERENCE     :     SYS_ICON   :=  0;
SYS_ICON_VIEW_C2       :     SYS_ICON   :=  1;
```

```
SYS_ICON_MESSAGE        :     SYS_ICON   :=  2;
SYS_ICON_BUILD_C2       :     SYS_ICON   :=  3;
SYS_ICON_AIDS           :     SYS_ICON   :=  4;
SYS_ICON_CONTROL        :     SYS_ICON   :=  5;
-- Icon Name
subtype SYS_ICON_NAME is SYS_TEXT(1..7); -- This includes space for 6
                                         -- characters plus terminating Null

-- Types of tools in the tool window
type SYS_TOOLS is (NO_TOOL, SCRATCH_PAD, TSTM, CALCULATOR,
                   TASK_ORGANIZATION, FORM);

-- Range of ID's for elements of a window
subtype SYS_WINDOW_ELE_ID is INTEGER;
SYS_NO_WINDOW           :   constant SYS_WINDOW_ELE_ID := -1;
SYS_ROOT_WINDOW         :   constant SYS_WINDOW_ELE_ID := 0;

-- Global value for indicating no subpanel
SYS_NULL_SUBPANEL : constant SYS_WINDOW_ELE_ID := 0;

-- Input types returned from window utilities
type SYS_WINDOW_INPUT  is  range 0..20;
for SYS_WINDOW_INPUT'SIZE use 2*SYS_BITS_IN_BYTE;
SYS_INPUT_NONE          :    constant SYS_WINDOW_INPUT  :=  0;
SYS_INPUT_TERMINATE     :    constant SYS_WINDOW_INPUT  :=  1;
SYS_INPUT_MENU_SELECT   :    constant SYS_WINDOW_INPUT  :=  2;
SYS_INPUT_CHECKBOX      :    constant SYS_WINDOW_INPUT  :=  3;
SYS_INPUT_SCROLLBAR     :    constant SYS_WINDOW_INPUT  :=  4;
SYS_INPUT_MESSAGE       :    constant SYS_WINDOW_INPUT  :=  5;
SYS_INPUT_BUTTON        :    constant SYS_WINDOW_INPUT  :=  6;
SYS_INPUT_MOUSE_PRESS   :    constant SYS_WINDOW_INPUT  :=  7;
SYS_INPUT_MOUSE_RELEASE :    constant SYS_WINDOW_INPUT  :=  8;
SYS_INPUT_TRAVERSAL     :    constant SYS_WINDOW_INPUT  :=  9;
SYS_INPUT_EXPOSURE      :    constant SYS_WINDOW_INPUT  := 10;
SYS_INPUT_OPEN          :    constant SYS_WINDOW_INPUT  := 11;
SYS_INPUT_RESIZE        :    constant SYS_WINDOW_INPUT  := 12;
SYS_INPUT_CLOSE         :    constant SYS_WINDOW_INPUT  := 13;
SYS_INPUT_SAVE          :    constant SYS_WINDOW_INPUT  := 14;
SYS_INPUT_RESET         :    constant SYS_WINDOW_INPUT  := 15;
SYS_INPUT_PUSH_BUTTON   :    constant SYS_WINDOW_INPUT  := 16;
SYS_INPUT_RADIO_BUTTON  :    constant SYS_WINDOW_INPUT  := 17;

-- Input types returned from the digital map control system
SYS_INPUT_MAP           :    constant SYS_WINDOW_INPUT  := 20;

-- Input type codes returned from window utilities
type    SYS_ACTION_COUNT  is  range 0..5;
subtype SYS_WINDOW_VALUE  is  INTEGER;
subtype SYS_BUTTON_COUNT  is  SYS_WINDOW_VALUE range 0..2;
SYS_VALUE_RIGHT_BUTTON  :    constant SYS_WINDOW_VALUE  :=  0;
SYS_VALUE_MIDDLE_BUTTON :    constant SYS_WINDOW_VALUE  :=  1;
SYS_VALUE_LEFT_BUTTON   :    constant SYS_WINDOW_VALUE  :=  2;

-- Traversal type codes returned
SYS_TRAVERSE_NEXT : constant SYS_WINDOW_VALUE := 1;
SYS_TRAVERSE_PREV : constant SYS_WINDOW_VALUE := 2;
```

```
SYS_TRAVERSE_UP   : constant SYS_WINDOW_VALUE := 3;
SYS_TRAVERSE_DOWN : constant SYS_WINDOW_VALUE := 4;

-- Button Pressed Down or Up actions
SYS_RIGHT_BUTTON_DOWN   :    constant SYS_ACTION_COUNT  :=  0;
SYS_MIDDLE_BUTTON_DOWN  :    constant SYS_ACTION_COUNT  :=  1;
SYS_LEFT_BUTTON_DOWN    :    constant SYS_ACTION_COUNT  :=  2;
SYS_RIGHT_BUTTON_UP     :    constant SYS_ACTION_COUNT  :=  3;
SYS_MIDDLE_BUTTON_UP    :    constant SYS_ACTION_COUNT  :=  4;
SYS_LEFT_BUTTON_UP      :    constant SYS_ACTION_COUNT  :=  5;

-- Input types returned from the digital map input utiliity
SYS_MAP_CHANGE          :    constant SYS_WINDOW_VALUE  :=  0;
SYS_BLUEFOR_UNIT_CHANGE :    constant SYS_WINDOW_VALUE  :=  1;
SYS_BLUEFOR_UNIT_DEACT  :    constant SYS_WINDOW_VALUE  :=  2;
SYS_OPFOR_UNIT_CHANGE   :    constant SYS_WINDOW_VALUE  :=  3;
SYS_OPFOR_UNIT_DEACT    :    constant SYS_WINDOW_VALUE  :=  4;
SYS_CNTRL_MSR_CHANGE    :    constant SYS_WINDOW_VALUE  :=  5;
SYS_CNTRL_MSR_DELETE    :    constant SYS_WINDOW_VALUE  :=  6;
SYS_OBSTACLE_CHANGE     :    constant SYS_WINDOW_VALUE  :=  7;
SYS_OBSTACLE_DELETE     :    constant SYS_WINDOW_VALUE  :=  8;
SYS_OPPLAN_CHANGE       :    constant SYS_WINDOW_VALUE  :=  9;
SYS_WORK_OPPLAN_CHANGE  :    constant SYS_WINDOW_VALUE  :=  10;

-- Input data returned from the window utilities
type SYS_WINDOW_DATA_COUNT is range 1..4;
type SYS_WINDOW_DATA is array (SYS_WINDOW_DATA_COUNT) of SYS_WINDOW_VALUE;
type SYS_FIELD_TYPE is range 1..2;
for SYS_FIELD_TYPE'SIZE use SYS_BITS_IN_BYTE;
SYS_STRING_FIELD        :    constant SYS_FIELD_TYPE := 1;
SYS_NUMBER_FIELD        :    constant SYS_FIELD_TYPE := 2;

type   SYS_PIXEL        is     range -32767..32767;
for SYS_PIXEL'SIZE use 2*SYS_BITS_IN_BYTE;

-- Number of columns and rows in a window
subtype SYS_WINDOW_PIXEL is SYS_PIXEL range -1024..2048;
subtype SYS_WINDOW_COLUMN is SYS_WINDOW_PIXEL;
subtype SYS_WINDOW_ROW    is SYS_WINDOW_PIXEL;
SYS_NULL_COLUMN :   constant SYS_WINDOW_PIXEL := 0;
SYS_NULL_ROW:       constant SYS_WINDOW_PIXEL := 0;
type SYS_WINDOW_LOCATION is record
   X  :  SYS_WINDOW_COLUMN := 0;
   Y  :  SYS_WINDOW_ROW    := 0;
end record;
type SYS_RECTANGLE is record
   X  :  SYS_WINDOW_COLUMN;
   Y  :  SYS_WINDOW_ROW;
   WIDTH :   SYS_WINDOW_COLUMN;
   HEIGHT:   SYS_WINDOW_ROW;
end record;

-- Width and height of a virtual image in pixels.
subtype SYS_IMAGE_PIXEL is SYS_PIXEL;
subtype SYS_IMAGE_COLUMN  is SYS_IMAGE_PIXEL;
subtype SYS_IMAGE_ROW     is SYS_IMAGE_PIXEL;
```

```
   type SYS_IMAGE_LOCATION is record
      X  :  SYS_IMAGE_COLUMN := 0;
      Y  :  SYS_IMAGE_ROW    := 0;
   end record;
   type SYS_GRID_LABEL  is  range 0..99;

   -- Digital map scales
   type SYS_MAP_SCALES is
       (S1_40000, S1_80000, S1_160000, S1_400000, S1_800000);

   -- Digital map background types
   type SYS_MAP_BACKGROUND is
       (CROSS_COUNTRY_MOVE, ELEVATION_BANDED, SHADED_RELIEF, THREE_D,
        VEGETATION, NO_BACKGROUND);

   -- Coordinate system sizes
   subtype SYS_UTM_LETTER   is   INTEGER  range 1..2;
   type SYS_COORDINATE      is     range -99999999..99999999;
   subtype SYS_UTM_COORD    is   SYS_COORDINATE  range 0..99999;

   -- Color lookup table size
   type SYS_COLOR_TABLE  is     range 0..255;
   for SYS_COLOR_TABLE'SIZE use 2*SYS_BITS_IN_BYTE;
   type SYS_COLOR        is     range 0..255;
   for SYS_COLOR'SIZE use 4*SYS_BITS_IN_BYTE;
   type SYS_COLOR_PLANE  is     range 1..8;
   for SYS_COLOR_PLANE'SIZE use SYS_BITS_IN_BYTE;
   type SYS_COLOR_MASK   is     range INTEGER'FIRST..INTEGER'LAST;
   for SYS_COLOR_MASK'SIZE use 4*SYS_BITS_IN_BYTE;
   type SYS_BITS_DEEP    is     range 8..32;
   for SYS_BITS_DEEP'SIZE use SYS_BITS_IN_BYTE;
   type SYS_MAX_PLANES   is     range 0..8;
   for SYS_MAX_PLANES'SIZE use SYS_BITS_IN_BYTE;

   -- Hexadecimal bit images
   type SYS_HEXADECIMAL is ('0','1','2','3','4','5','6','7','8','9','A','B','C',
         'D','E','F');
   for SYS_HEXADECIMAL'SIZE use SYS_BITS_IN_NIBBLE;
   for SYS_HEXADECIMAL use ('0'=>0, '1'=>1, '2'=>2, '3'=>3, '4'=>4, '5'=>5,
        '6'=>6, '7'=>7, '8'=>8, '9'=>9, 'A'=>10, 'B'=>11, 'C'=>12, 'D'=>13,
        'E'=>14, 'F'=>15);

   -- Color lookup update flags
   type SYS_LUT_STATUS   is     range 0..2;
   SYS_LUT_NO_CHANGE        :      SYS_LUT_STATUS   :=  0;
   SYS_LUT_HILITE           :      SYS_LUT_STATUS   :=  1;
   SYS_LUT_UNHILITE         :      SYS_LUT_STATUS   :=  2;

   -- Color Image Action flags
   type SYS_COLOR_ACTION   is   range 0..16;
   for SYS_COLOR_ACTION'SIZE use SYS_BITS_IN_BYTE;
   SYS_COPY_IMAGE           :      SYS_COLOR_ACTION := 3;
   SYS_OR_IMAGE             :      SYS_COLOR_ACTION := 7;

-- EDDIC communications limitations
```

```
-- List of stations in EDDIC
type SYS_PARTICIPANTS is (G2, G3, G4, EXPERIMENTER);

-- List of processes in the EDDIC network
type SYS_EDDIC_PROCESSES is
    (G2_REFERENCE_1, G2_REFERENCE_2, G2_REFERENCE_3, G2_REFERENCE_4,
     G2_REFERENCE_5, G2_REFERENCE_6, G2_REFERENCE_7,
     G2_VIEW_C2_1  , G2_VIEW_C2_2  , G2_VIEW_C2_3  , G2_VIEW_C2_4  ,
     G2_VIEW_C2_5  , G2_VIEW_C2_6  , G2_VIEW_C2_7  ,
     G2_MESSAGE_1  , G2_MESSAGE_2  , G2_MESSAGE_3  , G2_MESSAGE_4  ,
     G2_MESSAGE_5  , G2_MESSAGE_6  , G2_MESSAGE_7  ,
     G2_BUILD_C2_1 , G2_BUILD_C2_2 , G2_BUILD_C2_3 , G2_BUILD_C2_4 ,
     G2_BUILD_C2_5 , G2_BUILD_C2_6 , G2_BUILD_C2_7 ,
     G2_AIDS_1     , G2_AIDS_2     , G2_AIDS_3     , G2_AIDS_4     ,
     G2_AIDS_5     , G2_AIDS_6     , G2_AIDS_7     ,
     G2_CONTROL_1  , G2_CONTROL_2  , G2_CONTROL_3  , G2_CONTROL_4  ,
     G2_CONTROL_5  , G2_CONTROL_6  , G2_CONTROL_7  , G2_HELP       ,
     G3_REFERENCE_1, G3_REFERENCE_2, G3_REFERENCE_3, G3_REFERENCE_4,
     G3_REFERENCE_5, G3_REFERENCE_6, G3_REFERENCE_7,
     G3_VIEW_C2_1  , G3_VIEW_C2_2  , G3_VIEW_C2_3  , G3_VIEW_C2_4  ,
     G3_VIEW_C2_5  , G3_VIEW_C2_6  , G3_VIEW_C2_7  ,
     G3_MESSAGE_1  , G3_MESSAGE_2  , G3_MESSAGE_3  , G3_MESSAGE_4  ,
     G3_MESSAGE_5  , G3_MESSAGE_6  , G3_MESSAGE_7  ,
     G3_BUILD_C2_1 , G3_BUILD_C2_2 , G3_BUILD_C2_3 , G3_BUILD_C2_4 ,
     G3_BUILD_C2_5 , G3_BUILD_C2_6 , G3_BUILD_C2_7 ,
     G3_AIDS_1     , G3_AIDS_2     , G3_AIDS_3     , G3_AIDS_4     ,
     G3_AIDS_5     , G3_AIDS_6     , G3_AIDS_7     ,
     G3_CONTROL_1  , G3_CONTROL_2  , G3_CONTROL_3  , G3_CONTROL_4  ,
     G3_CONTROL_5  , G3_CONTROL_6  , G3_CONTROL_7  , G3_HELP       ,
     G4_REFERENCE_1, G4_REFERENCE_2, G4_REFERENCE_3, G4_REFERENCE_4,
     G4_REFERENCE_5, G4_REFERENCE_6, G4_REFERENCE_7,
     G4_VIEW_C2_1  , G4_VIEW_C2_2  , G4_VIEW_C2_3  , G4_VIEW_C2_4  ,
     G4_VIEW_C2_5  , G4_VIEW_C2_6  , G4_VIEW_C2_7  ,
     G4_MESSAGE_1  , G4_MESSAGE_2  , G4_MESSAGE_3  , G4_MESSAGE_4  ,
     G4_MESSAGE_5  , G4_MESSAGE_6  , G4_MESSAGE_7  ,
     G4_BUILD_C2_1 , G4_BUILD_C2_2 , G4_BUILD_C2_3 , G4_BUILD_C2_4 ,
     G4_BUILD_C2_5 , G4_BUILD_C2_6 , G4_BUILD_C2_7 ,
     G4_AIDS_1     , G4_AIDS_2     , G4_AIDS_3     , G4_AIDS_4     ,
     G4_AIDS_5     , G4_AIDS_6     , G4_AIDS_7     ,
     G4_CONTROL_1  , G4_CONTROL_2  , G4_CONTROL_3  , G4_CONTROL_4  ,
     G4_CONTROL_5  , G4_CONTROL_6  , G4_CONTROL_7  , G4_HELP       ,
     EX_REFERENCE_1, EX_REFERENCE_2, EX_REFERENCE_3, EX_REFERENCE_4,
     EX_REFERENCE_5, EX_REFERENCE_6, EX_REFERENCE_7,
     EX_VIEW_C2_1  , EX_VIEW_C2_2  , EX_VIEW_C2_3  , EX_VIEW_C2_4  ,
     EX_VIEW_C2_5  , EX_VIEW_C2_6  , EX_VIEW_C2_7  ,
     EX_MESSAGE_1  , EX_MESSAGE_2  , EX_MESSAGE_3  , EX_MESSAGE_4  ,
     EX_MESSAGE_5  , EX_MESSAGE_6  , EX_MESSAGE_7  ,
     EX_BUILD_C2_1 , EX_BUILD_C2_2 , EX_BUILD_C2_3 , EX_BUILD_C2_4 ,
     EX_BUILD_C2_5 , EX_BUILD_C2_6 , EX_BUILD_C2_7 ,
     EX_AIDS_1     , EX_AIDS_2     , EX_AIDS_3     , EX_AIDS_4     ,
     EX_AIDS_5     , EX_AIDS_6     , EX_AIDS_7     ,
     EX_CONTROL_1  , EX_CONTROL_2  , EX_CONTROL_3  , EX_CONTROL_4  ,
     EX_CONTROL_5  , EX_CONTROL_6  , EX_CONTROL_7  , EX_HELP       ,

     SITUATION_DB_MANAGER, C2_DB_MANAGER     , REFERENCE_DB_MANAGER,
     HELP_MANAGER        , CONTROL_MANAGER   , G2_STATION_MANAGER,
```

```
      G3_STATION_MANAGER   , G4_STATION_MANAGER, EX_STATION_MANAGER,

      C2_PRODUCT_ROUTER, CONTROL_ROUTER, REFERENCE_ROUTER, SITUATION_ROUTER);

for SYS_EDDIC_PROCESSES use
    (G2_REFERENCE_1=>1   , G2_REFERENCE_2=>2   , G2_REFERENCE_3=>3   ,
     G2_REFERENCE_4=>4   , G2_REFERENCE_5=>5   , G2_REFERENCE_6=>6   ,
     G2_REFERENCE_7=>7   ,
     G2_VIEW_C2_1=>11    , G2_VIEW_C2_2=>12    , G2_VIEW_C2_3=>13    ,
     G2_VIEW_C2_4=>14    , G2_VIEW_C2_5=>15    , G2_VIEW_C2_6=>16    ,
     G2_VIEW_C2_7=>17    ,
     G2_MESSAGE_1=>21    , G2_MESSAGE_2=>22    , G2_MESSAGE_3=>23    ,
     G2_MESSAGE_4=>24    , G2_MESSAGE_5=>25    , G2_MESSAGE_6=>26    ,
     G2_MESSAGE_7=>28    ,
     G2_BUILD_C2_1=>31   , G2_BUILD_C2_2=>32   , G2_BUILD_C2_3=>33   ,
     G2_BUILD_C2_4=>34   , G2_BUILD_C2_5=>35   , G2_BUILD_C2_6=>36   ,
     G2_BUILD_C2_7=>37   ,
     G2_AIDS_1=>41       , G2_AIDS_2=>42       , G2_AIDS_3=>43       ,
     G2_AIDS_4=>44       , G2_AIDS_5=>45       , G2_AIDS_6=>46       ,
     G2_AIDS_7=>47       ,
     G2_CONTROL_1=>51    , G2_CONTROL_2=>52    , G2_CONTROL_3=>53    ,
     G2_CONTROL_4=>54    , G2_CONTROL_5=>55    , G2_CONTROL_6=>56    ,
     G2_CONTROL_7=>57    , G2_HELP=>58         ,
     G3_REFERENCE_1=>61  , G3_REFERENCE_2=>62  , G3_REFERENCE_3=>63  ,
     G3_REFERENCE_4=>64  , G3_REFERENCE_5=>65  , G3_REFERENCE_6=>66  ,
     G3_REFERENCE_7=>67  ,
     G3_VIEW_C2_1=>71    , G3_VIEW_C2_2=>72    , G3_VIEW_C2_3=>73    ,
     G3_VIEW_C2_4=>74    , G3_VIEW_C2_5=>75    , G3_VIEW_C2_6=>76    ,
     G3_VIEW_C2_7=>77    ,
     G3_MESSAGE_1=>81    , G3_MESSAGE_2=>82    , G3_MESSAGE_3=>83    ,
     G3_MESSAGE_4=>84    , G3_MESSAGE_5=>85    , G3_MESSAGE_6=>86    ,
     G3_MESSAGE_7=>87    ,
     G3_BUILD_C2_1=>91   , G3_BUILD_C2_2=>92   , G3_BUILD_C2_3=>93   ,
     G3_BUILD_C2_4=>94   , G3_BUILD_C2_5=>95   , G3_BUILD_C2_6=>96   ,
     G3_BUILD_C2_7=>97   ,
     G3_AIDS_1=>101      , G3_AIDS_2=>102      , G3_AIDS_3=>103      ,
     G3_AIDS_4=>104      , G3_AIDS_5=>105      , G3_AIDS_6=>106      ,
     G3_AIDS_7=>107      ,
     G3_CONTROL_1=>111   , G3_CONTROL_2=>112   , G3_CONTROL_3=>113   ,
     G3_CONTROL_4=>114   , G3_CONTROL_5=>115   , G3_CONTROL_6=>116   ,
     G3_CONTROL_7=>117   , G3_HELP=>118        ,
     G4_REFERENCE_1=>121, G4_REFERENCE_2=>122, G4_REFERENCE_3=>123,
     G4_REFERENCE_4=>124, G4_REFERENCE_5=>125, G4_REFERENCE_6=>126,
     G4_REFERENCE_7=>127,
     G4_VIEW_C2_1=>131   , G4_VIEW_C2_2=>132   , G4_VIEW_C2_3=>133   ,
     G4_VIEW_C2_4=>134   , G4_VIEW_C2_5=>135   , G4_VIEW_C2_6=>136   ,
     G4_VIEW_C2_7=>138   ,
     G4_MESSAGE_1=>141   , G4_MESSAGE_2=>142   , G4_MESSAGE_3=>143   ,
     G4_MESSAGE_4=>144   , G4_MESSAGE_5=>145   , G4_MESSAGE_6=>146   ,
     G4_MESSAGE_7=>147   ,
     G4_BUILD_C2_1=>151  , G4_BUILD_C2_2=>152  , G4_BUILD_C2_3=>153  ,
     G4_BUILD_C2_4=>154  , G4_BUILD_C2_5=>155  , G4_BUILD_C2_6=>156  ,
     G4_BUILD_C2_7=>157  ,
     G4_AIDS_1=>161      , G4_AIDS_2=>162      , G4_AIDS_3=>163      ,
     G4_AIDS_4=>164      , G4_AIDS_5=>165      , G4_AIDS_6=>166      ,
     G4_AIDS_7=>167      ,
```

```
        G4_CONTROL_1=>171   , G4_CONTROL_2=>172   , G4_CONTROL_3=>173   ,
        G4_CONTROL_4=>174   , G4_CONTROL_5=>175   , G4_CONTROL_6=>176   ,
        G4_CONTROL_7=>177   , G4_HELP=>178        ,
        EX_REFERENCE_1=>181, EX_REFERENCE_2=>182, EX_REFERENCE_3=>183,
        EX_REFERENCE_4=>184, EX_REFERENCE_5=>185, EX_REFERENCE_6=>186,
        EX_REFERENCE_7=>187,
        EX_VIEW_C2_1=>191   , EX_VIEW_C2_2=>192   , EX_VIEW_C2_3=>193   ,
        EX_VIEW_C2_4=>194   , EX_VIEW_C2_5=>195   , EX_VIEW_C2_6=>196   ,
        EX_VIEW_C2_7=>198   ,
        EX_MESSAGE_1=>201   , EX_MESSAGE_2=>202   , EX_MESSAGE_3=>203   ,
        EX_MESSAGE_4=>204   , EX_MESSAGE_5=>205   , EX_MESSAGE_6=>206   ,
        EX_MESSAGE_7=>207   ,
        EX_BUILD_C2_1=>211 , EX_BUILD_C2_2=>212 , EX_BUILD_C2_3=>213 ,
        EX_BUILD_C2_4=>214 , EX_BUILD_C2_5=>215 , EX_BUILD_C2_6=>216 ,
        EX_BUILD_C2_7=>217 ,
        EX_AIDS_1=>221      , EX_AIDS_2=>222      , EX_AIDS_3=>223      ,
        EX_AIDS_4=>224      , EX_AIDS_5=>225      , EX_AIDS_6=>226      ,
        EX_AIDS_7=>227      ,
        EX_CONTROL_1=>231   , EX_CONTROL_2=>232   , EX_CONTROL_3=>233   ,
        EX_CONTROL_4=>234   , EX_CONTROL_5=>235   , EX_CONTROL_6=>236   ,
        EX_CONTROL_7=>237   , EX_HELP=>238        ,

        SITUATION_DB_MANAGER=>241, C2_DB_MANAGER=>242,
        REFERENCE_DB_MANAGER=>243, HELP_MANAGER =>244, CONTROL_MANAGER=>245,
        G2_STATION_MANAGER=>246, G3_STATION_MANAGER=>247,
        G4_STATION_MANAGER=>248, EX_STATION_MANAGER=>249,

        C2_PRODUCT_ROUTER=>250, CONTROL_ROUTER=>251, REFERENCE_ROUTER=>252,
        SITUATION_ROUTER=>253);

    for SYS_EDDIC_PROCESSES'SIZE use SYS_BITS_IN_BYTE;

    --  Position of a process within the process list
    type SYS_PROCESS_POSITION is range 0..255;

    -- Client (socket) counts, IDs and indexes
    type SYS_CLIENT         is range 0..31;
    for SYS_CLIENT'SIZE use 4*SYS_BITS_IN_BYTE;

-- EDDIC system limitations

    -- Maximum length of a environment string passed from the operating
    -- system to the software.
    subtype SYS_ENV_STRING     is INTEGER range 1..80;

    -- Error code ranges
    type SYS_ERROR          is range 0..127;
    for SYS_ERROR'SIZE use SYS_BITS_IN_BYTE;
    SYS_NO_ERROR               :   SYS_ERROR   := 0;
    SYS_EXCEPTION              :   exception;
    SYS_LUT_EXCEPTION          :   exception;
    SYS_SDB_IO_EXCEPTION       :   exception;
    SYS_SDB_SEND_EXCEPTION     :   exception;
    SYS_SDB_UPDT_EXCEPTION     :   exception;
    SYS_TOT_EXCEPTION          :   exception;
    SYS_TSB_EXCEPTION          :   exception;
```

```
      SYS_UCC_EXCEPTION          :   exception;
      SYS_UCE_EXCEPTION          :   exception;
      SYS_UCM_EXCEPTION          :   exception;
      SYS_UED_EXCEPTION          :   exception;
      SYS_UFM_EXCEPTION          :   exception;
      SYS_UIN_EXCEPTION          :   exception;
      SYS_UME_EXCEPTION          :   exception;
      SYS_UMP_EXCEPTION          :   exception;
      SYS_UNT_EXCEPTION          :   exception;
      SYS_UOB_EXCEPTION          :   exception;
      SYS_UOE_EXCEPTION          :   exception;
      SYS_UTM_EXCEPTION          :   exception;
      SYS_UUE_EXCEPTION          :   exception;
      SYS_UUX_EXCEPTION          :   exception;
      SYS_UWN_EXCEPTION          :   exception;
      SYS_UIW_EXCEPTION          :   exception;

      -- Number of seconds a process can be suspended
      type SYS_DELAY          is range 0..3600;
      for SYS_DELAY'SIZE use 4*SYS_BITS_IN_BYTE;

      -- Dates and times
      type SYS_DAY            is range 1..31;
      type SYS_TIME           is range 0..2359;
      type SYS_YEAR           is range 0..9999;
      type SYS_MONTH             is (JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG,
                                     SEP, OCT, NOV, DEC);
      type SYS_HOUR           is range 0..9999;
      subtype SYS_MINUTE_TOTAL   is INTEGER;
      type SYS_PERCENT        is range 0..100;

      -- EDDIC Date and Time
      type SYS_DATE_TIME is
         record
            SYS_MINUTE        :       SYS_MINUTE_TOTAL;
         end record;

   end SYSTEM_PACKAGE;
```

```
--cpc package specification name: TSTM_DB
--
--cpc description: The TSTM_DB cpc describes the objects that are used
--                 for interacting with the TSTM Feedback Module.
--
--cpc design notes:
--
--cpc package author: Bruce Packard
--                    Science Applications International Corporation
--                    424 Delaware, Suite C3
--                    Leavenworth, KS 66048
--

with SYSTEM_PACKAGE;        use SYSTEM_PACKAGE;

package TSTM_DB is

   type TSTM_PARTICIPANTS  is  (G2, G3, G4);
   type TSTM_MATRIX_DATA   is  (MATRIX, OCOKA, COA);
   type TSTM_FBACK_ACTION  is  (NONE, INITIAL, COLUMN_COMPLETE);
   type TSTM_PHASE         is  (PRETEST, TRAINING, GROUP_RESULT, FINAL_SOLUTION,
                                POSTTEST, TERMINATION);

   -- TSTM Matrix limitation
   type TSTM_COA_VALUE     is   (' ', '0', '1', '2', '3');
   type TSTM_OCOKA_VALUE   is   (' ', '0', '1', '2', '3', '4', '5');
   subtype TSTM_NUM_ROW          is  SYS_WINDOW_ROW     range  1..5;
   subtype TSTM_NUM_OCOKA        is  SYS_WINDOW_COLUMN  range  1..5;
   subtype TSTM_NUM_COA          is  SYS_WINDOW_COLUMN  range  1..3;
   subtype TSTM_TITLE_LEN        is  INTEGER  range  1..40;
   subtype TSTM_ROW_HDR_LEN      is  INTEGER  range  1..20;
   subtype TSTM_COL_TITLE_LEN    is  INTEGER  range  1..30;
   subtype TSTM_ROW_TITLE_LEN    is  INTEGER  range  1..30;
   subtype TSTM_FBK_TITLE_LEN    is  INTEGER  range  1..25;

   -- Number of records in TSTM databases
   subtype TSTM_NUM_FBACK_DESC    is  SYS_DB_SIZE  range  0..200;
   subtype TSTM_NUM_TEXT          is  SYS_DB_SIZE  range  0..500;
   subtype TSTM_NUM_MATRIX_DESC   is  SYS_DB_SIZE  range  0..3;
   subtype TSTM_NUM_MATRIX_VAL    is  SYS_DB_SIZE  range  0..1000;
   subtype TSTM_NUM_OCOKA_VAL     is  SYS_DB_SIZE  range  0..50;

   -- Matrix Definitions
   type TSTM_COA_VALUES    is  array (TSTM_NUM_COA)    of TSTM_COA_VALUE;
   type TSTM_ROW_VALUES    is  array (TSTM_NUM_OCOKA)  of TSTM_COA_VALUES;
   type TSTM_COL_VALUES    is  array (TSTM_NUM_ROW)    of TSTM_COA_VALUES;
   type TSTM_MATRIX_VALUES is  array (TSTM_NUM_ROW)    of TSTM_ROW_VALUES;
   type TSTM_OCOKA_VALUES  is  array (TSTM_NUM_OCOKA)  of TSTM_OCOKA_VALUE;

   -- Feedback Description record
   type TSTM_FBACK_DESC_TYPE is
      record
         TSTM_PARTICIPANT       :  TSTM_PARTICIPANTS;
         TSTM_TYPE              :  TSTM_MATRIX_DATA;
         TSTM_ROW               :  TSTM_NUM_ROW;
```

```
      TSTM_COL                : TSTM_NUM_OCOKA;
      TSTM_MTRX_EXPERT        : TSTM_COA_VALUES;
      TSTM_OCOKA_EXPERT       : TSTM_OCOKA_VALUES;
      TSTM_START              : TSTM_NUM_TEXT;
      TSTM_END                : TSTM_NUM_TEXT;
end record;

TSTM_FBACK_DESC_REC           : TSTM_FBACK_DESC_TYPE;

-- Feedback Text record
TSTM_TEXT_SIZE                : SYS_PRODUCT_LENGTH   := 252;

type TSTM_TEXT_TYPE  is
   record
      TSTM_NUMBER_CHAR        : SYS_PRODUCT_LENGTH  range 0..
                                TSTM_TEXT_SIZE;
      TSTM_TEXT               : SYS_TEXT (1..TSTM_TEXT_SIZE);
   end record;
type TSTM_TEXT_POINT          is access TSTM_TEXT_TYPE;
TSTM_TEXT_REC                 : TSTM_TEXT_POINT :=  new TSTM_TEXT_TYPE;

-- Matrix Description Record
type TSTM_COL_REC is
   record
      TSTM_TITLE              :   SYS_TEXT (TSTM_COL_TITLE_LEN);
      TSTM_FBACK_COL          :   TSTM_NUM_OCOKA;
      TSTM_FBACK              :   TSTM_FBACK_ACTION;
      TSTM_FBACK_TITLE        :   SYS_TEXT (TSTM_FBK_TITLE_LEN);
end record;

type TSTM_COL_ARRAY   is  array (TSTM_NUM_OCOKA) of TSTM_COL_REC;
type TSTM_COL_POINT   is  access TSTM_COL_ARRAY;

type TSTM_ROW_REC is
   record
      TSTM_TITLE              :   SYS_TEXT (TSTM_ROW_TITLE_LEN);
      TSTM_FBACK_ROW          :   TSTM_NUM_ROW;
      TSTM_FBACK              :   TSTM_FBACK_ACTION;
      TSTM_FBACK_TITLE        :   SYS_TEXT (TSTM_FBK_TITLE_LEN);
end record;

type TSTM_ROW_ARRAY   is  array (TSTM_NUM_ROW) of TSTM_ROW_REC;
type TSTM_ROW_POINT   is  access TSTM_ROW_ARRAY;

type TSTM_MATRIX_DESC_TYPE is
   record
      TSTM_TITLE              : SYS_TEXT (TSTM_TITLE_LEN);
      TSTM_ROW_HEADER         : SYS_TEXT (TSTM_ROW_HDR_LEN);
      TSTM_COA_COUNT          : TSTM_NUM_COA;
      TSTM_COL_COUNT          : TSTM_NUM_OCOKA;
      TSTM_COL                : TSTM_COL_ARRAY;
      TSTM_ROW_COUNT          : TSTM_NUM_ROW;
      TSTM_ROW                : TSTM_ROW_ARRAY;
      TSTM_OCOKA_FBACK        : TSTM_FBACK_ACTION;
      TSTM_COA_FBACK          : TSTM_FBACK_ACTION;
end record;
```

```
TSTM_MATRIX_DESC_REC          :  TSTM_MATRIX_DESC_TYPE;

-- Matrix Value record
type TSTM_MATRIX_VAL_TYPE is
   record
      TSTM_PARTICIPANT        :  TSTM_PARTICIPANTS;
      TSTM_TIME               :  SYS_DATE_TIME;
      TSTM_TYPE               :  TSTM_MATRIX_DATA;
      TSTM_ROW                :  TSTM_NUM_ROW;
      TSTM_COL                :  TSTM_NUM_OCOKA;
      TSTM_VALUE              :  TSTM_COA_VALUES;
end record;

TSTM_MATRIX_VAL_REC           :  TSTM_MATRIX_VAL_TYPE;

-- OCOKA Value record
type TSTM_OCOKA_VAL_TYPE is
   record
      TSTM_PARTICIPANT        :  TSTM_PARTICIPANTS;
      TSTM_TIME               :  SYS_DATE_TIME;
      TSTM_VALUE              :  TSTM_OCOKA_VALUES;
end record;

TSTM_OCOKA_VAL_REC            :  TSTM_OCOKA_VAL_TYPE;

-- TSTM MESSAGE RECORDS

-- TSTM Matrix definition record
type TSTM_INITIAL_MATRIX is
   record
      TSTM_LAYOUT             :  TSTM_MATRIX_DESC_TYPE;
      TSTM_VAL                :  TSTM_MATRIX_VALUES;
      TSTM_OCOKA_VAL          :  TSTM_OCOKA_VALUES;
      TSTM_COA_VAL            :  TSTM_COA_VALUES;
end record;

-- Column Feedback record
type TSTM_COLUMN_FEEDBACK is
   record
      TSTM_REQUESTOR          :  SYS_EDDIC_PROCESSES;
      TSTM_COL_NUMBER         :  TSTM_NUM_OCOKA;
      TSTM_VAL                :  TSTM_COL_VALUES;
end record;

-- Row Feedback record
type TSTM_ROW_FEEDBACK is
   record
      TSTM_REQUESTOR          :  SYS_EDDIC_PROCESSES;
      TSTM_ROW_NUMBER         :  TSTM_NUM_ROW;
      TSTM_COA_COUNT          :  TSTM_NUM_COA;
      TSTM_VAL                :  TSTM_ROW_VALUES;
end record;

-- OCOKA Feedback record
type TSTM_OCOKA_FEEDBACK is
```

```
      record
         TSTM_REQUESTOR          :  SYS_EDDIC_PROCESSES;
         TSTM_VAL                :  TSTM_OCOKA_VALUES;
   end record;

   -- COA Feedback record
   type TSTM_COA_FEEDBACK is
      record
         TSTM_REQUESTOR          :  SYS_EDDIC_PROCESSES;
         TSTM_VAL                :  TSTM_COA_VALUES;
   end record;

   -- Matrix Save record
   type TSTM_MATRIX_SAVE is
      record
         TSTM_REQUESTOR          :  SYS_EDDIC_PROCESSES;
         TSTM_ROW_COUNT          :  TSTM_NUM_ROW;
         TSTM_OCOKA_COUNT        :  TSTM_NUM_OCOKA;
         TSTM_COA_COUNT          :  TSTM_NUM_COA;
         TSTM_VAL                :  TSTM_MATRIX_VALUES;
         TSTM_OCOKA_VAL          :  TSTM_OCOKA_VALUES;
         TSTM_COA_VAL            :  TSTM_COA_VALUES;
   end record;

end TSTM_DB;
```

UED Utility Package Specifications

The following package specifications are contained in the EDDIC general purpose utility function:

```
TSB_LOCATION
UED_EDDIC_MATH_UTIL
UED_LIST
UED_QUEUE
UED_STRING_UTILITIES
```

```
--CPC package specification name:
--     TSB_LOCATION
--
--CPC description:
--     TSB_LOCATION CPC is the Tree Structure Builder, written in the "Ada"
--     programming language, which defines the variables and variable types
--     needed to determine general hierarchical tree structure elements X-Y
--     Locations.
--
--CPC design notes:
--     1.)  This package has generic formal parameters.
--     2.)  The generic parameter is an application dependent data structure
--     the likes of which are of no concern to this package.  If the application
--     wants to associate some data with each element, this is the place to put
--     it.  If not the application must create a dummy structure so the package
--     can be instantiated.
--     3.)  This package can raise the following exceptions:
--          SYS_TSB_EXCEPTION.
--
--CPC package author:
--     Richard T. Zarse      30 Aug 1988
--     Science Applications International Corporation (SAIC)
--     424 Delaware, Suite C-3
--     Leavenworth, KS  66048   (913) 651-7925
--
--
with SYSTEM_PACKAGE;     use SYSTEM_PACKAGE;

generic
   type APPL_DEP_DATA  is private;  -- Application dependent data

package TSB_LOCATION is

   type TSB_TREE_DEPTH   is  range 0..10;

   type TREE_RECORD;
   type TREE_RECORD_PTR is access TREE_RECORD;
   type TREE_RECORD is record
      UL               :  SYS_IMAGE_LOCATION;
      CENTER           :  SYS_IMAGE_LOCATION;
      WIDTH            :  SYS_IMAGE_COLUMN := 25;
      HEIGHT           :  SYS_IMAGE_ROW    := 30;
      CHILD            :  TREE_RECORD_PTR  := null;
      SIBLING          :  TREE_RECORD_PTR  := null;
      CHLDRN_HRZ_2_ME  :  BOOLEAN := True;
      CHLDRN_VRT_2_ME  :  BOOLEAN := False;
      SIBLNG_HRZ_2_ME  :  BOOLEAN := True;
      SIBLNG_VRT_2_ME  :  BOOLEAN := False;
      A_D_D            :  APPL_DEP_DATA;
   end record;

   TSB_X_SPACING       : constant SYS_IMAGE_COLUMN := 25;
   TSB_Y_SPACING       : constant SYS_IMAGE_ROW    := 30;
   TSB_HALF_X_SPACING  : constant SYS_IMAGE_COLUMN := TSB_X_SPACING / 2;
   TSB_HALF_Y_SPACING  : constant SYS_IMAGE_ROW    := TSB_Y_SPACING / 2;
```

```
-- ##############################################################################
   procedure TSB_FIND_XY_LOC (TREE_ELMNT          : in   TREE_RECORD_PTR;
                              VRT_CHLDRN_R_LEGAL  : in   BOOLEAN;
                              VRT_SIBLNG_R_LEGAL  : in   BOOLEAN;
                              TREE_SIZ            : out  SYS_IMAGE_LOCATION);
   --
   --CPM description:
   --    This module, as part of the Tree Structure Builder, determines (Finds)
   --    the X and Y Locations of the given element and all of its siblings
   --    and children.
   --
   --CPM design notes:
   --    1.)  This module is called passing in the hierarchically first or
   --    oldest sibling and all of its children and siblings locations are
   --    determined as well.
   --    2.)  One of the attributes in the element structure is a pointer to
   --    its first child; another attribute points to its next sibling.
   --    Using these two attributes of the structure a forward pointing link
   --    list can be built.  A parent with multiple children points to its
   --    first child and all the other children are pointed to by that childs
   --    siblings, etc, etc.
   --    3.)  Before this module is called the entire link list must be
   --    established and the width and height attributes must be set for each
   --    element.
   --
   --formal parameters
   --IN     TREE_ELMNT          - The hierarchically first Tree Element whose
   --                            location is desired.
   --IN     VRT_CHLDRN_R_LEGAL  - A boolean which tells if it is Legal to
   --                            display Children Vertically.
   --                  = True    - It's legal to display children vertically for
   --                            resultant horizontal space savings.
   --                  = False   - All children will be displayed horizontally.
   --IN     VRT_SIBLNG_R_LEGAL  - A boolean which tells if it is Legal to
   --                            display remaining Siblings Vertically.
   --                            ie: There are six children in the family, the
   --                            last (youngest) four siblings have no
   --                            children; then they are candidates to be
   --                            displayed vertically after the last sibling
   --                            with children.
   --                  = True    - It's legal to display siblings vertically for
   --                            resultant horizontal space savings.
   --                  = False   - All siblings will be displayed horizontally.
   --OUT    TREE_SIZ            - The Size (x & y), in pixels, of the current
   --                            Tree.
   --end formal parameters;
   --

-- ##############################################################################
   procedure TSB_DISPLAY_CONNECTING_LINES (TREE_ELMNT : in  TREE_RECORD_PTR;
                                           WINDOW_ID  : in  SYS_WINDOW_ELE_ID;
                                           OFFSET     : in  SYS_IMAGE_LOCATION;
                                           LUT_COLOR  : in  SYS_COLOR;
                                           PLANE_MASK : in  SYS_COLOR_MASK);
   --
```

```
    --CPM description:
    --     This module, as part of the Tree Structure Builder, Displays the Lines
    --     which Connect parent to child and sibling to sibling for the given
    --     hierarchical tree.
    --
    --CPM design notes:
    --     1.)  This module is called passing in the hierarchically first or
    --     oldest sibling and all of its children and siblings locations are
    --     determined as well.
    --     2.)  Before this module is called all of tree elements must already
    --     have been placed by TSB_FIND_XY_LOC.
    --     3.)  The format used here for connecting children and siblings is
    --     that of a normal general tree with one exception.  Based on certain
    --     rules in TSB_FIND_XY_LOC some parents may display their children
    --     vertically under the parent (stacked up and down), instead of
    --     horizontally centered under the parent (placed side by side).
    --     4.)  This procedure is recursive.
    --
    --formal parameters
    --IN      TREE_ELMNT   - The hierarchically first Tree Element whose lines are
    --                       to be drawn.
    --IN      WINDOW_ID    - The Id of the Window to display the lines in.
    --IN      OFFSET       - The number of X & Y pixels, within the window, to
    --                       Offset these lines.
    --IN      LUT_COLOR    - The index into the color LookUp Table for the Color
    --                       of the lines.
    --IN      PLANE_MASK   - A bit map representation of the Planes to be affected
    --                       by the lines.  Value can be obtained from
    --                       "UIW_PLANE_MASK".
    --end formal parameters;
    --
end TSB_LOCATION;
```

```
--cpc package specification name: UED_EDDIC_MATH_UTIL
--
--cpc description: UED_EDDIC_MATH_UTIL contains all-purpose Ada math utility
--                 procedures that are required throughout the EDDIC system.
--
--cpc design notes:
--    This package raises the SYS_UED_EXCEPTION when an exception is detected.
--
--cpc package author: Bruce J. Packard
--                   Science Applications International Corporation
--                   424 Delaware, Suite C3
--                   Leavenworth, KS 66048
--

with SYSTEM_PACKAGE;          use SYSTEM_PACKAGE;
with SDB_SITUATION_DB;        use SDB_SITUATION_DB;
package UED_EDDIC_MATH_UTIL is


   function UED_BLUEFOR_ECH_RANK (FIRST_UNIT  :  in  SDB_BLUE_TASK_RECORD;
                                  SECOND_UNIT :  in  SDB_BLUE_TASK_RECORD) return
                                  BOOLEAN;
   --
   --cpm procedure name:  UED_BLUEFOR_ECH_RANK
   --
   --cpm description:  Determines if the FIRST_UNIT should be positioned before
   --                  the second unit in a task organization structure. If it
   --                  should, this function returns true, otherwise, it returns
   --                  false.
   --
   --formal parameters
   --IN   FIRST_UNIT       The description of the first unit.
   --
   --IN   SECOND_UNIT      The description of the second unit.
   --


   function UED_DIST (X_POINT_1       :   in      INTEGER;
                      Y_POINT_1       :   in      INTEGER;
                      X_POINT_2       :   in      INTEGER;
                      Y_POINT_2       :   in      INTEGER) return FLOAT;

   function UED_DIST (X_POINT_1       :   in      SYS_PIXEL;
                      Y_POINT_1       :   in      SYS_PIXEL;
                      X_POINT_2       :   in      SYS_PIXEL;
                      Y_POINT_2       :   in      SYS_PIXEL) return FLOAT;

   function UED_DIST (X_POINT_1       :   in      SYS_COORDINATE;
                      Y_POINT_1       :   in      SYS_COORDINATE;
                      X_POINT_2       :   in      SYS_COORDINATE;
                      Y_POINT_2       :   in      SYS_COORDINATE) return FLOAT;
   --
   --cpm procedure name: UED_DIST
   --
   --cpm description: Computes the distance between two points
```

```
--
--formal parameters
--IN    X_POINT_1       X coordinate of first point
--
--IN    Y_POINT_1       Y coordinate of first point
--
--IN    X_POINT_2       X coordinate of second point
--
--IN    Y_POINT_2       Y coordinate of second point
--


function UED_DIST_POINT_TO_LINE (
                    X_POINT          :   in      INTEGER;
                    Y_POINT          :   in      INTEGER;
                    X_LINE_POINT_1   :   in      INTEGER;
                    Y_LINE_POINT_1   :   in      INTEGER;
                    X_LINE_POINT_2   :   in      INTEGER;
                    Y_LINE_POINT_2   :   in      INTEGER) return FLOAT;

function UED_DIST_POINT_TO_LINE (
                    X_POINT          :   in      SYS_PIXEL;
                    Y_POINT          :   in      SYS_PIXEL;
                    X_LINE_POINT_1   :   in      SYS_PIXEL;
                    Y_LINE_POINT_1   :   in      SYS_PIXEL;
                    X_LINE_POINT_2   :   in      SYS_PIXEL;
                    Y_LINE_POINT_2   :   in      SYS_PIXEL) return FLOAT;

function UED_DIST_POINT_TO_LINE (
                    X_POINT          :   in      SYS_COORDINATE;
                    Y_POINT          :   in      SYS_COORDINATE;
                    X_LINE_POINT_1   :   in      SYS_COORDINATE;
                    Y_LINE_POINT_1   :   in      SYS_COORDINATE;
                    X_LINE_POINT_2   :   in      SYS_COORDINATE;
                    Y_LINE_POINT_2   :   in      SYS_COORDINATE) return FLOAT;
--
--cpm procedure name: UED_DIST_POINT_TO_LINE
--
--cpm description: Computes the distance between a point and a line segment
--                defined by two points.
--
--formal parameters
--IN    X_POINT           The X coordinate of the Point.
--
--IN    Y_POINT           The Y coordinate of the Point.
--
--IN    X_LINE_POINT_1    The X coordinate of the start of the line segment.
--
--IN    Y_LINE_POINT_1    The Y coordinate of the start of the line segment.
--
--IN    X_LINE_POINT_2    The X coordinate of the end of the line segment.
--
--IN    Y_LINE_POINT_2    The Y coordinate of the end of the line segment.
--
```

```
procedure UED_INTERSECT_LINES (X_POINT_1       :   in      INTEGER;
                               Y_POINT_1       :   in      INTEGER;
                               X_POINT_2       :   in      INTEGER;
                               Y_POINT_2       :   in      INTEGER;
                               X_POINT_3       :   in      INTEGER;
                               Y_POINT_3       :   in      INTEGER;
                               X_POINT_4       :   in      INTEGER;
                               Y_POINT_4       :   in      INTEGER;
                               INTERSECTION    :   out     BOOLEAN;
                               X_INTERSECT     :   out     INTEGER;
                               Y_INTERSECT     :   out     INTEGER);

procedure UED_INTERSECT_LINES (X_POINT_1       :   in      SYS_PIXEL;
                               Y_POINT_1       :   in      SYS_PIXEL;
                               X_POINT_2       :   in      SYS_PIXEL;
                               Y_POINT_2       :   in      SYS_PIXEL;
                               X_POINT_3       :   in      SYS_PIXEL;
                               Y_POINT_3       :   in      SYS_PIXEL;
                               X_POINT_4       :   in      SYS_PIXEL;
                               Y_POINT_4       :   in      SYS_PIXEL;
                               INTERSECTION    :   out     BOOLEAN;
                               X_INTERSECT     :   out     SYS_PIXEL;
                               Y_INTERSECT     :   out     SYS_PIXEL);

procedure UED_INTERSECT_LINES (X_POINT_1       :   in      SYS_COORDINATE;
                               Y_POINT_1       :   in      SYS_COORDINATE;
                               X_POINT_2       :   in      SYS_COORDINATE;
                               Y_POINT_2       :   in      SYS_COORDINATE;
                               X_POINT_3       :   in      SYS_COORDINATE;
                               Y_POINT_3       :   in      SYS_COORDINATE;
                               X_POINT_4       :   in      SYS_COORDINATE;
                               Y_POINT_4       :   in      SYS_COORDINATE;
                               INTERSECTION    :   out     BOOLEAN;
                               X_INTERSECT     :   out     SYS_COORDINATE;
                               Y_INTERSECT     :   out     SYS_COORDINATE);
--
--cpm procedure name: UED_INTERSECT_LINES
--
--cpm description: Calculates the intersection of two lines specified by
--                 endpoints.
--
-- formal parameters
-- IN    X_POINT_1     The x coordinate of one of the endpoints of the
--                     first line segment.
-- IN    Y_POINT_1     The y coordinate of one of the endpoints of the
--                     first line segment.
-- IN    X_POINT_2     The x coordinate of the other endpoint of the
--                     first line segment.
-- IN    Y_POINT_2     The x coordinate of the other endpoint of the
--                     first line segment.
-- IN    X_POINT_3     The x coordinate of one of the endpoints of the
--                     second line segment.
-- IN    Y_POINT_3     The y coordinate of one of the endpoints of the
--                     second line segment.
-- IN    X_POINT_4     The x coordinate of the other endpoint of the
--                     second line segment.
```

```
-- IN     Y_POINT_4      The y coordinate of the other endpoint of the
--                       second line segment.
-- OUT    INTERSECTION   A logical indicating whether an intersection was
--                       found for the two input line segments.
-- OUT    X_INTERSECT    The x coordinate of the intersection point.
-- OUT    Y_INTERSECT    The y coordinate of the intersection point.
--


procedure UED_LINE_ANGLE (X_POINT_1       :   in      INTEGER;
                          Y_POINT_1       :   in      INTEGER;
                          X_POINT_2       :   in      INTEGER;
                          Y_POINT_2       :   in      INTEGER;
                          SIN_LINE_ANGLE  :   out     FLOAT;
                          COS_LINE_ANGLE  :   out     FLOAT);

procedure UED_LINE_ANGLE (X_POINT_1       :   in      SYS_PIXEL;
                          Y_POINT_1       :   in      SYS_PIXEL;
                          X_POINT_2       :   in      SYS_PIXEL;
                          Y_POINT_2       :   in      SYS_PIXEL;
                          SIN_LINE_ANGLE  :   out     FLOAT;
                          COS_LINE_ANGLE  :   out     FLOAT);

procedure UED_LINE_ANGLE (X_POINT_1       :   in      SYS_COORDINATE;
                          Y_POINT_1       :   in      SYS_COORDINATE;
                          X_POINT_2       :   in      SYS_COORDINATE;
                          Y_POINT_2       :   in      SYS_COORDINATE;
                          SIN_LINE_ANGLE  :   out     FLOAT;
                          COS_LINE_ANGLE  :   out     FLOAT);
--
--cpm procedure name: UED_LINE_ANGLE
--
--cpm description: Computes the sine and the cosine of the angle formed by a
--                 line and the X-axis.
--
-- formal parameters
-- IN     X_POINT_1      The x coordinate of one of the end points of the
--                       line.
-- IN     Y_POINT_1      The y coordinate of one of the end points of the
--                       line.
-- IN     X_POINT_2      The x coordinate of the other endpoint of the line.
-- IN     Y_POINT_2      The y coordinate of the other endpoint of the line.
-- OUT    SIN_LINE_ANGLE The sine of the angle formed by the input line and
--                       the x axis.
-- OUT    COS_LINE_ANGLE The cosine of the angle formed by the input line and
--                       the x axis.
--


procedure UED_INTERSECT_LINE_SEGS (
                                X_POINT_1      :   in      INTEGER;
                                Y_POINT_1      :   in      INTEGER;
                                X_POINT_2      :   in      INTEGER;
                                Y_POINT_2      :   in      INTEGER;
                                X_POINT_3      :   in      INTEGER;
                                Y_POINT_3      :   in      INTEGER;
```

```
                                X_POINT_4       :   in      INTEGER;
                                Y_POINT_4       :   in      INTEGER;
                                INTERSECTION    :   out     BOOLEAN;
                                X_INTERSECT     :   out     INTEGER;
                                Y_INTERSECT     :   out     INTEGER);

procedure UED_INTERSECT_LINE_SEGS (
                                X_POINT_1       :   in      SYS_PIXEL;
                                Y_POINT_1       :   in      SYS_PIXEL;
                                X_POINT_2       :   in      SYS_PIXEL;
                                Y_POINT_2       :   in      SYS_PIXEL;
                                X_POINT_3       :   in      SYS_PIXEL;
                                Y_POINT_3       :   in      SYS_PIXEL;
                                X_POINT_4       :   in      SYS_PIXEL;
                                Y_POINT_4       :   in      SYS_PIXEL;
                                INTERSECTION    :   out     BOOLEAN;
                                X_INTERSECT     :   out     SYS_PIXEL;
                                Y_INTERSECT     :   out     SYS_PIXEL);

procedure UED_INTERSECT_LINE_SEGS (
                                X_POINT_1       :   in      SYS_COORDINATE;
                                Y_POINT_1       :   in      SYS_COORDINATE;
                                X_POINT_2       :   in      SYS_COORDINATE;
                                Y_POINT_2       :   in      SYS_COORDINATE;
                                X_POINT_3       :   in      SYS_COORDINATE;
                                Y_POINT_3       :   in      SYS_COORDINATE;
                                X_POINT_4       :   in      SYS_COORDINATE;
                                Y_POINT_4       :   in      SYS_COORDINATE;
                                INTERSECTION    :   out     BOOLEAN;
                                X_INTERSECT     :   out     SYS_COORDINATE;
                                Y_INTERSECT     :   out     SYS_COORDINATE);
--
--cpm procedure name: UED_INTERSECT_LINE_SEGS
--
--cpm description: Calculates the intersection of two line segments
--                 specified by endpoints.
--
-- formal parameters
-- IN    X_POINT_1     The x coordinate of one of the endpoints of the
--                     first line segment.
-- IN    Y_POINT_1     The y coordinate of one of the endpoints of the
--                     first line segment.
-- IN    X_POINT_2     The x coordinate of the other endpoint of the
--                     first line segment.
-- IN    Y_POINT_2     The x coordinate of the other endpoint of the
--                     first line segment.
-- IN    X_POINT_3     The x coordinate of one of the endpoints of the
--                     second line segment.
-- IN    Y_POINT_3     The y coordinate of one of the endpoints of the
--                     second line segment.
-- IN    X_POINT_4     The x coordinate of the other endpoint of the
--                     second line segment.
-- IN    Y_POINT_4     The y coordinate of the other endpoint of the
--                     second line segment.
-- OUT   INTERSECTION  A logical indicating whether an intersection was
--                     found for the two input line segments. True if the
```

```
--                          intersection point is within the line segments. False
--                          if the lines don't intersect or the intersection
--                          is not within the line segments.
-- OUT    X_INTERSECT       The x coordinate of the intersection point.
-- OUT    Y_INTERSECT       The y coordinate of the intersection point.
--

procedure UED_OFFSET_POINT (X_POINT         :   in      INTEGER;
                            Y_POINT         :   in      INTEGER;
                            SIN_LINE_ANGLE  :   in      FLOAT;
                            COS_LINE_ANGLE  :   in      FLOAT;
                            OFFSET_ANGLE    :   in      FLOAT;
                            OFFSET_DISTANCE:    in      FLOAT;
                            OFFSET_X        :   out     INTEGER;
                            OFFSET_Y        :   out     INTEGER);

procedure UED_OFFSET_POINT (X_POINT         :   in      SYS_PIXEL;
                            Y_POINT         :   in      SYS_PIXEL;
                            SIN_LINE_ANGLE  :   in      FLOAT;
                            COS_LINE_ANGLE  :   in      FLOAT;
                            OFFSET_ANGLE    :   in      FLOAT;
                            OFFSET_DISTANCE:    in      FLOAT;
                            OFFSET_X        :   out     SYS_PIXEL;
                            OFFSET_Y        :   out     SYS_PIXEL);

procedure UED_OFFSET_POINT (X_POINT         :   in      SYS_COORDINATE;
                            Y_POINT         :   in      SYS_COORDINATE;
                            SIN_LINE_ANGLE  :   in      FLOAT;
                            COS_LINE_ANGLE  :   in      FLOAT;
                            OFFSET_ANGLE    :   in      FLOAT;
                            OFFSET_DISTANCE:    in      FLOAT;
                            OFFSET_X        :   out     SYS_COORDINATE;
                            OFFSET_Y        :   out     SYS_COORDINATE);
--
--cpm procedure name: UED_OFFSET_POINT
--
--cpm description: Offsets a point by a specified distance and angle from
--                the original point on a line specified by its angle from
--                the x-axis.
--
-- formal parameters
-- IN     X_POINT          The x coordinate of the point to be offset.
-- IN     Y_POINT          The y coordinate of the point to be offset.
-- IN     SIN_LINE_ANGLE   The sine of the angle formed by the line
--                         containing the point and the x axis.
-- IN     COS_LINE_ANGLE   The cosine of the angle formed by the line
--                         containing the point and the x axis.
-- IN     OFFSET_ANGLE     The angle at which the point is to be offset
--                         from the line.
-- IN     OFFSET_DISTANCE  The distance from the line which the point is
--                         to be offset.
-- OUT    OFFSET_X         The x coordinate of the resultant point.
-- OUT    OFFSET_Y         The y coordinate of the resultant point.
--
```

```
function UED_OPFOR_ECH_RANK (FIRST_UNIT  :  in  SDB_OPFOR_TASK_RECORD;
                             SECOND_UNIT :  in  SDB_OPFOR_TASK_RECORD) return
                             BOOLEAN;
--
--cpm procedure name:  UED_OPFOR_ECH_RANK
--
--cpm description:  Determines if the FIRST_UNIT should be positioned before
--                  the second unit in a task organization structure. If it
--                  should, this function returns true, otherwise, it returns
--                  false.
--
--formal parameters
--IN   FIRST_UNIT       The description of the first unit.
--
--IN   SECOND_UNIT      The description of the second unit.
--


procedure UED_POINT_LINE_XING (
                    X_POINT          :   in      INTEGER;
                    Y_POINT          :   in      INTEGER;
                    X_LINE_POINT_1   :   in      INTEGER;
                    Y_LINE_POINT_1   :   in      INTEGER;
                    X_LINE_POINT_2   :   in      INTEGER;
                    Y_LINE_POINT_2   :   in      INTEGER;
                    X_XING           :   out     INTEGER;
                    Y_XING           :   out     INTEGER);
procedure UED_POINT_LINE_XING (
                    X_POINT          :   in      SYS_PIXEL;
                    Y_POINT          :   in      SYS_PIXEL;
                    X_LINE_POINT_1   :   in      SYS_PIXEL;
                    Y_LINE_POINT_1   :   in      SYS_PIXEL;
                    X_LINE_POINT_2   :   in      SYS_PIXEL;
                    Y_LINE_POINT_2   :   in      SYS_PIXEL;
                    X_XING           :   out     SYS_PIXEL;
                    Y_XING           :   out     SYS_PIXEL);

procedure UED_POINT_LINE_XING (
                    X_POINT          :   in      SYS_COORDINATE;
                    Y_POINT          :   in      SYS_COORDINATE;
                    X_LINE_POINT_1   :   in      SYS_COORDINATE;
                    Y_LINE_POINT_1   :   in      SYS_COORDINATE;
                    X_LINE_POINT_2   :   in      SYS_COORDINATE;
                    Y_LINE_POINT_2   :   in      SYS_COORDINATE;
                    X_XING           :   out     SYS_COORDINATE;
                    Y_XING           :   out     SYS_COORDINATE);
--
--cpm procedure name: UED_POINT_LINE_XING
--
--cpm description: Computes the intersection of a line defined by two points
--                 and by a perpendicular line that passes through a point.
--
--formal parameters
--IN   X_POINT          The X coordinate of the Point.
--
```

```
   --IN    Y_POINT           The Y coordinate of the Point.
   --
   --IN    X_LINE_POINT_1    The X coordinate of the start of the line segment.
   --
   --IN    Y_LINE_POINT_1    The Y coordinate of the start of the line segment.
   --
   --IN    X_LINE_POINT_2    The X coordinate of the end of the line segment.
   --
   --IN    Y_LINE_POINT_2    The Y coordinate of the end of the line segment.
   --
   --OUT   X_XING            X coordinate of the Intersection Point.
   --
   --OUT   Y_XING            Y coordinate of the Intersection Point.
   --

end UED_EDDIC_MATH_UTIL;
```

```
--cpc package specification name: UED_List
--
--cpc description: UED_List contains a generic list system.
--
--cpc exceptions:
--  UED_No_More_Space_In_List   Signalled when no more information can
--                              be put into the list.
--
--  UED_Beyond_End_Of_List      Signalled when user attempts to access
--                              information beyond the end of the list.
--
--cpc design notes:
--    The list always maintains a pointer to the current item in a list
--    and operates with respect to the current position.  An insertion operation--
  always causes the newly inserted item to be the current item. A query of
--    the list's contents always sets the current position to the beginning of
--    the list.  A deletion or retrieval sets the current position to the next
--    available item.
--
--cpc package author: Laura M. McClanahan
--                    Science Applications International Corporation
--                    424 Delaware, Suite C3
--                    Leavenworth, KS 66048

with System_Package;   use System_Package;
generic
   type List_Item_Type is private;

package UED_List is

   UED_No_More_Space_In_List, UED_Beyond_End_Of_List : exception;

   type UED_List_Contents is array (SYS_DB_SIZE range <>) of List_Item_Type;
   type UED_List_Content_Ptr is access UED_List_Contents;


   procedure UED_Delete_List_Item;
   --
   --cpm description: This procedure deletes the current item from the list.
   --                 The current position of the list is set to the next
   --                 available item.
   --
   --formal parameters
   --None
   --end formal parameters;


   function UED_End_Of_List return boolean;
   --
   --cpm description: This function returns "true" if there are no more
   --                 items in the list.  Otherwise, it returns "false".
   --
   --formal parameters
   --None
   --end formal parameters;
```

```
procedure UED_Get_Next_Item_From_List (Information : out List_Item_Type);
--
--cpm description: This procedure retrieves the next item from the list.
--                 It retrieves the first item if this procedure was
--                 preceded by a call on "Go_To_Beginning_Of_List".
--
--formal parameters
--OUT   Information   The data item corresponding to the current position
--                    pointer of the list.
--end formal parameters;


procedure UED_Go_To_Beginning_Of_List;
--
--cpm description: This procedure resets the list pointer to the beginning
--                 of the list.
--
--formal parameters
--None
--end formal parameters;


procedure UED_Insert_After_List_Item (Information : in List_Item_Type);
--
--cpm description: This procedure places information after the current
--                 item in the list.
--
--formal parameters
--IN   Information   The data to be inserted into the list.
--end formal parameters;


procedure UED_Insert_Before_List_Item (Information : in List_Item_Type);
--
--cpm description: This procedure places information before the current
--                 item in the list.
--
--formal parameters
--IN   Information   The data to be inserted into the list.
--end formal parameters;


function UED_List_Count return SYS_DB_SIZE;
--
--cpm description: UED_List_Count returns the number of items currently
--                 queued.
--
--formal parameters
--None
--end formal parameters;


procedure UED_Query_List (List_Contents :   in out   UED_List_Content_Ptr);
--
```

```
    --cpm description:  UED_Query_List returns an array of the List containing
    --                  pointers to all the items.  Note the array should be
    --                  allocated by the application to the total item count
    --                  which may be obtained via UED_List_Count.
    --
    --formal parameters
    --IN OUT   List_Contents  The array of the List and its item pointers.
    --end formal parameters;


    procedure UED_Set_List_Current_Item (Information :   in   List_Item_Type);
    --
    --cpm description:  UED_Set_List_Current_Item sets the given item as the
    --                  current item.
    --
    --formal parameters
    --IN   Information  The information in the list to be considered the
    --                  current item.
    --end parameters;


end UED_List;
```

```
--cpc package specification name: UED_QUEUE
--
--cpc description: UED_QUEUE contains a generic queue system.
--
--cpc exceptions:
-- UED_Queue_Underflow   Raised whenever an attempt is made to remove
--                       an item off of an empty queue.
--
--SYS_UED_EXCEPTION      Raised whenever a system CONSTRAINT, NUMERIC, or
--                       STORAGE error is raised.
--
--cpc design notes:
--
--cpc package author: Laura M. McClanahan
--                   Science Applications International Corporation
--                   424 Delaware, Suite C3
--                   Leavenworth, KS 66048

with SYSTEM_PACKAGE;   use SYSTEM_PACKAGE;

generic
   type Queue_Item_Type is private;

package UED_Queue is

   type UED_Queue_Contents is array (SYS_DB_SIZE range <>) of Queue_Item_Type;
   type UED_Queue_Content_Ptr is access UED_Queue_Contents;

   UED_Queue_Underflow : exception;


   function UED_Queue_Count return SYS_DB_SIZE;
   --
   --cpm description: UED_Queue_Count returns the number of items currently
   --                 queued.
   --
   --formal parameters
   --None
   --end formal parameters;


   function UED_Queue_Empty return Boolean;
   --
   --cpm description: UED_Queue_Empty returns "true" if the queue is empty;
   --                 otherwise, it returns "false".
   --
   --formal parameters
   --NONE
   --end formal parameters;


   procedure UED_Queue_Insert (Information :  in  Queue_Item_Type);
   --
   --cpm description: UED_Queue_Insert pushes an item into the queue.
   --
```

```
    --formal parameters
    --IN   Information      The information to be pushed into the queue.
    --
    -- end formal parameters;


    procedure UED_Queue_Delete (Information :  out   Queue_Item_Type);
    --
    --cpm description: UED_Queue_Delete deletes an item from the queue.
    --
    --formal parameters
    --OUT  Information      The information to be deleted from the queue.
    --
    -- end formal parameters;


    procedure UED_Queue_Peek (Information : out   Queue_Item_Type);
    --
    --cpm description: UED_Queue_Peek peeks at the next item on the queue,
    --                 without deleting the information from the queue.
    --
    --formal parameters
    --OUT   Information     The information peeked from the next item on the
    --                      queue.
    --
    --end formal parameters;


    procedure UED_Queue_Query (Contents :   in out   UED_Queue_Content_Ptr);
    --
    --cpm description:  UED_Queue_Query fills the array pointed to by the input
    --                  pointer with all the information currently queued.
    --                  The access pointer must already be allocated to the
    --                  current number of items in the queue; obtained via
    --                  UED_Queue_Count.
    --
    --formal parameters
    --IN OUT   Contents    The pointer to an array containing all the information
    --                     currently queued.
    --
    -- end formal parameters;

end UED_Queue;
```

```
--cpc package specification name: UED_STRING_UTILITIES
--
--cpc description:  UED_STRING_UTILITIES contains the string utilities
--                  used throughout the EDDIC system.
--
--cpc design notes:
--    This package raises the SYS_UED_EXCEPTION when a exception is detected.
--
--cpc package author: Laura M. McClanahan
--                    Science Applications International Corporation
--                    424 Delaware, Suite C3
--                    Leavenworth, KS 66048
--
with SYSTEM_PACKAGE;   use SYSTEM_PACKAGE;

package UED_STRING_UTILITIES is


   procedure UED_COUNT_LINES (TEXT          :   in      SYS_TEXT_PTR;
                              WIDTH         :   out     SYS_WINDOW_COLUMN;
                              HEIGHT        :   out     SYS_PRODUCT_LENGTH);
   --
   --cpm procedure name: UED_COUNT_LINES
   --
   --cpm description: Counts the number of lines a in string buffer and
   --                 determines the width of the longest line.
   --
   --formal parameters
   --IN   TEXT          Textual Buffer
   --
   --OUT  WIDTH         Length of the longest line in the buffer (characters)
   --
   --OUT  HEIGHT        Number of lines in the buffer
   --


   procedure UED_INTEGER_STRING (INTEGER_VALUE   :   in      INTEGER;
                                 ZERO_FILLED     :   in      BOOLEAN;
                                 STRING_FIELD    :   in out  STRING);
   --
   --cpm procedure name:  UED_INTEGER_STRING
   --
   --cpm description:  Converts an integer into a string of specified length,
   --                  right justified within the string field given and
   --                  either blank filled or zero filled depending upon the
   --                  user's request.
   --
   --formal parameters
   --IN   INTEGER_VALUE    The integer value to be converted into a string.
   --
   --IN   ZERO_FILLED      The logical indicating whether the resultant
   --                      string should include leading zeroes or not.
   --                      TRUE = After the string is right justified within
   --                             the given string field, zero fill any
   --                             leading blanks
```

```
    --                          FALSE = Do not zero fill any leading blanks
    --
    --IN-OUT STRING_FIELD      The string field to contain the resultant
    --                          conversion of the integer into characters.
    --


    function UED_STRING_SEARCH (TEXT:    in   STRING;
                                WORD:    in   STRING;
                                START_INDEX:   in   Positive := 1)
             return Natural;


    function UED_STRING_SEARCH (TEXT:    in   SYS_TEXT_PTR;
                                WORD:    in   STRING;
                                START_INDEX:   in   Positive := 1)
             return Natural;


    function UED_STRING_SEARCH (TEXT:          in   SYS_TEXT_PTR;
                                WORD:          in   SYS_TEXT_PTR;
                                START_INDEX:   in   Positive := 1)
             return Natural;
    --
    --cpm procedure name:  UED_STRING_SEARCH
    --
    --cpm description: UED_STRING_SEARCH provides a string search implementation
    --                  of the Boyer_Moore approach as written by David P. Wood
    --                  and David Turcaso in the article "Implementing a Faster
    --                  String Search Algorithm in Ada" published in the 1988
    --                  May/June issue of Ada Letters.  The implementation here is
--                    the second implementation provided in the article, found
    --                  on pages 96 and 97 with the enhancement and correction
    --                  made by David Wood in his letter to the editors in the
    --                  November/December issue.
    --
    -- formal parameters:
    -- IN      TEXT         The string of text to be searched.
    --
    -- IN      WORD         The word or string to be found.
    --
    -- IN      START_INDEX  The index of the text buffer at which the search will
    --                      start.
    --
    -- end formal parameters;

end UED_STRING_UTILITIES;
```

UFM Utility Package Specifications

The following package specifications are contained in the form manager function:

UFM_FORM_FIELDS
UFM_FORM_MANAGER

```
--cpc package specification name:  UFM_FORM_FIELDS
--
--cpc description:  UFM_FORM_FIELDS provides the capabilities of creating,
--                  handling, and deleting the individual fields of a form.
--
--cpc design notes:
--
--cpc package author: Laura McClanahan
--                    Science Applications International Corporation
--                    424 Delaware, Suite C3
--                    Leavenworth, KS 66048
--
with System_Package;   use System_Package;

package UFM_Form_Fields is

   type UFM_Form_Editor is private;

   type UFM_Form_Src is private;


   procedure UFM_CHANGE_CHECKBOX_STATES (Checkbox_ID:   in   UFM_Form_Editor;
                                         Num_Fields:    in   SYS_MENU_BUTTON_INDEX;
                                         Start_Index:   in   SYS_MENU_BUTTON_INDEX;
                                         Status_Array:  in   SYS_MENU_BUTTON_STATUS_PTR;
                                         State_Flag:    in   BOOLEAN);
   --
   -- CPM description:  UFM_CHANGE_CHECKBOX_STATES changes one or more
   --                   checkbox states according to the input state flag.
   --
   -- formal parameters
   --IN   Checkbox_ID       The ID attached to the checkbox editor.
   --
   --IN   Num_Fields        The number of checkbox(es) states to be changed.
   --
   --IN   Start_Index       The correlating index of the checkbox which the
   --                       start of the array to the order the items were
   --                       originally created; the first element is always
   --                       zero.
   --
   --IN   Status_Array      The array of current status of the checkboxes to
   --                       be changed.
   --
   --IN   State_Flag        The flag indicating the state all the checkboxes
   --                       are to match.
   -- end formal parameters;


   procedure UFM_CHANGE_MEMO_TEXT (EDITOR_ID:          in   UFM_Form_Editor;
                                   MAX_BUFFER_SIZE:    in   SYS_PRODUCT_LENGTH;
                                   TEXT_BUFFER:        in   SYS_TEXT_PTR;
                                   BUFFER_SIZE:        in   SYS_PRODUCT_LENGTH);

   --
   -- CPM description: Changes the text buffer used by a form's memo.
```

```
--
-- formal parameters
--IN     EDITOR_ID          ID attached to the memo.
--
--IN     MAX_BUFFER_SIZE    Maximum number of pixels that the TEXT_BUFFER
--                          can hold.
--
--IN     TEXT_BUFFER        Buffer of the initial text to display in the memo.
--
--IN     BUFFER_SIZE        The number of pixels in TEXT_BUFFER.
-- end formal parameters;


procedure UFM_CHANGE_SCROLLBAR (SCROLLBAR_ID:   in   UFM_FORM_EDITOR;
                                DOC_SIZE:       in   SYS_IMAGE_PIXEL;
                                PIXEL_LENGTH:   in   SYS_WINDOW_PIXEL;
                                DISP_POSITION:  in   SYS_IMAGE_PIXEL;
                                SCROLL_INTRVL:  in   SYS_WINDOW_PIXEL);

--
-- CPM description: Changes the size of a scrollbar.
-- formal parameters
--IN     SCROLLBAR_ID       ID to attached to the scrollbar.
--                          This ID was defined by UFM_DEFINE_SCROLLBAR.
--
--IN     DOC_SIZE           The number of lines in the document buffer.
--
--IN     PIXEL_LENGTH       The number of pixels to be occupied by the
--                          scrollbar.
--
--IN     SCROLL_INTRVL      The number of pixels the work will be scrolled
--                          whenever the user selects an arrow button.  Note:
--                          The work will not be scrolled by these utilities
--                          but, this argument is required to calculate
--                          the interactive slidepositioning.
-- end formal parameters;


procedure UFM_DEFINE_BUTTON_WALK (EDITOR:          out  UFM_Form_Editor;
                                  FORM_SRC:        in   UFM_Form_Src;
                                  DEST_ID  :       in   SYS_WINDOW_ELE_ID;
                                  MENU_STRUCT_ID:  in   SYS_WINDOW_ELE_ID;
                                  MENU_INDEX:      in   SYS_WALKING_CELL;
                                  PIXEL_COL:       in   SYS_WINDOW_COLUMN;
                                  PIXEL_ROW:       in   SYS_WINDOW_ROW;
                                  PIXEL_WIDTH:     in   SYS_WINDOW_COLUMN;
                                  PIXEL_HEIGHT:    in   SYS_WINDOW_ROW;
                                  BUTTON_TEXT:     in   string);

--
-- CPM description: Defines a button  walking menu within a form.
--
--
-- formal parameters
--OUT    EDITOR             ID attached to the editor. This
--                          ID is required for all interactions with the editor.
```

```
--
--IN     FORM_SRC          The ID of the Source of the Form as output by
--                         UFM_INITIALIZE_FORM_FIELDS.
--
--IN     DEST_ID           ID attached to the destination that the editor is
--                         assigned to.  This is set to NULL when the
--                         destination is the RootWindow.
--
--IN     MENU_STRUCT_ID    The ID attached to the menu.
--
--IN     MENU_INDEX        The index into the Text_Array of the submenu to
--                         be activated for a particular window, if applicable.
--                         If the menu to be activated is not a walking menu,
--                         or is the top level of a walking menu, then this
--                         parameter should be set to NULL.
--
--
--IN     PIXEL_COL         Column number from within the window where the left
--                         side of the button shall be placed. Column 0 is at
--                         left of the window.
--
--IN     PIXEL_ROW         Row number from within the window where the top side
--                         of the button shall be placed. Row 0 is at the top
--                         of the window.
--
--IN     PIXEL_WIDTH       The number of columns to be occupied by the button.
--
--IN     PIXEL_HEIGHT      The number of rows to be occupied by the button.
--
--IN     BUTTON_TEXT       Textual string to display in the button.
-- end formal parameters;


procedure UFM_Define_Checkbox_Menu (Editor:    out   UFM_Form_Editor;
              FORM_SRC:         in   UFM_Form_Src;
              DEST_TYPE:        in   SYS_DESTINATION_TYPE;
              DEST_ID   :       in   SYS_WINDOW_ELE_ID;
              PIXEL_COL:        in   SYS_WINDOW_COLUMN;
              PIXEL_ROW:        in   SYS_WINDOW_ROW;
              NUM_FIELDS:       in   SYS_MENU_BUTTON_INDEX;
              NUM_COLS:         in   SYS_MENU_BUTTON_INDEX;
              LABELS:           in   SYS_MENU_BUTTON_LABEL;
              STATUS:           in   SYS_MENU_BUTTON_STATUS;
              PIXEL_WIDTH:      in   SYS_WINDOW_COLUMN := SYS_NULL_COLUMN;
              PIXEL_HEIGHT:     in   SYS_WINDOW_ROW := SYS_NULL_ROW);
--
-- CPM description:  This defines a menu where the user is allowed to
--                   make multiple selections.
--
--formal parameters:
--OUT    EDITOR            ID attached to the editor. This
--                         ID is required for all interactions with the editor.
--
--IN     FORM_SRC          The ID of the Source of the Form as output by
--                         UFM_INITIALIZE_FORM_FIELDS.
--
```

```
--IN    DEST_TYPE       The type of the destination for the editor, where:
--                      SYS_WINDOW_DEST = Window
--                      SYS_PANEL_DEST = Panel
--
--IN    DEST_ID         ID attached to the destination that the editor is
--                      assigned to.  This is set to NULL when the
--                      destination is the RootWindow.
--
--IN    PIXEL_COL        Column number from within the form where the left
--                      side of the menu shall be placed. Column 0 is at
--                      left of the form.
--
--IN    PIXEL_ROW        Row number from within the form where the top side
--                      of the menu shall be placed. Row 0 is at the top
--                      of the form.
--
--IN    NUM_FIELDS       The total number of checkbox buttons to be in the
--                      menu.
--
--IN    NUM_COLS         The number of columns the checkbox buttons are to be
--                      arranged in.
--
--IN    LABELS           Pointer to the array of label addresses for all
--                      the checkbox buttons.
--
--IN    STATUS           Pointer to the boolean array of statuses for all the
--                      checkbox buttons.
--
--IN    PIXEL_WIDTH      The number of pixel columns wide the checkbox editor
--                      is to be created.  If wish width to be calculated,
--                      use the default value of zero.
--
--IN    PIXEL_HEIGHT     The number of pixel rows height the checkbox editor
--                      is to be created.  If wish height to be calculated,
--                      use the default value of zero.
-- end formal parameters;


procedure UFM_Define_Map;
-- CPM description:
--formal parameters:
--
--end formal parameters;


procedure UFM_Define_Memo (Editor:    out   UFM_Form_Editor;
                           FORM_SRC:          in    UFM_Form_Src;
                           DEST_TYPE:         in    SYS_DESTINATION_TYPE;
                           DEST_ID  :         in    SYS_WINDOW_ELE_ID;
                           PIXEL_COL:         in    SYS_WINDOW_COLUMN;
                           PIXEL_ROW:         in    SYS_WINDOW_ROW;
                           PIXEL_WIDTH:       in    SYS_WINDOW_COLUMN;
                           PIXEL_HEIGHT:      in    SYS_WINDOW_ROW;
                           READ_ONLY:         in    BOOLEAN;
                           MAX_BUFFER_SIZE:   in    SYS_PRODUCT_LENGTH;
                           TEXT_BUFFER:       in    SYS_TEXT_PTR;
```

```
                                BUFFER_SIZE:        in   SYS_PRODUCT_LENGTH);
-- CPM description:  This procedure defines a memo area within a form.
--
--formal parameters:
--OUT    EDITOR           The ID attached to the memo.  This ID is required
--                        for all interactions with the memo.
--
--IN     FORM_SRC         The ID of the Source of the Form as output by
--                        UFM_INITIALIZE_FORM_FIELDS.
--
--IN     DEST_TYPE        The type of the destination for the editor, where:
--                        SYS_WINDOW_DEST = Window
--                        SYS_PANEL_DEST = Panel
--
--IN     DEST_ID          ID attached to the destination that the editor is
--                        assigned to.  This is set to NULL when the
--                        destination is the RootWindow.
--
--IN     PIXEL_COL        Column number from within the form where the left
--                        side of the memo shall be placed. Column 0 is at
--                        left of the form.
--
--IN     PIXEL_ROW        Row number from within the form where the top side
--                        of the memo shall be placed. Row 0 is at the top
--                        of the form.
--
--IN     PIXEL_WIDTH      The number of columns to be occupied by the memo.
--
--IN     PIXEL_HEIGHT     The number of rows to be occupied by the memo.
--
--IN     READ_ONLY        Flag indicating if the user has full editing
--                        capabilities or is limited to only scroll and copy
--                        operations.
--                                    true   =  Read only
--                                    false  =  Full edit
--
--IN     MAX_BUFFER_SIZE  Maximum number of pixels that the TEXT_BUFFER
--                        can hold.
--
--IN     TEXT_BUFFER      Buffer of the initial text to display in the memo.
--
--IN     BUFFER_SIZE      The number of pixels in TEXT_BUFFER.
-- end formal parameters;


procedure UFM_Define_Number_Field (Editor:    out    UFM_Form_Editor;
                                   FORM_SRC:          in       UFM_Form_Src;
                                   DEST_TYPE:         in       SYS_DESTINATION_TYPE;
                                   DEST_ID  :         in       SYS_WINDOW_ELE_ID;
                                   PIXEL_COL:         in       SYS_WINDOW_COLUMN;
                                   PIXEL_ROW:         in       SYS_WINDOW_ROW;
                                   LABEL:             in       STRING;
                                   LABEL_POSITION:    in       SYS_LABEL_POSITION;
                                   NUMBER_VARIABLE:   in out   STRING;
                                   MIN_NUMBER:        in       STRING;
                                   MAX_NUMBER:        in       STRING;
```

```
                                        MAX_CHARACTERS:    in       SYS_PRODUCT_LENGTH);
-- CPM description:  This procedure defines a number field within a form.
--
--formal parameters:
--OUT    Editor             The ID attached to the editor.  This ID is required
--                          all interactions with the number field.
--
--IN     FORM_SRC           The ID of the Source of the Form as output by
--                          UFM_INITIALIZE_FORM_FIELDS.
--
--IN     DEST_TYPE          The type of the destination for the editor, where:
--                          SYS_WINDOW_DEST = Window
--                          SYS_PANEL_DEST = Panel
--
--IN     DEST_ID            ID attached to the destination that the editor is
--                          assigned to.  This is set to NULL when the
--                          destination is the RootWindow.
--
--IN     PIXEL_COL          Column number from within the form where the left
--                          side of the editor shall be placed. Column 0 is at
--                          left of the form.
--
--IN     PIXEL_ROW          Row number from within the form where the top side
--                          of the editor shall be placed. Row 0 is at the top
--                          of the form.
--
--IN     LABEL              The optional label before the number field.  This
--                          should be set to NULL if no label will be displayed.
--
--IN     LABEL_POSITION     Value specifying whether the optional label should
--                          be placed to the left or the right of the number
--                          field.  The two valid settings for this field are:
--                              0 = Left aligned
--                              1 = Right aligned
--                          If no label is specified, this parameter will
--                          be ignored.
--
--INOUT NUMBER_VARIABLE     The address of the variable to store the
--                          input number at.  This variable may be
--                          initialized to some number value, which would
--                          be displayed.  This must be a NULL terminated
--                          string.
--
--IN     MIN_NUMBER         The string representing the minimum number
--                          to be allowed as input from the user.  This
--                          string must be MAX_CHARACTERS long with each
--                          digit of the string representing the minimum
--                          value for that digit and the string must be NULL
--                          terminated.
--
--IN     MAX_NUMBER         The string representing the maximum number to be
--                          allowed as input from the user.  This string must
--                          be MAX_CHARACTERS long with each digit of the string
--                          representing the maximum value for that digit and
--                          the string must be NULL terminated.
--
```

```
--IN    MAX_CHARACTERS    The maximum number of characters which will
--                        be allowed to be entered into the field.
--
-- end formal parameters;


procedure UFM_DEFINE_PUSHBUTTON (EDITOR:         out  UFM_Form_Editor;
                                 FORM_SRC:       in   UFM_Form_Src;
                                 DEST_TYPE:      in   SYS_DESTINATION_TYPE;
                                 DEST_ID  :      in   SYS_WINDOW_ELE_ID;
                                 PIXEL_COL:      in   SYS_WINDOW_COLUMN;
                                 PIXEL_ROW:      in   SYS_WINDOW_ROW;
                                 NUM_FIELDS:     in   SYS_MENU_BUTTON_INDEX;
                                 NUM_COLS:       in   SYS_MENU_BUTTON_INDEX;
                                 LABELS:         in   SYS_MENU_BUTTON_LABEL_PTR;
                                 DEFAULT_BUTTON: in   SYS_MENU_BUTTON_VALUES);
--
-- CPM description: Creates a pushbutton editor within the form.
--
-- formal parameters
--OUT   EDITOR           ID attached to the editor. This
--                       ID is required for all interactions with the editor.
--
--IN    FORM_SRC         The ID of the Source of the Form as output by
--                       UFM_INITIALIZE_FORM_FIELDS.
--
--IN    DEST_TYPE        The type of the destination for the editor, where:
--                       SYS_WINDOW_DEST = Window
--                       SYS_PANEL_DEST = Panel
--
--IN    DEST_ID          ID attached to the destination that the editor is
--                       assigned to.  This is set to NULL when the
--                       destination is the RootWindow.
--
--IN    PIXEL_COL        Column number from within the form where the left
--                       side of the editor shall be placed. Column 0 is at
--                       left of the form.
--
--IN    PIXEL_ROW        Row number from within the form where the top side
--                       of the editor shall be placed. Row 0 is at the top
--                       of the form.
--
--IN    NUM_FIELDS       The total number of pushbuttons to be in the
--                       editor.
--
--IN    NUM_COLS         The number of columns the pushbuttons are to be
--                       arranged in.
--
--IN    LABELS           Address of the array of label addresses for all the
--                       pushbuttons.
--
--IN    DEFAULT_BUTTON   The index into the pushbutton array of the button to
--                       be drawn "active" or displayed as the default
--                       button.  A value of SYS_NO_DEFAULT_BUTTON will
--                       disable this feature.
-- end formal parameters;
```

```
procedure UFM_Define_Radiobutton_Menu (Editor:    out   UFM_Form_Editor;
                                       FORM_SRC:        in   UFM_Form_Src;
                                       DEST_TYPE:       in   SYS_DESTINATION_TYPE;
                                       DEST_ID  :       in   SYS_WINDOW_ELE_ID;
                                       PIXEL_COL:       in   SYS_WINDOW_COLUMN;
                                       PIXEL_ROW:       in   SYS_WINDOW_ROW;
                                       NUM_FIELDS:      in   SYS_MENU_BUTTON_INDEX;
                                       NUM_COLS:        in   SYS_MENU_BUTTON_INDEX;
                                       LABELS:          in   SYS_MENU_BUTTON_LABEL;
                                       DEFAULT_BUTTON: in   SYS_MENU_BUTTON_INDEX);
--
-- CPM description:  This procedure defines a single selection menu within
--                    a form.  If the specified area is not large enough to
--                    have all the options visible to the user, a scrollbar
--                    will be added to provide the capability to scroll the
--                    options.
--
--formal parameters:
--OUT   EDITOR          ID attached to the menu  editor. This
--                      ID is required for all interactions with the editor.
--
--IN    FORM_SRC        The ID of the Source of the Form as output by
--                      UFM_INITIALIZE_FORM_FIELDS.
--
--IN    DEST_TYPE       The type of the destination for the editor, where:
--                      SYS_WINDOW_DEST = Window
--                      SYS_PANEL_DEST = Panel
--
--IN    DEST_ID         ID attached to the destination that the editor is
--                      assigned to.  This is set to NULL when the
--                      destination is the RootWindow.
--
--IN    PIXEL_COL       Column number from within the form where the left
--                      side of the menu shall be placed. Column 0 is at
--                      left of the form.
--
--IN    PIXEL_ROW       Row number from within the form where the top side
--                      of the editor shall be placed. Row 0 is at the top
--                      of the form.
--
--IN    NUM_FIELDS      The total number of radiobuttons to be in the
--                      editor.
--
--IN    NUM_COLS        The number of columns the radiobuttons are to be
--                      arranged in.
--
--IN    LABELS          Address of the array of label addresses for all the
--                      radiobuttons.
--
--IN    DEFAULT_BUTTON  The index into the radiobutton array of the button to
--                      be drawn "active" or displayed as the default
--                      button.
-- end formal parameters;
```

```
procedure UFM_Define_Scrollbar (Editor:    out   UFM_Form_Editor;
                                FORM_SRC:        in    UFM_Form_Src;
                                DEST_TYPE:       in    SYS_DESTINATION_TYPE;
                                DEST_ID  :       in    SYS_WINDOW_ELE_ID;
                                ORIENTATION:     in    SYS_SB_DIRECTION;
                                PIXEL_COL:       in    SYS_WINDOW_COLUMN;
                                PIXEL_ROW:       in    SYS_WINDOW_ROW;
                                PIXEL_WIDTH:     in    SYS_WINDOW_PIXEL;
                                PIXEL_LENGTH:    in    SYS_WINDOW_PIXEL;
                                DOC_SIZE:        in    SYS_IMAGE_PIXEL;
                                DISP_POSITION:   in    SYS_IMAGE_PIXEL;
                                SCROLL_INTRVL:   in    SYS_WINDOW_PIXEL);
--
-- CPM description:  This provides the form with a scrollbar either at
--                   the side or bottom of the form.
--
--formal parameters:
--OUT    EDITOR          The ID attached to the scrollbar.  This ID is
--                      required for all interactions with the scrollbar.
--
--IN     FORM_SRC        The ID of the Source of the Form as output by
--                      UFM_INITIALIZE_FORM_FIELDS.
--
--IN     DEST_TYPE       The type of the destination for the editor, where:
--                      SYS_WINDOW_DEST = Window
--                      SYS_PANEL_DEST = Panel
--
--IN     DEST_ID         ID attached to the destination that the editor is
--                      assigned to.  This is set to SYS_ROOT_WINDOW when
--                      the destination is the RootWindow.
--
--IN     ORIENTATION     Direction of the scrollbar (Horizontal or Vertical)
--
--IN     PIXEL_COL       Column number from within the form where the left
--                      side of the scrollbar shall be placed. Column 0 is
--                      at the left of the form.
--
--IN     PIXEL_ROW       Row number from within the form where the top side
--                      of the scrollbar shall be placed. Row 0 is at the
--                      top of the form.
--
--IN     PIXEL_WIDTH     The number of pixels to be occupied by the
--                      scrollbar's width.
--
--IN     PIXEL_LENGTH    The number of pixels to be occupied by the
--                      scrollbar's length.
--
--IN     DOC_SIZE        The number of lines in the document buffer.
--
--IN     DISP_POSITION   The offset from the beginning of the work surface to
--                      first pixel visible to the user.
--
--IN     SCROLL_INTRVL   The number of pixels the work will be scrolled
--                      whenever the user selects an arrow button.  Note:
```

```
--                          The work will not be scrolled by these utilities
--                          but, this argument is required to calculate
--                          the interactive slidepositioning.
-- end formal parameters;


procedure UFM_Define_Static_Text (Editor:    out   UFM_Form_Editor;
                                  FORM_SRC:          in    UFM_Form_Src;
                                  DEST_TYPE:         in    SYS_DESTINATION_TYPE;
                                  DEST_ID  :         in    SYS_WINDOW_ELE_ID;
                                  PIXEL_COL:         in    SYS_WINDOW_COLUMN;
                                  PIXEL_ROW:         in    SYS_WINDOW_ROW;
                                  PIXEL_WIDTH:       in    SYS_WINDOW_COLUMN;
                                  PIXEL_HEIGHT:      in    SYS_WINDOW_ROW;
                                  TEXT:              in    SYS_TEXT_PTR;
                                  TEXT_ALIGNMENT:    in    SYS_TEXT_ALIGNMENT);
--
-- CPM description:    This procedure defines a static text area within a
--                     form.
--
--formal parameters:
--OUT    EDITOR              The ID attached to the static text area.  This ID is
--                           required for all interactions with the static text
--                           area.
--
--IN     FORM_SRC            The ID of the Source of the Form as output by
--                           UFM_INITIALIZE_FORM_FIELDS.
--
-- IN     DEST_TYPE          The type of the destination for the editor, where:
--                           SYS_WINDOW_DEST = Window
--                           SYS_PANEL_DEST = Panel
--
--IN     DEST_ID             ID attached to the destination that the editor is
--                           assigned to.  This is set to NULL when the
--                           destination is the RootWindow.
--
--IN     PIXEL_COL           Column number from within the form where the left
--                           side of the static text area shall be placed.
--                           Column 0 is at the left of the form.
--
--IN     PIXEL_ROW           Row number from within the form where the top side
--                           of the static text area shall be placed. Row 0 is
--                           at the top of the form.
--
--IN     PIXEL_WIDTH         The number of columns to be occupied by the static
--                           text area.
--
--IN     PIXEL_HEIGHT        The number of rows to be occupied by the static
--                           text area.
--
--IN     TEXT                Textual string to display in the button.
--
--IN     TEXT_ALIGNMENT      Alignment of the text within the static text area
--                           (CENTER_ALIGNED, LEFT_ALIGNED, RIGHT_ALIGNED,
--                            NO_ALIGNMENT)
-- end formal parameters;
```

```
procedure UFM_Define_String_Field (Editor:    out   UFM_Form_Editor;
                                   FORM_SRC:        in      UFM_Form_Src;
                                   DEST_TYPE:       in      SYS_DESTINATION_TYPE;
                                   DEST_ID  :       in      SYS_WINDOW_ELE_ID;
                                   PIXEL_COL:       in      SYS_WINDOW_COLUMN;
                                   PIXEL_ROW:       in      SYS_WINDOW_ROW;
                                   LABEL:           in      STRING;
                                   LABEL_POSITION:  in      SYS_LABEL_POSITION;
                                   STRING_VARIABLE: in out  STRING;
                                   MAX_CHARACTERS:  in      SYS_PRODUCT_LENGTH);
--
-- CPM description:  This procedure defines a string field within a form.
--
--formal parameters:
--OUT    EDITOR           The ID attached to the string field.  This ID is
--                       required for all interactions with the string field.
--
--IN     FORM_SRC         The ID of the Source of the Form as output by
--                       UFM_INITIALIZE_FORM_FIELDS.
--
--IN     DEST_TYPE        The type of the destination for the editor, where:
--                       SYS_WINDOW_DEST = Window
--                       SYS_PANEL_DEST = Panel
--
--IN     DEST_ID          ID attached to the destination that the editor is
--                       assigned to.  This is set to NULL when the
--                       destination is the RootWindow.
--
--IN     PIXEL_COL        Column number from within the form where the left
--                       side of the editor shall be placed. Column 0 is at
--                       left of the form.
--
--IN     PIXEL_ROW        Row number from within the form where the top side
--                       of the editor shall be placed. Row 0 is at the top
--                       of the form.
--
--IN     LABEL            The optional label before the string field.  This
--                       should be set to NULL if no label will be displayed.
--
--IN     LABEL_POSITION   Value specifying whether the optional label should
--                       be placed to the left or the right of the number
--                       field.  The two valid settings for this field are:
--                           0 = Left aligned
--                           1 = Right aligned
--                       If no label is specified, this parameter will
--                       be ignored by the editor.
--
--INOUT STRING_VARIABLE  The address of the variable to store the
--                       input string at.  This variable may be
--                       initialized to some string value, which would
--                       be displayed.  This must be a NULL terminated
--                       string.
--
--IN     MAX_CHARACTERS   The maximum number of characters which will
```

```
--                          be allowed to be entered into the field.
--
-- end formal parameters;


procedure UFM_DELETE_CHECKBOX_MENU (CHECKBOX_ID    :    in  UFM_Form_Editor);
--
-- CPM description: UFM_DELETE_CHECKBOX_MENU deletes a multiple selection
--                  menu that has been defined by UFM_DEFINE_CHECKBOX_MENU.
--
-- formal parameters
--IN    CHECKBOX_ID       The ID of the checkbox menu to delete.
-- end formal parameters;


procedure UFM_Delete_Form_Fields (Form_Src:   in out   UFM_Form_Src);

-- CPM description:  This procedure deletes a form's fields.
--
--formal parameters:
--IN    FORM_SRC          The ID of the Source of the Form as output by
--                        UFM_INITIALIZE_FORM_FIELDS.
--
--end formal parameters;


procedure UFM_DELETE_MEMO (EDITOR_ID    :     in   UFM_Form_Editor);
--
-- CPM description: UFM_DELETE_MEMO deletes a memo that is defined by
--                  UFM_DEFINE_MEMO.
--
-- formal parameters
--IN    EDITOR_ID         The ID of the editor to delete.
--
-- end formal parameters;


procedure UFM_DELETE_NUMBER_FIELD (
                              EDITOR_ID    :     in   UFM_Form_Editor);
--
-- CPM description: Deletes an numeric field from within a form that
--                  is defined by UFM_DEFINE_NUMBER_FIELD.
--
-- formal parameters
--IN    EDITOR_ID         The ID of the editor to delete.
--
-- end formal parameters;


procedure UFM_DELETE_PUSHBUTTON (PUSHBUTTON_ID    :   in   UFM_Form_Editor);
--
-- CPM description:  UFM_DELETE_PUSHBUTTON deletes a pushbutton editor that
--                   is defined by UFM_DEFINE_PUSHBUTTON.
--
-- formal parameters
--IN    PUSHBUTTON_ID     The ID of the pushbutton editor.
```

```
-- end formal parameters;


procedure UFM_DELETE_RADIOBUTTON_MENU (
                            RADIOBUTTON_ID   :   in   UFM_Form_Editor);
--
-- CPM description:  UFM_DELETE_RADIOBUTTON _MENU deletes a radiobutton menu
--                   that is defined by UFM _DEFINE_RADIOBUTTON_MENU.
--
-- formal parameters
--IN    RADIOBUTTON_ID   The ID of the radiobutton editor.
-- end formal parameters;


procedure UFM_DELETE_SCROLLBAR (SCROLLBAR_ID:     in   UFM_Form_Editor);
--
-- CPM description: UFM_DELETE_SCROLLBAR deletes a scrollbar that is defined
--                   by UFM_DEFINE_SCROLLBAR.
--
-- formal parameters
--IN    SCROLLBAR_ID     The ID of the scrollbar to delete.
--
-- end formal parameters;


procedure UFM_DELETE_STATIC_TEXT (STATIC_ID   :     in   UFM_Form_Editor);
--
-- CPM description: UFM_DELETE_STATIC_TEXT deletes static text that is
--                  defined in a form by UFM_DEFINE_STATIC_TEXT.
--
-- formal parameters
--IN    STATIC_ID        The ID of the static text to delete.
--
-- end formal parameters;


procedure UFM_DELETE_STRING_FIELD (
                              EDITOR_ID   :     in   UFM_Form_Editor);
--
-- CPM description: Deletes an string field editor that
--                  is defined by UFM_DEFINE_STRING_FIELD.
--
-- formal parameters
--IN    EDITOR_ID        The ID of the editor to delete.
--
-- end formal parameters;


procedure UFM_INITIALIZE_FORM_FIELDS (Form_Src :   out   UFM_Form_Src;
                                      DEST_TYPE:   in    SYS_DESTINATION_TYPE;
                                      DEST_ID:     in    SYS_WINDOW_ELE_ID);
--
-- CPM description:  This procedure initializes the form fields for a
--                   particular form.
--
--formal parameters:
```

```
        --OUT   Form_Src      The ID to be used in defining any form field(s)
        --                    within a particular form.
        --
The     --IN    DEST_TYPE         type of the destination for the editor, where:
        --                    SYS_WINDOW_DEST = Window
        --                    SYS_PANEL_DEST = Panel
        --
        --IN    DEST_ID       ID attached to the destination that the editor is
        --                    assigned to.
        --
        -- end formal parameters;


        procedure UFM_Input (INPUT_TYPE     :   in out   SYS_WINDOW_INPUT;
                             INPUT_WINDOW_ID:   in out   SYS_WINDOW_ELE_ID;
                             INPUT_VALUE    :   in out   SYS_WINDOW_VALUE;
                             INPUT_DATA     :   in out   SYS_WINDOW_DATA;
                             INPUT_EDITOR   :   out   UFM_Form_Editor);
        --
        -- CPM description:  Receives user input and internet messages and
        --                   determines if it is within the form.  If it is
        --                   it processes it and returns appropriate data to
        --                   the application software.
        --
        --formal parameters:
        --IN OUT   INPUT_TYPE       Type of input returned from the window system
        --
        --IN OUT   INPUT_WINDOW_ID  The id of the window which received input.
        --
        --IN OUT   INPUT_VALUE      The value of the input that accompanies the type
        --
        --IN OUT   INPUT_DATA       The value of the data that accompanies the type
        --                        and input values, if appropriate.
        --
        --OUT   INPUT_EDITOR     The ID of the editor associated with the input
        --                       if appropriate.
        --
        --  The following table lists the output returned to the application
        --  for its own processing:
        --                       window-
        -- input_type              id       type_code     data
        -- ----------            -------    ---------     --------
        -- 1 Exit                n/a        n/a           n/a
        -- 2 Menu                X          Menu_Id       menu_index
        -- 3 Checkbox            X          Editor_Id     Checkbox_index
        -- 4 Scrollbar           X          Editor_Id     SlidePosition
        -- 5 XrFILE              n/a        fd            n/a
        -- 6 ButtonWindow        X          n/a           n/a
        -- 7 Mouse Button        X          Button:       window_type:
        --   Pressed                          0 = R       1 = window
        --                                    1 = M       2 = panel
        --                                    2 = L       3 = button
        --                                                x, y
        -- 8 Mouse Button        X          Button:       window_type:
        --   Released                         0 = R       1 = window
        --                                    1 = M       2 = panel
```

```
--                                        2 = L          3 = button
--                                                       x, y
-- 9 Field Traversal   X                  Editor_id      editor_type:
--                                                       1 = string_field
--                                                       2 = Number_field
--                                                       type of traversal:
--                                                       1 - Next
--                                                       2 - Previous
--                                                       3 - Up
--                                                       4 - Down
--10 Exposure          X                  n/a            x, y, width, height
--11 Open Window       n/a                n/a            n/a
--12 Window Resized    n/a                n/a            n/a
--13 Close Window      n/a                n/a            n/a
--14 XrEEDIT_SAVE      X                  Editor_Id      bufferCount
--15 XrEEDIT_RESET     X                  Editor_Id      n/a
--16 Pushbutton        X                  Editor_Id      Button_index
--17 Radiobutton       X                  Editor_Id      Active_index,
--                                                       Previous_Index
--
--end formal parameters;

procedure UFM_MOVE_CHECKBOX_MENU (CHECKBOX_ID:     in out   UFM_Form_Editor;
                                  PIXEL_COL:       in   SYS_WINDOW_COLUMN;
                                  PIXEL_ROW:       in   SYS_WINDOW_ROW);

--
-- CPM description: Changes the location of a checkbox menu within a form.
--
-- formal parameters
--IN     CHECKBOX_ID        ID attached to the checkbox menu.
--
--IN     PIXEL_COL          Column number from within the form where the left
--                          side of the editor shall be placed. Column 0 is at
--                          left of the form.
--
--IN     PIXEL_ROW          Row number from within the form where the top side
--                          of the editor shall be placed. Row 0 is at the top
--                          of the form.
-- end formal parameters;

procedure UFM_MOVE_MEMO (EDITOR:            in out UFM_Form_Editor;
                         PIXEL_COL:         in   SYS_WINDOW_COLUMN;
                         PIXEL_ROW:         in   SYS_WINDOW_ROW);

--
-- CPM description: Changes the location of a memo within a form.
--
-- formal parameters
--IN     EDITOR             ID attached to the editor. This
--                          ID is required for all interactions with the editor.
--
--IN     PIXEL_COL          Column number from within the form where the left
--                          side of the editor shall be placed. Column 0 is at
```

```
--                          left of the form.
--
--IN     PIXEL_ROW           Row number from within the form where the top side
--                          of the editor shall be placed. Row 0 is at the top
--                          of the form.
-- end formal parameters;


procedure UFM_MOVE_NUMBER_FIELD (
                              EDITOR:          in out  UFM_Form_Editor;
                              PIXEL_COL:       in      SYS_WINDOW_COLUMN;
                              PIXEL_ROW:       in      SYS_WINDOW_ROW);

--
-- CPM description: Changes the location of a numeric field within a form.
--
-- formal parameters
--IN     EDITOR              ID attached to the editor. This
--                          ID is required for all interactions with the editor.
--
--IN     PIXEL_COL           Column number from within the form where the left
--                          side of the editor shall be placed. Column 0 is at
--                          left of the form.
--
--IN     PIXEL_ROW           Row number from within the form where the top side
--                          of the editor shall be placed. Row 0 is at the top
--                          of the form.
-- end formal parameters;


procedure UFM_MOVE_PUSHBUTTON (PUSHBUTTON_ID:     in out   UFM_Form_Editor;
                               PIXEL_COL:         in   SYS_WINDOW_COLUMN;
                               PIXEL_ROW:         in   SYS_WINDOW_ROW);

--
-- CPM description: Changes the location of a pushbutton editor within a form.
--
-- formal parameters
--IN     PUSHBUTTON_ID       ID attached to the pushbutton editor to move.
--
--IN     PIXEL_COL           Column number from within the form where the left
--                          side of the editor shall be placed. Column 0 is at
--                          left of the form.
--
--IN     PIXEL_ROW           Row number from within the form where the top side
--                          of the editor shall be placed. Row 0 is at the top
--                          of the form.
-- end formal parameters;


procedure UFM_MOVE_RADIOBUTTON_MENU (
                              RADIOBUTTON_ID:    in out   UFM_Form_Editor;
                              PIXEL_COL:         in       SYS_WINDOW_COLUMN;
                              PIXEL_ROW:         in       SYS_WINDOW_ROW);
```

```
--
-- CPM description: Changes the location of a radiobutton menu within a form.
--
-- formal parameters
--IN     RADIOBUTTON_ID    ID attached to the radiobutton editor to move.
--
--IN     PIXEL_COL         Column number from within the form where the left
--                         side of the editor shall be placed. Column 0 is at
--                         left of the form.
--
--IN     PIXEL_ROW         Row number from within the form where the top side
--                         of the editor shall be placed. Row 0 is at the top
--                         of the form.
-- end formal parameters;


procedure UFM_MOVE_SCROLLBAR (
                            SCROLLBAR_ID:      in out   UFM_Form_Editor;
                            PIXEL_COL:         in       SYS_WINDOW_COLUMN;
                            PIXEL_ROW:         in       SYS_WINDOW_ROW);

--
-- CPM description: Changes the location of a scrollbar within the form.
--
-- formal parameters
--IN     SCROLLBAR_ID      ID attached to the scrollbar.
--                         This ID is required for all interactions with the
--                         scrollbar.
--
--IN     PIXEL_COL         Column number from within the form where the left
--                         side of the scrollbar shall be placed. Column 0 is
--                         at left of the form.
--
--IN     PIXEL_ROW         Row number from within the form where the top side
--                         of the scrollbar shall be placed. Row 0 is at the
--                         top of the form.
-- end formal parameters;


procedure UFM_MOVE_STATIC_TEXT (
                            TEXT_ID:           in out   UFM_Form_Editor;
                            PIXEL_COL:         in       SYS_WINDOW_COLUMN;
                            PIXEL_ROW:         in       SYS_WINDOW_ROW);

--
-- CPM description: Changes the location of static text within a form.
--
-- formal parameters
--IN     EDITOR_ID         ID attached to the text. This
--                         ID is required for all interactions with the text.
--
--IN     PIXEL_COL         Column number from within the form where the left
--                         side of the text shall be placed. Column 0 is at
--                         left of the form.
--
```

```
--IN    PIXEL_ROW        Row number from within the form where the top side
--                       of the text shall be placed. Row 0 is at the top
--                       of the form.
-- end formal parameters;


procedure UFM_MOVE_STRING_FIELD (
                           EDITOR:        in  out   UFM_Form_Editor;
                           PIXEL_COL:     in        SYS_WINDOW_COLUMN;
                           PIXEL_ROW:     in        SYS_WINDOW_ROW);

--
-- CPM description: Changes the location of a string field editor within a
--                  form.
--
-- formal parameters
--IN    EDITOR           ID attached to the editor. This
--                       ID is required for all interactions with the editor.
--
--IN    PIXEL_COL        Column number from within the form where the left
--                       side of the editor shall be placed. Column 0 is at
--                       left of the form.
--
--IN    PIXEL_ROW        Row number from within the form where the top side
--                       of the editor shall be placed. Row 0 is at the top
--                       of the form.
-- end formal parameters;


procedure UFM_QUERY_CHECKBOX_SIZE (CHECKBOX_ID:     in   UFM_Form_Editor;
                                   PIXEL_COL:         out  SYS_WINDOW_COLUMN;
                                   PIXEL_ROW:         out  SYS_WINDOW_ROW);
--
-- CPM description: Returns the number of pixel columns and rows that
--                  a checkbox menu within a form occupies.
--
-- formal parameters
--IN    CHECKBOX_ID      ID attached to the menu.
--
--OUT   PIXEL_COL        Number of pixel columns in the menu.
--
--OUT   PIXEL_ROW        Number of pixel rows in the menu.
-- end formal parameters;

procedure UFM_QUERY_MEMO_SIZE (EDITOR_ID:          in   UFM_Form_Editor;
                               PIXEL_COL:          out  SYS_WINDOW_COLUMN;
                               PIXEL_ROW:          out  SYS_WINDOW_ROW);
--
-- CPM description: Returns the number of pixel columns and rows that
--                  a memo within the form occupies.
--
-- formal parameters
--IN    EDITOR_ID        ID attached to the memo.
--
--OUT   PIXEL_COL        Number of pixel columns in the memo.
--
```

```
--OUT   PIXEL_ROW       Number of pixel rows in the memo.
-- end formal parameters;


procedure UFM_QUERY_NUMBER_FIELD_SIZE (
                                    EDITOR_ID:          in   UFM_Form_Editor;
                                    PIXEL_COL:          out  SYS_WINDOW_COLUMN;
                                    PIXEL_ROW:          out  SYS_WINDOW_ROW);
--
-- CPM description: Returns the number of pixel columns and rows that
--                  an numeric field editor within the form occupies.
--
-- formal parameters
--IN    EDITOR_ID         ID attached to the editor.
--
--OUT   PIXEL_COL         Number of pixel columns in the editor.
--
--OUT   PIXEL_ROW         Number of pixel rows in the editor.
-- end formal parameters;


procedure UFM_QUERY_PUSHBUTTON_SIZE (PUSHBUTTON_ID:   in   UFM_Form_Editor;
                                    PIXEL_COL:          out  SYS_WINDOW_COLUMN;
                                    PIXEL_ROW:          out  SYS_WINDOW_ROW);
--
-- CPM description: Returns the number of pixel columns and rows that
--                  a pushbutton editor within the form occupies.
--
-- formal parameters
--IN    PUSHBUTTON_ID   ID attached to the editor.
--
--OUT   PIXEL_COL       Number of pixel columns in the editor.
--
--OUT   PIXEL_ROW       Number of pixel rows in the editor.
-- end formal parameters;

   p355X

procedure UFM_QUERY_RADIOBUTTON_SIZE (
                                    RADIOBUTTON_ID:     in   UFM_Form_Editor;
                                    PIXEL_COL:          out  SYS_WINDOW_COLUMN;
                                    PIXEL_ROW:          out  SYS_WINDOW_ROW);
--
-- CPM description: Returns the number of pixel columns and rows that
--                  a radiobutton menu within the form occupies.
--
-- formal parameters
--IN    RADIOBUTTON_ID  ID attached to the menu.
--
--OUT   PIXEL_COL       Number of pixel columns in the menu.
--
--OUT   PIXEL_ROW       Number of pixel rows in the menu.
-- end formal parameters;


procedure UFM_QUERY_SCROLLBAR_SIZE (
```

```
                                        SCROLLBAR_ID:       in   UFM_Form_Editor;
                                        PIXEL_COL:          out  SYS_WINDOW_COLUMN;
                                        PIXEL_ROW:          out  SYS_WINDOW_ROW);
--
-- CPM description: Returns the number of pixel columns and rows that
--                  a scrollbar within the form occupies.
--
-- formal parameters
--IN    SCROLLBAR_ID     ID attached to the scrollbar.
--
--OUT   PIXEL_COL        Number of pixel columns in the scrollbar.
--
--OUT   PIXEL_ROW        Number of pixel rows in the scrollbar.
-- end formal parameters;


procedure UFM_QUERY_STRING_FIELD_SIZE (
                                        EDITOR_ID:          in   UFM_Form_Editor;
                                        PIXEL_COL:          out  SYS_WINDOW_COLUMN;
                                        PIXEL_ROW:          out  SYS_WINDOW_ROW);
--
-- CPM description: Returns the number of pixel columns and rows that
--                  a string field editor within the form occupies.
--
-- formal parameters
--IN    EDITOR_ID        ID attached to the editor.
--
--OUT   PIXEL_COL        Number of pixel columns in the editor.
--
--OUT   PIXEL_ROW        Number of pixel rows in the editor.
-- end formal parameters;


procedure UFM_RESIZE_CHECKBOX_MENU(CHECKBOX_ID:     in   UFM_Form_Editor;
                                   PIXE_COL:        in   SYS_WINDOW_COLUMN;
                                   PIXEL_ROW:       in   SYS_WINDOW_ROW;
                                   PIXEL_WIDTH:     in   SYS_WINDOW_COLUMN;
                                   PIXEL_HEIGHT:    in   SYS_WINDOW_ROW);

--
-- CPM description: Changes the size of a checkbox menu within the form.
--
-- formal parameters
--IN    CHECKBOX_ID      ID of the menu.
--
--IN    PIXEL_COL        Column number from within the form where the left
--                       side of the menu shall be placed. Column 0 is at
--                       left of the form.
--
--IN    PIXEL_ROW        Row number from within the form where the top side
--                       of the menu shall be placed. Row 0 is at the top
--                       of the form.
--
--IN    PIXEL_WIDTH      The number of columns to be occupied by the menu.
--
--IN    PIXEL_HEIGHT     The number of rows to be occupied by the menu.
```

```
-- end formal parameters;

procedure UFM_RESIZE_MEMO (EDITOR:            in out   UFM_Form_Editor;
                           PIXEL_COL:         in       SYS_WINDOW_COLUMN;
                           PIXEL_ROW:         in       SYS_WINDOW_ROW;
                           PIXEL_WIDTH:       in       SYS_WINDOW_COLUMN;
                           PIXEL_HEIGHT:      in       SYS_WINDOW_ROW);

--
-- CPM description: Changes the size of a memo within a form.
--
-- formal parameters
--IN    EDITOR           ID of the memo.
--
--IN    PIXEL_COL        Column number from within the form where the left
--                       side of the memo shall be placed. Column 0 is at
--                       left of the form.
--
--IN    PIXEL_ROW        Row number from within the form where the top side
--                       of the memo shall be placed. Row 0 is at the top
--                       of the form.
--
--IN    PIXEL_WIDTH      The number of columns to be occupied by the memo.
--
--IN    PIXEL_HEIGHT     The number of rows to be occupied by the memo.
-- end formal parameters;


procedure UFM_RESIZE_NUMBER_FIELD (
                           EDITOR:            in out   UFM_Form_Editor;
                           PIXEL_COL:         in       SYS_WINDOW_COLUMN;
                           PIXEL_ROW:         in       SYS_WINDOW_ROW;
                           PIXEL_WIDTH:       in       SYS_WINDOW_COLUMN;
                           PIXEL_HEIGHT:      in       SYS_WINDOW_ROW);

--
-- CPM description: Changes the size of a numeric field editor within a form.
--
-- formal parameters
--IN    EDITOR           ID of the editor.
--
--IN    PIXEL_COL        Column number from within the form where the left
--                       side of the editor shall be placed. Column 0 is at
--                       left of the form.
--
--IN    PIXEL_ROW        Row number from within the form where the top side
--                       of the editor shall be placed. Row 0 is at the top
--                       of the form.
--
--IN    PIXEL_WIDTH      The number of columns to be occupied by the editor.
--
--IN    PIXEL_HEIGHT     The number of rows to be occupied by the editor.
-- end formal parameters;
```

```
procedure UFM_RESIZE_PUSHBUTTON (
                              PUSHBUTTON_ID:      in   UFM_Form_Editor;
                              PIXEL_COL:          in   SYS_WINDOW_COLUMN;
                              PIXEL_ROW:          in   SYS_WINDOW_ROW;
                              PIXEL_WIDTH:        in   SYS_WINDOW_COLUMN;
                              PIXEL_HEIGHT:       in   SYS_WINDOW_ROW);

--
-- CPM description: Changes the size of a pushbutton editor within a form.
--
-- formal parameters
--IN    PUSHBUTTON_ID     ID of the pushbutton editor.
--
--IN    PIXEL_COL         Column number from within the form where the left
--                        side of the editor shall be placed. Column 0 is at
--                        left of the form.
--
--IN    PIXEL_ROW         Row number from within the form where the top side
--                        of the editor shall be placed. Row 0 is at the top
--                        of the form.
--
--IN    PIXEL_WIDTH       The number of columns to be occupied by the editor.
--
--IN    PIXEL_HEIGHT      The number of rows to be occupied by the editor.
-- end formal parameters;


procedure UFM_RESIZE_RADIOBUTTON_MENU (
                              RADIOBUTTON_ID:     in   UFM_Form_Editor;
                              PIXEL_COL:          in   SYS_WINDOW_COLUMN;
                              PIXEL_ROW:          in   SYS_WINDOW_ROW;
                              PIXEL_WIDTH:        in   SYS_WINDOW_COLUMN;
                              PIXEL_HEIGHT:       in   SYS_WINDOW_ROW);

--
-- CPM description: Changes the size of a radiobutton menu in a form.
--
-- formal parameters
--IN    RADIOBUTTON_ID    ID of the radiobutton menu.
--
--IN    PIXEL_COL         Column number from within the form where the left
--                        side of the menu shall be placed. Column 0 is at
--                        left of the form.
--
--IN    PIXEL_ROW         Row number from within the form where the top side
--                        of the menu shall be placed. Row 0 is at the top
--                        of the form.
--
--IN    PIXEL_WIDTH       The number of columns to be occupied by the menu.
--
--IN    PIXEL_HEIGHT      The number of rows to be occupied by the menu.
-- end formal parameters;


procedure UFM_RESIZE_STRING_FIELD (
                              EDITOR:             in out  UFM_Form_Editor;
```

```
                                    PIXEL_COL:          in      SYS_WINDOW_COLUMN;
                                    PIXEL_ROW:          in      SYS_WINDOW_ROW;
                                    PIXEL_WIDTH:        in      SYS_WINDOW_COLUMN;
                                    PIXEL_HEIGHT:       in      SYS_WINDOW_ROW);

    --
    -- CPM description: Changes the size of a string field editor in a form.
    --
    -- formal parameters
    --IN     EDITOR              ID of the editor.
    --
    --IN     PIXEL_COL           Column number from within the form where the left
    --                           side of the editor shall be placed. Column 0 is at
    --                           left of the form.
    --
    --IN     PIXEL_ROW           Row number from within the form where the top side
    --                           of the editor shall be placed. Row 0 is at the top
    --                           of the form.
    --
    --IN     PIXEL_WIDTH         The number of columns to be occupied by the editor.
    --
    --IN     PIXEL_HEIGHT        The number of rows to be occupied by the editor.
    -- end formal parameters;


private

   type UFM_Node_Type is (Row_Head, Row_Element);

   type UFM_Form_Node (Code: UFM_Node_Type);

   type UFM_Form_Editor is access UFM_Form_Node;

   type UFM_Form_Matrix_Head is
      record
         Dest_Type:    SYS_DESTINATION_TYPE;
         Dest_ID:      SYS_WINDOW_ELE_ID;
         Head:         UFM_Form_Editor;
   end record;

   type UFM_Form_Src is access UFM_Form_Matrix_Head;

   type UFM_Form_Field is (Button, CheckBox, Map, PushButton, RadioButton,
                           ScrollBar, Static_Text, Memo, Numeric, Alpha_String);
-- NOTE:
--    the traversal editors must be listed at end between Memo and Alpha_String
--    See the package body for type UFM_Traversal_Fields

   type UFM_Form_Node (Code: UFM_Node_Type) is
      record
         Form_Src:       UFM_Form_Src;
         Dest_Type:      SYS_Destination_Type;
         Dest_ID:        SYS_WINDOW_ELE_ID;
         Previous:       UFM_Form_Editor;
         Next_Editor:    UFM_Form_Editor;
         Field_Ptr:      SYS_Window_Ele_Id;
```

```
            Field_Type:     UFM_Form_Field;
            Field_Rect:     SYS_Rectangle;
            case Code is
               when Row_Head =>
                  Next_Row:    UFM_Form_Editor;
                  Prev_Row:    UFM_Form_Editor;
               when Row_Element =>
                  null;
            end case;
         end record;

end UFM_Form_Fields;
```

```
--cpc package specification name:  UFM_FORM_MANAGER
--
--cpc description:   UFM_FORM_MANAGER is the utility package for defining and
--                   managing forms.
--
--cpc design notes:
--    This package raises SYS_UFM_EXCEPTION when a system error is encountered.
--
--cpc package author: Laura McClanahan
--                   Science Applications International Corporation
--                   424 Delaware, Suite C3
--                   Leavenworth, KS 66048
--
with CM_System;          use CM_System;
with Map_System;         use Map_System;
with OBS_System;         use OBS_System;
with System_Package;     use System_Package;
with Unit_System;        use Unit_System;
package UFM_Form_Manager is

   subtype UFM_Form_Buffer is SYS_TEXT_PTR;
   subtype UFM_ID is SYS_TEXT_PTR;
   type UFM_Field_Type is (MEMO_TEXT, NUMERIC_FIELD, STRING_FIELD,
                           RADIO_BUTTON, CHECKLIST, BUTTON_WALK, FORM_WALK,
                           MULTIPLE_SELECT_MENU, SINGLE_SELECT_MENU,
                           DIGITAL_MAP, PUSH_BUTTON, STATIC_TEXT);
   type UFM_Editor_Status (Field: UFM_Field_Type) is
      record
         case Field is
           when MEMO_TEXT =>
              CONTENT    :   SYS_TEXT_PTR;
              SIZE       :   SYS_PRODUCT_LENGTH;
           when NUMERIC_FIELD =>
              N_VALUE :   SYS_TEXT_PTR;
           when STRING_FIELD =>
              S_VALUE :   SYS_TEXT_PTR;
           when RADIO_BUTTON =>
              STATE :   BOOLEAN;
           when CHECKLIST =>
              STATES :   SYS_MENU_BUTTON_STATUS_PTR;
           when BUTTON_WALK | FORM_WALK =>
              RET_VAL:   SYS_WALKING_CELL_VALUE;
           when MULTIPLE_SELECT_MENU =>
              Status:  SYS_MENU_BUTTON_STATUS_PTR;
           when SINGLE_SELECT_MENU =>
              Default : SYS_MENU_BUTTON_INDEX;
           when DIGITAL_MAP =>
              WIDTH          :   SYS_WINDOW_COLUMN;
              HEIGHT         :   SYS_WINDOW_ROW;
              DATE_TIME      :   SYS_DATE_TIME;
              OPLAN          :   SYS_OPPLAN;
              MAP_OPTIONS    :   MAP_MAP_OPTIONS;
              BLUEFOR_UNITS  :   UNIT_OPTIONS;
              CM_OPTIONS     :   CM_CNTRL_MSR_OPTIONS;
              OBS_OPTIONS    :   OBS_OBSTACLE_OPTIONS;
```

```
            OPFOR_UNITS    :    UNIT_OPTIONS;
         when PUSH_BUTTON =>
            PUSHED:    SYS_MENU_BUTTON_VALUES;
         when STATIC_TEXT =>
            NULL;    -- The UID field is used to store the static text string
      end case;
end record;

type UFM_Editor_Stat_Ptr is access UFM_Editor_Status;


procedure UFM_Read_Form_File (File_Name:       in    String;
                              Form_Buffer:    out    UFM_Form_Buffer);
--
-- CPM description:  UFM_Read_Form_File reads in a file designated by the
--                   input environment variable and puts it into a form
--                   ascii buffer for either validation or defining a form.
--
-- formal parameters:
--IN      File_Name       The name of the data file containing a form
--                        description.
--
--OUT     Form_Buffer     The buffer containing the form description as read
--                        from the data file designated by the Env_Variable.
-- end formal parameters;


procedure UFM_Validate_Form (Form_Description:    in    UFM_Form_Buffer);
--
-- CPM description:  UFM_Define_Form defines a form from an ASCII buffer.
--
-- formal parameters:
--IN    Form_Description    The description of the form to be created.
--
-- end formal parameters;


task type Form_Manager_Task is
   entry Define_Form (
               Pixel_Width:          in     SYS_WINDOW_COLUMN;
               Pixel_Height:         in     SYS_WINDOW_ROW;
               Form_Description:     in     UFM_Form_Buffer;
               Process_ID:           in     SYS_EDDIC_PROCESSES;
               Form_ID:              out    SYS_WINDOW_ELE_ID;
               Map_Window:           in     BOOLEAN := FALSE;
               Parent_Window:        in     SYS_WINDOW_ELE_ID := SYS_ROOT_WINDOW;
               Parent_Window_X:      in     SYS_WINDOW_COLUMN := 0;
               Parent_Window_Y:      in     SYS_WINDOW_ROW := 0);

   --
   -- CPM description:  UFM_Define_Form defines a form from an ASCII buffer.
   --
   -- formal parameters:
   --IN    Pixel_Width          The pixel width of the form.
   --
   --IN    Pixel_Height         The pixel height of the form.
```

```
--
--IN   Form_Description    The description of the form to be created.
--
--IN   Process_ID          The ID of the calling process.
--
--OUT  Form_ID             The id given the form object.
--
--IN   MAP_WINDOW          The logical indicating whether the form
--                         window should be mapped upon creation or not.
--                         If it is not, the application can make the
--                         form window be visible later via a call
--                         to UWN_MAP_WINDOW.
--
--IN   Parent_Window       The ID of the window to which the form
--                         manager window will be a subwindow to.  The
--                         default is the root window thus making the
--                         form a popup window.
--
--IN   Parent_Window_X     The pixel column of the parent window where the
--                         form window's origin will be placed.  The
--                         default is zero, where the window may be moved
--                         via UWN_MOVE_WINDOW.
--
--IN   Parent_Window_Y     The pixel row of the parent window where the
--                         form window's origin will be placed. The
--                         default is zero.
--
-- end formal parameters;


entry Delete_Form;

-- CPM description:  This entry deletes a form's editors.
--
--formal parameters
-- NONE
--end formal parameters;


entry Process_Input (INPUT_TYPE   :  in   SYS_WINDOW_INPUT;
                     INPUT_WINDOW_ID: in  SYS_WINDOW_ELE_ID;
                     INPUT_VALUE    :  in  SYS_WINDOW_VALUE;
                     INPUT_DATA     :  in  SYS_WINDOW_DATA;
                     Input_Processed: out Boolean;
                     Field_ID       :  out UFM_ID;
                     Field_Type     :  out UFM_Field_Type);

--
-- CPM description: Receives user input and internet messages and
--                  determines if it is within the form.  If it is
--                  it processes it and returns appropriate data to
--                  the application software.
--
-- formal parameters
--IN INPUT_TYPE       Type of input returned from the window
--                    system
```

```
--
--IN INPUT_WINDOW_ID  The id of the window which received the
--                    input.
--
--IN INPUT_VALUE      The value of the input that accompanies
--                    the type
--
--IN INPUT_DATA       The value of the data that accompanies
--                    the type and input values, if appropriate.
--
--OUT Input_Processed  A boolean flag indicating whether the
--                    input was processed.
--
--OUT Field_ID         The ID of the field which received input.
--                    This ID may be NULL if it is not
--                    input within the form's fields but still
--                    processed by this task.
--
--OUT Field_Type       The type of field which received input.
--
-- end formal parameters;


entry Query_Form_Description (Form_Buffer:   out UFM_Form_Buffer);
      --
      -- CPM description: Returns a buffer containing the editor
      --                  description section of the form.
      --
      -- formal parameters
      --OUT   Form_Buffer     The ASCII buffer contains the editor
      --                      description section of the form.
      --
      -- end formal parameters;


entry Query_Form_Editor_Status (Editor_ID:   in   UFM_ID;
                                Editor_Stat:   in   UFM_Editor_Stat_Ptr);
--
-- CPM description: Returns the status of a form's editor.
--
-- formal parameters
--IN    Editor_ID       The ID of the editor whose status is being
--                      queried.
--
--OUT   Editor_Stat     The status of the editor.
--
-- end formal parameters;


entry Query_Form_Size (Pixel_Width:      out   SYS_WINDOW_COLUMN;
                       Pixel_Height:     out   SYS_WINDOW_ROW);
--
-- CPM description: Returns the number of pixel columns and rows that
--                  a form occupies.  Note this is not the number of pixel
--                  columns and rows necessarily visible to the user but
--                  the total required if it was entirely visible.
```

```
      --
      --OUT   Pixel_Width     Number of pixel columns in the form.
      --
      --OUT   Pixel_Height    Number of pixel rows in the form.
      --
      -- end formal parameters;

      entry Resize_Form (Pixel_Col:          in    SYS_WINDOW_COLUMN;
                         Pixel_Row:          in    SYS_WINDOW_ROW;
                         Pixel_Width:        in    SYS_WINDOW_COLUMN;
                         Pixel_Height:       in    SYS_WINDOW_ROW);

      --
      -- CPM description: Changes the size of a form.
      --
      -- formal parameters
      --IN    Pixel_Col         Column number from within the window where the
      --                        left side of the form shall be placed. Column 0
      --                        is at left of the window.
      --
      --IN    Pixel_Row         Row number from within the window where the top
      --                        side of the form shall be placed. Row 0 is at the
      --                        top of the window.
      --
      --IN    Pixel_Width       The number of columns to be occupied by the form.
      --
      --IN    Pixel_Height      The number of rows to be occupied by the form.
      --
      -- end formal parameters;

      entry Terminate_Form_Task;
      --
      -- CPM description:  Terminates the form task.
      --
      -- formal parameters
      -- NONE
      -- end formal parameters;

   end;

end UFM_Form_Manager;
```

UIN Utility Package Specifications

The following package specification is contained in the Internet communications function:

UIN_INTERNET_COMMUNICATIONS

```
--CPC package specification name:
--     UIN_INTERNET_COMMUNICATIONS
--
--CPC description:
--     UIN_INTERNET_COMMUNICATIONS CPC is set of Utility communications.
--     primitives, written in the "Ada" programming language, which allows
--     processes to communicate with each other using an InterNet protocol.
--     These primitives work both within one processor and over an ethernet
--     network.
--
--CPC design notes:
--     1.)  This package can raise the following exceptions:
--          SYS_UIN_EXCEPTION.
--
--CPC package author:
--     Bruce J. Packard
--     Science Applications International Corporation (SAIC)
--     424 Delaware, Suite C-3
--     Leavenworth, KS  66048   (913) 651-7925
--

with SYSTEM_PACKAGE;  use SYSTEM_PACKAGE;
with MSG_MESSAGE;     use MSG_MESSAGE;

package UIN_INTERNET_COMMUNICATIONS is

   -- List of client socket numbers.
   type UIN_CLIENT_NUMB_ARRAY is array (SYS_CLIENT) of SYS_CLIENT;
   type UIN_CLIENT_NUMB_PTR   is access UIN_CLIENT_NUMB_ARRAY;

   -- List of client ID's.
   type UIN_CLIENT_ID_ARRAY is array (SYS_CLIENT) of SYS_EDDIC_PROCESSES;
   type UIN_CLIENT_ID_PTR   is access UIN_CLIENT_ID_ARRAY;

   -- Host and Service name definition.
   subtype UIN_HOST_TYPE is string (SYS_ENV_STRING);
   subtype UIN_SERV_TYPE is string (SYS_ENV_STRING);

   -- Peek and no peek flags.
   UIN_PEEK              :  BOOLEAN     :=  true;
   UIN_NO_PEEK           :  BOOLEAN     :=  false;

-- ##########################################################################
   procedure UIN_CLIENT_CONNECT_SERVER (HOST_ID       : in   UIN_HOST_TYPE;
                                        SERVICE_ID    : in   UIN_SERV_TYPE;
                                        MSTR_SOCK_NUM : out  SYS_CLIENT);
   --
   --CPM description:
   --     This module allows a Client (user process) to Connect to the
   --     InterNet master (Server) socket, returning the master socket number.
   --
   --CPM design notes:
   --     1.)  None.
   --
   --formal parameters
```

```
   --IN     HOST_ID       - A string which the environment equates to the
   --                       name (Id) of the Host (server) machine.
   --IN     SERVICE_ID    - A string which the environment equates to the
   --                       Service Id (INET port number).
   --OUT    MSTR_SOCK_NUM - A pointer to the server (Master) Socket Number.
   --end formal parameters;

-- ##############################################################################
   procedure UIN_CLOSE_SOCKET (CSN_INDEX       : in      SYS_CLIENT;
                               CLIENT_SOCK_NUM : in      UIN_CLIENT_NUMB_PTR;
                               CLIENT_DISP_NUM : in      UIN_CLIENT_ID_PTR;
                               NUM_CLIENTS     : in out  SYS_CLIENT);
   --
   --CPM description:
   --    This module Closes the specified Internet client Socket and remove
   --    it from the list of client sockets.
   --
   --CPM design notes:
   --    1.)  None.
   --
   --formal parameters
   --IN     CSN_INDEX       - The  array Index of the Client Socket Number
   --                         being closed.
   --I/O    CLIENT_SOCK_NUM - The list of Client Socket Numbers.
   --I/O    CLIENT_DISP_NUM - The list of Client Display Numbers.  This is
   --                         machine number of the corresponding client
   --                         socket number.
   --I/O    NUM_CLIENTS     - The pointer to the actual Number of Client
   --                         sockets currently in the system.
   --end formal parameters;

-- ##############################################################################
   procedure UIN_ESTABLISH_SERVER (HOST_ID       : in      UIN_HOST_TYPE;
                                   SERVICE_ID    : in      UIN_SERV_TYPE;
                                   MSTR_SOCK_NUM : in out  SYS_CLIENT);
   --
   --CPM description:
   --    This module sets up and opens an InterNet Server returning the
   --    master socket number.
   --
   --CPM design notes:
   --    1.)  None.
   --
   --formal parameters
   --IN     HOST_ID       - A string which the environment equates to the
   --                       name (Id) of the Host (server) machine.
   --IN     SERVICE_ID    - A string which the environment equates to the
   --                       Service Id (INET port number).
   --OUT    MSTR_SOCK_NUM - A pointer to the server (Master) Socket Number.
   --end formal parameters;

-- ##############################################################################
   procedure UIN_FLUSH_MSG (SOCK_NUM  : in   SYS_CLIENT;
                            FLUSH_LEN : out  MSG_MESSAGE_LEN);
   --
   --CPM description:
```

```
   --    This module Flushes a Message from the InterNet buffer system.
   --
   --CPM design notes:
   --    1.)  None.
   --
   --formal parameters
   --IN     SOCK_NUM     - The Socket Number to read from.
   --OUT    FLUSH_LEN    - The length of the message flushed if it worked, and
   --                      the error number if the flush failed.
   --end formal parameters;

-- ##############################################################################
   procedure UIN_RECV_MSG (PEEK_FLAG     : in      BOOLEAN;
                           FROM_SOCK_NUM : in      SYS_CLIENT;
                           MSG_LEN       : in out  MSG_MESSAGE_LEN;
                           MSG           : in      MSG_MESSAGE_POINT);
   --
   --CPM description:
   --    This module sneaks a peek at, or Receives a Message which is being
   --    buffered in the InterNet system.
   --
   --CPM design notes:
   --    1.)  None.
   --
   --formal parameters
   --IN     PEEK_FLAG     - A Flag which tells this module whether to actually
   --                      receive the message or just "peek" at the first
   --                      "msg_len" bytes.
   --                     = TRUE  - just peek at the message.
   --                     = FALSE - read the entire message.
   --IN     FROM_SOCK_NUM - The Socket Number to read From.
   --I/O    MSG_LEN       - The number of bytes to read, or peek at, on the way
   --                      in and the number of bytes received, or the error
   --                      number if the received failed, on the way out.
   --OUT    MSG           - The Message received.
   --end formal parameters;

-- ##############################################################################
   procedure UIN_SEND_MSG (TO_SOCK_NUM : in      SYS_CLIENT;
                           MSG         : in      MSG_MESSAGE_POINT;
                           MSG_LEN     : in out  MSG_MESSAGE_LEN);
   --
   --CPM description:
   --    This module Sends a Message across the InterNet system.
   --
   --CPM design notes:
   --    1.)  None.
   --
   --formal parameters
   --IN     TO_SOCK_NUM - The Socket Number to write To.
   --IN     MSG         - The Message to write.
   --I/O    MSG_LEN     - The number of bytes to write on the way in and the
   --                    number of bytes written, or the error number if the
   --                    received failed, on the way out.
   --end formal parameters;
```

```
-- ##############################################################################
   procedure UIN_SERVER_CONNECT_CLIENT (
                MSTR_SOCK_NUM    : in      SYS_CLIENT;
                MAX_CLIENTS      : in      SYS_CLIENT;
                NUM_CLIENTS      : in out  SYS_CLIENT;
                CLIENT_SOCK_NUM  : in      UIN_CLIENT_NUMB_PTR;
                CLIENT_DISP_NUM  : in      UIN_CLIENT_ID_PTR);
   --
   --CPM description:
   --    This module allows the Server to Connect (accept) a client socket,
   --    returning the socket number.
   --
   --CPM design notes:
   --    1.)  None.
   --
   --formal parameters
   --IN      MSTR_SOCK_NUM    - The server (Master) Socket Number.
   --IN      MAX_CLIENTS      - The Maximum number of Clients allowed in the
   --                           system.
   --I/O     NUM_CLIENTS      - A pointer to the actual Number of Client sockets
   --                           currently in the system.
   --OUT     CLIENT_SOCK_NUM  - The list of Client Socket Numbers.
   --OUT     CLIENT_DISP_NUM  - The list of Display Numbers for each Client,
   --                           related to the corresponding "client_sock_num".
   --end formal parameters;

-- ##############################################################################
   procedure UIN_SERVER_WAIT (MSTR_SOCK_NUM     : in   SYS_CLIENT;
                              NUM_CLIENTS       : in   SYS_CLIENT;
                              CLIENT_SOCK_NUM   : in   UIN_CLIENT_NUMB_PTR;
                              CALLING_SOCK_NUM  : out  SYS_CLIENT;
                              SOCKET_INDEX      : out  SYS_CLIENT);
   --
   --CPM description:
   --    This module causes the Server program to Wait for a response from
   --    one of the clients on the InterNet.
   --
   --CPM design notes:
   --    1.)  None.
   --
   --formal parameters
   --IN      MSTR_SOCK_NUM     - The server (Master) Socket Number.
   --IN      NUM_CLIENTS       - The actual Number of Client sockets currently
   --                            in the system.
   --IN      CLIENT_SOCK_NUM   - The list of Client Socket Numbers.
   --OUT     CALLING_SOCK_NUM  - A pointer to the Number of the Socket who just
   --                            Called the server.
   --OUT     SOCKET_INDEX      - A pointer to the Client Socket Number array
   --                            Index, for the client who just called.
   --end formal parameters;

end UIN_INTERNET_COMMUNICATIONS;
```

UIW Utility Package Specifications

The following package specifications are contained in the color image display function:

```
UIW_GENERIC
UIW_IMAGE_WINDOW
```

```
--CPC package specification name:
--     UIW_GENERIC
--
--CPC description:
--     UIW_GENERIC CPC is a set of Utility color graphics primitives, written in
--     the "Ada" programming language, which allows programs to perform color
--     Imaging functions within the X Windows system.
--
--CPC design notes:
--     1.)  This package must be instantiated with its generic formal parameters.
--     2.)  This package can raise the following exceptions:
--          SYS_UIW_EXCEPTION.
--
--CPC package author:
--     Bruce J. Packard
--     Science Applications International Corporation (SAIC)
--     424 Delaware, Suite C-3
--     Leavenworth, KS  66048   (913) 651-7925
--
with SYSTEM_PACKAGE;              use SYSTEM_PACKAGE;

generic

   -- Types of buffers that can be used by the UUX I/O utilities
   type UIW_IMAGE_BUFFER is private;
   type UIW_IMAGE_POINTER is access UIW_IMAGE_BUFFER;

package UIW_GENERIC is

-- ##########################################################################
   procedure UIW_CREATE_PIXMAP (SIZE_X    : in   SYS_WINDOW_COLUMN;
                                SIZE_Y    : in   SYS_WINDOW_ROW;
                                BIT_IMAGE : in   UIW_IMAGE_POINTER;
                                COLOR     : in   SYS_COLOR;
                                PIXMAP_ID : out  SYS_WINDOW_ELE_ID);
   --
   --CPM description:
   --     This module Creates a Pixmap out of bitmapped data.
   --
   --CPM design notes:
   --     1.)  The bit image must be in memory order (Bits 0 - 15) for each 16
   --     bit word.
   --     2.)  The pixmap is displayed and erased with UIW_DISPLAY_BIT_IMAGE.
   --     3.)  The pixmap must be removed from memory with UIW_FREE_PIXMAP, when
   --     the pixel image is no longer required (see UIW_FREE_PIXMAP).
   --
   --formal parameters
   --IN      SIZE_X    - The Size of the image in the X direction.
   --IN      SIZE_Y    - The Size of the image in the Y direction.
   --IN      BIT_IMAGE - The Bit Image to transform.  The image is organized in
   --                    rows from the top to the bottom.  Each row contains
   --                    "SIZE_X" bits and there are "SIZE_Y" rows in the image.
   --IN      COLOR     - The index into the color lookup table for the Color
   --                    assigned to the on bits in this pixmap.
   --OUT     PIXMAP_ID - The Id assigned to this Pixmap.  This id is required
   --                    for displaying and freeing the pixmap.
```

```
    --end formal parameters;
    --

-- ##############################################################################
    procedure UIW_DISPLAY_IMAGE (WINDOW_ID       : in  SYS_WINDOW_ELE_ID;
                                 BITS_DEEP       : in  SYS_BITS_DEEP;
                                 SUB_ADD_FLAG    : in  BOOLEAN;
                                 DISPLAY_FUNTION : in  SYS_COLOR_ACTION;
                                 PIXEL_UL_X      : in  SYS_WINDOW_COLUMN;
                                 PIXEL_UL_Y      : in  SYS_WINDOW_ROW;
                                 SIZE_X          : in  SYS_WINDOW_COLUMN;
                                 SIZE_Y          : in  SYS_WINDOW_ROW;
                                 IMAGE           : in  UIW_IMAGE_POINTER;
                                 PLANE_MASK      : in  SYS_COLOR_MASK);
    --
    --CPM description:
    --    This module Displays or erases a raster image in the specified planes.
    --
    --CPM design notes:
    --    1.)  Image depths (BITS_DEEP) of 1 should use UIW_DISPLAY_BIT_IMAGE.
    --    2.)  The only image depth (BITS_DEEP) currently supported is 8.
    --
    --formal parameters
    --IN     WINDOW_ID       - The Id of the Window to display the image in.  It
    --                         can be obtained by calling UWM_QUERY_WINDOW_ID.
    --IN     BITS_DEEP       - The Depth of each pixel value in the raster image.
    --                        = 8 - Byte image.
    --IN     SUB_ADD_FLAG    - Image Subtraction or Addition Flag.  During
    --                         subtraction, the bits set in the raster image
    --                         shall be subtracted out of the selected planes.
    --                         During addition, the bits set in the raster image
    --                         shall be added into the selected planes.
    --                        = 0 - Subtract the image.
    --                        = 1 - Add the image.
    --IN     DISPLAY_FUNTION - The means of adding/subtracting the image to the
    --                         displayed image (and, or, copy...).
    --IN     PIXEL_UL_X      - The window X coordinate of the Upper Left corner
    --                         of the image.
    --IN     PIXEL_UL_Y      - The window Y coordinate of the Upper Left corner
    --                         of the image.
    --IN     SIZE_X          - The Size of the image in the X direction.
    --IN     SIZE_Y          - The Size of the image in the Y direction.
    --IN     IMAGE           - The raster Image to display/erase.  The image is
    --                         organized in rows from the top to the bottom.
    --                         Each row contains "SIZE_X" elements and there are
    --                         "SIZE_Y" rows in the image.  Each element of the
    --                         image occupies "BITS_DEEP" bits.
    --IN     PLANE_MASK      - A bit map representation of the Planes to be
    --                         affected by the image.  Value can be obtained from
    --                         "UIW_PLANE_MASK".
    --end formal parameters;
    --

-- ##############################################################################
    procedure UIW_16BIT_SWAP (WIDTH      : in  SYS_WINDOW_COLUMN;
                              HEIGHT     : in  SYS_WINDOW_ROW;
```

```
                                BIT_IMAGE : in  UIW_IMAGE_POINTER);
    --
    --CPM description:
    --    This module Swaps the Bits of 16 bit words, an order X Windows happens
    --    to prefer.
    --
    --CPM design notes:
    --    1.)  bit 0 -> bit 15;   bit 15 -> bit 0
    --         bit 1 -> bit 14;   bit 14 -> bit 1 ...
    --
    --formal parameters
    --IN      WIDTH     - The Width of the bit image buffer.
    --IN      HEIGHT    - The Height of the bit image buffer.
    --I/O     BIT_IMAGE - Buffer containing the bit image.
    --end formal parameters;
    --

end UIW_GENERIC;
```

```
--CPC package specification name:
--     UIW_IMAGE_WINDOW
--
--CPC description:
--     UIW_IMAGE_WINDOW CPC is a set of color graphics primitives, written in
--     the "Ada" programming language, which allows programs to perform color
--     imaging functions within the X Windows system.
--
--CPC design notes:
--     1.)  This package can raise the following exceptions:
--          SYS_UIW_EXCEPTION.
--
--CPC package author:
--     Bruce J. Packard
--     Science Applications International Corporation (SAIC)
--     424 Delaware, Suite C-3
--     Leavenworth, KS  66048   (913) 651-7925
--
with SYSTEM_PACKAGE;              use SYSTEM_PACKAGE;

package UIW_IMAGE_WINDOW is

   -- Array for storing contiguous line segment points
   type UIW_X_POINTS  is   array (SYS_DB_SIZE range <>) of SYS_IMAGE_COLUMN;
   type UIW_Y_POINTS  is   array (SYS_DB_SIZE range <>) of SYS_IMAGE_ROW;
   subtype UIW_Brush_Width is SYS_PIXEL range 1..5;
   UIW_Single_Brush_Width :  constant UIW_Brush_Width := 1;
   UIW_Double_Brush_Width :  constant UIW_Brush_Width := 2;

-- ##############################################################################
   procedure UIW_DISPLAY_BIT_IMAGE (WINDOW_ID       : in  SYS_WINDOW_ELE_ID;
                                    SUB_ADD_FLAG    : in  BOOLEAN;
                                    DISPLAY_FUNTION : in  SYS_COLOR_ACTION;
                                    PIXEL_UL_X      : in  SYS_WINDOW_COLUMN;
                                    PIXEL_UL_Y      : in  SYS_WINDOW_ROW;
                                    SIZE_X          : in  SYS_WINDOW_COLUMN;
                                    SIZE_Y          : in  SYS_WINDOW_ROW;
                                    PIXMAP_ID       : in  SYS_WINDOW_ELE_ID;
                                    PLANE_MASK      : in  SYS_COLOR_MASK);
   --
   --CPM description:
   --     This module Displays or erases a Bit Image (pixmap) in the
   --     specified planes.
   --
   --CPM design notes:
   --     1.)  The pixmap is created by UIW_CREATE_PIXMAP.
   --
   --formal parameters
   --IN      WINDOW_ID       - The Id of the Window to display the image in.  It
   --                          can be obtained by calling UWM_QUERY_WINDOW_ID.
   --IN      SUB_ADD_FLAG    - Image Subtraction or Addition Flag.  During
   --                          subtraction, the bits set in the raster image
   --                          shall be subtracted out of the selected planes.
   --                          During addition, the bits set in the raster image
   --                          shall be added into the selected planes.
   --                        = 0 - Subtract the image.
```

```
--                          = 1 - Add the image.
--IN     DISPLAY_FUNTION - The means of adding/subtracting the image to the
--                          displayed image (and, or, copy...).
--IN     PIXEL_UL_X      - The window X coordinate of the Upper Left corner
--                          of the image.
--IN     PIXEL_UL_Y      - The window Y coordinate of the Upper Left corner
--                          of the image.
--IN     SIZE_X          - The Size of the image in the X direction.
--IN     SIZE_Y          - The Size of the image in the Y direction.
--IN     PIXMAP_ID       - The Pixmap Id returned from UIW_CREATE_PIXMAP.
--IN     PLANE_MASK      - A bit map representation of the Planes to be
--                          affected by the image.  Value can be obtained from
--                          "UIW_PLANE_MASK".
--end formal parameters;
--

-- ##########################################################################
   procedure UIW_DISPLAY_CIRCLE (WINDOW_ID    : in  SYS_WINDOW_ELE_ID;
                                 SUB_ADD_FLAG : in  BOOLEAN;
                                 CENTER_X     : in  SYS_IMAGE_COLUMN;
                                 CENTER_Y     : in  SYS_IMAGE_ROW;
                                 RADIUS       : in  SYS_WINDOW_COLUMN;
                                 COLOR        : in  SYS_COLOR;
                                 PLANE_MASK   : in  SYS_COLOR_MASK);
   --
   --CPM description:
   --    This module Displays or erases a Circle in the specified planes.
   --
   --CPM design notes:
   --    1.)  None.
   --
   --formal parameters
   --IN     WINDOW_ID    - The Id of the Window to display the circle in.  It
   --                      can be obtained by calling UWM_QUERY_WINDOW_ID.
   --IN     SUB_ADD_FLAG - Image Subtraction or Addition Flag.  During
   --                      subtraction, the bits set in the raster image shall
   --                      be subtracted out of the selected planes.  During
   --                      addition, the bits set in the raster image shall be
   --                      added into the selected planes.
   --                    = 0 - Subtract the circle.
   --                    = 1 - Add the cirlce.
   --IN     CENTER_X     - The window X coordinate of the Center of the circle.
   --IN     CENTER_Y     - The window Y coordinate of the Center of the circle.
   --IN     RADIUS       - The Radius of the circle, in pixels.
   --IN     COLOR        - The index into the color lookup table for the Color
   --                      of the circle.
   --IN     PLANE_MASK   - A bit map representation of the Planes to be
   --                      affected by the circle.  Value can be obtained from
   --                      "UIW_PLANE_MASK".
   --end formal parameters;
   --

-- ##########################################################################
   procedure UIW_DISPLAY_LINE (WINDOW_ID    : in  SYS_WINDOW_ELE_ID;
                               SUB_ADD_FLAG : in  BOOLEAN;
                               LINE_START_X : in  SYS_IMAGE_COLUMN;
```

```
                                LINE_START_Y : in  SYS_IMAGE_ROW;
                                LINE_END_X   : in  SYS_IMAGE_COLUMN;
                                LINE_END_Y   : in  SYS_IMAGE_ROW;
                                BRUSH_WIDTH  : in  UIW_BRUSH_WIDTH;
                                COLOR        : in  SYS_COLOR;
                                PLANE_MASK   : in  SYS_COLOR_MASK);
    --
    --CPM description:
    --    This module Displays or erases a Line in the specified planes.
    --
    --CPM design notes:
    --    1.)  None.
    --
    --formal parameters
    --IN     WINDOW_ID     - The Id of the Window to display the line in.  It
    --                       can be obtained by calling UWM_QUERY_WINDOW_ID.
    --IN     SUB_ADD_FLAG  - Image Subtraction or Addition Flag.  During
    --                       subtraction, the bits set in the raster image shall
    --                       be subtracted out of the selected planes.  During
    --                       addition, the bits set in the raster image shall be
    --                       added into the selected planes.
    --                     = 0 - Subtract the line.
    --                     = 1 - Add the line.
    --IN     LINE_START_X  - The window X coordinate of the Start of the Line.
    --IN     LINE_START_Y  - The window Y coordinate of the Start of the Line.
    --IN     LINE_END_X    - The window X coordinate of the End of the Line.
    --IN     LINE_END_Y    - The window Y coordinate of the End of the Line.
    --IN     BRUSH_WIDTH   - The thickness (Width in pixels) of the lines.
    --IN     COLOR         - The index into the color lookup table for the Color
    --                       of the line.
    --IN     PLANE_MASK    - A bit map representation of the Planes to be
    --                       affected by the line.  Value can be obtained from
    --                       "UIW_PLANE_MASK".
    --end formal parameters;
    --

-- ##########################################################################
   procedure UIW_DISPLAY_LINES (WINDOW_ID      : in  SYS_WINDOW_ELE_ID;
                                SUB_ADD_FLAG   : in  BOOLEAN;
                                X_POINTS       : in  UIW_X_POINTS;
                                Y_POINTS       : in  UIW_Y_POINTS;
                                NUMBER_POINTS  : in  SYS_DB_SIZE;
                                BRUSH_WIDTH    : in  UIW_BRUSH_WIDTH;
                                COLOR          : in  SYS_COLOR;
                                PLANE_MASK     : in  SYS_COLOR_MASK);
    --
    --CPM description:
    --    This module Displays or erases contiguous Line segments in the
    --    specified planes.
    --
    --CPM design notes:
    --    1.)  This module will draw single or multiple line segments.
    --
    --formal parameters
    --IN     WINDOW_ID     - The Id of the Window to display the lines in.  It
    --                       can be obtained by calling UWM_QUERY_WINDOW_ID.
```

```
--IN     SUB_ADD_FLAG  - Image Subtraction or Addition Flag.  During
--                       subtraction, the bits set in the raster image shall
--                       be subtracted out of the selected planes.  During
--                       addition, the bits set in the raster image shall be
--                       added into the selected planes.
--                     = 0 - Subtract the lines.
--                     = 1 - Add the lines.
--IN     X_POINTS      - The list of window X coordinate Points in the
--                      contiguous line segments.
--IN     Y_POINTS      - The list of window Y-coordinate Points in the
--                      contiguous line segments.
--IN     NUMBER_POINTS - The Number of Points in the list.  This will
--                      produce (number_points - 1) line segments.
--                    >= 2 and fit in a 32 bit integer.
--IN     BRUSH_WIDTH   - The thickness (Width in pixels) of the lines.
--IN     COLOR         - The index into the color lookup table for the Color
--                       of the lines.
--IN     PLANE_MASK    - A bit map representation of the Planes to be
--                       affected by the line.  Value can be obtained from
--                       "UIW_PLANE_MASK".
--end formal parameters;
--

-- ##############################################################################
procedure UIW_DISPLAY_SYMBOL (WINDOW_ID    : in  SYS_WINDOW_ELE_ID;
                              FONT_ID      : in  SYS_WINDOW_ELE_ID;
                              SUB_ADD_FLAG : in  BOOLEAN;
                              PIXEL_COLUMN : in  SYS_IMAGE_COLUMN;
                              PIXEL_ROW    : in  SYS_IMAGE_ROW;
                              SYMBOL_VALUE : in  SYS_WINDOW_ELE_ID;
                              COLOR        : in  SYS_COLOR;
                              PLANE_MASK   : in  SYS_COLOR_MASK);
--
--CPM description:
--    This module Displays or erases a font Symbol in the specified planes.
--
--CPM design notes:
--    1.)  The font must be initialized with UIW_INIT_FONT before an element
--    can be displayed.
--
--formal parameters
--IN     WINDOW_ID     - The Id of the Window to display the symbol in.  It
--                       can be obtained by calling UWM_QUERY_WINDOW_ID.
--IN     FONT_ID       - The Id of the symbol Font.  Value is output from
--                       "UIW_INIT_FONT".
--IN     SUB_ADD_FLAG  - Image Subtraction or Addition Flag.  During
--                       subtraction, the bits set in the raster image shall
--                       be subtracted out of the selected planes.  During
--                       addition, the bits set in the raster image shall be
--                       added into the selected planes.
--                     = 0 - Subtract the symbol.
--                     = 1 - Add the symbol.
--IN     PIXEL_COLUMN  - The Pixel Column of the upper left corner of the
--                       symbol.
--IN     PIXEL_ROW     - The Pixel Row of the upper left corner of the
--                       symbol.
```

```
    --IN      SYMBOL_VALUE   - The integer Value of the Symbol to be displayed.
    --IN      COLOR          - The index into the color lookup table for the Color
    --                         of the symbol.
    --IN      PLANE_MASK     - A bit map representation of the Planes to be
    --                         affected by the symbol.  Value can be obtained from
    --                         "UIW_PLANE_MASK".
    --end formal parameters;
    --

-- ##########################################################################
    procedure UIW_DISPLAY_TEXT (WINDOW_ID     : in  SYS_WINDOW_ELE_ID;
                                FONT_ID       : in  SYS_WINDOW_ELE_ID;
                                SUB_ADD_FLAG  : in  BOOLEAN;
                                PIXEL_COLUMN  : in  SYS_IMAGE_COLUMN;
                                PIXEL_ROW     : in  SYS_IMAGE_ROW;
                                TEXT_STRING   : in  STRING;
                                COLOR         : in  SYS_COLOR;
                                PLANE_MASK    : in  SYS_COLOR_MASK);
    --
    --CPM description:
    --    This module Displays or erases a Text string in the specified planes.
    --
    --CPM design notes:
    --    1.)  The font must be initialized with UIW_INIT_FONT before a string
    --    can be displayed.
    --
    --formal parameters
    --IN      WINDOW_ID      - The Id of the Window to display the text string in.
    --                         It can be obtained by calling UWM_QUERY_WINDOW_ID.
    --IN      FONT_ID        - The Id of the text Font.  Value is output from
    --                         "UIW_INIT_FONT".
    --IN      SUB_ADD_FLAG   - Image Subtraction or Addition Flag.  During
    --                         subtraction, the bits set in the raster image shall
    --                         be subtracted out of the selected planes.  During
    --                         addition, the bits set in the raster image shall be
    --                         added into the selected planes.
    --                       = 0 - Subtract the text.
    --                       = 1 - Add the text.
    --IN      PIXEL_COLUMN   - The Pixel Column of the upper left corner of the
    --                         text.
    --IN      PIXEL_ROW      - The Pixel Row of the upper left corner of the text.
    --IN      TEXT_STRING    - The String of Text to be displayed.
    --IN      COLOR          - The index into the color lookup table for the Color
    --                         of the text string.
    --IN      PLANE_MASK     - A bit map representation of the Planes to be
    --                         affected by the text string.  Value can be obtained
    --                         from "UIW_PLANE_MASK".
    --end formal parameters;
    --

-- ##########################################################################
    procedure UIW_ERASE_PLANES (WINDOW_ID  : in  SYS_WINDOW_ELE_ID;
                                PIXEL_UL_X : in  SYS_WINDOW_COLUMN;
                                PIXEL_UL_Y : in  SYS_WINDOW_ROW;
                                SIZE_X     : in  SYS_WINDOW_COLUMN;
                                SIZE_Y     : in  SYS_WINDOW_ROW;
```

```
                                PLANE_MASK : in  SYS_COLOR_MASK);
    --
    --CPM description:
    --     This module Erases everything in a given rectangular image out of the
    --     specified Planes.
    --
    --CPM design notes:
    --     1.)  None.
    --
    --formal parameters
    --IN      WINDOW_ID  - The Id of the Window to erase the planes in.  It can
    --                     be obtained by calling UWM_QUERY_WINDOW_ID.
    --IN      PIXEL_UL_X - The window X coordinate of the Upper Left corner of
    --                     the image.
    --IN      PIXEL_UL_Y - The window Y coordinate of the Upper Left corner of
    --                     the image.
    --IN      SIZE_X     - The Size of the image in the X direction.
    --IN      SIZE_Y     - The Size of the image in the Y direction.
    --IN      PLANE_MASK - A bit map representation of the Planes to be affected
    --                     by the image.  Value can be obtained from
    --                     "UIW_PLANE_MASK".
    --end formal parameters;
    --

-- ##############################################################################
   procedure UIW_FLUSH_BUFFER;
   --
   --CPM description:
   --     This module Flushes the graphics command Buffer.
   --
   --CPM design notes:
   --     1.)  X Windows buffers its commands and flushes that buffer after
   --     certain commands or when the buffer is full.  Therefore this module
   --     only needs to be called when a previous command must be seen
   --     immediately.
   --
   --formal parameters
   --        None
   --end formal parameters;
   --

-- ##############################################################################
   procedure UIW_FREE_PIXMAP (PIXMAP_ID : in  SYS_WINDOW_ELE_ID);
   --
   --CPM description:
   --     This module Frees up the memory allocated to a Pixmap back in
   --     UIW_CREATE_PIXMAP.
   --
   --CPM design notes:
   --     1.)  In EDDIC the contours pixmaps should be freed after each block is
   --     displayed, but the unit symbology pixmaps can be defined once and left
   --     for the duration of the run.
   --
   --formal parameters
   --IN      PIXMAP_ID - The Pixmap Id returned from UIW_CREATE_PIXMAP.
   --end formal parameters;
```

```
   --

-- ##############################################################################
   procedure UIW_INIT_FONT (FONT_NAME   : in   STRING;
                            FONT_ID     : out  SYS_WINDOW_ELE_ID;
                            FONT_HEIGHT : out  SYS_WINDOW_ROW;
                            FONT_WIDTH  : out  SYS_WINDOW_COLUMN);
   --
   --CPM description:
   --    This module Initializes a specified Font.
   --
   --CPM design notes:
   --    1.)  Fonts are only initialized once.
   --    2.)  It is legal to have multiple fonts in a single process.
   --
   --formal parameters
   --IN     FONT_NAME   - The string containing the Font's directory and Name.
   --OUT    FONT_ID     - The Id of the Font as returned by the X system.
   --OUT    FONT_HEIGHT - The Height, in pixels, of a Font character.
   --OUT    FONT_WIDTH  - The Width, in pixels, of a Font character.
   --end formal parameters;
   --

-- ##############################################################################
   procedure UIW_INIT_LOOKUP_TABLE (MAX_PLANES : in  SYS_MAX_PLANES);
   --
   --CPM description:
   --    This module Initializes (allocates space for) the color Lookup Table.
   --
   --CPM design notes:
   --    1.)  The lookup table is only initialized once.
   --
   --formal parameters
   --IN     MAX_PLANES - The Maximum number of color Planes currently allowed
   --                   in the system.
   --end formal parameters;
   --

-- ##############################################################################
   procedure UIW_LOAD_LOOKUP_TABLE (LUT_INDEX    : in  SYS_COLOR_TABLE;
                                    RED_INTENS   : in  SYS_COLOR;
                                    GREEN_INTENS : in  SYS_COLOR;
                                    BLUE_INTENS  : in  SYS_COLOR);
   --
   --CPM description:
   --    This module Loads color values into the color Lookup Table.
   --
   --CPM design notes:
   --    1.)  The display is not altered by calling this module; the display
   --    is altered by calling UIW_STORE_LOOKUP_TABLE.
   --
   --formal parameters
   --IN    LUT_INDEX    - The Index into the Lookup Table to load.  Zero is
   --                     the first cell in the lookup table.
   --IN    RED_INTENS   - The Intensity for Red.
   --IN    GREEN_INTENS - The Intensity for Green.
```

```
--IN    BLUE_INTENS  - The Intensity for Blue.
--end formal parameters;
--

-- ##############################################################################
procedure UIW_MOVE_IMAGE (WINDOW_ID      : in  SYS_WINDOW_ELE_ID;
                          OLD_PIXEL_UL_X : in  SYS_WINDOW_COLUMN;
                          OLD_PIXEL_UL_Y : in  SYS_WINDOW_ROW;
                          NEW_PIXEL_UL_X : in  SYS_WINDOW_COLUMN;
                          NEW_PIXEL_UL_Y : in  SYS_WINDOW_ROW;
                          SIZE_X         : in  SYS_WINDOW_COLUMN;
                          SIZE_Y         : in  SYS_WINDOW_ROW);
--
--CPM description:
--    This module Moves a raster Image from one location in a window to
--    another location within the same window.
--
--CPM design notes:
--    1.)  None.
--
--formal parameters
--IN     WINDOW_ID      - The Id of the Window the image is in.  It can be
--                        obtained by calling UWM_QUERY_WINDOW_ID.
--IN     OLD_PIXEL_UL_X - The window X coordinate of the Upper Left corner
--                        of the source image.
--IN     OLD_PIXEL_UL_Y - The window Y coordinate of the Upper Left corner
--                        of the source image.
--IN     NEW_PIXEL_UL_X - The window X coordinate of the Upper Left corner
--                        of the destination image.
--IN     NEW_PIXEL_UL_Y   The window Y coordinate of the Upper Left corner
--                        of the destination image.
--IN     SIZE_X         - The Size of the image in the X direction.
--IN     SIZE_Y         - The Size of the image in the Y direction.
--end formal parameters;
--

-- ##############################################################################
procedure UIW_PLANE_MASK (START_PLANE : in   SYS_COLOR_PLANE;
                          END_PLANE   : in   SYS_COLOR_PLANE;
                          PLANE_MASK  : out  SYS_COLOR_MASK);
--
--CPM description:
--    This module calculates a bit map representation (Mask) of the Planes
--     requested by the user for later use.
--
--CPM design notes:
--    1.)  None.
--
--formal parameters
--IN     START_PLANE - The Plane number of the lowest plane to be affected
--                     by the image.  Bit 1 of the raster image shall be
--                     loaded into this plane.  Plane numbers start at 1.
--IN     END_PLANE   - The Plane number of the highest plane to be affected
--                     by the image.  Image bits that are greater than
--                     (end_plane - start_plane + 1) shall be ignored.
--OUT    PLANE_MASK  - A bit map representation of the Planes which the
```

```
--                      user would like to affect in a future window call.
--end formal parameters;
--

-- ##############################################################################
procedure UIW_RUBBERBAND_LINE (WINDOW_ID    : in   SYS_WINDOW_ELE_ID;
                               FROM_POINT_X : in   SYS_WINDOW_COLUMN;
                               FROM_POINT_Y : in   SYS_WINDOW_ROW;
                               COLOR        : in   SYS_COLOR;
                               PLANE_MASK   : in   SYS_COLOR_MASK;
                               END_POINT_X  : out  SYS_WINDOW_COLUMN;
                               END_POINT_Y  : out  SYS_WINDOW_ROW);
--
--CPM description:
--    This module draws a Rubberband Line in the specified window from
--    the specified point to the cursor and returns the end point selected
--    by the user.
--
--CPM design notes:
--    1.)  If the user moves the cursor outside the window and selects the
--    point, the end point coordinates are the lines window boundry crossing.
--    2.)  If the user moves the cursor outside the window and selects the
--    point, the rubberband line is not drawn upon return.
--
--formal parameters
--IN     WINDOW_ID    - The Id of the Window the line is in.  It can be
--                      obtained by calling UWM_QUERY_WINDOW_ID.
--IN     FROM_POINT_X - The window X coordinate of the Point the lines
--                      rubberbanding emanates From.
--IN     FROM_POINT_Y - The window Y coordinate of the Point the lines
--                      rubberbanding emanates From.
--IN     COLOR        - The index into the color lookup table for the Color
--                      of the line.
--IN     PLANE_MASK   - A bit map representation of the Planes to be
--                      affected by the line.  Value can be obtained from
--                      "UIW_PLANE_MASK".
--OUT    END_POINT_X  - The window X coordinate of the lines End Point as
--                      selected by the user.
--OUT    END_POINT_Y  - The window Y coordinate of the lines End Point as
--                      selected by the user.
--end formal parameters;

-- ##############################################################################
procedure UIW_STORE_LOOKUP_TABLE;
--
--CPM description:
--    This module Stores the color Lookup Table.
--
--CPM design notes:
--    1.)  Calling this module alters the display provided some of the
--    values were changed with UIW_LOAD_LOOKUP_TABLE.
--
--formal parameters
--       None.
--end formal parameters;
--
```

```
end UIW_IMAGE_WINDOW;
```

UTM Utility Package Specifications

The following package specifications are contained in the tactical map function:

```
CM_SYSTEM
MAP_SYSTEM
OBS_SYSTEM
UCC_COORD_CONVERT
UCE_CNTRL_MSR_EDITOR
UCM_CONTROL_MEASURE
UME_MAP_EDITOR
UMP_MAP
UNIT_SYSTEM
UNT_UNIT
UOB_OBSTACLE
UOE_OBSTACLE_EDITOR
UTM_TACTICAL_MAP
UUE_UNIT_EDITOR
```

```
--cpc package specification name: CM_SYSTEM
--
--cpc description: Defines types and objects that are common to the control
--                 measure display system.
--
--cpc design notes:
--
--cpc package author: Bruce Packard
--                    Science Applications International Corporation
--                    424 Delaware, Suite C3
--                    Leavenworth, KS 66048
--
with SYSTEM_PACKAGE;          use SYSTEM_PACKAGE;
with SDB_SITUATION_DB;        use SDB_SITUATION_DB;

package CM_SYSTEM is

   -- Control measure display options
   type CM_CNTRL_MSR_OPTIONS is
      record
         CM_BLUE_EAC                 :       BOOLEAN;
         CM_BLUE_CORP                :       BOOLEAN;
         CM_BLUE_DIV                 :       BOOLEAN;
         CM_BLUE_BDE                 :       BOOLEAN;
         CM_BLUE_BN                  :       BOOLEAN;
         CM_BLUE_CO                  :       BOOLEAN;
         CM_BLUE_POINT               :       BOOLEAN;
         CM_BLUE_LINE                :       BOOLEAN;
         CM_BLUE_AREA                :       BOOLEAN;
         CM_BLUE_ROUTE               :       BOOLEAN;
         CM_BLUE_CROSSING            :       BOOLEAN;
         CM_BLUE_FIRE_PLAN           :       BOOLEAN;
         CM_BLUE_MAP_FEAT            :       BOOLEAN;
         CM_OPFOR_ARMY               :       BOOLEAN;
         CM_OPFOR_DIV                :       BOOLEAN;
         CM_OPFOR_RGMT               :       BOOLEAN;
         CM_OPFOR_BN                 :       BOOLEAN;
         CM_OPFOR_CO                 :       BOOLEAN;
         CM_OPFOR_POINT              :       BOOLEAN;
         CM_OPFOR_LINE               :       BOOLEAN;
         CM_OPFOR_AREA               :       BOOLEAN;
         CM_OPFOR_ROUTE              :       BOOLEAN;
         CM_OPFOR_CROSSING           :       BOOLEAN;
         CM_OPFOR_FIRE_PLAN          :       BOOLEAN;
         CM_OPFOR_MAP_FEAT           :       BOOLEAN;
   end record;

   -- Displayed control measures
   CM_CNTRL_MSR_COUNT              :    SDB_CONTROL_MEASURE_ID;
   CM_CURR_CNTRL_MSR               :    SDB_CONTROL_MSR_POINT;
   CM_CNTRL_MSR_DISPLAYED      :    array (SDB_CONTROL_MEASURE_ID) of BOOLEAN;

   -- Displayed point control measures
   CM_CNTRL_MSR_PNT_COUNT          :    SDB_CONTROL_MEASURE_ID;
   CM_CURR_CNTRL_MSR_PNT           :    SDB_CNTRL_POINT_POINT;
   CM_CNTRL_MSR_PNT_DISPLAYED  :    array (SDB_CONTROL_MEASURE_ID) of BOOLEAN;
```

```
    -- Selected Control Measure point
    CM_SELECTED_BL_POINT            :    SDB_CONTROL_MEASURE_PT;
    CM_SELECTED_OP_POINT            :    SDB_CONTROL_MEASURE_PT;

    -- Control Measure definition objects
    CM_DEFINING_CNTRL_MSR           :    BOOLEAN;
    CM_DEFINE_BUTTON_ID             :    SYS_WINDOW_ELE_ID;
    CM_DEFINE_MENU_PANEL            :    SYS_WINDOW_ELE_ID;
    CM_CNTRL_MSR_DEFINITION         :    SDB_CONTROL_MEASURE_REC;
    CM_CNTRL_MSR_POINT_DEF          :    SDB_CNTRL_MSR_POINT_REC;

    -- Current Control Measure display options
    CM_CURR_CNTRL_MSR_OPTION        :    CM_CNTRL_MSR_OPTIONS;

    -- Control measure popup menus (Blue and OPFOR)
    CM_BL_MENU_ID                   :    SYS_WINDOW_ELE_ID;
    CM_BL_MENU_START                :    SYS_POP_UP_START_PTR := new
                                         SYS_POP_UP_START (SYS_CM_MENU);
    CM_BL_MENU_LENGTH               :    SYS_POP_UP_LENGTH_PTR := new
                                         SYS_POP_UP_LENGTH (SYS_CM_MENU);
    CM_BL_POP_UP_TEXT               :    SYS_MENU_TEXT_PTR := new
                                         SYS_MENU_TEXT (SYS_CM_CELL);
    CM_BL_POP_UP_CHILD              :    SYS_POP_UP_CHILD_PTR := new
                                         SYS_POP_UP_CHILD (SYS_CM_CELL);
    CM_BL_POP_UP_OPTION             :    SYS_CM_OPTION_PTR := new
                                         SYS_CM_OPTION_ARRAY (SYS_CM_CELL);
    CM_OP_MENU_ID                   :    SYS_WINDOW_ELE_ID;
    CM_OP_MENU_START                :    SYS_POP_UP_START_PTR := new
                                         SYS_POP_UP_START (SYS_CM_MENU);
    CM_OP_MENU_LENGTH               :    SYS_POP_UP_LENGTH_PTR := new
                                         SYS_POP_UP_LENGTH (SYS_CM_MENU);
    CM_OP_POP_UP_TEXT               :    SYS_MENU_TEXT_PTR := new
                                         SYS_MENU_TEXT (SYS_CM_CELL);
    CM_OP_POP_UP_CHILD              :    SYS_POP_UP_CHILD_PTR := new
                                         SYS_POP_UP_CHILD (SYS_CM_CELL);
    CM_OP_POP_UP_OPTION             :    SYS_CM_OPTION_PTR := new
                                         SYS_CM_OPTION_ARRAY (SYS_CM_CELL);

end CM_SYSTEM;
```

```
--cpc package specification name: MAP_SYSTEM
--
--cpc description: Defines types and objects that are common to the map system.
--
--cpc design notes:
--
--cpc package author: Bruce Packard
--                    Science Applications International Corporation
--                    424 Delaware, Suite C3
--                    Leavenworth, KS 66048
--
with SYSTEM_PACKAGE;            use SYSTEM_PACKAGE;
with SDB_SITUATION_DB;          use SDB_SITUATION_DB;
with LUT_SYSTEM;                use LUT_SYSTEM;
with UED_LIST;
with UNCHECKED_DEALLOCATION;

package MAP_SYSTEM is

   -- Lookup table types for each background type
   subtype MAP_BACK_RANGE is INTEGER range 0..5;
   MAP_BACK_LUT    :   array (MAP_BACK_RANGE) of LUT_BACKGROUND := (
         LUT_SHADE_VEG, LUT_SHADE_VEG, LUT_SHADE_VEG, LUT_SHADE_VEG,
         LUT_SHADE_VEG, LUT_NONE);

   -- Digital map scales
   subtype MAP_SCALE_RANGE is INTEGER range 0..5;
   MAP_SCALE_NAMES     :   array (SYS_MAP_SCALES) of SYS_WINDOW_NAME := (
        S1_40000 =>  "MAP SCALE: 1:40000 " & ASCII.NUL & "          ",
        S1_80000 =>  "MAP SCALE: 1:80000 " & ASCII.NUL & "          ",
        S1_160000 => "MAP SCALE: 1:160000 " & ASCII.NUL & "         ",
        S1_400000 => "MAP SCALE: 1:400000 " & ASCII.NUL & "         ",
        S1_800000 => "MAP SCALE: 1:800000 " & ASCII.NUL & "         ");

   -- Map display options
   -- Status flags indicate if feature is highlighted by class or displayed in
   -- a single color
   type MAP_MAP_OPTIONS is
      record
         MAP_BACK_TYPE      :          SYS_MAP_BACKGROUND;
         MAP_LUT            :          LUT_BACKGROUND;
         MAP_MAP_SCALE      :          SYS_MAP_SCALES;
         MAP_GRID_STATUS    :          BOOLEAN;
         MAP_CONTOUR_STATUS:           BOOLEAN;
         MAP_CENTER_X       :          SYS_COORDINATE;
         MAP_CENTER_Y       :          SYS_COORDINATE;
   end record;

   -- Map panel description
   type MAP_PANEL_DESCRIPTOR is
      record
         -- Id assigned to the digital map panel
         MAP_PANEL_ID       :          SYS_WINDOW_ELE_ID;
         -- Id returned from the window system
         MAP_WINDOW_ID      :          SYS_WINDOW_ELE_ID;
         -- Id of the map windows parent
```

```
      MAP_PARENT_ID     :          SYS_WINDOW_ELE_ID;
      -- Size of the panel in the X direction
      MAP_PIXEL_X       :          SYS_WINDOW_COLUMN;
      -- Size of the panel in the Y direction
      MAP_PIXEL_Y       :          SYS_WINDOW_ROW;
      -- World coordinates of the panels lower left corner
      MAP_WORLD_LL_X    :          SYS_COORDINATE;
      MAP_WORLD_LL_Y    :          SYS_COORDINATE;
end record;

-- Map Contour Database description
type MAP_CONT_DB_DESCRIPTOR is
   record
      -- Number of record columns in the database
      MAP_RECORD_X      :          SYS_DB_SIZE;
      -- Number of record rows in the database
      MAP_RECORD_Y      :          SYS_DB_SIZE;
      -- The size of a map block in the X direction
      MAP_BLOCK_SIZE_X  :          SYS_IMAGE_COLUMN;
      -- The size of a map block in the Y direction
      MAP_BLOCK_SIZE_Y  :          SYS_IMAGE_ROW;
end record;

-- Elevation Database description
type MAP_ELEV_DB_DESCRIPTOR is
   record
      -- Number of meters per pixel
      MAP_SCALE         :          FLOAT;
      -- Size of the digital map in the X direction
      MAP_PIXEL_X       :          SYS_IMAGE_COLUMN;
      -- Size of the digital map in the Y direction
      MAP_PIXEL_Y       :          SYS_IMAGE_ROW;
      -- Number of record columns in the database
      MAP_RECORD_X      :          SYS_DB_SIZE;
      -- Number of record rows in the database
      MAP_RECORD_Y      :          SYS_DB_SIZE;
      -- The size of a map block in the X direction
      MAP_BLOCK_SIZE_X  :          SYS_IMAGE_COLUMN;
      -- The size of a map block in the Y direction
      MAP_BLOCK_SIZE_Y  :          SYS_IMAGE_ROW;
      -- World coordinates of the elevation lower left corner
      MAP_WORLD_LL_X    :          SYS_COORDINATE;
      MAP_WORLD_LL_Y    :          SYS_COORDINATE;
end record;

-- Map Database description
type MAP_DB_DESCRIPTOR is
   record
      -- Number of meters per pixel
      MAP_SCALE         :          FLOAT;
      -- Size of the digital map in the X direction
      MAP_PIXEL_X       :          SYS_IMAGE_COLUMN;
      -- Size of the digital map in the Y direction
      MAP_PIXEL_Y       :          SYS_IMAGE_ROW;
      -- Number of record columns in the database
      MAP_RECORD_X      :          SYS_DB_SIZE;
```

```
      -- Number of record rows in the database
      MAP_RECORD_Y       :          SYS_DB_SIZE;
      -- The size of a map block in the X direction
      MAP_BLOCK_SIZE_X   :          SYS_IMAGE_COLUMN;
      -- The size of a map block in the Y direction
      MAP_BLOCK_SIZE_Y   :          SYS_IMAGE_ROW;
      -- World coordinates of the digital map upper left corner
      MAP_WORLD_UL_X     :          SYS_COORDINATE;
      MAP_WORLD_UL_Y     :          SYS_COORDINATE;
      -- Grid interval
      MAP_GRID_INTRVL    :          SYS_COORDINATE;
end record;

-- Map Selection Point description
type MAP_SELECT_PT_DESCRIPTOR is
   record
      MAP_PIXEL_X        :          SYS_IMAGE_COLUMN;
      MAP_PIXEL_Y        :          SYS_IMAGE_ROW;
      MAP_WORLD_X        :          SYS_COORDINATE;
      MAP_WORLD_Y        :          SYS_COORDINATE;
end record;

-- Map Masks
MAP_CONTOUR_MASK              :     SYS_COLOR_MASK;
MAP_BACKGROUND_MASK           :     SYS_COLOR_MASK;
MAP_OVERLAY_MASK              :     SYS_COLOR_MASK;
MAP_GRID_MASK                 :     SYS_COLOR_MASK;
MAP_RED_MASK                  :     SYS_COLOR_MASK;
MAP_BLUE_MASK                 :     SYS_COLOR_MASK;

-- Current Map display options
MAP_CURR_MAP_OPTION           :     MAP_MAP_OPTIONS;

-- Current map panel
MAP_CURR_PANEL                :     MAP_PANEL_DESCRIPTOR;

-- Current contour map database
MAP_CURR_CONT_DB              :     MAP_CONT_DB_DESCRIPTOR;

-- Current Elevation database
MAP_ELEV_DB                   :     MAP_ELEV_DB_DESCRIPTOR;

-- Current map database
MAP_CURR_DB                   :     MAP_DB_DESCRIPTOR;

-- Last user selection point on the map
MAP_CURR_SELECT_PT            :     MAP_SELECT_PT_DESCRIPTOR;

-- Digital map parameter file names
MAP_CONTOUR_FILE              :     array (MAP_SCALE_RANGE) of
                                    string (SYS_NAME_SIZE);
MAP_VEGETATION_FILE           :     array (MAP_SCALE_RANGE) of
                                    string (SYS_NAME_SIZE);
MAP_SHADED_FILE               :     array (MAP_SCALE_RANGE) of
                                    string (SYS_NAME_SIZE);
MAP_CCM_FILE                  :     array (MAP_SCALE_RANGE) of
```

```
                                    string (SYS_NAME_SIZE);
MAP_ELEVATION_FILE          :       array (MAP_SCALE_RANGE) of
                                    string (SYS_NAME_SIZE);
MAP_3D_FILE                 :       array (MAP_SCALE_RANGE) of
                                    string (SYS_NAME_SIZE);

-- Map Control menus
MAP_MENU_ID                 :       SYS_WINDOW_ELE_ID;
MAP_MENU_START              :       SYS_POP_UP_START_PTR := new
                                    SYS_POP_UP_START (SYS_MAP_MENU);
MAP_MENU_LENGTH             :       SYS_POP_UP_LENGTH_PTR := new
                                    SYS_POP_UP_LENGTH (SYS_MAP_MENU);
MAP_POP_UP_TEXT             :       SYS_MENU_TEXT_PTR := new
                                    SYS_MENU_TEXT (SYS_MAP_CELL);
MAP_POP_UP_CHILD            :       SYS_POP_UP_CHILD_PTR := new
                                    SYS_POP_UP_CHILD (SYS_MAP_CELL);
MAP_POP_UP_OPTION           :       SYS_MAP_CONTROL_PTR := new
                                    SYS_MAP_CONTROL_ARRAY (SYS_MAP_CELL);

-- Map fonts
MAP_SYMBOLOGY_FONT_ID       :       SYS_WINDOW_ELE_ID;
MAP_SYMBOLOGY_FONT_HEIGHT   :       SYS_WINDOW_ROW;
MAP_SYMBOLOGY_FONT_WIDTH    :       SYS_WINDOW_COLUMN;
MAP_TEXT_FONT_ID            :       SYS_WINDOW_ELE_ID;
MAP_TEXT_FONT_HEIGHT        :       SYS_WINDOW_COLUMN;
MAP_TEXT_FONT_WIDTH         :       SYS_WINDOW_ROW;

-- Station Control Process (SCL) for this workstation and control router
-- socket number
MAP_STATION_CONTROL         :       SYS_EDDIC_PROCESSES;
MAP_CALLING_PROCESS         :       SYS_EDDIC_PROCESSES;
MAP_CONTROL_SOCKET          :       SYS_CLIENT;

-- LIST OF OBJECTS ON THE DIGITAL MAP
type MAP_POINT_LIMIT is range 0..15;
type MAP_LIST_OF_POINTS is array (MAP_POINT_LIMIT) of SYS_IMAGE_LOCATION;

-- Type of objects on the map
type MAP_OBJECT_TYPE is (POINT, CIRCLE, RECTANGLE, LINE, POLYGON);
type MAP_OVERLAYS is (BLUEFOR_UNIT, OPFOR_UNIT, BLUEFOR_CNTRL_MSR,
     OPFOR_CNTRL_MSR, BLUEFOR_CNTRL_MSR_PNT, OPFOR_CNTRL_MSR_PNT,
     BLUEFOR_OBSTACLE, OPFOR_OBSTACLE);

-- Description of the different kinds of objects
type MAP_OBJECT_REC (OBJECT    : MAP_OBJECT_TYPE) is
   record

      case OBJECT is

         when POINT =>

            PT_LOC          :  SYS_IMAGE_LOCATION;

         when CIRCLE =>

            CENTER          :  SYS_IMAGE_LOCATION;
```

```
            RADIUS          :  SYS_PIXEL;

         when RECTANGLE =>

            UL              :  SYS_IMAGE_LOCATION;
            WIDTH           :  SYS_IMAGE_COLUMN;
            HEIGHT          :  SYS_IMAGE_ROW;

         when LINE =>

            LINE_PTS        :  MAP_POINT_LIMIT;
            LINE_LOC        :  MAP_LIST_OF_POINTS;

         when POLYGON =>

            POL_PTS         :  MAP_POINT_LIMIT;
            POL_LOC         :  MAP_LIST_OF_POINTS;

      end case;

end record;

type MAP_OBJECT_POINTER is access MAP_OBJECT_REC;

-- Description of the record that is in the object list
type MAP_OBJECT_DESC is
   record
      NAME                 :  STRING (1..21);
      OVERLAY              :  MAP_OVERLAYS;
      OVERLAY_INDEX        :  INTEGER;
      MENU_ID              :  SYS_WINDOW_ELE_ID;
      OBJECT               :  MAP_OBJECT_POINTER;

end record;
package OBJECT_LIST  is  new  UED_LIST (MAP_OBJECT_DESC);

-- Instantiate packages for storage deallocation
procedure OBJECT_DISPOSE is new UNCHECKED_DEALLOCATION (
         MAP_OBJECT_REC, MAP_OBJECT_POINTER);

-- Temporary storage for the Name, Overlay Type, Overlay Index, and Menu ID.
-- These should be loaded before the drawing application (UUE, UCE, UOE) is
-- called.
MAP_OBJECT_NAME          :  STRING (1..21);
MAP_OVERLAY_TYPE         :  MAP_OVERLAYS;
MAP_OVERLAY_INDEX        :  INTEGER;
MAP_OVERLAY_MENU         :  SYS_WINDOW_ELE_ID;

-- List of the digital map popup menus
type MAP_POPUP_MENU_DESC is
   record
      MENU_ID            :  SYS_WINDOW_ELE_ID;
      MENU_START         :  SYS_POP_UP_START_PTR;
      MENU_LENGTH        :  SYS_POP_UP_LENGTH_PTR;
      MENU_TEXT          :  SYS_MENU_TEXT_PTR;
      MENU_CHILD         :  SYS_POP_UP_CHILD_PTR;
```

```
end record;
package MAP_MENU_LIST  is  new  UED_LIST (MAP_POPUP_MENU_DESC);

-- Flag to indicate if the object list should be rebuilt
MAP_BUILD_LIST           :  BOOLEAN;

-- Description of the current selected object
MAP_SELECTED_OBJECT          :     MAP_OBJECT_DESC;

-- List of objects on the map that are too close to determine which one
-- was selected by the user.
MAP_DECLUTTER_THRESH         :    FLOAT   := 9.0;
type MAP_DECLUTTER_LIST_LIMIT is range 0..20;
type MAP_DECLUTTER_REC is
   record
      OBJECT_DESC        :  MAP_OBJECT_DESC;
      OBJECT_POINT       :  MAP_POINT_LIMIT;
end record;
MAP_DECLUTTER_COUNT      :  MAP_DECLUTTER_LIST_LIMIT;
MAP_DECLUTTER_LIST       :  array (MAP_DECLUTTER_LIST_LIMIT) of
                            MAP_DECLUTTER_REC;

-- Situation data objects
MAP_OPPLAN_ID                :  SYS_OPPLAN;
MAP_DATE_TIME                :  SYS_DATE_TIME;
MAP_SIT_SOCKET               :  SYS_CLIENT;

procedure MAP_DELETE_OBJECT (
              OVERLAY_TYPE        :   in      MAP_OVERLAYS;
              OVERLAY_INDEX       :   in      INTEGER);
--
-- CPM description: Deletes an object from the object list
--
-- formal parameters
--IN    OVERLAY_TYPE     Type of object to delete
--
--IN    OVERLAY_INDEX    Index for the object to delete
--

procedure MAP_FIND_OBJECT (
              OBJECT_FOUND        :   out     BOOLEAN;
              OVERLAY_TYPE        :   out     MAP_OVERLAYS;
              OBJECT_SELECTED     :   out     MAP_OBJECT_DESC;
              POINT_SELECTED      :   out     MAP_POINT_LIMIT);
--
-- CPM description: Finds the nearest object to a selected point
--
-- formal parameters
--OUT   OBJECT_FOUND     Logical flag indicating if an object was found
--                       within the search threshold.
--
--OUT   OVERLAY_TYPE     Type of object found
--
--OUT   OBJECT_SELECTED  Description of the selected object
--
--OUT   POINT_SELECTED   The index of the point selected for objects that
```

```
    --                          have multiple points.
    --

    function MAP_OBJECT_IN_WINDOW (
                OBJECT_DESC    :    in      MAP_OBJECT_DESC) return BOOLEAN;
    --
    -- CPM description: Determines if an object is within the map window
    --
    -- formal parameters
    --IN     OBJECT_DESC       Description of the object
    --

    procedure MAP_PURGE_OBJECT_LIST;
    --
    -- CPM description: Purges the overlay object list
    --
    -- formal parameters
    --None
    --

end MAP_SYSTEM;
--
-- Test Object definition package
--
with SYSTEM_PACKAGE;     use SYSTEM_PACKAGE;
with SDB_SITUATION_DB;   use SDB_SITUATION_DB;

package OBJECT is

   type OBJECT_LOCATION is
      record
         X                    :  SYS_IMAGE_COLUMN;
         Y                    :  SYS_IMAGE_ROW;
   end record;

   type POINT_LIMIT is range 0..15;
   type LIST_OF_POINTS is array (POINT_LIMIT) of OBJECT_LOCATION;


   type OBJECT_TYPE is (POINT, CIRCLE, RECTANGLE, LINE, POLYGON);

   type OBJECT_REC (OBJECT    : OBJECT_TYPE) is
      record

         case OBJECT is

            when POINT =>

               PT_LOC         :  OBJECT_LOCATION;

            when CIRCLE =>

               CENTER         :  OBJECT_LOCATION;
               RADIUS         :  SYS_PIXEL;

            when RECTANGLE =>
```

```
                UL              :  OBJECT_LOCATION;
                LR              :  OBJECT_LOCATION;

             when LINE =>

                LINE_LOC        :  LIST_OF_POINTS;

             when POLYGON =>

                POL_LOC         :  LIST_OF_POINTS;

          end case;

    end record;

    type OBJECT_POINTER is access OBJECT_REC;

    type OBJECT_DESC is
       record
          NAME                  :  STRING (1..21);
          OBJ_TYPE              :  OBJECT_TYPE;
          MENU_ID               :  SYS_WINDOW_ELE_ID;
          OBJECT                :  OBJECT_POINTER;

    end record;

end OBJECT;
```

```
--cpc package specification name: OBS_SYSTEM
--
--cpc description: Defines types and objects that are common to the obstacle
--                 display system.
--
--cpc design notes:
--
--cpc package author: Bruce Packard
--                    Science Applications International Corporation
--                    424 Delaware, Suite C3
--                    Leavenworth, KS 66048
--
with SYSTEM_PACKAGE;          use SYSTEM_PACKAGE;
with SDB_SITUATION_DB;        use SDB_SITUATION_DB;

package OBS_SYSTEM is

   -- Obstacle display options
   type OBS_OBSTACLE_OPTIONS is
      record
         OBS_BLUEFOR                  :       BOOLEAN;
         OBS_OPFOR                    :       BOOLEAN;
   end record;

   -- Displayed obstacles
   OBS_OBSTACLE_COUNT           :       SDB_OBSTACLE_ID;
   OBS_CURR_OBSTACLE            :       SDB_OBSTACLE_POINT;
   OBS_OBSTACLE_DISPLAYED       :       array (SDB_OBSTACLE_ID) of BOOLEAN;

   -- Current obstacle display options
   OBS_CURR_OBSTACLE_OPTION     :    OBS_OBSTACLE_OPTIONS;

   -- Obstacle popup menus (Blue and OPFOR)
   OBS_BL_MENU_ID               :    SYS_WINDOW_ELE_ID;
   OBS_BL_MENU_START            :    SYS_POP_UP_START_PTR := new
                                     SYS_POP_UP_START (SYS_OBS_MENU);
   OBS_BL_MENU_LENGTH           :    SYS_POP_UP_LENGTH_PTR := new
                                     SYS_POP_UP_LENGTH (SYS_OBS_MENU);
   OBS_BL_POP_UP_TEXT           :    SYS_MENU_TEXT_PTR := new
                                     SYS_MENU_TEXT (SYS_OBS_CELL);
   OBS_BL_POP_UP_CHILD          :    SYS_POP_UP_CHILD_PTR := new
                                     SYS_POP_UP_CHILD (SYS_OBS_CELL);
   OBS_BL_POP_UP_OPTION         :    SYS_OBS_OPTION_PTR := new
                                     SYS_OBS_OPTION_ARRAY (SYS_OBS_CELL);
   OBS_OP_MENU_ID               :    SYS_WINDOW_ELE_ID;
   OBS_OP_MENU_START            :    SYS_POP_UP_START_PTR := new
                                     SYS_POP_UP_START (SYS_OBS_MENU);
   OBS_OP_MENU_LENGTH           :    SYS_POP_UP_LENGTH_PTR := new
                                     SYS_POP_UP_LENGTH (SYS_OBS_MENU);
   OBS_OP_POP_UP_TEXT           :    SYS_MENU_TEXT_PTR := new
                                     SYS_MENU_TEXT (SYS_OBS_CELL);
   OBS_OP_POP_UP_CHILD          :    SYS_POP_UP_CHILD_PTR := new
                                     SYS_POP_UP_CHILD (SYS_OBS_CELL);
   OBS_OP_POP_UP_OPTION         :    SYS_OBS_OPTION_PTR := new
                                     SYS_OBS_OPTION_ARRAY (SYS_OBS_CELL);
```

```
end OBS_SYSTEM;
```

```
--cpc package specification name: UCC_COORD_CONVERT
--
--cpc description: UCC_COORD_CONVERT contains the utilities to perform the
--                 following conversions:
--                      World Coordinate   to   Military Grid
--                      World Coordinate   to   Pixel
--                      Military Grid      to   World Coordinate
--                      Pixel              to   World Coordinate
--
--cpc design notes:
--    This package raises the SYS_UCC_EXCEPTION when an exception is detected.
--
--cpc package author: Bruce Packard
--                   Science Applications International Corporation
--                   424 Delaware, Suite C3
--                   Leavenworth, KS 66048
--
with SYSTEM_PACKAGE;              use SYSTEM_PACKAGE;

package UCC_COORD_CONVERT is

   procedure UCC_DEFINE_MAP_AREA (
                     WORLD_LL_X      :     in    SYS_COORDINATE;
                     WORLD_LL_Y      :     in    SYS_COORDINATE;
                     WORLD_UR_X      :     in    SYS_COORDINATE;
                     WORLD_UR_Y      :     in    SYS_COORDINATE);
   --
   -- CPM description: Informs the conversion software of the lower left and
   --                  upper right corners of the digital map.
   --
   -- formal parameters
   --IN    WORLD_LL_X       The X coordinate of the lower left corner of the
   --                       digital map in world coordinates.
   --
   --IN    WORLD_LL_Y       The Y coordinate of the lower left corner of the
   --                       digital map in world coordinates.
   --
   --IN    WORLD_UR_X       The X coordinate of the upper right corner of the
   --                       digital map in world coordinates.
   --
   --IN    WORLD_UR_X       The X coordinate of the upper right corner of the
   --                       digital map in world coordinates.
   --

  procedure UCC_DEFINE_MAP_DISPLAY (
                     MAP_SCALE       :     in    FLOAT;
                     WORLD_LL_X      :     in    SYS_COORDINATE;
                     WORLD_LL_Y      :     in    SYS_COORDINATE;
                     PIXEL_LL_X      :     in    SYS_WINDOW_COLUMN;
                     PIXEL_LL_Y      :     in    SYS_WINDOW_ROW;
                     PIXEL_UR_X      :     in    SYS_WINDOW_COLUMN;
                     PIXEL_UR_Y      :     in    SYS_WINDOW_ROW);
  --
  -- CPM description: Informs the conversion software of the lower left and
  --                  upper right corners of the digital map display panel.
  --
```

```
-- formal parameters
--IN     MAP_SCALE          The number of meters per pixel for the map
--
--IN     WORLD_LL_X         The X coordinate of the lower left corner of the
--                          digital map display in world coordinates.
--
--IN     WORLD_LL_Y         The Y coordinate of the lower left corner of the
--                          digital map display in world coordinates.
--
--IN     PIXEL_LL_X         The X coordinate of the lower left corner of the
--                          digital map display in pixels.
--
--IN     PIXEL_LL_Y         The Y coordinate of the lower left corner of the
--                          digital map display in pixels.
--
--IN     PIXEL_UR_X         The X coordinate of the upper right corner of the
--                          digital map display in pixels.
--
--IN     PIXEL_UR_X         The X coordinate of the upper right corner of the
--                          digital map display in pixels.
--

procedure UCC_MIL_GRID_TO_WORLD (
                    UTM_LETTER      :     in    string;
                    UTM_X           :     in    SYS_UTM_COORD;
                    UTM_Y           :     in    SYS_UTM_COORD;
                    WORLD_X         :     out   SYS_COORDINATE;
                    WORLD_Y         :     out   SYS_COORDINATE);
--
-- CPM description: Converts military grid coordinates to world coordinates.
--
-- formal parameters
--IN     UTM_LETTER         The two-letter designation part of the military
--                          grid coordinate.
--
--IN     UTM_X              The military grid X coordinate.
--
--IN     UTM_Y              The military grid Y coordinate.
--
--OUT    WORLD_X            The World X Coordinate.
--
--OUT    WORLD_Y            The World Y Coordinate.
--

procedure UCC_PIXEL_TO_WORLD (
                    PIXEL_X         :     in    SYS_PIXEL;
                    PIXEL_Y         :     in    SYS_PIXEL;
                    WORLD_X         :     out   SYS_COORDINATE;
                    WORLD_Y         :     out   SYS_COORDINATE);
--
-- CPM description: Converts pixel coordinates to world coordinates.
--
-- formal parameters
--IN     PIXEL_X            The pixel X coordinate.
--
--IN     PIXEL_Y            The pixel Y coordinate.
```

```
    --
    --OUT    WORLD_X          The World X Coordinate.
    --
    --OUT    WORLD_Y          The World Y Coordinate.
    --

    procedure UCC_WORLD_TO_MIL_GRID (
                        WORLD_X          :     in    SYS_COORDINATE;
                        WORLD_Y          :     in    SYS_COORDINATE;
                        UTM_LETTER       :     out   string;
                        UTM_X            :     out   SYS_UTM_COORD;
                        UTM_Y            :     out   SYS_UTM_COORD);
    --
    -- CPM description: Converts world coordinates to military grid coordinates.
    --
    -- formal parameters
    --IN     WORLD_X          The World X Coordinate.
    --
    --IN     WORLD_Y          The World Y Coordinate.
    --
    --OUT    UTM_LETTER       The two-letter designation part of the military
    --                       grid coordinate.
    --
    --OUT    UTM_X            The military grid X coordinate.
    --
    --OUT    UTM_Y            The military grid Y coordinate.
    --

    procedure UCC_WORLD_TO_PIXEL (
                        WORLD_X          :     in    SYS_COORDINATE;
                        WORLD_Y          :     in    SYS_COORDINATE;
                        PIXEL_X          :     out   SYS_PIXEL;
                        PIXEL_Y          :     out   SYS_PIXEL);
    --
    -- CPM description: Converts world coordinates to pixel coordinates.
    --
    -- formal parameters
    --IN     WORLD_X          The World X Coordinate.
    --
    --IN     WORLD_Y          The World Y Coordinate.
    --
    --OUT    PIXEL_X          The pixel X coordinate.
    --
    --OUT    PIXEL_Y          The pixel Y coordinate.
    --


end UCC_COORD_CONVERT;
```

```
    --
    --OUT    WORLD_X          The World X Coordinate.
    --
    --OUT    WORLD_Y          The World Y Coordinate.
    --

    procedure UCC_WORLD_TO_MIL_GRID (
                        WORLD_X          :     in    SYS_COORDINATE;
                        WORLD_Y          :     in    SYS_COORDINATE;
                        UTM_LETTER       :     out   string;
                        UTM_X            :     out   SYS_UTM_COORD;
                        UTM_Y            :     out   SYS_UTM_COORD);
    --
    -- CPM description: Converts world coordinates to military grid coordinates.
    --
    -- formal parameters
    --IN     WORLD_X          The World X Coordinate.
    --
    --IN     WORLD_Y          The World Y Coordinate.
    --
    --OUT    UTM_LETTER       The two-letter designation part of the military
    --                       grid coordinate.
    --
    --OUT    UTM_X            The military grid X coordinate.
    --
    --OUT    UTM_Y            The military grid Y coordinate.
    --

    procedure UCC_WORLD_TO_PIXEL (
                        WORLD_X          :     in    SYS_COORDINATE;
                        WORLD_Y          :     in    SYS_COORDINATE;
                        PIXEL_X          :     out   SYS_PIXEL;
                        PIXEL_Y          :     out   SYS_PIXEL);
    --
    -- CPM description: Converts world coordinates to pixel coordinates.
    --
    -- formal parameters
    --IN     WORLD_X          The World X Coordinate.
    --
    --IN     WORLD_Y          The World Y Coordinate.
    --
    --OUT    PIXEL_X          The pixel X coordinate.
    --
    --OUT    PIXEL_Y          The pixel Y coordinate.
    --


end UCC_COORD_CONVERT;
```

```
--cpc package specification name: UCE_CNTRL_MSR_EDITOR
--
--cpc description: UCE_CNTRL_MSR_EDITOR contains the low level control
--                 measure for displaying specific types of control measures.
--
--cpc design notes:
--    This package raises the SYS_UCE_EXCEPTION when an exception is detected.
--
--cpc package author: Bruce Packard
--                   Science Applications International Corporation
--                   424 Delaware, Suite C3
--                   Leavenworth, KS 66048
--
with SYSTEM_PACKAGE;              use SYSTEM_PACKAGE;
with SDB_SITUATION_DB;            use SDB_SITUATION_DB;
with UCC_COORD_CONVERT;           use UCC_COORD_CONVERT;
with UIW_IMAGE_WINDOW;            use UIW_IMAGE_WINDOW;

package UCE_CNTRL_MSR_EDITOR is

   procedure UCE_STATUS (
                  ADD_FLAG          :     in    BOOLEAN;
                  NAME              :     in    STRING;
                  ECHELON           :     in    SDB_FORCE_ECHELON;
                  CM_TYPE           :     in    SDB_CONTROL_MEASURE_TYPE) ;
   --
   -- CPM description: Displays the status of a control measure.
   --
   -- formal parameters
   --IN    ADD_FLAG          Add or erase the control measure flag
   --                        True = Add;  False = Erase
   --
   --IN    NAME              The name of the control measure.
   --
   --IN    ECHELON           The echelon of the control measure.
   --
   --IN    CM_TYPE           The type of the control measure.
   --


-- The following are considered area control measure routines:

      procedure UCE_AREA_OF_OPER (
                     ADD_FLAG          :     in    BOOLEAN;
                     CNTRL_MSR_DESC    :     in    SDB_CONTROL_MEASURE_REC);
      --
      -- CPM description: Displays an area of operations on the digital map.
      --
      -- formal parameters
      --IN    ADD_FLAG          Add or erase the control measure flag
      --                        True = Add;  False = Erase
      --
      --IN    CNTRL_MSR_DESC    The description of the area of operations.
      --

      procedure UCE_ASSEMBLY_AREA (
```

```
                ADD_FLAG          :     in    BOOLEAN;
                CNTRL_MSR_DESC    :     in    SDB_CONTROL_MEASURE_REC);
--
-- CPM description: Displays an assembly area on the digital map.
--
-- formal parameters
--IN    ADD_FLAG          Add or erase the control measure flag
--                       True = Add;  False = Erase
--
--IN    CNTRL_MSR_DESC    The description of the assembly area.
--

procedure UCE_ATTACK_POSITION (
                ADD_FLAG          :     in    BOOLEAN;
                CNTRL_MSR_DESC    :     in    SDB_CONTROL_MEASURE_REC);
--
-- CPM description: Displays an attack position on the digital map.
--
-- formal parameters
--IN    ADD_FLAG          Add or erase the control measure flag
--                       True = Add;  False = Erase
--
--IN    CNTRL_MSR_DESC    The description of the attack position.
--

procedure UCE_BATTLE_POSITION (
                ADD_FLAG          :     in    BOOLEAN;
                CNTRL_MSR_DESC    :     in    SDB_CONTROL_MEASURE_REC);
--
-- CPM description: Displays a battle position on the digital map.
--
-- formal parameters
--IN    ADD_FLAG          Add or erase the control measure flag
--                       True = Add;  False = Erase
--
--IN    CNTRL_MSR_DESC    The description of the battle position.
--

procedure UCE_BDE_SUPT_AREA (
                ADD_FLAG          :     in    BOOLEAN;
                CNTRL_MSR_DESC    :     in    SDB_CONTROL_MEASURE_REC);
--
-- CPM description: Displays the brigade support area on the digital map.
--
-- formal parameters
--IN    ADD_FLAG          Add or erase the control measure flag
--                       True = Add;  False = Erase
--
--IN    CNTRL_MSR_DESC    The description of the brigade support area.
--

procedure UCE_BN_SUPT_AREA (
                ADD_FLAG          :     in    BOOLEAN;
                CNTRL_MSR_DESC    :     in    SDB_CONTROL_MEASURE_REC);
--
-- CPM description: Displays the battalion support area on the digital map.
```

```
--
-- formal parameters
--IN     ADD_FLAG          Add or erase the control measure flag
--                         True = Add;  False = Erase
--
--IN     CNTRL_MSR_DESC    The description of the battalion support area.
--

procedure UCE_DIV_SUPT_AREA (
                ADD_FLAG          :     in    BOOLEAN;
                CNTRL_MSR_DESC    :     in    SDB_CONTROL_MEASURE_REC);
--
-- CPM description: Displays the division support area on the digital map.
--
-- formal parameters
--IN     ADD_FLAG          Add or erase the control measure flag
--                         True = Add;  False = Erase
--
--IN     CNTRL_MSR_DESC    The description of the division support area.
--

procedure UCE_DROP_ZONE (
                ADD_FLAG          :     in    BOOLEAN;
                CNTRL_MSR_DESC    :     in    SDB_CONTROL_MEASURE_REC);
--
-- CPM description: Displays a drop zone on the digital map.
--
-- formal parameters
--IN     ADD_FLAG          Add or erase the control measure flag
--                         True = Add;  False = Erase
--
--IN     CNTRL_MSR_DESC    The description of the drop zone.
--

procedure UCE_FREE_FIRE_AREA (
                ADD_FLAG          :     in    BOOLEAN;
                CNTRL_MSR_DESC    :     in    SDB_CONTROL_MEASURE_REC);
--
-- CPM description: Displays a free fire area on the digital map.
--
-- formal parameters
--IN     ADD_FLAG          Add or erase the control measure flag
--                         True = Add;  False = Erase
--
--IN     CNTRL_MSR_DESC    The description of the free fire area.
--

procedure UCE_LANDING_ZONE (
                ADD_FLAG          :     in    BOOLEAN;
                CNTRL_MSR_DESC    :     in    SDB_CONTROL_MEASURE_REC);
--
-- CPM description: Displays a landing zone on the digital map.
--
-- formal parameters
--IN     ADD_FLAG          Add or erase the control measure flag
--                         True = Add;  False = Erase
```

```
--
--IN     CNTRL_MSR_DESC    The description of the landing zone.
--

procedure UCE_NO_FIRE_AREA (
                ADD_FLAG          :     in     BOOLEAN;
                CNTRL_MSR_DESC    :     in     SDB_CONTROL_MEASURE_REC);
--
-- CPM description: Displays a no fire area on the digital map.
--
-- formal parameters
--IN     ADD_FLAG          Add or erase the control measure flag
--                         True = Add;  False = Erase
--
--IN     CNTRL_MSR_DESC    The description of the no fire area.
--

procedure UCE_OBJECTIVE (
                ADD_FLAG          :     in     BOOLEAN;
                CNTRL_MSR_DESC    :     in     SDB_CONTROL_MEASURE_REC);
--
-- CPM description: Displays an objective on the digital map.
--
-- formal parameters
--IN     ADD_FLAG          Add or erase the control measure flag
--                         True = Add;  False = Erase
--
--IN     CNTRL_MSR_DESC    The description of the objective.
--

procedure UCE_RESTRICT_FIRE_AREA (
                ADD_FLAG          :     in     BOOLEAN;
                CNTRL_MSR_DESC    :     in     SDB_CONTROL_MEASURE_REC);
--
-- CPM description: Displays a restricted fire area on the digital map.
--
-- formal parameters
--IN     ADD_FLAG          Add or erase the control measure flag
--                         True = Add;  False = Erase
--
--IN     CNTRL_MSR_DESC    The description of the restricted fire area.
--

procedure UCE_ZONE_OF_ACTION (
                ADD_FLAG          :     in     BOOLEAN;
                CNTRL_MSR_DESC    :     in     SDB_CONTROL_MEASURE_REC);
--
-- CPM description: Displays a zone of action on the digital map.
--
-- formal parameters
--IN     ADD_FLAG          Add or erase the control measure flag
--                         True = Add;  False = Erase
--
--IN     CNTRL_MSR_DESC    The description of the zone of action.
--
```

```
-- end area control measures;

-- The following are considered crossing control measure routines:

      procedure UCE_ASSAULT_CROSS (
                    ADD_FLAG          :     in    BOOLEAN;
                    CNTRL_MSR_DESC    :     in    SDB_CNTRL_MSR_POINT_REC);
      --
      -- CPM description: Displays an assault crossing on the digital map.
      --
      -- formal parameters
      --IN    ADD_FLAG          Add or erase the control measure flag
      --                       True = Add;  False = Erase
      --
      --IN    CNTRL_MSR_DESC    The description of the assault crossing.
      --

      procedure UCE_RAFT_SITE (
                    ADD_FLAG          :     in    BOOLEAN;
                    CNTRL_MSR_DESC    :     in    SDB_CNTRL_MSR_POINT_REC);
      --
      -- CPM description: Displays a raft site on the digital map.
      --
      -- formal parameters
      --IN    ADD_FLAG          Add or erase the control measure flag
      --                       True = Add;  False = Erase
      --
      --IN    CNTRL_MSR_DESC    The description of the raft site.
      --

--  end crossing control measures;

-- The following are considered fire plan control measure routines:

      procedure UCE_GROUP_OF_TARGETS (
                    ADD_FLAG          :     in    BOOLEAN;
                    CNTRL_MSR_DESC    :     in    SDB_CONTROL_MEASURE_REC);
      --
      -- CPM description: Displays a group of targets on the digital map.
      --
      -- formal parameters
      --IN    ADD_FLAG          Add or erase the control measure flag
      --                       True = Add;  False = Erase
      --
      --IN    CNTRL_MSR_DESC    The description of the group of targets.
      --

--  end fire plan control measures;

-- The following are considered line control measure routines:

      procedure UCE_BOUNDARY (
```

```
                ADD_FLAG          :     in    BOOLEAN;
                CNTRL_MSR_DESC    :     in    SDB_CONTROL_MEASURE_REC);
--
-- CPM description: Displays a boundary on the digital map.
--
-- formal parameters
--IN    ADD_FLAG          Add or erase the control measure flag
--                       True = Add;  False = Erase
--
--IN    CNTRL_MSR_DESC    The description of the boundary.
--

procedure UCE_BRIDGEHEAD_LINE (
                ADD_FLAG          :     in    BOOLEAN;
                CNTRL_MSR_DESC    :     in    SDB_CONTROL_MEASURE_REC);
--
-- CPM description: Displays a bridgehead line on the digital map.
--
-- formal parameters
--IN    ADD_FLAG          Add or erase the control measure flag
--                       True = Add;  False = Erase
--
--IN    CNTRL_MSR_DESC    The description of the bridgehead line.
--

procedure UCE_COA_LINE (
                ADD_FLAG          :     in    BOOLEAN;
                CNTRL_MSR_DESC    :     in    SDB_CONTROL_MEASURE_REC);
--
-- CPM description: Displays a COA line on the digital map.
--
-- formal parameters
--IN    ADD_FLAG          Add or erase the control measure flag
--                       True = Add;  False = Erase
--
--IN    CNTRL_MSR_DESC    The description of the bridgehead line.
--

procedure UCE_COORD_FIRE_LINE (
                ADD_FLAG          :     in    BOOLEAN;
                CNTRL_MSR_DESC    :     in    SDB_CONTROL_MEASURE_REC);
--
-- CPM description: Displays a coordination fire line on the digital map.
--
-- formal parameters
--IN    ADD_FLAG          Add or erase the control measure flag
--                       True = Add;  False = Erase
--
--IN    CNTRL_MSR_DESC    The description of the coordination fire line.
--

procedure UCE_FEBA (
                ADD_FLAG          :     in    BOOLEAN;
                CNTRL_MSR_DESC    :     in    SDB_CONTROL_MEASURE_REC);
--
-- CPM description: Displays the forward edge of the battle area on the
```

```
--                  digital map.
--
-- formal parameters
--IN    ADD_FLAG          Add or erase the control measure flag
--                       True = Add;  False = Erase
--
--IN    CNTRL_MSR_DESC    The description of the FEBA.
--

procedure UCE_FIRE_SUP_COORD_LN (
                ADD_FLAG          :     in    BOOLEAN;
                CNTRL_MSR_DESC    :     in    SDB_CONTROL_MEASURE_REC);
--
-- CPM description: Displays a fire support coordination line on the
--                  digital map.
--
-- formal parameters
--IN    ADD_FLAG          Add or erase the control measure flag
--                       True = Add;  False = Erase
--
--IN    CNTRL_MSR_DESC    The description of the fire support coordination
--                       line.
--

procedure UCE_FLOT (
                ADD_FLAG          :     in    BOOLEAN;
                CNTRL_MSR_DESC    :     in    SDB_CONTROL_MEASURE_REC);
--
-- CPM description: Displays a forward line of troops on the digital map.
--
-- formal parameters
--IN    ADD_FLAG          Add or erase the control measure flag
--                       True = Add;  False = Erase
--
--IN    CNTRL_MSR_DESC    The description of the forward line of troops.
--

procedure UCE_HOLDING_LINE (
                ADD_FLAG          :     in    BOOLEAN;
                CNTRL_MSR_DESC    :     in    SDB_CONTROL_MEASURE_REC);
--
-- CPM description: Displays a holding line on the digital map.
--
-- formal parameters
--IN    ADD_FLAG          Add or erase the control measure flag
--                       True = Add;  False = Erase
--
--IN    CNTRL_MSR_DESC    The description of the holding line.
--

procedure UCE_LIGHT_LINE (
                ADD_FLAG          :     in    BOOLEAN;
                CNTRL_MSR_DESC    :     in    SDB_CONTROL_MEASURE_REC);
--
-- CPM description: Displays a light line on the digital map.
--
```

```
-- formal parameters
--IN    ADD_FLAG          Add or erase the control measure flag
--                       True = Add;  False = Erase
--
--IN    CNTRL_MSR_DESC    The description of the light line.
--

procedure UCE_LIMIT_OF_ADV (
                ADD_FLAG          :     in    BOOLEAN;
                CNTRL_MSR_DESC    :     in    SDB_CONTROL_MEASURE_REC);
--
-- CPM description: Displays the limit of advance on the digital map.
--
-- formal parameters
--IN    ADD_FLAG          Add or erase the control measure flag
--                       True = Add;  False = Erase
--
--IN    CNTRL_MSR_DESC    The description of the limit of advance.
--

procedure UCE_LINE_OF_CONTACT (
                ADD_FLAG          :     in    BOOLEAN;
                CNTRL_MSR_DESC    :     in    SDB_CONTROL_MEASURE_REC);
--
-- CPM description: Displays a line of contact on the digital map.
--
-- formal parameters
--IN    ADD_FLAG          Add or erase the control measure flag
--                       True = Add;  False = Erase
--
--IN    CNTRL_MSR_DESC    The description of the line of contact.
--

procedure UCE_LINE_OF_DEPART (
                ADD_FLAG          :     in    BOOLEAN;
                CNTRL_MSR_DESC    :     in    SDB_CONTROL_MEASURE_REC);
--
-- CPM description: Displays a line of departure on the digital map.
--
-- formal parameters
--IN    ADD_FLAG          Add or erase the control measure flag
--                       True = Add;  False = Erase
--
--IN    CNTRL_MSR_DESC    The description of the line of departure.
--

procedure UCE_PHASE_LINE (
                ADD_FLAG          :     in    BOOLEAN;
                CNTRL_MSR_DESC    :     in    SDB_CONTROL_MEASURE_REC);
--
-- CPM description: Displays a phase line on the digital map.
--
-- formal parameters
--IN    ADD_FLAG          Add or erase the control measure flag
--                       True = Add;  False = Erase
--
```

```
        --IN    CNTRL_MSR_DESC   The description of the phase line.
        --

        procedure UCE_RESTRICT_FIRE_LINE (
                        ADD_FLAG          :     in    BOOLEAN;
                        CNTRL_MSR_DESC    :     in    SDB_CONTROL_MEASURE_REC);
        --
        -- CPM description: Displays a restricted fire line on the digital map.
        --
        -- formal parameters
        --IN    ADD_FLAG          Add or erase the control measure flag
        --                       True = Add;  False = Erase
        --
        --IN    CNTRL_MSR_DESC    The description of the restricted fire line.
        --

--  end line control measures;


-- The following are considered map feature control measure routines:

        procedure UCE_AIR_FIELD (
                        ADD_FLAG          :     in    BOOLEAN;
                        CNTRL_MSR_DESC    :     in    SDB_CNTRL_MSR_POINT_REC);
        --
        -- CPM description: Displays an air field on the digital map.
        --
        -- formal parameters
        --IN    ADD_FLAG          Add or erase the control measure flag
        --                       True = Add;  False = Erase
        --
        --IN    CNTRL_MSR_DESC    The description of the air field.
        --

        procedure UCE_BRIDGE (
                        ADD_FLAG          :     in    BOOLEAN;
                        CNTRL_MSR_DESC    :     in    SDB_CNTRL_MSR_POINT_REC);
        --
        -- CPM description: Displays a bridge on the digital map.
        --
        -- formal parameters
        --IN    ADD_FLAG          Add or erase the control measure flag
        --                       True = Add;  False = Erase
        --
        --IN    CNTRL_MSR_DESC    The description of the bridge.
        --

        procedure UCE_BUILDING (
                        ADD_FLAG          :     in    BOOLEAN;
                        CNTRL_MSR_DESC    :     in    SDB_CNTRL_MSR_POINT_REC);
        --
        -- CPM description: Displays a building on the digital map.
        --
        -- formal parameters
        --IN    ADD_FLAG          Add or erase the control measure flag
        --                       True = Add;  False = Erase
```

```
--
--IN    CNTRL_MSR_DESC   The description of the building.
--

procedure UCE_CITY (
                ADD_FLAG          :     in    BOOLEAN;
                CNTRL_MSR_DESC    :     in    SDB_CNTRL_MSR_POINT_REC);
--
-- CPM description: Displays a city on the digital map.
--
-- formal parameters
--IN    ADD_FLAG          Add or erase the control measure flag
--                       True = Add;  False = Erase
--
--IN    CNTRL_MSR_DESC    The description of the city.
--

procedure UCE_LAKE (
                ADD_FLAG          :     in    BOOLEAN;
                CNTRL_MSR_DESC    :     in    SDB_CNTRL_MSR_POINT_REC);
--
-- CPM description: Displays a lake on the digital map.
--
-- formal parameters
--IN    ADD_FLAG          Add or erase the control measure flag
--                       True = Add;  False = Erase
--
--IN    CNTRL_MSR_DESC    The description of the lake.
--

procedure UCE_MAP_REF_POINT (
                ADD_FLAG          :     in    BOOLEAN;
                CNTRL_MSR_DESC    :     in    SDB_CNTRL_MSR_POINT_REC);
--
-- CPM description: Displays a map reference point on the digital map.
--
-- formal parameters
--IN    ADD_FLAG          Add or erase the control measure flag
--                       True = Add;  False = Erase
--
--IN    CNTRL_MSR_DESC    The description of the map reference point.
--

procedure UCE_MOUNTAIN_PEAK (
                ADD_FLAG          :     in    BOOLEAN;
                CNTRL_MSR_DESC    :     in    SDB_CNTRL_MSR_POINT_REC);
--
-- CPM description: Displays a mountain peak on the digital map.
--
-- formal parameters
--IN    ADD_FLAG          Add or erase the control measure flag
--                       True = Add;  False = Erase
--
--IN    CNTRL_MSR_DESC    The description of the mountain peak.
--
```

```
      procedure UCE_ROAD_INTERSECT (
                     ADD_FLAG          :     in    BOOLEAN;
                     CNTRL_MSR_DESC    :     in    SDB_CNTRL_MSR_POINT_REC);
      --
      -- CPM description: Displays a road intersection on the digital map.
      --
      -- formal parameters
      --IN    ADD_FLAG          Add or erase the control measure flag
      --                       True = Add;  False = Erase
      --
      --IN    CNTRL_MSR_DESC    The description of the road intersection.
      --

      procedure UCE_TOWN (
                     ADD_FLAG          :     in    BOOLEAN;
                     CNTRL_MSR_DESC    :     in    SDB_CNTRL_MSR_POINT_REC);
      --
      -- CPM description: Displays a town on the digital map.
      --
      -- formal parameters
      --IN    ADD_FLAG          Add or erase the control measure flag
      --                       True = Add;  False = Erase
      --
      --IN    CNTRL_MSR_DESC    The description of the town.
      --

      procedure UCE_VILLAGE (
                     ADD_FLAG          :     in    BOOLEAN;
                     CNTRL_MSR_DESC    :     in    SDB_CNTRL_MSR_POINT_REC);
      --
      -- CPM description: Displays a village on the digital map.
      --
      -- formal parameters
      --IN    ADD_FLAG          Add or erase the control measure flag
      --                       True = Add;  False = Erase
      --
      --IN    CNTRL_MSR_DESC    The description of the village.
      --

--  end map feature control measures;


-- The following are considered point control measures:

      procedure UCE_CHECKPOINT (
                     ADD_FLAG          :     in    BOOLEAN;
                     CNTRL_MSR_DESC    :     in    SDB_CNTRL_MSR_POINT_REC);
      --
      -- CPM description: Displays a checkpoint on the digital map.
      --
      -- formal parameters
      --IN    ADD_FLAG          Add or erase the control measure flag
      --                       True = Add;  False = Erase
      --
      --IN    CNTRL_MSR_DESC    The description of the checkpoint.
      --
```

```
procedure UCE_COLLECT_POINT (
                ADD_FLAG          :     in    BOOLEAN;
                CNTRL_MSR_DESC    :     in    SDB_CNTRL_MSR_POINT_REC);
--
-- CPM description: Displays a collection point on the digital map.
--
-- formal parameters
--IN    ADD_FLAG          Add or erase the control measure flag
--                        True = Add;  False = Erase
--
--IN    CNTRL_MSR_DESC    The description of the collection point.
--

procedure UCE_CONTACT_POINT (
                ADD_FLAG          :     in    BOOLEAN;
                CNTRL_MSR_DESC    :     in    SDB_CNTRL_MSR_POINT_REC);
--
-- CPM description: Displays a contact point on the digital map.
--
-- formal parameters
--IN    ADD_FLAG          Add or erase the control measure flag
--                        True = Add;  False = Erase
--
--IN    CNTRL_MSR_DESC    The description of the contact point.
--

procedure UCE_COORD_POINT (
                ADD_FLAG          :     in    BOOLEAN;
                CNTRL_MSR_DESC    :     in    SDB_CNTRL_MSR_POINT_REC);
--
-- CPM description: Displays a coordination point on the digital map.
--
-- formal parameters
--IN    ADD_FLAG          Add or erase the control measure flag
--                        True = Add;  False = Erase
--
--IN    CNTRL_MSR_DESC    The description of the coordination point.
--

procedure UCE_CRITICAL_EVENT (
                ADD_FLAG          :     in    BOOLEAN;
                CNTRL_MSR_DESC    :     in    SDB_CNTRL_MSR_POINT_REC);
--
-- CPM description: Displays a critical event point on the digital map.
--
-- formal parameters
--IN    ADD_FLAG          Add or erase the control measure flag
--                        True = Add;  False = Erase
--
--IN    CNTRL_MSR_DESC    The description of the coordination point.
--

procedure UCE_LINK_UP_POINT (
                ADD_FLAG          :     in    BOOLEAN;
                CNTRL_MSR_DESC    :     in    SDB_CNTRL_MSR_POINT_REC);
```

```
--
-- CPM description: Displays a linkup point on the digital map.
--
-- formal parameters
--IN     ADD_FLAG          Add or erase the control measure flag
--                         True = Add;  False = Erase
--
--IN     CNTRL_MSR_DESC    The description of the link up point.
--

procedure UCE_PASSAGE_POINT (
                ADD_FLAG          :     in    BOOLEAN;
                CNTRL_MSR_DESC    :     in    SDB_CNTRL_MSR_POINT_REC);
--
-- CPM description: Displays a passage point on the digital map.
--
-- formal parameters
--IN     ADD_FLAG          Add or erase the control measure flag
--                         True = Add;  False = Erase
--
--IN     CNTRL_MSR_DESC    The description of the passage point.
--

procedure UCE_POINT_OF_DEPART (
                ADD_FLAG          :     in    BOOLEAN;
                CNTRL_MSR_DESC    :     in    SDB_CNTRL_MSR_POINT_REC);
--
-- CPM description: Displays a point of departure on the digital map.
--
-- formal parameters
--IN     ADD_FLAG          Add or erase the control measure flag
--                         True = Add;  False = Erase
--
--IN     CNTRL_MSR_DESC    The description of the point of departure.
--

procedure UCE_RELEASE_POINT (
                ADD_FLAG          :     in    BOOLEAN;
                CNTRL_MSR_DESC    :     in    SDB_CNTRL_MSR_POINT_REC);
--
-- CPM description: Displays a release point on the digital map.
--
-- formal parameters
--IN     ADD_FLAG          Add or erase the control measure flag
--                         True = Add;  False = Erase
--
--IN     CNTRL_MSR_DESC    The description of the release point.
--

procedure UCE_START_POINT (
                ADD_FLAG          :     in    BOOLEAN;
                CNTRL_MSR_DESC    :     in    SDB_CNTRL_MSR_POINT_REC);
--
-- CPM description: Displays a start point on the digital map.
--
-- formal parameters
```

```
      --IN    ADD_FLAG          Add or erase the control measure flag
      --                        True = Add;  False = Erase
      --
      --IN    CNTRL_MSR_DESC    The description of the start point.
      --

      procedure UCE_STRONG_POINT (
                     ADD_FLAG          :     in    BOOLEAN;
                     CNTRL_MSR_DESC    :     in    SDB_CNTRL_MSR_POINT_REC);
      --
      -- CPM description: Displays a strong point on the digital map.
      --
      -- formal parameters
      --IN    ADD_FLAG          Add or erase the control measure flag
      --                        True = Add;  False = Erase
      --
      --IN    CNTRL_MSR_DESC    The description of the strong point.
      --

      procedure UCE_TRAFFIC_CNTRL_POINT (
                     ADD_FLAG          :     in    BOOLEAN;
                     CNTRL_MSR_DESC    :     in    SDB_CNTRL_MSR_POINT_REC);
      --
      -- CPM description: Displays a traffic control point on the digital map.
      --
      -- formal parameters
      --IN    ADD_FLAG          Add or erase the control measure flag
      --                        True = Add;  False = Erase
      --
      --IN    CNTRL_MSR_DESC    The description of the traffic control point.
      --

--  end point control measures;


-- The following are considered to be route control measure routines:

      procedure UCE_AIR_AXIS_OF_ADV (
                     ADD_FLAG          :     in    BOOLEAN;
                     CNTRL_MSR_DESC    :     in    SDB_CONTROL_MEASURE_REC);
      --
      -- CPM description: Displays an air axis of advance on the digital map.
      --
      -- formal parameters
      --IN    ADD_FLAG          Add or erase the control measure flag
      --                        True = Add;  False = Erase
      --
      --IN    CNTRL_MSR_DESC    The description of the air axis of advance.
      --

      procedure UCE_AIR_CORRIDOR (
                     ADD_FLAG          :     in    BOOLEAN;
                     CNTRL_MSR_DESC    :     in    SDB_CONTROL_MEASURE_REC);
      --
      -- CPM description: Displays an air corridor on the digital map.
      --
```

```
-- formal parameters
--IN     ADD_FLAG          Add or erase the control measure flag
--                         True = Add;  False = Erase
--
--IN     CNTRL_MSR_DESC    The description of the air corridor.
--

procedure UCE_AXIS_OF_ADV_MAIN (
                ADD_FLAG          :     in    BOOLEAN;
                CNTRL_MSR_DESC    :     in    SDB_CONTROL_MEASURE_REC);
--
-- CPM description: Displays the main axis of advance on the digital map.
--
-- formal parameters
--IN     ADD_FLAG          Add or erase the control measure flag
--                         True = Add;  False = Erase
--
--IN     CNTRL_MSR_DESC    The description of the main axis of advance.
--

procedure UCE_AXIS_OF_ADV_SUPT (
                ADD_FLAG          :     in    BOOLEAN;
                CNTRL_MSR_DESC    :     in    SDB_CONTROL_MEASURE_REC);
--
-- CPM description: Displays the support axis of advance on the digital map.
--
-- formal parameters
--IN     ADD_FLAG          Add or erase the control measure flag
--                         True = Add;  False = Erase
--
--IN     CNTRL_MSR_DESC    The description of the support axis of advance.
--

procedure UCE_DIRECT_OF_ATTACK (
                ADD_FLAG          :     in    BOOLEAN;
                CNTRL_MSR_DESC    :     in    SDB_CONTROL_MEASURE_REC);
--
-- CPM description: Displays the direction of attack on the digital map.
--
-- formal parameters
--IN     ADD_FLAG          Add or erase the control measure flag
--                         True = Add;  False = Erase
--
--IN     CNTRL_MSR_DESC    The description of the direction of attack.
--

procedure UCE_FEINT (
                ADD_FLAG          :     in    BOOLEAN;
                CNTRL_MSR_DESC    :     in    SDB_CONTROL_MEASURE_REC);
--
-- CPM description: Displays the FEINT on the digital map.
--
-- formal parameters
--IN     ADD_FLAG          Add or erase the control measure flag
--                         True = Add;  False = Erase
--
```

```
        --IN    CNTRL_MSR_DESC   The description of the FEINT.
        --

        procedure UCE_MAIN_SUPPLY_RTE (
                        ADD_FLAG          :     in    BOOLEAN;
                        CNTRL_MSR_DESC    :     in    SDB_CONTROL_MEASURE_REC);
        --
        -- CPM description: Displays the main supply route on the digital map.
        --
        -- formal parameters
        --IN    ADD_FLAG          Add or erase the control measure flag
        --                        True = Add;  False = Erase
        --
        --IN    CNTRL_MSR_DESC    The description of the main supply route.
        --

        procedure UCE_ROUTE (
                        ADD_FLAG          :     in    BOOLEAN;
                        CNTRL_MSR_DESC    :     in    SDB_CONTROL_MEASURE_REC);
        --
        -- CPM description: Displays a route on the digital map.
        --
        -- formal parameters
        --IN    ADD_FLAG          Add or erase the control measure flag
        --                        True = Add;  False = Erase
        --
        --IN    CNTRL_MSR_DESC    The description of the route.
        --

--  end route control measures;

end UCE_CNTRL_MSR_EDITOR;
```

```
--cpc package specification name: UCM_CONTROL_MEASURE
--
--cpc description: UCM_CONTROL_MEASURE is the intermediate level control measure
--                 display package that is responsible for displaying and
--                 erasing control measures on the digital map.
--
--cpc design notes:
--    This package raises the SYS_UCM_EXCEPTION when an exception is detected.
--
--cpc package author: Bruce Packard
--                   Science Applications International Corporation
--                   424 Delaware, Suite C3
--                   Leavenworth, KS 66048
--
with SYSTEM_PACKAGE;              use SYSTEM_PACKAGE;
with SDB_SITUATION_DB;            use SDB_SITUATION_DB;
with UWN_WINDOW_SYSTEM;           use UWN_WINDOW_SYSTEM;

package UCM_CONTROL_MEASURE is

   procedure UCM_DEFINE_AREA (
                CNTRL_MSR_TYPE      :   in      SDB_CONTROL_MEASURE_TYPE;
                CNTRL_MSR_SIDE      :   in      SDB_SIDE_TYPE;
                ECH_WINDOW          :  out      SYS_WINDOW_ELE_ID);
   --
   -- CPM description: Controls the user interface to define a new area control
   --                  measure.
   --
   -- formal parameters
   --IN    CNTRL_MSR_TYPE     Location type of the control measures to define
   --
   --IN    CNTRL_MSR_SIDE     Force of the control measures to define
   --
   --OUT   ECH_WINDOW         Id of the echelon selection menu
   --

   procedure UCM_DEFINE_CROSSING (
                CNTRL_MSR_TYPE      :   in      SDB_CONTROL_MEASURE_TYPE;
                CNTRL_MSR_SIDE      :   in      SDB_SIDE_TYPE;
                ECH_WINDOW          :  out      SYS_WINDOW_ELE_ID);
   --
   -- CPM description: Controls the user interface to define a new crossing
   --                  control measure.
   --
   -- formal parameters
   --IN    CNTRL_MSR_TYPE     Location type of the control measures to define
   --
   --IN    CNTRL_MSR_SIDE     Force of the control measures to define.
   --
   --OUT   ECH_WINDOW         Id of the echelon selection menu
   --

   procedure UCM_DEFINE_FIRE_PLAN (
                CNTRL_MSR_TYPE      :   in      SDB_CONTROL_MEASURE_TYPE;
                CNTRL_MSR_SIDE      :   in      SDB_SIDE_TYPE;
                ECH_WINDOW          :  out      SYS_WINDOW_ELE_ID);
```

```
--
-- CPM description: Controls the user interface to define a new fire plan
--                  control measure.
--
-- formal parameters
--IN    CNTRL_MSR_TYPE     Location type of the control measures to define
--
--IN    CNTRL_MSR_SIDE     Force of the control measures to define
--
--OUT   ECH_WINDOW         Id of the echelon selection menu
--

procedure UCM_DEFINE_LINE (
              CNTRL_MSR_TYPE      :   in      SDB_CONTROL_MEASURE_TYPE;
              CNTRL_MSR_SIDE      :   in      SDB_SIDE_TYPE;
              ECH_WINDOW          :  out      SYS_WINDOW_ELE_ID);
--
-- CPM description: Controls the user interface to define a new line
--                  control measure.
--
-- formal parameters
--IN    CNTRL_MSR_TYPE     Location type of the control measures to define
--
--IN    CNTRL_MSR_SIDE     Force of the control measures to define
--
--OUT   ECH_WINDOW         Id of the echelon selection menu
--

procedure UCM_DEFINE_MAP_FEAT (
              CNTRL_MSR_TYPE      :   in      SDB_CONTROL_MEASURE_TYPE;
              CNTRL_MSR_SIDE      :   in      SDB_SIDE_TYPE;
              SCALE_WINDOW        :  out      SYS_WINDOW_ELE_ID);
--
-- CPM description: Controls the user interface to define a new map feature
--                  control measure.
--
-- formal parameters
--IN    CNTRL_MSR_TYPE     Location type of the control measures to define
--
--IN    CNTRL_MSR_SIDE     Force of the control measures to define
--
--OUT   SCALE_WINDOW       Id of the map scale selection menu
--

procedure UCE_DEFINE_NEXT_POINT (
             X_PIXEL      :   in      SYS_IMAGE_COLUMN;
             Y_PIXEL      :   in      SYS_IMAGE_ROW);
--
-- CPM description: Defines the cursor location as the next point in a
--                  control measure definition.
--
--
-- formal parameters
--IN    X_PIXEL     Pixel X coordinate of the selected point
--
--IN    Y_PIXEL     Pixel Y coordinate of the selected point
```

```
--
procedure UCM_DEFINE_POINT (
              CNTRL_MSR_TYPE      :   in      SDB_CONTROL_MEASURE_TYPE;
              CNTRL_MSR_SIDE      :   in      SDB_SIDE_TYPE;
              ECH_WINDOW          :  out      SYS_WINDOW_ELE_ID);
--
-- CPM description: Controls the user interface to define a new point
--                  control measure.
--
-- formal parameters
--IN    CNTRL_MSR_TYPE     Location type of the control measures to define
--
--IN    CNTRL_MSR_SIDE     Force of the control measures to define
--
--OUT   ECH_WINDOW         Id of the echelon selection menu
--

procedure UCM_DEFINE_ROUTE (
              CNTRL_MSR_TYPE      :   in      SDB_CONTROL_MEASURE_TYPE;
              CNTRL_MSR_SIDE      :   in      SDB_SIDE_TYPE;
              ECH_WINDOW          :  out      SYS_WINDOW_ELE_ID);
--
-- CPM description: Controls the user interface to define a new route
--                  control measure.
--
-- formal parameters
--IN    CNTRL_MSR_TYPE     Location type of the control measures to define
--
--IN    CNTRL_MSR_SIDE     Force of the control measures to define
--
--OUT   ECH_WINDOW         Id of the echelon selection menu
--

procedure UCM_DISPLAY_CNTRL_MSR (
              CNTRL_MSR_ECHELON   :   in      SDB_FORCE_ECHELON;
              CNTRL_MSR_TYPE      :   in      SDB_CONTROL_MEASURE_LOC_TYPE;
              CNTRL_MSR_SIDE      :   in      SDB_SIDE_TYPE);
--
-- CPM description: Displays the control measures of a given type and echelon
--                  and belonging to a specified force
--
-- formal parameters
--IN    CNTRL_MSR_ECHELON  Echelon of the control measures to display
--
--IN    CNTRL_MSR_TYPE     Location type of the control measures to display
--
--IN    CNTRL_MSR_SIDE     Force of the control measures to display
--

procedure UCM_DELETE_CNTRL_MSR (
              CNTRL_MSR_IND       :   in      SDB_CONTROL_MEASURE_ID);
--
-- CPM description: Deletes a control measure from the display
--
-- formal parameters
--IN    CNTRL_MSR_IND      Index of the control measure to delete
```

```
--
procedure UCM_ERASE_CNTRL_MSR (
              CNTRL_MSR_ECHELON    :   in      SDB_FORCE_ECHELON;
              CNTRL_MSR_TYPE       :   in      SDB_CONTROL_MEASURE_LOC_TYPE;
              CNTRL_MSR_SIDE       :   in      SDB_SIDE_TYPE);
--
-- CPM description: Erases the control measures of a given type and echelon
--                  and belonging to a specified force
--
-- formal parameters
--IN    CNTRL_MSR_ECHELON  Echelon of the control measures to erase
--
--IN    CNTRL_MSR_TYPE     Location type of the control measures to erase
--
--IN    CNTRL_MSR_SIDE     Force of the control measures to erase
--

procedure UCM_INITIALIZE_CNTRL_MSR;
--
-- CPM description: Initializes the control measure display system.
--
-- formal parameters
--      None
--

procedure UCM_MOVE_CNTRL_MSR (
                  CNTRL_MSR_ID    :     in    SDB_CONTROL_MEASURE_ID;
                  CNTRL_MSR_REC   :     in    SDB_CONTROL_MEASURE_REC);
--
-- CPM description: Updates the location of a control measure
--
-- formal parameters
--      CNTRL_MSR_ID       Id of the control measure to move
--
--      CNTRL_MSR_REC      New description of the control measure
--

procedure UCM_MOVE_CNTRL_MSR_PNT (
                  CNTRL_MSR_ID    :     in    SDB_CONTROL_MEASURE_ID;
                  CNTRL_MSR_REC   :     in    SDB_CNTRL_MSR_POINT_REC);
--
-- CPM description: Updates the location of a point control measure
--
-- formal parameters
--      CNTRL_MSR_ID       Id of the control measure to move
--
--      CNTRL_MSR_REC      New description of the control measure
--

procedure UCM_MOVE_DEFINE_CNTRL_MSR (
                  PIXEL_X         :     in    SYS_IMAGE_COLUMN;
                  PIXEL_Y         :     in    SYS_IMAGE_ROW);
--
-- CPM description: Changes the location of a control measure that is being
--                  defined.
```

```
--
-- formal parameters
--IN     PIXEL_X          The number of pixels to move in the X direction.
--
--IN     PIXEL_Y          The number of pixels to move in the Y direction.
--

procedure UCM_PROCESS_DEFINE_BUTTON (
                    BUTTON_INDEX    :     in    SYS_WINDOW_VALUE;
                    ECH_WINDOW      :     out   SYS_WINDOW_ELE_ID;
                    SEND_TO_APPL    :     out   BOOLEAN);
--
-- CPM description: processes a button selection from the menu definition
--                  button.
--
-- formal parameters
--       BUTTON_INDEX       Index into the menu buttons
--
--OUT    ECH_WINDOW          Id of the echelon selection menu
--
--OUT    SEND_TO_APPL       Indicator if the control measure record should be
--                          transferred to the calling application.  This
--                          is set to true when the DONE button is hit.
--

procedure UCM_PROCESS_BL_ECH_MENU (
              MENU_STATUS          :  in       UWN_BUTTON_MENU_OUTPUT;
              ECH_WINDOW           :  in       SYS_WINDOW_ELE_ID);
--
-- CPM description: processes the completion of a BLUEFOR echelon menu
--
-- formal parameters
--       MENU_STATUS       Completion status of the menu
--
--       ECH_WINDOW        Id of the echelon menu
--

procedure UCM_PROCESS_OP_ECH_MENU (
              MENU_STATUS          :  in       UWN_BUTTON_MENU_OUTPUT;
              ECH_WINDOW           :  in       SYS_WINDOW_ELE_ID);
--
-- CPM description: processes the completion of a OPFOR echelon menu
--
-- formal parameters
--       MENU_STATUS       Completion status of the menu
--
--       ECH_WINDOW        Id of the echelon menu
--

procedure UCM_PROCESS_SCALE_MENU (
              MENU_STATUS          :  in       UWN_BUTTON_MENU_OUTPUT;
              ECH_WINDOW           :  in       SYS_WINDOW_ELE_ID);
--
-- CPM description: processes the completion of a map scale menu
--
-- formal parameters
```

```
    --       MENU              Description of the map scale menu
    --
    --       MENU_STATUS       Completion status of the menu
    --
    --       ECH_WINDOW        Id of the echelon menu
    --

    procedure UCM_RESTORE_CNTRL_MSR;
    --
    -- CPM description: Restores the control measure displays that were destroyed
    --                  by overlapping windows.
    --
    -- formal parameters
    --       None
    --

end UCM_CONTROL_MEASURE;
```

```
--cpc package specification name: UME_MAP_EDITOR
--
--cpc description: UME_MAP_EDITOR contains the low level digital map utilities
--                 for reading map images from the database and displaying them
--                 in the appropriate image planes.
--
--cpc design notes:
--    This package raises the SYS_UME_EXCEPTION when an exception is detected.
--
--cpc package author: Bruce Packard
--                    Science Applications International Corporation
--                    424 Delaware, Suite C3
--                    Leavenworth, KS 66048
--
with SYSTEM_PACKAGE;              use SYSTEM_PACKAGE;

package UME_MAP_EDITOR is

   procedure UME_CLOSE_CONT_FILE (
                     FILE_DESC      :     in    SYS_FILE_DESC);
   --
   -- CPM description: Closes a contour image database.
   --
   -- formal parameters
   --IN    FILE_DESC        The file descriptor for the contour database. It
   --                       is returned from UME_OPEN_CONT_FILE.
   --

   procedure UME_CLOSE_ELEV_FILE (
                     FILE_DESC      :     in    SYS_FILE_DESC);
   --
   -- CPM description: Closes a elevation database.
   --
   -- formal parameters
   --IN    FILE_DESC        The file descriptor for the elevation database. It
   --                       is returned from UME_OPEN_ELEV_FILE.
   --

   procedure UME_CLOSE_MAP_FILE (
                     FILE_DESC      :     in    SYS_FILE_DESC);
   --
   -- CPM description: Closes a map image database.
   --
   -- formal parameters
   --IN    FILE_DESC        The file descriptor for the map image database. It
   --                       is returned from UME_OPEN_MAP_FILE.
   --

   procedure UME_DEFINE_MAP_COORD (
                     PANEL_ID       :     in    SYS_WINDOW_ELE_ID);
   --
   -- CPM description: Defines the map coordinates of the panel. This routine
   --                  should be called whenever the map panel changes or the
   --                  map scale changes.
   --
   -- formal parameters
```

```
--IN    PANEL_ID          The id assigned to the digital map panel.
--

procedure UME_DETERMINE_CONT_FILE (
                  FILE_NAME      :     out   STRING);
--
-- CPM description: Determines the name of the contour map parameter file for
--                  the current map scale.
--
-- formal parameters
--OUT   FILE_NAME         The name of the contour image database.
--

procedure UME_DETERMINE_CONT_BLOCK (
                  UL_PIXEL_X     :     in    SYS_WINDOW_COLUMN;
                  UL_PIXEL_Y     :     in    SYS_WINDOW_ROW;
                  PIXEL_WIDTH    :     in    SYS_WINDOW_COLUMN;
                  PIXEL_HEIGHT   :     in    SYS_WINDOW_ROW;
                  START_BLOCK    :     out   SYS_DB_SIZE;
                  NUMBER_COLUMN  :     out   SYS_DB_SIZE;
                  NUMBER_ROW     :     out   SYS_DB_SIZE;
                  ROW_INCREMENT  :     out   SYS_DB_SIZE;
                  NEW_PIXEL_X    :     out   SYS_WINDOW_COLUMN;
                  NEW_PIXEL_Y    :     out   SYS_WINDOW_ROW);
--
-- CPM description: Determines the blocks or records to retrieve from the
--                  contour image database. The columns run from left to
--                  right and the rows run from top to bottom.
--
-- formal parameters
--IN    UL_PIXEL_X         Upper left map panel X coordinate where the contour
--                         image is to be displayed.
--
--IN    UL_PIXEL_Y         Upper left map panel Y coordinate where the contour
--                         image is to be displayed.
--
-- IN   PIXEL_WIDTH        The width of the area where the contours are to be
--                         displayed.
--
-- IN   PIXEL_HEIGHT       The height of the area where the contours are to be
--                         displayed.
--
--OUT   START_BLOCK        The starting record number.
--
--OUT   NUMBER_COLUMN      The number of columns to read.
--
--OUT   NUMBER_ROW         The number of rows to read.
--
--OUT   ROW_INCREMENT      The amount to increment the record number for
--                         each contour row.
--
--OUT   NEW_PIXEL_X        Upper left map panel X coordinate where the contour
--                         blocks should actually be displayed
--
--OUT   NEW_PIXEL_Y        Upper left map panel Y coordinate where the contour
--                         blocks should actually be displayed
```

```
--

procedure UME_DETERMINE_ELEV (
                  PIXEL_X       :     in    SYS_WINDOW_COLUMN;
                  PIXEL_Y       :     in    SYS_WINDOW_ROW;
                  FILE_DESC     :     in    SYS_FILE_DESC;
                  ELEV          :     out   SYS_COORDINATE);
--
-- CPM description: Determines the elevation of a given coordinate
--
-- formal parameters
--IN    PIXEL_X           X coordinate to be evaluated
--
--IN    PIXEL_Y           Y coordinate to be evaluated
--
--IN    FILE_DESC         The file descriptor for the elevation database. It
--                        is returned from UME_OPEN_ELEV_FILE.
--
-- IN   ELEV              Elevation at the desired location.
--

procedure UME_DETERMINE_GRID_INTRVL (
                  START_VERT_PIXEL :     out   SYS_WINDOW_COLUMN;
                  VERT_INCREMENT   :     out   SYS_WINDOW_COLUMN;
                  VERT_GRID_NUMB   :     out   SYS_GRID_LABEL;
                  VERT_NUMB_INC    :     out   SYS_GRID_LABEL;
                  START_HORZ_PIXEL :     out   SYS_WINDOW_ROW;
                  HORZ_INCREMENT   :     out   SYS_WINDOW_ROW;
                  HORZ_GRID_NUMB   :     out   SYS_GRID_LABEL;
                  HORZ_NUMB_INC    :     out   SYS_GRID_LABEL);
--
-- CPM description: Determines the start pixel and the pixel increment for
--                  drawing and labeling the UTM grid line. The start pixel
--                  is located in the upper left corner of the digital map
--                  panel.
--
-- formal parameters
--OUT   START_VERT_PIXEL  The digital map panel X coordinate where the
--                        vertical grid lines should start.
--
--OUT   VERT_INCREMENT    The distance to the next vertical grid line in
--                        pixels.
--
--OUT   VERT_GRID_NUMB    The number to display on the first vertical grid.
--
--OUT   VERT_NUMB_INC     The amount to increment the grid label for each
--                        vertical grid line
--
--OUT   START_HORZ_PIXEL  The digital map panel Y coordinate where the
--                        horizontal grid lines should start.
--
--OUT   HORZ_INCREMENT    The distance to the next horizontal grid line in
--                        pixels.
--
--OUT   VERT_GRID_NUMB    The number to display on the first horizontal grid.
--
```

```
--OUT   HORZ_NUMB_INC      The amount to increment the grid label for each
--                         horizontal grid line
--

procedure UME_DETERMINE_MAP_FILE (
                  FILE_NAME      :     out   STRING);
--
-- CPM description: Determines the name of the map parameter for
--                  the current map scale and type.
--
-- formal parameters
--OUT   FILE_NAME         The name of the contour image database.
--

procedure UME_DETERMINE_MAP_BLOCK (
                  UL_PIXEL_X     :     in    SYS_WINDOW_COLUMN;
                  UL_PIXEL_Y     :     in    SYS_WINDOW_ROW;
                  PIXEL_WIDTH    :     in    SYS_WINDOW_COLUMN;
                  PIXEL_HEIGHT   :     in    SYS_WINDOW_ROW;
                  START_BLOCK    :     out   SYS_DB_SIZE;
                  NUMBER_COLUMN  :     out   SYS_DB_SIZE;
                  NUMBER_ROW     :     out   SYS_DB_SIZE;
                  ROW_INCREMENT  :     out   SYS_DB_SIZE;
                  NEW_PIXEL_X    :     out   SYS_WINDOW_COLUMN;
                  NEW_PIXEL_Y    :     out   SYS_WINDOW_ROW);
--
-- CPM description: Determines the blocks or records to retrieve from the
--                  map image database. The columns run from left to
--                  right and the rows run from top to bottom.
--
-- formal parameters
--IN    UL_PIXEL_X         Upper left map panel X coordinate where the map
--                         image is to be displayed.
--
--IN    UL_PIXEL_Y         Upper left map panel Y coordinate where the map
--                         image is to be displayed.
--
-- IN   PIXEL_WIDTH        The width of the area where the map is to be
--                         displayed.
--
-- IN   PIXEL_HEIGHT       The height of the area where the map is to be
--                         displayed.
--
--OUT   START_BLOCK        The starting record number.
--
--OUT   NUMBER_COLUMN      The number of columns to read.
--
--OUT   NUMBER_ROW         The number of rows to read.
--
--OUT   ROW_INCREMENT      The amount to increment the record number for
--                         each map row.
--
--OUT   NEW_PIXEL_X        Upper left map panel X coordinate where the map
--                         blocks should actually be displayed
--
--OUT   NEW_PIXEL_Y        Upper left map panel Y coordinate where the map
```

```
--                          blocks should actually be displayed
--

procedure UME_DISPLAY_CONT_BLOCKS (
                    PANEL_ID       :     in    SYS_WINDOW_ELE_ID;
                    FILE_DESC      :     in    SYS_FILE_DESC;
                    START_BLOCK    :     in    SYS_DB_SIZE;
                    NUMBER_COLUMN  :     in    SYS_DB_SIZE;
                    NUMBER_ROW     :     in    SYS_DB_SIZE;
                    ROW_INCREMENT  :     in    SYS_DB_SIZE;
                    PIXEL_START_X  :     in    SYS_WINDOW_COLUMN;
                    PIXEL_START_Y  :     in    SYS_WINDOW_ROW);
--
-- CPM description: Displays the contour bit image blocks on the digital map.
--
-- formal parameters
--IN     PANEL_ID           The id of the digital map panel.
--
--IN     FILE_DESC          The file descriptor for the contour database. It
--                          is returned from UME_OPEN_CONT_FILE.
--
--IN     START_BLOCK        The starting record number.
--
--IN     NUMBER_COLUMN      The number of columns to read.
--
--IN     NUMBER_ROW         The number of rows to read.
--
--IN     ROW_INCREMENT      The amount to increment the record number for
--                          each contour row.
--
--IN     PIXEL_START_X      Upper left map panel X coordinate where the contour
--                          blocks should actually be displayed
--
--IN     PIXEL_START_Y      Upper left map panel Y coordinate where the contour
--                          blocks should actually be displayed
--

procedure UME_DISPLAY_MAP_BLOCKS (
                    PANEL_ID       :     in    SYS_WINDOW_ELE_ID;
                    FILE_DESC      :     in    SYS_FILE_DESC;
                    START_BLOCK    :     in    SYS_DB_SIZE;
                    NUMBER_COLUMN  :     in    SYS_DB_SIZE;
                    NUMBER_ROW     :     in    SYS_DB_SIZE;
                    ROW_INCREMENT  :     in    SYS_DB_SIZE;
                    PIXEL_START_X  :     in    SYS_WINDOW_COLUMN;
                    PIXEL_START_Y  :     in    SYS_WINDOW_ROW);
--
-- CPM description: Displays a the map bit image blocks on the digital map.
--
-- formal parameters
--IN     PANEL_ID           The id of the digital map panel.
--
--IN     FILE_DESC          The file descriptor for the contour database. It
--                          is returned from UME_OPEN_MAP_FILE.
--
--IN     START_BLOCK        The starting record number.
```

```
--
--IN     NUMBER_COLUMN      The number of columns to read.
--
--IN     NUMBER_ROW         The number of rows to read.
--
--IN     ROW_INCREMENT      The amount to increment the record number for
--                          each contour row.
--
--IN     PIXEL_START_X      Upper left map panel X coordinate where the contour
--                          blocks should actually be displayed
--
--IN     PIXEL_START_Y      Upper left map panel Y coordinate where the contour
--                          blocks should actually be displayed
--

procedure UME_DRAW_HORIZONTAL_GRID (
                  PANEL_ID       :     in    SYS_WINDOW_ELE_ID;
                  PIXEL_Y        :     in    SYS_WINDOW_ROW);
--
-- CPM description: Draws a horizontal grid line on the digital map.
--
-- formal parameters
--IN     PANEL_ID           The id of the digital map panel.
--
--IN     PIXEL_Y            Digital map panel Y coordinate where the grid line
--                          is to be displayed.

procedure UME_DRAW_VERTICAL_GRID (
                  PANEL_ID       :     in    SYS_WINDOW_ELE_ID;
                  PIXEL_X        :     in    SYS_WINDOW_COLUMN);
--
-- CPM description: Draws a vertical grid line on the digital map.
--
-- formal parameters
--IN     PANEL_ID           The id of the digital map panel.
--
--IN     PIXEL_X            Digital map panel X coordinate where the grid line
--                          is to be displayed.

procedure UME_INIT_MAP_SYSTEM (
                  CONTOUR_FILE   :     in    STRING;
                  MAP_FILE       :     in    STRING);
--
-- CPM description: Initializes the contour and map system.
--
-- formal parameters
--IN     CONTOUR_FILE       The name of the file containing the contour map
--                          initialization.
--
--IN     MAP_FILE           The name of the file containing the map
--                          initialization.
--

procedure UME_LABEL_HORIZONTAL_GRID (
                  PANEL_ID       :     in    SYS_WINDOW_ELE_ID;
                  PIXEL_Y        :     in    SYS_WINDOW_ROW;
```

```
                    GRID_LABEL     :     in     SYS_GRID_LABEL);
--
-- CPM description: Labels a horizontal grid line on the digital map.
--
-- formal parameters
--IN     PANEL_ID          The id of the digital map panel.
--
--IN     PIXEL_Y           Digital map panel Y coordinate where the grid label
--                         is to be displayed.
--
--IN     GRID_LABEL        The number label to display on the grid
--

procedure UME_LABEL_VERTICAL_GRID (
                    PANEL_ID       :     in     SYS_WINDOW_ELE_ID;
                    PIXEL_X        :     in     SYS_WINDOW_COLUMN;
                    GRID_LABEL     :     in     SYS_GRID_LABEL);
--
-- CPM description: Labels a vertical grid line on the digital map.
--
-- formal parameters
--IN     PANEL_ID          The id of the digital map panel.
--
--IN     PIXEL_X           Digital map panel X coordinate where the grid label
--                         is to be displayed.
--
--IN     GRID_LABEL        The number label to display on the grid
--

procedure UME_OPEN_CONT_FILE (
                    FILE_NAME      :     in     STRING;
                    FILE_DESC      :     out    SYS_FILE_DESC);
--
-- CPM description: Opens the contour image database.
--
-- formal parameters
--IN     FILE_NAME         The name of the contour parameter file
--
--OUT    FILE_DESC         The file descriptor for the contour file. This
--                         descriptor is required for other I/O operations.
--

procedure UME_OPEN_ELEV_FILE (
                    FILE_NAME      :     in     STRING;
                    FILE_DESC      :     out    SYS_FILE_DESC);
--
-- CPM description: Opens the elevation database.
--
-- formal parameters
--IN     FILE_NAME         The name of the elevation parameter file
--
--OUT    FILE_DESC         The file descriptor for the elevation file. This
--                         descriptor is required for other I/O operations.
--

procedure UME_OPEN_MAP_FILE (
```

```
                    FILE_NAME       :     in    STRING;
                    FILE_DESC       :     out   SYS_FILE_DESC);
    --
    -- CPM description: Opens the map image database.
    --
    -- formal parameters
    --IN     FILE_NAME          The name of the map parameter file
    --
    --OUT    FILE_DESC          The file descriptor for the map file. This
    --                          descriptor is required for other I/O operations.
    --

end UME_MAP_EDITOR;
```

```
--cpc package specification name: UMP_MAP
--
--cpc description: UMP_MAP is the intermediate level digital map package that
--                 is responsible for displaying and erasing the digital map
--                 and the digital map features.
--
--cpc design notes:
--
--cpc package author: Bruce Packard
--                    Science Applications International Corporation
--                    424 Delaware, Suite C3
--                    Leavenworth, KS 66048
--
with SYSTEM_PACKAGE;      use  SYSTEM_PACKAGE;
with MSG_MESSAGE;         use  MSG_MESSAGE;
with MAP_SYSTEM;          use  MAP_SYSTEM;

package UMP_MAP is

   UMP_LUT_MESSAGE         :      MSG_MESSAGE_POINT := new MSG_VAR_MESSAGES (
                                  MSG_LUT_UPDATE);

   procedure UMP_SEND_LUT_UPDATE;
   --
   -- CPM description: Sends a color lookup table update to the station
   --                  control process (SCL)
   --
   -- formal parameters
   --        None
   --

   procedure UMP_DISPLAY_CONTOURS;
   --
   -- CPM description: Displays contours on the digital map
   --
   -- formal parameters
   --        None
   --

   procedure UMP_DISPLAY_GRIDS;
   --
   -- CPM description: Displays grid lines on the digital map
   --
   -- formal parameters
   --        None
   --

   procedure UMP_DISPLAY_MAP;
   --
   -- CPM description: Displays digital map background image
   --
   -- formal parameters
   --        None
   --

 procedure UMP_ELEVATION_QUERY;
```

```
--
-- CPM description: Display the elevation of a given point
--
-- formal parameters
--        None
--

procedure UMP_ERASE_CONTOURS;
--
-- CPM description: Erases contours from the digital map
--
-- formal parameters
--        None
--

procedure UMP_ERASE_GRIDS;
--
-- CPM description: Erases grids from the digital map
--
-- formal parameters
--        None
--

procedure UMP_ERASE_MAP;
--
-- CPM description: Erases digital map background image
--
-- formal parameters
--        None
--

procedure UMP_HILITE_HYDRO;
--
-- CPM description: Highlights the individual classes of hydrography.
--
-- formal parameters
--        None
--

procedure UMP_HILITE_MISC;
--
-- CPM description: Highlights the individual classes of miscellaneous
--                  features.
--
-- formal parameters
--        None
--

procedure UMP_HILITE_ROAD;
--
-- CPM description: Highlights the individual classes of roads.
--
-- formal parameters
--        None
--
```

```
procedure UMP_HILITE_URBAN;
--
-- CPM description: Highlights the individual classes of urban areas.
--
-- formal parameters
--        None
--

procedure UMP_INITIALIZE_MAP;
--
-- CPM description: Initializes the map system.
--
-- formal parameters
--        None
--

procedure UMP_NEW_BACK_TYPE (BACKGROUND  :   in   SYS_MAP_BACKGROUND);
--
-- CPM description: Updates the color lookup table for a change in
--                  background type.
--
-- formal parameters
--IN    BACKGROUND - New Background Type
--

procedure UMP_RESTORE_MAP (
                  PANEL_ID      :      in    SYS_WINDOW_ELE_ID;
                  PIXEL_X       :      in    SYS_WINDOW_COLUMN;
                  PIXEL_Y       :      in    SYS_WINDOW_ROW;
                  PIXEL_WIDTH   :      in    SYS_WINDOW_COLUMN;
                  PIXEL_HEIGHT  :      in    SYS_WINDOW_ROW);
--
-- CPM description: Restores the display of the map image.  This is required
--                  during scrolling operations and when a portion of the
--                  digital map panel has been exposed.
--
-- formal parameters
--IN    PANEL_ID         The id of the digital map panel.
--
--IN    PIXEL_X          The window X coordinate of the upper left corner of
--                       the digital map restore area.
--
--IN    PIXEL_Y          The window Y coordinate of the upper left corner of
--                       the digital map restore area.
--
--IN    PIXEL_WIDTH      The width of the digital map restore area in pixels.
--
--IN    PIXEL_HEIGHT     The height of the digital map restore area in
--                       pixels.
--

procedure UMP_UNHILITE_HYDRO;
--
-- CPM description: Sets all classes of hydrography to a single color.
--
-- formal parameters
```

```
    --        None
    --

    procedure UMP_UNHILITE_MISC;
    --
    -- CPM description: Sets all classes of miscellaneous features to a single
    --                  color.
    --
    -- formal parameters
    --        None
    --

    procedure UMP_UNHILITE_ROAD;
    --
    -- CPM description: Sets all classes of roads to a single color.
    --
    -- formal parameters
    --        None
    --

    procedure UMP_UNHILITE_URBAN;
    --
    -- CPM description: Sets all classes of urban areas to a single color.
    --
    -- formal parameters
    --        None
    --

end UMP_MAP;
```

```
--cpc package specification name: UNIT_SYSTEM
--
--cpc description: Defines types and objects that are common to the unit display
--                 system.
--
--cpc design notes:
--
--cpc package author: Bruce Packard
--                    Science Applications International Corporation
--                    424 Delaware, Suite C3
--                    Leavenworth, KS 66048
--
with SYSTEM_PACKAGE;          use SYSTEM_PACKAGE;
with SDB_SITUATION_DB;        use SDB_SITUATION_DB;

package UNIT_SYSTEM is

   -- Unit data that is to be displayed in the unit summary status box
   type UNIT_STATUS_DATA is
      record
         UNIT_NAME         :       STRING (SDB_UNIT_NAME_LEN);
         UNIT_ECHELON      :       SDB_FORCE_ECHELON;
         UNIT_TYPE         :       SDB_UNIT_TYPE;
         LOCATION          :       SDB_LOCATION_REC;
         PERCENT_STR       :       SYS_PERCENT;
   end record;

   -- Unit display options
   type UNIT_OPTIONS is
      record
         UNIT_DIV          :       BOOLEAN;
         UNIT_BDE          :       BOOLEAN;
         UNIT_RGMT         :       BOOLEAN;
         UNIT_BN           :       BOOLEAN;
         UNIT_CO           :       BOOLEAN;
         UNIT_CBT_COMMIT   :       BOOLEAN;
         UNIT_CS_REINF     :       BOOLEAN;
         UNIT_CSS_ARTIL    :       BOOLEAN;
         UNIT_NAME         :       BOOLEAN;
         UNIT_SYMBOL       :       BOOLEAN;
   end record;

   -- Current BLUEFOR units displayed
   UNIT_BLUEFOR_COUNT             :       SDB_BLUEFOR_UNIT_ID;
   UNIT_CURRENT_BLUEFOR           :       SDB_LOCATION_LIST_POINT;
   UNIT_BLUEFOR_DISPLAYED         :   array (SDB_BLUEFOR_UNIT_ID) of BOOLEAN;

   -- Current OPFOR units displayed
   UNIT_OPFOR_COUNT               :       SDB_OPFOR_UNIT_ID;
   UNIT_CURRENT_OPFOR             :       SDB_LOCATION_LIST_POINT;
   UNIT_OPFOR_DISPLAYED           :   array (SDB_OPFOR_UNIT_ID) of BOOLEAN;

   -- Current BLUEFOR Unit and OPFOR Unit display options
   UNIT_CURR_BLUEFOR_OPTION       :    UNIT_OPTIONS;
   UNIT_CURR_OPFOR_OPTION         :    UNIT_OPTIONS;
```

```
    -- Unit popup menus (Blue and OPFOR)
    UNIT_BL_MENU_ID                 :    SYS_WINDOW_ELE_ID;
    UNIT_BL_MENU_START              :    SYS_POP_UP_START_PTR := new
                                         SYS_POP_UP_START (SYS_UNIT_MENU);
    UNIT_BL_MENU_LENGTH             :    SYS_POP_UP_LENGTH_PTR := new
                                         SYS_POP_UP_LENGTH (SYS_UNIT_MENU);
    UNIT_BL_POP_UP_TEXT             :    SYS_MENU_TEXT_PTR := new
                                         SYS_MENU_TEXT (SYS_UNIT_CELL);
    UNIT_BL_POP_UP_CHILD            :    SYS_POP_UP_CHILD_PTR := new
                                         SYS_POP_UP_CHILD (SYS_UNIT_CELL);
    UNIT_BL_POP_UP_OPTION           :    SYS_UNIT_OPTION_PTR := new
                                         SYS_UNIT_OPTION_ARRAY (SYS_UNIT_CELL);
    UNIT_OP_MENU_ID                 :    SYS_WINDOW_ELE_ID;
    UNIT_OP_MENU_START              :    SYS_POP_UP_START_PTR := new
                                         SYS_POP_UP_START (SYS_UNIT_MENU);
    UNIT_OP_MENU_LENGTH             :    SYS_POP_UP_LENGTH_PTR := new
                                         SYS_POP_UP_LENGTH (SYS_UNIT_MENU);
    UNIT_OP_POP_UP_TEXT             :    SYS_MENU_TEXT_PTR := new
                                         SYS_MENU_TEXT (SYS_UNIT_CELL);
    UNIT_OP_POP_UP_CHILD            :    SYS_POP_UP_CHILD_PTR := new
                                         SYS_POP_UP_CHILD (SYS_UNIT_CELL);
    UNIT_OP_POP_UP_OPTION           :    SYS_UNIT_OPTION_PTR := new
                                         SYS_UNIT_OPTION_ARRAY (SYS_UNIT_CELL);

end UNIT_SYSTEM;
```

```
--cpc package specification name: UNT_UNIT
--
--cpc description: UNT_UNIT is the intermediate level unit display package that
--                 is responsible for displaying and erasing units on the
--                 digital map.
--
--cpc design notes:
--    This package raises the SYS_UNT_EXCEPTION when an exception is detected.
--
--cpc package author: Bruce Packard
--                   Science Applications International Corporation
--                   424 Delaware, Suite C3
--                   Leavenworth, KS 66048
--
with SDB_SITUATION_DB;          use SDB_SITUATION_DB;

package UNT_UNIT is

   procedure UNT_DISPLAY_BLUEFOR_UNIT (
                UNIT_ECHELON        :   in      SDB_FORCE_ECHELON;
                BATTLE_FUNC         :   in      SDB_BATTLE_FUNCTION);
   --
   -- CPM description: Displays a BLUEFOR units of a given echelon
   --
   -- formal parameters
   --IN    UNIT_ECHELON     Echelon of the unit to display
   --
   --IN    BATTLE_FUNC      Combat, CS, or CSS units
   --

   procedure UNT_DISPLAY_OPFOR_UNIT (
                UNIT_ECHELON        :   in      SDB_FORCE_ECHELON;
                BATTLE_FUNC         :   in      SDB_BATTLE_FUNCTION);
   --
   -- CPM description: Displays OPFOR units of a given echelon
   --
   -- formal parameters
   --IN    UNIT_ECHELON     Echelon of the unit to display
   --
   --IN    BATTLE_FUNC      Committed, Reinforcing, or Artillery units
   --

   procedure UNT_DISPLAY_OPFOR_STATUS (
                UNIT_IND            :   in      SDB_OPFOR_UNIT_ID);
   --
   -- CPM description: Displays a summary status report for a OPFOR. This
   --                  is generated when the user selects the status option
   --                  for a unit displayed on the digital map
   --
   -- formal parameters
   --IN    UNIT_IND          Index into the UNIT_CURRENT_OPFOR array of the
   --                       unit to display a status report on
   --
   --

   procedure UNT_ERASE_BLUEFOR_UNIT (
```

```
                UNIT_ECHELON        :   in      SDB_FORCE_ECHELON;
                BATTLE_FUNC         :   in      SDB_BATTLE_FUNCTION);
--
-- CPM description: Erases BLUEFOR units of a given echelon
--
-- formal parameters
--IN     UNIT_ECHELON      Echelon of the unit to erase
--
--IN     BATTLE_FUNC       Combat, CS, or CSS units
--

procedure UNT_ERASE_OPFOR_UNIT (
                UNIT_ECHELON        :   in      SDB_FORCE_ECHELON;
                BATTLE_FUNC         :   in      SDB_BATTLE_FUNCTION);
--
-- CPM description: Erases OPFOR units of a given echelon
--
-- formal parameters
--IN     UNIT_ECHELON      Echelon of the unit to erase
--
--IN     BATTLE_FUNC       Committed, Reinforcing, or Artillery units
--

procedure UNT_ERASE_OPFOR_STATUS;
--
-- CPM description: Erases a OPFOR unit summary status report
--
-- formal parameters
--      None
--

procedure UNT_INITIALIZE_UNITS;
--
-- CPM description: Initializes the unit display system
--
-- formal parameters
--      None
--

procedure UNT_MOVE_BLUEFOR_UNIT (
                    UNIT_IND        :   in    SDB_BLUEFOR_UNIT_ID;
                    UNIT_LOCATION   :   in    SDB_LOCATION_REC);
--
-- CPM description: Updates the location of a BLUEFOR unit
--
-- formal parameters
--      UNIT_IND          Index into the UNIT_CURRENT_BLUEFOR array of the
--                        unit to move
--
--      UNIT_LOCATION     New location of the unit
--

procedure UNT_MOVE_OPFOR_UNIT (
                    UNIT_IND        :   in    SDB_OPFOR_UNIT_ID;
                    UNIT_LOCATION   :   in    SDB_LOCATION_REC);
--
```

```
    -- CPM description: Updates the location of a BLUEFOR unit
    --
    -- formal parameters
    --      UNIT_IND        Index into the UNIT_CURRENT_OPFOR array of the
    --                      unit to move
    --
    --      UNIT_LOCATION   New location of the unit
    --

    procedure UNT_RESTORE_UNITS;
    --
    -- CPM description: Redisplays the BLUEFOR and OPFOR units.
    --
    -- formal parameters
    --      None
    --

end UNT_UNIT;
```

```
--cpc package specification name: UOB_OBSTACLE
--
--cpc description: UOB_OBSTACLE is the intermediate level obstacle
--                 display package that is responsible for displaying and
--                 erasing obstacle on the digital map.
--
--cpc design notes:
--    This package raises the SYS_UOB_EXCEPTION when an exception is detected.
--
--cpc package author: Bruce Packard
--                    Science Applications International Corporation
--                    424 Delaware, Suite C3
--                    Leavenworth, KS 66048
--
with SDB_SITUATION_DB;          use SDB_SITUATION_DB;

package UOB_OBSTACLE is

   procedure UOB_DISPLAY_OBSTACLE (
                OBSTACLE_SIDE       :   in      SDB_SIDE_TYPE);
   --
   -- CPM description: Displays the obstacles of a given force and echelon.
   --
   -- formal parameters
   --IN    OBSTACLE_SIDE     Force of the obstacles to display
   --

   procedure UOB_DELETE_OBSTACLE (
                OBSTACLE_IND        :   in      SDB_OBSTACLE_ID);
   --
   -- CPM description: Deletes an obstacle from the display
   --
   -- formal parameters
   --IN    CNTRL_MSR_IND      Index of the control measure to delete
   --

   procedure UOB_ERASE_OBSTACLE (
                OBSTACLE_SIDE       :   in      SDB_SIDE_TYPE);
   --
   -- CPM description: Erases the obstacles of a given force and echelon.
   --
   -- formal parameters
   --IN    OBSTACLE_SIDE     Force of the obstacles to erase
   --

   procedure UOB_INITIALIZE_OBSTACLE;
   --
   -- CPM description: Initializes the obstacle display system.
   --
   -- formal parameters
   --      None
   --

   procedure UOB_MOVE_OBSTACLE (
                     OBSTACLE_ID    :     in    SDB_OBSTACLE_ID;
                     OBSTACLE_REC   :     in    SDB_OBSTACLE_REC);
```

```
    --
    -- CPM description: Updates the location of an obstacle
    --
    -- formal parameters
    --      OBSTACLE_ID        Id of the obstacle to move
    --
    --      OBSTACLE_REC       New description of the obstacle
    --

    procedure UOB_RESTORE_OBSTACLE;
    --
    -- CPM description: Restores the obstacle displays that were destroyed
    --                  by overlapping windows.
    --
    -- formal parameters
    --      None
    --

end UOB_OBSTACLE;
```

```
--cpc package specification name: UOE_OBSTACLE_EDITOR
--
--cpc description: UOE_OBSTACLE_EDITOR contains the low level obstacle utilities
--                 for displaying specific types of obstacles.
--
--cpc design notes:
--    This package raises the SYS_UOE_EXCEPTION when an exception is detected.
--
--cpc package author: Bruce Packard
--                    Science Applications International Corporation
--                    424 Delaware, Suite C3
--                    Leavenworth, KS 66048
--
with SYSTEM_PACKAGE;                use SYSTEM_PACKAGE;
with SDB_SITUATION_DB;              use SDB_SITUATION_DB;
with TEXT_IO;                       use TEXT_IO;

package UOE_OBSTACLE_EDITOR is

   procedure UOE_ABATIS (
                     ADD_FLAG          :     in    BOOLEAN;
                     OBSTACLE_DESC     :     in    SDB_OBSTACLE_REC);
   --
   -- CPM description: Displays an abatis on the digital map.
   --
   -- formal parameters
   --IN    ADD_FLAG          Add or erase the obstacle flag
   --                        True = Add;  False = Erase
   --
   --IN    OBSTACLE_DESC     The description of the abatis.
   --

   procedure UOE_AT_DITCH (
                     ADD_FLAG          :     in    BOOLEAN;
                     OBSTACLE_DESC     :     in    SDB_OBSTACLE_REC);
   --
   -- CPM description: Displays an anti-tank ditch on the digital map.
   --
   -- formal parameters
   --IN    ADD_FLAG          Add or erase the obstacle flag
   --                        True = Add;  False = Erase
   --
   --IN    OBSTACLE_DESC     The description of the anti-tank ditch.
   --

   procedure UOE_BRIDGE_DEMO (
                     ADD_FLAG          :     in    BOOLEAN;
                     OBSTACLE_DESC     :     in    SDB_OBSTACLE_REC);
   --
   -- CPM description: Displays a bridge demolition on the digital map.
   --
   -- formal parameters
   --IN    ADD_FLAG          Add or erase the obstacle flag
   --                        True = Add;  False = Erase
   --
   --IN    OBSTACLE_DESC     The description of the bridge demolition.
```

```
--
procedure UOE_CHEMICAL (
                    ADD_FLAG          :     in    BOOLEAN;
                    OBSTACLE_DESC     :     in    SDB_OBSTACLE_REC);
--
-- CPM description: Displays a chemical obstacle on the digital map.
--
-- formal parameters
--IN     ADD_FLAG          Add or erase the obstacle flag
--                         True = Add;  False = Erase
--
--IN     OBSTACLE_DESC     The description of the chemical obstacle.
--

procedure UOE_CRATER (
                    ADD_FLAG          :     in    BOOLEAN;
                    OBSTACLE_DESC     :     in    SDB_OBSTACLE_REC);
--
-- CPM description: Displays a crater on the digital map.
--
-- formal parameters
--IN     ADD_FLAG          Add or erase the obstacle flag
--                         True = Add;  False = Erase
--
--IN     OBSTACLE_DESC     The description of the crater.
--

procedure UOE_DAM_DEMO (
                    ADD_FLAG          :     in    BOOLEAN;
                    OBSTACLE_DESC     :     in    SDB_OBSTACLE_REC);
--
-- CPM description: Displays a dam demolition on the digital map.
--
-- formal parameters
--IN     ADD_FLAG          Add or erase the obstacle flag
--                         True = Add;  False = Erase
--
--IN     OBSTACLE_DESC     The description of the dam demolition.
--

procedure UOE_FLOODING (
                    ADD_FLAG          :     in    BOOLEAN;
                    OBSTACLE_DESC     :     in    SDB_OBSTACLE_REC);
--
-- CPM description: Displays a flooding obstacle on the digital map.
--
-- formal parameters
--IN     ADD_FLAG          Add or erase the obstacle flag
--                         True = Add;  False = Erase
--
--IN     OBSTACLE_DESC     The description of the flooding obstacle.
--

procedure UOE_LOG_POSTS (
                    ADD_FLAG          :     in    BOOLEAN;
```

```
                    OBSTACLE_DESC     :     in    SDB_OBSTACLE_REC);
--
-- CPM description: Displays log posts on the digital map.
--
-- formal parameters
--IN    ADD_FLAG          Add or erase the obstacle flag
--                        True = Add;  False = Erase
--
--IN    OBSTACLE_DESC     The description of the log posts.
--

procedure UOE_MINEFIELD_AP (
                    ADD_FLAG          :     in    BOOLEAN;
                    OBSTACLE_DESC     :     in    SDB_OBSTACLE_REC);
--
-- CPM description: Displays a anti-personnel minefield on the digital map.
--
-- formal parameters
--IN    ADD_FLAG          Add or erase the obstacle flag
--                        True = Add;  False = Erase
--
--IN    OBSTACLE_DESC     The description of the anti-personnel minefield.
--

procedure UOE_MINEFIELD_AT (
                    ADD_FLAG          :     in    BOOLEAN:
                    OBSTACLE_DESC     :     in    SDB_OBSTACLE_REC);
--
-- CPM description: Displays a anti-tank minefield on the digital map.
--
-- formal parameters
--IN    ADD_FLAG          Add or erase the obstacle flag
--                        True = Add;  False = Erase
--
--IN    OBSTACLE_DESC     The description of the anti-tank minefield.
--

procedure UOE_MINEFIELD_AT_AP (
                    ADD_FLAG          :     in    BOOLEAN;
                    OBSTACLE_DESC     :     in    SDB_OBSTACLE_REC);
--
-- CPM description: Displays a anti-tank/anti-personnel minefield on the
--                  digital map.
--
-- formal parameters
--IN    ADD_FLAG          Add or erase the obstacle flag
--                        True = Add;  False = Erase
--
--IN    OBSTACLE_DESC     The description of the anti-tank/anti-personnel
--                        minefield.
--

procedure UOE_NUCLEAR (
                    ADD_FLAG          :     in    BOOLEAN;
                    OBSTACLE_DESC     :     in    SDB_OBSTACLE_REC);
--
```

```
-- CPM description: Displays a nuclear obstacle on the digital map.
--
-- formal parameters
--IN     ADD_FLAG          Add or erase the obstacle flag
--                         True = Add;  False = Erase
--
--IN     OBSTACLE_DESC     The description of nuclear obstacle.
--

procedure UOE_SCAT_MINE_AP (
                   ADD_FLAG          :     in    BOOLEAN;
                   OBSTACLE_DESC     :     in    SDB_OBSTACLE_REC);
--
-- CPM description: Displays a scattered anti-personnel minefield on the
--                   digital map.
--
-- formal parameters
--IN     ADD_FLAG          Add or erase the obstacle flag
--                         True = Add;  False = Erase
--
--IN     OBSTACLE_DESC     The description of scattered anti-personnel
--                         minefield.
--

procedure UOE_SCAT_MINE_AT (
                   ADD_FLAG          :     in    BOOLEAN;
                   OBSTACLE_DESC     :     in    SDB_OBSTACLE_REC);
--
-- CPM description: Displays a scattered anti-tank minefield on the digital
--                   map.
--
-- formal parameters
--IN     ADD_FLAG          Add or erase the obstacle flag
--                         True = Add;  False = Erase
--
--IN     OBSTACLE_DESC     The description of scattered anti-tank minefield.
--

procedure UOE_SCAT_MINE_AT_AP (
                   ADD_FLAG          :     in    BOOLEAN;
                   OBSTACLE_DESC     :     in    SDB_OBSTACLE_REC);
--
-- CPM description: Displays a scattered anti-tank/anti-personnel minefield
--                   on the digital map.
--
-- formal parameters
--IN     ADD_FLAG          Add or erase the obstacle flag
--                         True = Add;  False = Erase
--
--IN     OBSTACLE_DESC     The description of scattered anti-tank/
--                         anti-personnel minefield.
--

procedure UOE_STATUS (
                   ADD_FLAG          :     in    BOOLEAN;
                   OBS_TYPE          :     in    SDB_OBSTACLE_TYPE;
```

```
                    OBS_STATUS          :     in    SDB_OBSTACLE_STATUS);
    --
    -- CPM description: Displays the status of a obstacle.
    --
    -- formal parameters
    --IN    ADD_FLAG          Add or erase the obstacle flag
    --                       True = Add;  False = Erase
    --
    --IN    OBS_TYPE          The type of the obstacle.
    --
    --IN    OBS_STATUS        The status of the obstacle.
    --

    procedure UOE_TUNNEL_DEMO (
                    ADD_FLAG          :     in    BOOLEAN;
                    OBSTACLE_DESC     :     in    SDB_OBSTACLE_REC);
    --
    -- CPM description: Displays a tunnel demolition on the digital map.
    --
    -- formal parameters
    --IN    ADD_FLAG          Add or erase the obstacle flag
    --                       True = Add;  False = Erase
    --
    --IN    OBSTACLE_DESC     The description of tunnel demolition.
    --

    procedure UOE_WIRE (
                    ADD_FLAG          :     in    BOOLEAN;
                    OBSTACLE_DESC     :     in    SDB_OBSTACLE_REC);
    --
    -- CPM description: Displays a wire obstacle on the digital map.
    --
    -- formal parameters
    --IN    ADD_FLAG          Add or erase the obstacle flag
    --                       True = Add;  False = Erase
    --
    --IN    OBSTACLE_DESC     The description of wire obstacle.
    --

end UOE_OBSTACLE_EDITOR;
```

```
--cpc package specification name: UTM_TACTICAL_MAP
--
--cpc description: UTM_TACTICAL_MAP is the high level digital map package that
--                 is responsible for displaying the EDDIC specific displays
--                 such as units and control measures.
--
--cpc design notes:
--    This package raises the SYS_UTM_EXCEPTION when an exception is detected.
--
--cpc package author: Bruce Packard
--                    Science Applications International Corporation
--                    424 Delaware, Suite C3
--                    Leavenworth, KS 66048
--
with SYSTEM_PACKAGE;                use SYSTEM_PACKAGE;
with MAP_SYSTEM;                    use MAP_SYSTEM;
with UNIT_SYSTEM;                   use UNIT_SYSTEM;
with CM_SYSTEM;                     use CM_SYSTEM;
with OBS_SYSTEM;                    use OBS_SYSTEM;
with SDB_SITUATION_DB;              use SDB_SITUATION_DB;
with UWN_WALKING_MENU;

package UTM_TACTICAL_MAP is

   -- Generic instantiations for the digital map walking menus
   package MAP_WALK    is    new  UWN_WALKING_MENU (SYS_MAP_CONTROL,
            SYS_MAP_CONTROL_ARRAY, SYS_MAP_CONTROL_PTR);
   package CM_WALK     is    new  UWN_WALKING_MENU (SYS_CM_OPTION,
            SYS_CM_OPTION_ARRAY, SYS_CM_OPTION_PTR);
   package OBS_WALK    is    new  UWN_WALKING_MENU (SYS_OBS_OPTION,
            SYS_OBS_OPTION_ARRAY, SYS_OBS_OPTION_PTR);
   package UNIT_WALK   is    new  UWN_WALKING_MENU (SYS_UNIT_OPTION,
            SYS_UNIT_OPTION_ARRAY, SYS_UNIT_OPTION_PTR);

   procedure UTM_BLUEFOR_UNITS (
                     PANEL_ID       :     in    SYS_WINDOW_ELE_ID;
                     UNIT_COUNT     :     in    SDB_BLUEFOR_UNIT_ID;
                     UNIT_LOC_TABLE :     in    SDB_LOCATION_LIST_POINT;
                     UNIT_OPTION    :     in    UNIT_OPTIONS);
   --
   -- CPM description: Displays all the BLUEFOR units on the digital map
   --
   -- formal parameters
   --IN    PANEL_ID          The id of the digital map panel.
   --
   --IN    UNIT_COUNT        The number of units in the unit location table
   --
   --IN    UNIT_LOC_TABLE    Array of records defining the unit names, location
   --                        echelon and type.
   --
   --IN    UNIT_OPTION       The default unit display options
   --

   procedure UTM_BLUEFOR_UNIT_CHG (
                     PANEL_ID       :     in    SYS_WINDOW_ELE_ID;
                     UNIT_ID        :     in    SDB_BLUEFOR_UNIT_ID;
```

```
                    UNIT_LOCATION  :     in    SDB_LOCATION_REC);

--
-- CPM description: Changes the location of a BLUEFOR unit on the digital map
--
-- formal parameters
--IN     PANEL_ID          The id of the digital map panel.
--
--IN     UNIT_ID           The id of the changed unit
--
--IN     UNIT_LOCATION     Units new location
--

procedure UTM_CONTROL_MEASURE (
                    PANEL_ID        :     in    SYS_WINDOW_ELE_ID;
                    CNTRL_MSR_COUNT :     in    SDB_CONTROL_MEASURE_ID;
                    CNTRL_MSR_TABLE :     in    SDB_CONTROL_MSR_POINT;
                    CNTRL_MSR_OPTION:     in    CM_CNTRL_MSR_OPTIONS);
--
-- CPM description: Displays all the control measures on the digital map
--
-- formal parameters
--IN     PANEL_ID          The id of the digital map panel.
--
--IN     CNTRL_MSR_COUNT   The number of control measures in the control
--                         measure table
--
--IN     CNTRL_MSR_TABLE   Array of records defining the control measures.
--
--IN     CNTRL_MSR_OPTION Default display options for the control measures
--

procedure UTM_CONTROL_MEASURE_CHG (
                    PANEL_ID        :     in    SYS_WINDOW_ELE_ID;
                    CNTRL_MSR_ID    :     in    SDB_CONTROL_MEASURE_ID;
                    CNTRL_MSR_DESC  :     in    SDB_CONTROL_MEASURE_REC);
--
-- CPM description: Updates a control measure on the digital map
--
-- formal parameters
--IN     PANEL_ID          The id of the digital map panel.
--
--IN     CNTRL_MSR_ID      The id of the changed control measure
--
--IN     CNTRL_MSR_DESC    Description of the control measure.
--

procedure UTM_CNTRL_MSR_POINT (
                    PANEL_ID        :     in    SYS_WINDOW_ELE_ID;
                    CNTRL_MSR_COUNT :     in    SDB_CONTROL_MEASURE_ID;
                    CNTRL_MSR_TABLE :     in    SDB_CNTRL_POINT_POINT);
--
-- CPM description: Displays all the point control measures on the digital
--                  map
--
-- formal parameters
```

```
--IN    PANEL_ID          The id of the digital map panel.
--
--IN    CNTRL_MSR_COUNT   The number of point control measures in the control
--                        measure table
--
--IN    CNTRL_MSR_TABLE   Array of records defining the control measures.
--

procedure UTM_CNTRL_MSR_POINT_CHG (
                  PANEL_ID       :     in    SYS_WINDOW_ELE_ID;
                  CNTRL_MSR_ID   :     in    SDB_CONTROL_MEASURE_ID;
                  CNTRL_MSR_DESC :     in    SDB_CNTRL_MSR_POINT_REC);
--
-- CPM description: Updates a point control measure on the digital map
--
-- formal parameters
--IN    PANEL_ID          The id of the digital map panel.
--
--IN    CNTRL_MSR_ID      The id of the changed control measure
--
--IN    CNTRL_MSR_DESC    Description of the control measure.
--

procedure UTM_DEFINE_BLUE_CM_MENU (
                  STRUCT_ID        :        in     SYS_WINDOW_ELE_ID;
                  MENU_COUNT       :        in     SYS_MENU_TREE_LIMIT;
                  MENU_LIST        :        in     SYS_MENU_TREE_PTR;
                  ASSOCIATED_TABLE :        in     SYS_CM_OPTION_PTR);
--
-- CPM description: Defines the pop up menus to be used by the blue control
--                  measure menu software
--
-- formal parameters
--IN    MENU_TYPE         The type of menu to define
--
--IN    STRUCT_ID         The id to assign to the blue control measure  pop
--                        up menu.
--
--IN    MENU_COUNT        The number of menu selections in MENU_LIST.
--
--IN    MENU_LIST         A string array of the blue control measure walking
--                        menu in outline format. The first character of each
--                        line must be blank and menu children should be
--                        indented one character from its parent.
--
--IN    ASSOCIATED_TABLE  A table of ids to be associated with each menu
--                        element
-- end formal parameters;

procedure UTM_DEFINE_BLUE_OBS_MENU (
                  STRUCT_ID        :        in     SYS_WINDOW_ELE_ID;
                  MENU_COUNT       :        in     SYS_MENU_TREE_LIMIT;
                  MENU_LIST        :        in     SYS_MENU_TREE_PTR;
                  ASSOCIATED_TABLE :        in     SYS_OBS_OPTION_PTR);
--
-- CPM description: Defines the pop up menus to be used by the blue
```

```
--                    obstacle menu software
--
-- formal parameters
--IN     MENU_TYPE          The type of menu to define
--
--IN     STRUCT_ID          The id to assign to the blue obstacle pop up menu.
--
--IN     MENU_COUNT         The number of menu selections in MENU_LIST.
--
--IN     MENU_LIST          A string array of the blue obstacle walking
--                          menu in outline format. The first character of each
--                          line must be blank and menu children should be
--                          indented one character from its parent.
--
--IN     ASSOCIATED_TABLE   A table of ids to be associated with each menu
--                          element
-- end formal parameters;

procedure UTM_DEFINE_BLUE_UNIT_MENU (
                    STRUCT_ID        :        in     SYS_WINDOW_ELE_ID;
                    MENU_COUNT       :        in     SYS_MENU_TREE_LIMIT;
                    MENU_LIST        :        in     SYS_MENU_TREE_PTR;
                    ASSOCIATED_TABLE :        in     SYS_UNIT_OPTION_PTR);
--
-- CPM description: Defines the pop up menus to be used by the blue unit menu
--                   software
--
-- formal parameters
--IN     MENU_TYPE          The type of menu to define
--
--IN     STRUCT_ID          The id to assign to the blue unit pop up menu.
--
--IN     MENU_COUNT         The number of menu selections in MENU_LIST.
--
--IN     MENU_LIST          A string array of the blue unit walking menu
--                          in outline format. The first character of each
--                          line must be blank and menu children should be
--                          indented one character from its parent.
--
--IN     ASSOCIATED_TABLE   A table of ids to be associated with each menu
--                          element
-- end formal parameters;

procedure UTM_DEFINE_MAP_MENU (
                    STRUCT_ID        :        in     SYS_WINDOW_ELE_ID;
                    MENU_COUNT       :        in     SYS_MENU_TREE_LIMIT;
                    MENU_LIST        :        in     SYS_MENU_TREE_PTR;
                    ASSOCIATED_TABLE :        in     SYS_MAP_CONTROL_PTR);
--
-- CPM description: Defines the pop up menus to be used by the map control
--                  software
--
-- formal parameters
--IN     STRUCT_ID          The id to assign to the map control pop up menu.
--
--IN     MENU_COUNT         The number of menu selections in MENU_LIST.
```

```
--
--IN     MENU_LIST          A string array of the map control walking menu
--                          in outline format. The first character of each
--                          line must be blank and menu children should be
--                          indented one character from its parent.
--
--IN     ASSOCIATED_TABLE   A table of ids to be associated with each menu
--                          element
-- end formal parameters;

procedure UTM_DEFINE_MAP_PANEL (
                    WINDOW_ID     :      in    SYS_WINDOW_ELE_ID;
                    PANEL_ID      :      out   SYS_WINDOW_ELE_ID;
                    PIXEL_X       :      in    SYS_WINDOW_COLUMN;
                    PIXEL_Y       :      in    SYS_WINDOW_ROW;
                    PIXEL_WIDTH   :      in    SYS_WINDOW_COLUMN;
                    PIXEL_HEIGHT  :      in    SYS_WINDOW_ROW;
                    CALL_PROCESS  :      in    SYS_EDDIC_PROCESSES;
                    SCL_PROCESS   :      in    SYS_EDDIC_PROCESSES;
                    PART_OF_FORM  :      in    BOOLEAN := false);
--
-- CPM description: Defines the digital map panel
--
-- formal parameters
--IN     WINDOW_ID          The id of the window to contain the map panel.
--
--OUT    PANEL_ID           The id of the digital map panel.
--
--IN     PIXEL_X            The window X coordinate of the upper left corner of
--                          the digital map panel.
--
--IN     PIXEL_Y            The window Y coordinate of the upper left corner of
--                          the digital map panel.
--
--IN     PIXEL_WIDTH        The width of the digital map panel in pixels.
--
--IN     PIXEL_HEIGHT       The height of the digital map panel in pixels.
--
--IN     CALL_PROCESS       The ID of the calling process
--
--IN     SCL_PROCESS        The ID of the Station Control Process
--
--IN     PART_OF_FORM       Logical flag to indicate if the map panel is part
--                          of a form. This flag is used to determine if the
--                          current map scale should be displayed in the window
--                          top border. It is not displayed in a form.
--

procedure UTM_DEFINE_OPFOR_CM_MENU (
                    STRUCT_ID       :      in     SYS_WINDOW_ELE_ID;
                    MENU_COUNT      :      in     SYS_MENU_TREE_LIMIT;
                    MENU_LIST       :      in     SYS_MENU_TREE_PTR;
                    ASSOCIATED_TABLE :     in     SYS_CM_OPTION_PTR);
--
-- CPM description: Defines the pop up menus to be used by the OPFOR control
--                  measure menu software
```

```
--
-- formal parameters
--IN     MENU_TYPE          The type of menu to define
--
--IN     STRUCT_ID          The id to assign to the OPFOR control measure  pop
--                          up menu.
--
--IN     MENU_COUNT         The number of menu selections in MENU_LIST.
--
--IN     MENU_LIST          A string array of the OPFOR control measure walking
--                          menu in outline format. The first character of each
--                          line must be blank and menu children should be
--                          indented one character from its parent.
--
--IN     ASSOCIATED_TABLE   A table of ids to be associated with each menu
--                          element
-- end formal parameters;

procedure UTM_DEFINE_OPFOR_OBS_MENU (
                    STRUCT_ID        :        in     SYS_WINDOW_ELE_ID;
                    MENU_COUNT       :        in     SYS_MENU_TREE_LIMIT;
                    MENU_LIST        :        in     SYS_MENU_TREE_PTR;
                    ASSOCIATED_TABLE :        in     SYS_OBS_OPTION_PTR);
--
-- CPM description: Defines the pop up menus to be used by the OPFOR
--                   obstacle menu software
--
-- formal parameters
--IN     MENU_TYPE          The type of menu to define
--
--IN     STRUCT_ID          The id to assign to the OPFOR obstacle pop up menu.
--
--IN     MENU_COUNT         The number of menu selections in MENU_LIST.
--
--IN     MENU_LIST          A string array of the OPFOR obstacle walking
--                          menu in outline format. The first character of each
--                          line must be blank and menu children should be
--                          indented one character from its parent.
--
--IN     ASSOCIATED_TABLE   A table of ids to be associated with each menu
--                          element
-- end formal parameters;

procedure UTM_DEFINE_OPFOR_UNIT_MENU (
                    STRUCT_ID        :        in     SYS_WINDOW_ELE_ID;
                    MENU_COUNT       :        in     SYS_MENU_TREE_LIMIT;
                    MENU_LIST        :        in     SYS_MENU_TREE_PTR;
                    ASSOCIATED_TABLE :        in     SYS_UNIT_OPTION_PTR);
--
-- CPM description: Defines the pop up menus to be used by the OPFOR unit
--                   menu software
--
-- formal parameters
--IN     MENU_TYPE          The type of menu to define
--
--IN     STRUCT_ID          The id to assign to the OPFOR unit pop up menu.
```

```
--
--IN     MENU_COUNT          The number of menu selections in MENU_LIST.
--
--IN     MENU_LIST           A string array of the OPFOR unit walking menu
--                           in outline format. The first character of each
--                           line must be blank and menu children should be
--                           indented one character from its parent.
--
--IN     ASSOCIATED_TABLE    A table of ids to be associated with each menu
--                           element
-- end formal parameters;

procedure UTM_DEFINE_OPLAN (
                        OPPLAN_ID       : in  SYS_OPPLAN;
                        DATE_TIME       : in  SYS_DATE_TIME;
                        SOCKET          : in  SYS_CLIENT);
--
-- CPM description:  Defines the current Operational Plan and Date/Time
--                   for Situation Data retrievals.
--
-- formal parameters
--IN    OPPLAN_ID    Id of the current OPPLAN.
--
--IN    DATE_TIME    Date and time for the situation data requests.
--
--IN    SOCKET       The number of the socket for the situation DB manager.
--
-- end formal parameters;

procedure UTM_DELETE_MAP_MENUS (
                    PANEL_ID      :      in    SYS_WINDOW_ELE_ID);
--
-- CPM description: Deletes the digital map multiple selection menus
--
-- formal parameters
--IN     PANEL_ID          The id of the digital map panel.
--

procedure UTM_DELETE_MAP_PANEL (
                    PANEL_ID      :      in    SYS_WINDOW_ELE_ID);
--
-- CPM description: Deletes the digital map panel
--
-- formal parameters
--IN     PANEL_ID          The id of the digital map panel.
--

procedure UTM_ERASE_OVERLAY (
                    PANEL_ID      :      in    SYS_WINDOW_ELE_ID);
--
-- CPM description: Erases all the unit and control measure overlays
--
-- formal parameters
--IN     PANEL_ID          The id of the digital map panel.
--
```

```
procedure UTM_INPUT (
                    INPUT_TYPE          :     out       SYS_WINDOW_INPUT;
                    INPUT_WINDOW        :     out       SYS_WINDOW_ELE_ID;
                    INPUT_VALUE         :     out       SYS_WINDOW_VALUE;
                    INPUT_DATA          :     out       SYS_WINDOW_DATA);
--
-- CPM description: Processes user input when the digital map panel is
--                  displayed.  This procedure will intercept all map
--                  inputs and return the rest to the call procedure.
--
-- formal parameters
--OUT   INPUT_TYPE          Type of input returned from the window system
--
--OUT   INPUT_WINDOW        The id of the window which received the input.
--
--OUT   INPUT_VALUE         The value of the input that accompanies the type
--
--OUT   INPUT_DATA          The value of the data that accompanies the type
--
--  The following table lists the map specific output returned to the
--  application for its own processing:
--
--                     window-
-- input_type            id       value_code          data
-- ----------          -------    ----------          --------
--20 Map Input         X          0 Map                n/a
--                                1 Blue Unit Chg      (1) Unit Index
--                                2 Blue Unit Deact    (1) Unit Index
--                                3 OPFOR Unit Chg     (1) Unit Index
--                                4 OPFOR Unit Deact   (1) Unit Index
--                                5 Cntrl Msr Chg      (1) CM Index
--                                                     (2) 1 = Single Point
--                                                     (2) 2 = Multiple Point
--                                6 Cntrl Msr Del      (1) CM Index
--                                                     (2) 1 = Single Point
--                                                     (2) 2 = Multiple Point
--                                7 Obstacle Chg       (1) Obs Index
--                                8 Obstacle Del       (1) Obs Index
--                                9 Oplan Chg          n/a
--                               10 New Work Oplan     n/a

procedure UTM_MAP_BACKGROUND (
                    PANEL_ID       :      in     SYS_WINDOW_ELE_ID;
                    MAP_OPTION     :      in     MAP_MAP_OPTIONS);
--
-- CPM description: Displays the digital map background image
--
-- formal parameters
--IN    PANEL_ID          The id of the digital map panel.
--
--IN    MAP_OPTION        The initial map display options
--

procedure UTM_MOVE_MAP (
                    PANEL_ID       :      in     SYS_WINDOW_ELE_ID;
                    PIXEL_X        :      in     SYS_IMAGE_COLUMN;
```

```
                    PIXEL_Y       :      in    SYS_IMAGE_ROW);
--
-- CPM description: Changes the location of the displayed digital map.
--
-- formal parameters
--IN     PANEL_ID          The id of the digital map panel.
--
--IN     PIXEL_X           The number of pixels to move in the X direction.
--
--IN     PIXEL_Y           The number of pixels to move in the Y direction.
--

procedure UTM_OBSTACLE (
                    PANEL_ID       :      in    SYS_WINDOW_ELE_ID;
                    OBSTACLE_COUNT :      in    SDB_OBSTACLE_ID;
                    OBSTACLE_TABLE :      in    SDB_OBSTACLE_POINT;
                    OBSTACLE_OPTION:      in    OBS_OBSTACLE_OPTIONS);
--
-- CPM description: Displays all the obstacles on the digital map
--
-- formal parameters
--IN     PANEL_ID          The id of the digital map panel.
--
--IN     OBSTACLE_COUNT    The number of obstacles in the obstacle table
--
--IN     OBSTACLE_TABLE    Array of records defining the obstacles.
--
--IN     OBSTACLE_OPTION   The default obstacle display options
--

procedure UTM_OBSTACLE_CHG (
                    PANEL_ID       :      in    SYS_WINDOW_ELE_ID;
                    OBSTACLE_ID    :      in    SDB_OBSTACLE_ID;
                    OBSTACLE_DESC  :      in    SDB_OBSTACLE_REC);
--
-- CPM description: Change the display of an obstacle on the digital map
--
-- formal parameters
--IN     PANEL_ID          The id of the digital map panel.
--
--IN     OBSTACLE_ID       The id of the obstacle measure
--
--IN     OBSTACLE_DESC     Description of the obstacle.
--

procedure UTM_OPFOR_UNITS (
                    PANEL_ID       :      in    SYS_WINDOW_ELE_ID;
                    UNIT_COUNT     :      in    SDB_OPFOR_UNIT_ID;
                    UNIT_LOC_TABLE :      in    SDB_LOCATION_LIST_POINT;
                    UNIT_OPTION    :      in    UNIT_OPTIONS);
--
-- CPM description: Displays all the OPFOR units on the digital map
--
-- formal parameters
--IN     PANEL_ID          The id of the digital map panel.
--
```

```
    --IN    UNIT_COUNT        The number of units in the unit location table
    --
    --IN    UNIT_LOC_TABLE    Array of records defining the unit names, location
    --                        echelon and type.
    --
    --IN    UNIT_OPTION       The default unit display options
    --

    procedure UTM_OPFOR_UNIT_CHG (
                      PANEL_ID       :     in    SYS_WINDOW_ELE_ID;
                      UNIT_ID        :     in    SDB_OPFOR_UNIT_ID;
                      UNIT_LOCATION  :     in    SDB_LOCATION_REC);

    --
    -- CPM description: Changes the location of a OPFOR unit on the digital map
    --
    -- formal parameters
    --IN    PANEL_ID          The id of the digital map panel.
    --
    --IN    UNIT_ID           The id of the changed unit
    --
    --IN    UNIT_LOCATION     New location of the unit
    --

    procedure UTM_RESIZE_MAP_PANEL (
                      PANEL_ID       :     in    SYS_WINDOW_ELE_ID;
                      PIXEL_X        :     in    SYS_WINDOW_COLUMN;
                      PIXEL_Y        :     in    SYS_WINDOW_ROW;
                      PIXEL_WIDTH    :     in    SYS_WINDOW_COLUMN;
                      PIXEL_HEIGHT   :     in    SYS_WINDOW_ROW);
    --
    -- CPM description: Changes the size and location of the digital map panel
    --
    -- formal parameters
    --IN    PANEL_ID          The id of the digital map panel.
    --
    --IN    PIXEL_X           The window X coordinate of the upper left corner of
    --                        the digital map panel.
    --
    --IN    PIXEL_Y           The window Y coordinate of the upper left corner of
    --                        the digital map panel.
    --
    --IN    PIXEL_WIDTH       The width of the digital map panel in pixels.
    --
    --IN    PIXEL_HEIGHT      The height of the digital map panel in pixels.
    --

end UTM_TACTICAL_MAP;
```

```
--cpc package specification name: UUE_STATUS_REPORT
--
--cpc description: UUE_STATUS_REPORT displays the unit status reports for the
--                 task organization tool.
--
--cpc design notes:
--
--cpc package author: Bruce Packard
--                    Science Applications International Corporation
--                    424 Delaware, Suite C3
--                    Leavenworth, KS 66048
--
with SYSTEM_PACKAGE;                use SYSTEM_PACKAGE;
with SDB_SITUATION_DB;              use SDB_SITUATION_DB;
with TSB_LOCATION;

package UUE_STATUS_REPORT is

   package TSBL      is new TSB_LOCATION(SDB_BLUE_TASK_RECORD);  use TSBL;

   procedure UUE_DEFINE_STATUS_PIXMAP;

   --
   -- CPM description:  Defines the pixmaps required for the graphic status
   --                   reports.  This procedure should be call one time before
   --                   the status report tasks are called.
   --
   -- formal parameters
   --  None
   --
   -- end formal parameters;

   task type UUE_DETAIL_STATUS is
      entry INITIALIZE    (UNIT_ID          : in  SDB_UNIT;
                           UNIT_NAME        : in  STRING;
                           OPPLAN_ID        : in  SYS_OPPLAN;
                           DATE_TIME        : in  SYS_DATE_TIME;
                           SOCKET           : in  SYS_CLIENT;
                           PROCESS          : in  SYS_EDDIC_PROCESSES;
                           WINDOW           : out SYS_WINDOW_ELE_ID);
      --
      -- CPM description:  This entry point creates a popup window to display
      --                   a unit detail status report in and gets the
      --                   required data from the situation DB manager.
      --
      -- formal parameters
      --IN    UNIT_ID      Id of the unit to display the status report for.
      --
      --IN    UNIT_NAME    Name the unit to display the status report for.
      --
      --IN    OPPLAN_ID    Id of the current OPPLAN.
      --
      --IN    DATE_TIME    Date and time for the situation data requests.
      --
      --IN    SOCKET       The number of the socket for the situation DB manager.
      --
```

```
      --IN    PROCESS      The name of the parent process.
      --
      --OUT   WINDOW       The ID of the newly created popup window.
      --
      -- end formal parameters;

      entry PROCESS_INPUT (NEW_WINDOW_INPUT  : in  SYS_WINDOW_INPUT;
                           NEW_WINDOW_VALUE  : in  SYS_WINDOW_VALUE;
                           NEW_WINDOW_DATA   : in  SYS_WINDOW_DATA;
                           WINDOW_TERMINATED : out BOOLEAN);

      --
      -- CPM description:  This entry point processes and input that has
      --                   happened for the popup window created by INITIALIZE.
      --                   This entry point should be called for all input
      --                   from UWN that matches the window ID from INITIALIZE.
      --                   The WINDOW_TERMINATED flag is set to true if the
      --                   selected action causes the deletion of the popup
      --                   window.
      --
      -- formal parameters
      --IN    NEW_WINDOW_INPUT  Input type (See UWN_WINDOW_SYSTEM for a
      --                        complete description).
      --
      --IN    NEW_WINDOW_VALUE  Input value (See UWN_WINDOW_SYSTEM for a
      --                        complete description).
      --
      --IN    NEW_WINDOW_DATA   Input data (See UWN_WINDOW_SYSTEM for a
      --                        complete description).
      --
      --OUT   WINDOW_TERMINATED Window Termination flag
      --                        true  = Window was terminated
      --                        false = Window was not terminated.
      --
      -- end formal parameters;

      entry TERMINATE_TASK;
      --
      -- CPM description:  This entry point terminates popup status window.
      --

end;

task type UUE_SUMMARY_STATUS is
   entry INITIALIZE    (UNIT_DESC       : in  TREE_RECORD_PTR;
                        OPPLAN_ID       : in  SYS_OPPLAN;
                        DATE_TIME       : in  SYS_DATE_TIME;
                        SOCKET          : in  SYS_CLIENT;
                        PROCESS         : in  SYS_EDDIC_PROCESSES;
                        WINDOW          : out SYS_WINDOW_ELE_ID);
   --
   -- CPM description:  This entry point creates a popup window to display
   --                   a unit summary status report in and gets the
   --                   required data from the situation DB manager.
   --
   -- formal parameters
```

```
        --IN    UNIT_DESC    Description of the unit that the summary report was
        --                   requested for.
        --
        --IN    OPPLAN_ID    Id of the current OPPLAN.
        --
        --IN    DATE_TIME    Date and time for the situation data requests.
        --
        --IN    SOCKET       The number of the socket for the situation DB manager.
        --
        --IN    PROCESS      The name of the parent process.
        --
        --OUT   WINDOW       The ID of the newly created popup window.
        --
        -- end formal parameters;

        entry PROCESS_INPUT (NEW_WINDOW_INPUT  : in  SYS_WINDOW_INPUT;
                             NEW_WINDOW_VALUE  : in  SYS_WINDOW_VALUE;
                             NEW_WINDOW_DATA   : in  SYS_WINDOW_DATA;
                             WINDOW_TERMINATED : out BOOLEAN);

        --
        -- CPM description:  This entry point processes and input that has
        --                   happened for the popup window created by INITIALIZE.
        --                   This entry point should be called for all input
        --                   from UWN that matches the window ID from INITIALIZE.
        --                   The WINDOW_TERMINATED flag is set to true if the
        --                   selected action causes the deletion of the popup
        --                   window.
        --
        -- formal parameters
        --IN    NEW_WINDOW_INPUT  Input type (See UWN_WINDOW_SYSTEM for a
        --                        complete description).
        --
        --IN    NEW_WINDOW_VALUE  Input value (See UWN_WINDOW_SYSTEM for a
        --                        complete description).
        --
        --IN    NEW_WINDOW_DATA   Input data (See UWN_WINDOW_SYSTEM for a
        --                        complete description).
        --
        --OUT   WINDOW_TERMINATED Window Termination flag
        --                        true  = Window was terminated
        --                        false = Window was not terminated.
        --
        -- end formal parameters;

        entry TERMINATE_TASK;
        --
        -- CPM description:  This entry point terminates popup status window.
        --

    end;

end UUE_STATUS_REPORT;
```

```
--cpc package specification name: UUE_UNIT_EDITOR
--
--cpc description: UUE_UNIT_EDITOR contains the low level unit display utilities
--                 for determining what echelon and unit type symbol to display
--                 for a specific unit.
--
--cpc design notes:
--    This package raises the SYS_UUE_EXCEPTION when an exception is detected.
--
--cpc package author: Bruce Packard
--                   Science Applications International Corporation
--                   424 Delaware, Suite C3
--                   Leavenworth, KS 66048
--
with SYSTEM_PACKAGE;                use SYSTEM_PACKAGE;
with UNIT_SYSTEM;                   use UNIT_SYSTEM;
with SDB_SITUATION_DB;              use SDB_SITUATION_DB;

package UUE_UNIT_EDITOR is

      -- Unit Type to unit symbol font conversion table
      UNIT_SYMBOL_INDEX :    Array (SDB_UNIT_TYPE'FIRST..SDB_UNIT_TYPE'LAST) of
                             INTEGER :=
                                (AIRBORNE => 35,
                                 AIR_ASSAULT => 36,
                                 AIR_DEFENSE => 15,
                                 AIR_DEFENSE_MISSLE => 16,
                                 ANTI_ARMOR => 18,
                                 ARMOR_CAV => 20,
                                 ARMOR_TANK => 19,
                                 ARTY_TOWED => 22,
                                 ARTY_SP => 23,
                                 ATTACK_HELICOPTER => 26,
                                 AVIATION => 25,
                                 AVIATION_FW => 27,
                                 AVIATION_RW => 28,
                                 BAND => 44,
                                 CAV_RECON => 21,
                                 CHEMICAL => 29,
                                 CIVIL_AFFAIRS => 30,
                                 COMBINED_ARMS_ARMY => 31,
                                 ENGINEER => 32,
                                 FINANCE => 33,
                                 INF_MECHANIZED => 37,
                                 INF_MOTORIZED => 38,
                                 MAINTENANCE => 39,
                                 MEDICAL => 40,
                                 MILITARY_INTEL => 41,
                                 MILITARY_POLICE => 43,
                                 ORDNANCE => 45,
                                 PERS_SVC => 46,
                                 PSYCH_OPNS => 47,
                                 QUARTERMASTER => 48,
                                 ROCKET_ARTILLERY => 24,
                                 SIGNAL => 49,
                                 SPECIAL_FORCES => 51,
```

```
                                SPT_COM => 52,
                                SUPPLY_SERVICES => 50,
                                SURF_TO_SURF_MISSLE => 17,
                                TRANSPORTATION => 53);

   -- Echelon to echelon symbol font conversion table
   ECHELON_SYMBOL_INDEX   :   Array (SDB_FORCE_ECHELON'FIRST..
                                     SDB_FORCE_ECHELON'LAST) of INTEGER :=
                              (ARMY_GROUP..FRONT => 10,
                               ARMY => 9,
                               CORPS => 8,
                               DIVISION => 7,
                               BRIGADE => 6,
                               REGIMENT..GROUP => 5,
                               BATTALION..SQUADRON => 4,
                               COMPANY..TROOP => 3,
                               PLATOON => 2,
                               SECTION => 1,
                               SQUAD..TEAM => 0);

procedure UUE_ECHELON_SYMBOL (
                  DISPLAY_FLAG  :     in    BOOLEAN;
                  SIDE_TYPE     :     in    SDB_SIDE_TYPE;
                  ECHELON       :     in    SDB_FORCE_ECHELON;
                  PIXEL_X       :     in    SYS_IMAGE_COLUMN;
                  PIXEL_Y       :     in    SYS_IMAGE_ROW);
--
-- CPM description: Displays a unit echelon symbol on the digital map.
--
-- formal parameters
--IN    DISPLAY_FLAG      Flag to indicate if the symbol is begin drawn or
--                        erased. (True = Draw;  False = Erase)
--
--IN    SIDE_TYPE         The side which the echelon symbol will be
--                        representing.
--
--IN    ECHELON           The echelon.
--
--IN    PIXEL_X           Digital map panel X coordinate where the upper left
--                        corner of the echelon symbol is to be displayed.
--
--IN    PIXEL_Y           Digital map panel Y coordinate where the upper left
--                        corner of the echelon is to be displayed.
--
--

procedure UUE_UNIT_NAME (
                  DISPLAY_FLAG  :     in    BOOLEAN;
                  SIDE_TYPE     :     in    SDB_SIDE_TYPE;
                  NAME          :     in    STRING;
                  UNIT_LOCATION :     in    SDB_LOCATION_REC);
--
-- CPM description: Displays a unit name on the digital map.
--
-- formal parameters
--
```

```
    --IN    DISPLAY_FLAG        Flag to indicate if the symbol is begin drawn or
    --                          erased. (True = Draw;  False = Erase)
    --
    --IN    SIDE_TYPE           The side the unit symbol will be representing.
    --
    --IN    NAME                The name of the unit.
    --
    --IN    UNIT_LOCATION       Record of the unit's location.
    --

    procedure UUE_STATUS_BOX (
                        STATUS          :     in    UNIT_STATUS_DATA);
    --
    -- CPM description: Retrieves the unit type symbol for a specific unit type.
    --
    -- formal parameters
    --IN    STATUS              Unit data to display in the status box.
    --

    procedure UUE_UNIT_SYMBOL (
                        DISPLAY_FLAG  :     in    BOOLEAN;
                        SIDE_TYPE     :     in    SDB_SIDE_TYPE;
                        UNIT_TYPE     :     in    SDB_UNIT_TYPE;
                        UNIT_ECHELON  :     in    SDB_FORCE_ECHELON;
                        UNIT_LOCATION :     in    SDB_LOCATION_REC);
    --
    -- CPM description: Displays a unit type symbol on the digital map.
    --
    -- formal parameters
    --
    --IN    DISPLAY_FLAG        Flag to indicate if the symbol is begin drawn or
    --                          erased. (True = Draw;  False = Erase)
    --
    --IN    SIDE_TYPE           The side the unit symbol will be representing.
    --
    --IN    UNIT_TYPE           Type of the unit symbol to be displayed.
    --
    --IN    UNIT_ECHELON        Echelon of the unit symbol to be displayed.
    --
    --IN    UNIT_LOCATION       Record of the unit's location.
    --

end UUE_UNIT_EDITOR;
```

UUX Utility Package Specifications

The following package specifications are included in the Unix utility function:

UUX_IO
UUX_UTIL

```
--CPC package specification name:
--     UUX_IO
--
--CPC description:
--     UUX_IO CPC is a set of input/output primitives, written in the "Ada"
--     programming language, which allow programs access to low level
--     input/output.
--
--CPC design notes:
--     1.)  This package must be instantiated with its generic formal parameters.
--     2.)  This package can raise the following exceptions:
--          SYS_UUX_EXCEPTION.
--
--CPC package author:
--     Bruce J. Packard
--     Science Applications International Corporation (SAIC)
--     424 Delaware, Suite C-3
--     Leavenworth, KS  66048   (913) 651-7925
--

with SYSTEM_PACKAGE;         use SYSTEM_PACKAGE;

generic

   -- Types of buffers that can be used by the UUX I/O utilities.
   type UUX_IO_BUFFER is private;
   type UUX_IO_POINTER is access UUX_IO_BUFFER;

package UUX_IO is

   -- Input/Output parameters.
   type UUX_IO_OPERATION   is range 0..2;
   for  UUX_IO_OPERATION'SIZE use SYS_BITS_IN_BYTE;
   type UUX_IO_FORMAT      is range 0..1;
   for  UUX_IO_FORMAT'SIZE use SYS_BITS_IN_BYTE;
   UUX_IO_READ                : UUX_IO_OPERATION  := 0;
   UUX_IO_WRITE               : UUX_IO_OPERATION  := 1;
   UUX_IO_APPEND              : UUX_IO_OPERATION  := 2;
   UUX_IO_FIXED               : UUX_IO_FORMAT     := 0;
   UUX_IO_VARIABLE            : UUX_IO_FORMAT     := 1;

-- ####################' ##########################################################
   procedure UUX_BINARY_READ (FILE_DESC     : in  SYS_FILE_DESC;
                              OFFSET        : in  SYS_DB_SIZE;
                              RECORD_LENGTH : in  SYS_DB_SIZE;
                              FORMAT        : in  UUX_IO_FORMAT;
                              BUFFER        : in  UUX_IO_POINTER);
   --
   --CPM description:
   --     This module performs a binary (unformatted) read on a specific record
   --     of the specified file, which was opened by UUX_OPEN_FILE.
   --
   --CPM design notes:
   --     1.)  None.
   --
   --formal parameters
```

```
    --IN     FILE_DESC     - A pointer to the file descriptor returned from
    --                       UUX_OPEN_FILE.
    --IN     OFFSET        - The offset from the beginning of the file (Starts
    --                       at one).  For fixed length record files the offset
    --                       units are records.  For variable length record
    --                       files the offset units are bytes.
    --IN     RECORD_LENGTH - Number of bytes in this record to be read.
    --IN     FORMAT        - File format.
    --                     = 0 - Fixed length records.
    --                     = 1 - Variable length records.
    --OUT    BUFFER        - Pointer to the Buffer that was read.
    --end formal parameters;

-- ##############################################################################
    procedure UUX_BINARY_WRITE (FILE_DESC     : in  SYS_FILE_DESC;
                                OFFSET        : in  SYS_DB_SIZE;
                                RECORD_LENGTH : in  SYS_DB_SIZE;
                                FORMAT        : in  UUX_IO_FORMAT;
                                BUFFER        : in  UUX_IO_POINTER);
    --
    --CPM description:
    --    This module performs a binary (unformatted) write on a specific record
    --    of the specified file, which was opened by UUX_OPEN_FILE.
    --
    --CPM design notes:
    --    1.)  None.
    --
    --formal parameters
    --IN     FILE_DESC     - A pointer to the file descriptor returned from
    --                       UUX_OPEN_FILE.
    --IN     OFFSET        - The offset from the beginning of the file (Starts
    --                       at one).  For fixed length record files the offset
    --                       units are records.  For variable length record
    --                       files the offset units are bytes.
    --IN     RECORD_LENGTH - Number of bytes in this record to be written.
    --IN     FORMAT        - File format.
    --                     = 0 - Fixed length records.
    --                     = 1 - Variable length records.
    --IN     BUFFER        - Pointer to the Buffer to write to.
    --end formal parameters;

-- ##############################################################################
    procedure UUX_CLOSE_FILE (FILE_DESC : in  SYS_FILE_DESC);
    --
    --CPM description:
    --    This module closes a file opened by UUX_OPEN_FILE.
    --
    --CPM design notes:
    --    1.)  None.
    --
    --formal parameters
    --IN     FILE_DESC - A pointer to the file descriptor returned from
    --                   UUX_OPEN_FILE.
    --end formal parameters;

-- ##############################################################################
```

```
    procedure UUX_OPEN_FILE (FILE_NAME      : in   STRING;
                             FILE_OPERATION : in   UUX_IO_OPERATION;
                             FILE_DESC      : out  SYS_FILE_DESC);
    --
    --CPM description:
    --    This module opens a file for the performing of binary reads and writes.
    --
    --CPM design notes:
    --    1.)  None.
    --
    --formal parameters
    --IN     FILE_NAME       - The name of the file to be opened.
    --IN     FILE_OPERATION  - A flag that tells which Mode to Open the file.
    --                       = 0 - Read only.
    --                       = 1 - Read, write, and create if needed.
    --                       = 2 - Append.
    --OUT    FILE_DESC       - File descriptor assigned to the open file.
    --end formal parameters;

end UUX_IO;
```

```
--CPC package specification name:
--     UUX_UTIL
--
--CPC description:
--     UUX_UTIL CPC is a set of Utility primitives, written in the "Ada"
--     programming language, which allow programs to access UNIX operating
--     system commands.
--
--CPC design notes:
--     1.)  This package can raise the following exceptions:
--          SYS_UUX_EXCEPTION.
--
--CPC package author:
--     Bruce J. Packard
--     Science Applications International Corporation (SAIC)
--     424 Delaware, Suite C-3
--     Leavenworth, KS  66048    (913) 651-7925
--

with SYSTEM_PACKAGE;          use SYSTEM_PACKAGE;

package UUX_UTIL is

-- ##############################################################################
   procedure UUX_GETENV (ENV_STRING    : in      string;
                         RESULT_STRING : in out  string);
   --
   --CPM description:
   --     This module searches the Unix Environment list and returns (Gets) the
   --     evaluated, requested string.
   --
   --CPM design notes:
   --     1.)  None.
   --
   --formal parameters
   --IN      ENV_STRING     - The string that was created by a setenv.
   --OUT     RESULT_STRING - The evaluated Environment String.
   --end formal parameters;

-- ##############################################################################
   procedure UUX_SETENV (ENV_STRING    : in  string;
                         VALUE_STRING : in  string);
   --
   --CPM description:
   --     This module sets a Unix Environment variable to the requested string.
   --
   --CPM design notes:
   --     1.)  None.
   --
   --formal parameters
   --IN      ENV_STRING     - The environment variable string name.
   --IN      VALUE_STRING   - The value to set the environment variable to.
   --end formal parameters;

-- ##############################################################################
   procedure UUX_SYSTEM (CMD_STRING : in  string);
```

```
    --
    --CPM description:
    --     This module executes a Unix System call.
    --
    --CPM design notes:
    --     1.)  None.
    --
    --formal parameters
    --IN      CMD_STRING  - Command string to execute in the UNIX environment.
    --end formal parameters;

-- ##############################################################################
    procedure UUX_WAIT (SECONDS_TO_WAIT : in      SYS_DELAY;
                        SECONDS_WAITED  : in out  SYS_DELAY);
    --
    --CPM description:
    --     This module suspends a process for a specified period of time.
    --
    --CPM design notes:
    --     1.)  None.
    --
    --formal parameters
    --IN      SECONDS_TO_WAIT  - The number of seconds to suspend the process.
    --OUT     SECONDS_WAITED   - The number of seconds actually suspend.
    --end formal parameters;

end UUX_UTIL;
```

UWN_Utility Package Specifications

The following package specifications are included in the windowing system function:

```
DML_DSPL_MENU_LAYOUT
UWN_WALKING_MENU
UWN_WINDOW_SYSTEM
```

```
--CPC package specification name:
--     DML_DSPL_MENU_LAYOUT
--
--CPC description:
--     DML_DSPL_MENU_LAYOUT CPC is the Display Menu Layout, written in the "Ada"
--     programming language, which defines the variables and variable types
--     needed to draw walking and/or multiple selection menus hierarchy,
--     graphically.
--
--CPC design notes:
--     1.)  This package can raise the following exceptions:
--          SYS_UWN_EXCEPTION.
--
--CPC package author:
--     Richard T. Zarse      30 Mar 1989
--     Science Applications International Corporation (SAIC)
--     424 Delaware, Suite C-3
--     Leavenworth, KS  66048    (913) 651-7925
--
with SYSTEM_PACKAGE;      use SYSTEM_PACKAGE;
with TSB_LOCATION;

package DML_DSPL_MENU_LAYOUT is

   task type DML_DSPL_MENU_LAYOUT_TASK  is

-- ##############################################################################
      entry DSPL_INIT_MENU  (FILENAME     : in   SYS_TEXT_PTR;
                             FILENAMELEN  : in   SYS_ENV_STRING;
                             UPPERLEFT    : in   SYS_WINDOW_LOCATION;
                             MAXWINSIZE   : in   SYS_WINDOW_LOCATION;
                             FONTID       : in   SYS_WINDOW_ELE_ID;
                             FONTWIDTH    : in   SYS_WINDOW_COLUMN;
                             FONTHEIGHT   : in   SYS_WINDOW_ROW;
                             LUTCOLOR     : in   SYS_COLOR;
                             PLANEMASK    : in   SYS_COLOR_MASK;
                             WINDOW_ID    : out  SYS_WINDOW_ELE_ID;
                             SUBWINDOW_ID : out  SYS_WINDOW_ELE_ID);
      --
      --CPM description:
      --     This entry point creates a popup window which Displays the Initial
      --     chosen Menu.
      --
      --CPM design notes:
      --     1.)  None.
      --
      --formal parameters
      --IN      FILENAME      - The Name of the menu File which is to be
      --                        displayed.
      --IN      FILENAMELEN   - The actual Length (number of characters) of the
      --                        File Name.
      --IN      UPPERLEFT     - The structure containing the Upper Left, X & Y
      --                        location of the displaying Window.
      --IN      MAXWINSIZE    - The Maximum allowable Size, X & Y, of the
      --                        displaying Window.
      --IN      FONTID        - The Id of the display Font.
```

```
        --IN      FONTWIDTH     - The Width of an element in the Font.
        --IN      FONTHEIGHT    - The Height of an element in the Font.
        --IN      LUTCOLOR      - An index into the Color LookUp Table for
        --                        displaying and drawing the menu.
        --IN      PLANEMASK     - A bitmap Mask of the Planes to be affected in
        --                        displaying and drawing the menu.
        --OUT     WINDOW_ID     - The Id of the newly created popup Window.
        --OUT     SUBWINDOW_ID - The Id of the SubWindow inside the popup window,
        --                        where the picture is actually displayed.
        --end formal parameters;

    -- ##########################################################################
        entry PROCESS_INPUT (WINDOW_INPUT      : in   SYS_WINDOW_INPUT;
                             WINDOW_VALUE      : in   SYS_WINDOW_VALUE;
                             WINDOW_DATA       : in   SYS_WINDOW_DATA;
                             WINDOW_TERMINATED : out  BOOLEAN);
        --
        --CPM description:
        --    This entry point Processes any Input that has happened in/to the
        --    popup window.  This entry point will be called for all input from
        --    UWN that matches the window ID.
        --
        --CPM design notes:
        --    1.)  None.
        --
        --formal parameters
        --IN      WINDOW_INPUT      - The type of Input.
        --IN      WINDOW_VALUE      - The Value of the input.
        --IN      WINDOW_DATA       - The input Data.
        --        (See UWN_WINDOW_SYSTEM for a complete description of these 3).
        --OUT     WINDOW_TERMINATED - Window Termination flag
        --                           = true  - Window was terminated
        --                           = false - Window was not terminated.
        --
        --end formal parameters;

    -- ##########################################################################
        entry TERMINATE_TASK;
        --
        --CPM description:
        --    This entry point Terminates the display menu window.
        --
        --CPM design notes:
        --    1.)  None.
        --
        --formal parameters
        --       None.
        --end formal parameters;
        --
      end DML_DSPL_MENU_LAYOUT_TASK;

    end DML_DSPL_MENU_LAYOUT;
```

```
--cpc package specification name: UWN_WALKING_MENU
--
--cpc description: EDDIC Walking menu utilities.
--
--cpc design notes:
--    This package raises the SYS_UWN_EXCEPTION when an exception is detected.
--
--cpc package author: Bruce Packard
--                    Science Applications International Corporation
--                    424 Delaware, Suite C3
--                    Leavenworth, KS 66048
--
with SYSTEM_PACKAGE;          use SYSTEM_PACKAGE;
with UWN_WINDOW_SYSTEM;       use UWN_WINDOW_SYSTEM;
with UED_LIST;

generic

   type UWN_ASSOCIATED_TYPE is (<>);
   type UWN_ASSOCIATED_ARRAY is array (SYS_WALKING_CELL range <>) of
        UWN_ASSOCIATED_TYPE;
   type UWN_ASSOCIATED_POINTER  is   access UWN_ASSOCIATED_ARRAY;

package UWN_WALKING_MENU is

   -- Types for multiple selection menus defined in the walking menu
   type UWN_MULTIPLE_MENU is
      record
         MENU_OPTION       :  UWN_ASSOCIATED_TYPE;
         MENU_TEXT         :  STRING (SYS_POP_UP_TEXT);
         MENU_ON_OPTION    :  UWN_ASSOCIATED_TYPE;
         MENU_OFF_OPTION   :  UWN_ASSOCIATED_TYPE;
   end record;
   package UWN_MULT is new UED_LIST (UWN_MULTIPLE_MENU);
   type UWN_ASSOCIATED_LIST is  array (SYS_MENU_BUTTON_INDEX range <>) of
                           UWN_ASSOCIATED_TYPE;
   type UWN_ASSOCIATED_LIST_PTR is access UWN_ASSOCIATED_LIST;

   procedure UWN_BUILD_WALKING_MENU (
                       TREE_ELE_COUNT    :        in     SYS_MENU_TREE_LIMIT;
                       MENU_TREE         :        in     SYS_MENU_TREE_PTR;
                       ASSOCIATED_TABLE  :        in     UWN_ASSOCIATED_POINTER;
                       MENU_START        :        in     SYS_POP_UP_START_PTR;
                       MENU_LENGTH       :        in     SYS_POP_UP_LENGTH_PTR;
                       POP_UP_TEXT       :        in     SYS_MENU_TEXT_PTR;
                       POP_UP_CHILD      :        in     SYS_POP_UP_CHILD_PTR;
                       SORT_ASSOCIATED   :        in     UWN_ASSOCIATED_POINTER);
   --
   -- CPM description: Builds the walking menu structures from a string array
   --                  of the menu tree structure. Each branch should be
   --                  indented one character from its parent. This procedure
   --                  also builds an associated table of ids so that an id
   --                  can be assigned to each menu element.
   --
   -- formal parameters
   --IN    TREE_ELE_COUNT   The number of entries in the menu tree structure.
```

```
--
--IN     MENU_TREE          String array of the menu tree structure. Each branch
--                          must be indented one character from its parent The
--                          first menu item should start in column 3.
--
--IN     ASSOCIATED_TABLE   A table of ids to be associated with each menu
--                          element
--
--INOUT  MENU_START         Index into TEXT_ARRAY for the start of each pop-up
--                          menu in the walking menu.
--
--INOUT  MENU_LENGTH        Number of cells in each pop-up menu
--
--OUT    POP_UP_TEXT        Text for each cell of each pop-up menu in the
--                          walking menu
--
--OUT    POP_UP_CHILD       Pop-up index of the pop-up menu that is the child
--                          of each pop-up menu cell index into START_ARRAY
--                          and LENGTH_ARRAY;
--
--OUT    SORT_ASSOCIATED    Table of ids associated with each element in
--                          popup text array.
-- end formal parameters;

procedure UWN_BUILD_MULTIPLE (
                    MENU_NAME        :     in     SYS_TEXT_PTR;
                    MENU_COUNT       :     in     SYS_MENU_BUTTON_INDEX;
                    MENU             :     out    UWN_BUTTON_MENU_PTR;
                    ON_ACTIONS       :     out    UWN_ASSOCIATED_LIST_PTR;
                    OFF_ACTIONS      :     out    UWN_ASSOCIATED_LIST_PTR);
--
-- CPM description: Uses the list created by UWN_READ_WALKING_MENU to load
--                  a multiple selection menu record.
--
-- formal parameters
--IN     MENU_NAME          The name to put in the menu title bar.
--
--IN     MENU_COUNT         The number of items in the multiple selection menu.
--
--OUT    MENU               The description of the multiple selection menu.
--
--OUT    ON_ACTIONS         The options to perform for on selections.
--
--OUT    OFF_ACTIONS        The options to perform for off selections.
--
-- end formal parameters;

function UWN_MENU_COUNT return SYS_MENU_BUTTON_INDEX;
--
-- CPM description: Uses the list created by UWN_READ_WALKING_MENU to
--                  determine the number of elements in a multiple selection
--                  menu.
--

procedure UWN_READ_WALKING_MENU (
                    FILE_NAME        :        in     STRING;
```

```
                        TREE_ELE_COUNT   :        out    SYS_MENU_TREE_LIMIT;
                        MENU_TREE        :        in     SYS_MENU_TREE_PTR;
                        ASSOCIATED_TABLE :        in     UWN_ASSOCIATED_POINTER);
    --
    -- CPM description: Reads a walking menu structure from a ASCII file.
    --                  The text that is to appear in the menu must start in
    --                  column 3 and each submenu selection must be indented 1
    --                  column. The associated variable must start in column 35.
    --
    -- formal parameters
    --IN    FILE_NAME        The name of the menu description file.
    --
    --OUT   TREE_ELE_COUNT   The number of entries in the menu tree structure.
    --
    --OUT   MENU_TREE        String array of the menu tree structure. Each branch
    --                       must be indented one character from its parent The
    --                       first menu item should start in column 3.
    --
    --OUT   ASSOCIATED_TABLE A table of ids to be associated with each menu
    --                       element
    --
    -- end formal parameters;

end UWN_WALKING_MENU;
```

```
--cpc package specification name: UWN_WINDOW_SYSTEM
--
--cpc description: UWN_WINDOW_SYSTEM is the Ada version of the EDDIC window
--                 utilities using the X-window protocol. This package is
--                 an intermediate level between the applications software
--                 and the C based utilities (CWN);
--
--cpc design notes:
--
--cpc package author: Bruce Packard
--                    Science Applications International Corporation
--                    424 Delaware, Suite C3
--                    Leavenworth, KS 66048
--
with SYSTEM;   use SYSTEM;
with SYSTEM_PACKAGE;   use SYSTEM_PACKAGE;

package UWN_WINDOW_SYSTEM is

   -- array of allowed buttons and allowed button actions
   type UWN_BUTTON_ALLOWED    is  array (SYS_BUTTON_COUNT) of BOOLEAN;
   type UWN_BUTTON_ACTION     is  array (SYS_ACTION_COUNT) of BOOLEAN;

   -- types for use in the button menu manager

   subtype UWN_MENU_OPERATIONS is SYS_MENU_BUTTON_INDEX range 0..3;
      -- where each index is:
   UWN_EXIT_MENU          : constant UWN_MENU_OPERATIONS := 0;
   UWN_CANCEL_MENU        : constant UWN_MENU_OPERATIONS := 1;
   UWN_SET_ALL_MENU       : constant UWN_MENU_OPERATIONS := 2;
   UWN_CLEAR_ALL_MENU     : constant UWN_MENU_OPERATIONS := 3;

   -- Window mapping constants.
   UWN_MAP        : constant BOOLEAN  := True;
   UWN_DONT_MAP   : constant BOOLEAN  := False;

   type UWN_MENU_OPERATIONS_ARRAY is array (UWN_MENU_OPERATIONS) of Boolean;
   type UWN_MENU_OPERATIONS_PTR is access UWN_MENU_OPERATIONS_ARRAY;

   type UWN_MENU_BUTTON_TYPES is (CHECKBOX_BUTTON, RADIOBUTTON);

   type UWN_Button_Menu_Record (Button_Type : UWN_MENU_BUTTON_TYPES) is
      record
         Header :        SYS_TEXT_PTR;
         Total :         SYS_MENU_BUTTON_INDEX;
         Columns :       SYS_MENU_BUTTON_INDEX;
         Vis_Rows :      SYS_MENU_BUTTON_INDEX;
         Labels:         SYS_MENU_BUTTON_LABEL_PTR;
         Operations:     UWN_MENU_OPERATIONS_PTR;
         case Button_Type is
           when CHECKBOX_BUTTON =>
              Status:         SYS_MENU_BUTTON_STATUS_PTR;
           when RADIOBUTTON =>
              Default_RadioButton : SYS_MENU_BUTTON_INDEX;
         end case;
   end record;
```

```
type UWN_BUTTON_MENU_PTR is access UWN_Button_Menu_Record;

type UWN_BUTTON_MENU_OUTPUT is (DONE, CANCEL, NO_ACTION_REQUIRED);

type UWN_RECTANGLE_ARRAY is array (SYS_MENU_BUTTON_INDEX range <>) of
     SYS_RECTANGLE;

type UWN_RECTANGLE_ARRAY_PTR is access UWN_RECTANGLE_ARRAY;


procedure UWN_ACTIVATE_EDITOR (EDITOR_ID:   in   SYS_WINDOW_ELE_ID);
--
-- CPM description:  This routine activates an existing editor.  It is
--                   provided basically for traversing from a string field
--                   or numeric field to an editor.
--
-- formal parameters
--IN   EDITOR_ID   The id of the editor to activate.
--
-- end formal parameters;


procedure UWN_ACTIVATE_MENU (MENU_STRUCT_ID:  in  SYS_WINDOW_ELE_ID;
                             MENU_INDEX:      in  SYS_WALKING_CELL;
                             WINDOW_TYPE:     in  SYS_WINDOW_TYPE;
                             WINDOW_ID:       in  SYS_WINDOW_ELE_ID);
--
-- CPM description: This routine activates an already defined popup menu for
--                  either:
--                     a. A defined  window,
--                     b. a displayed panel (via cwn_end_panel),
--                     c. or, a defined button (via cwn_define_button).
--                  It also specifies the mode for posting the menu.
-- formal parameters
--IN    MENU_STRUCT_ID    The id of the menu structure given by the
--                        application at the time of the menu definition.
--
--IN    MENU_INDEX        The index into the Text_Array of the submenu to
--                        be activated for a particular window, if applicable.
--                        If the menu to be activated is not a walking menu,
--                        or is the top level of a walking menu, then this
--                        parameter should be set to NULL.
--
--IN    WINDOW_TYPE       The type of window the menu will be activated for,
--                        where:
--                          SYS_WINDOW          = a defined window
--                          SYS_DISPLAY_PANEL   = a displayed panel
--                          SYS_DEFINED_BUTTON  = defined button
--
--IN    WINDOW_ID         The id given at the time of the window type's
--                        creation where:
--                        If window_type is SYS_WINDOW and window_id is 0,
--                        then the menu will be activated for the RootWindow
--                        or (Display).  Otherwise, the menu will be activated
--                        for the matching window_id.
```

```
--                          If window_type = SYS_DISPLAY_PANEL, the id should
--                          be the panel id.
--                          If window_type = SYS_DEFINED_BUTTON, the id should
--                          be the button id.
-- end formal parameters;


procedure UWN_ACTIVATE_NUMBER_FIELD (
                                NUMBER_FIELD_ID: in  SYS_WINDOW_ELE_ID);
--
-- CPM description: This routine activates an existing number field.  It is
--                  provided basically for traversing from one number field
--                  to another.
--
-- formal parameters
--IN    NUMBER_FIELD_ID  The id of the numeric field to move to.
-- end formal parameters;


procedure UWN_ACTIVATE_STRING_FIELD (
                                STRING_FIELD_ID: in  SYS_WINDOW_ELE_ID);
--
-- CPM description: This routine activates an existing string field.  It is
--                  provided basically for traversing from one string field
--                  to another.
--
-- formal parameters
--IN    STRING_FIELD_ID  The id of the string field to move to.
--
-- end formal parameters;


procedure UWN_ADD_INPUT_SOCKET (SOCKET_ID:  in   SYS_CLIENT);

--
-- CPM description: UWN_ADD_INPUT_SOCKET adds a socket id to be watched by
--                  UWN_INPUT.  When a message is received on this socket,
--                  UWN_INPUT returns type SYS_INPUT_MESSAGE along with the
--                  socket ID. The applications software is responsible for
--                  reading the message.
--
-- formal parameters
--IN    SOCKET_ID        ID of the socket to watch for input.
-- end formal parameters;


procedure UWN_BUTTON_MENU_INPUT (INPUT_TYPE    :   in    SYS_WINDOW_INPUT;
                           MENU_WINDOW_ID  :   in    SYS_WINDOW_ELE_ID;
                           INPUT_VALUE     :   in    SYS_WINDOW_VALUE;
                           INPUT_DATA      :   in    SYS_WINDOW_DATA;
                           SELECTION_STATUS:   out   UWN_BUTTON_MENU_OUTPUT);
--
-- CPM description:   UWN_BUTTON_MENU_INPUT processes input performed
--                    within a button menu.
--
-- formal parameters
```

```
 --IN   INPUT_TYPE         Type of input returned from the window system
 --
 --IN   MENU_WINDOW_ID     The id of the menu window which received input.
 --
 --IN   INPUT_VALUE        The value of the input that accompanies the type
 --
 --IN   INPUT_DATA         The value of the data that accompanies the type
 --                        and input values, if appropriate.
 --
 --OUT  SELECTION_STATUS   Indicates status of user's selection process.
 --                         = CANCEL if user opted to cancel selection
 --                         = EXIT if user exited selection process where
 --                           the selection or selections made of the button
 --                           menu will be reflected in the input status
 --                           buffer or default_radiobutton, appropriately.
 --                         = NO_ACTION_REQUIRED if user simply selected
 --                           on button or scrollbar.
 --
 -- end formal parameters;

procedure UWN_CHANGE_BUTTON_LABEL (BUTTON_ID:        in   SYS_WINDOW_ELE_ID;
                                   BUTTON_TEXT:      in   string);

 --
 -- CPM description: UWN_CHANGE_BUTTON_LABEL changes the text displayed inside
 --                  a button created with UWN_DEFINE_BUTTON.
 --
 -- formal parameters
 --IN    BUTTON_ID         ID attached to the button.
 --
 --IN    BUTTON_TEXT       Textual string to display in the button.
 -- end formal parameters;


 procedure UWN_CHANGE_CHECKBOX_STATES (Checkbox_ID:   in   SYS_WINDOW_ELE_ID;
                         Num_Fields:    in   SYS_MENU_BUTTON_INDEX;
                         Start_Index:   in   SYS_MENU_BUTTON_INDEX;
                         Status_Array:  in out  SYS_MENU_BUTTON_STATUS_PTR;
                         State_Flag:    in   BOOLEAN);
 --
 -- CPM description:  CWN_CHANGE_CHECKBOX_STATES changes one or more
 --                   checkbox states according to the input state flag.
 --
 -- formal parameters
 --IN   Checkbox_ID       The ID attached to the checkbox editor.
 --
 --IN   Num_Fields        The number of checkbox(es) states to be changed.
 --
 --IN   Start_Index       The correlating index of the checkbox which the
 --                       start of the array to the order the items were
 --                       originally created; the first element is always
 --                       zero.
 --
 --IN   Status_Array      The array of current status of the checkboxes to
 --                       be changed.
```

```
--
--IN   State_Flag          The flag indicating the state all the checkboxes
--                         are to match.
-- end formal parameters;

procedure UWN_CHANGE_EDITOR_TEXT (EDITOR_ID:         in   SYS_WINDOW_ELE_ID;
                                  MAX_BUFFER_SIZE:   in   SYS_PRODUCT_LENGTH;
                                  TEXT_BUFFER:       in   SYS_TEXT_PTR;
                                  BUFFER_SIZE:       in   SYS_PRODUCT_LENGTH);

--
-- CPM description: Changes the text buffer used by the window full page
--                  text editor.
--
-- formal parameters
--IN    EDITOR_ID          ID attached to the editor.
--
--IN    MAX_BUFFER_SIZE    Maximum number of pixels that the TEXT_BUFFER
--                         can hold.
--
--IN    TEXT_BUFFER        Buffer of the initial text to display in the editor.
--
--IN    BUFFER_SIZE        The number of pixels in TEXT_BUFFER.
-- end formal parameters;


procedure UWN_CHANGE_ICON_LABEL (ICON_LABEL:       in   SYS_ICON_NAME);

--
-- CPM description: UWN_CHANGE_ICON_LABEL changes the icon label displayed
--                  in the window's icon.
--
-- formal parameters
--IN    ICON_LABEL          Textual string to display in the icon.

-- end formal parameters;


procedure UWN_CHANGE_SCROLLBAR (SCROLLBAR_ID:    in   SYS_WINDOW_ELE_ID;
                                DOC_SIZE:        in   SYS_PIXEL;
                                PIXEL_LENGTH:    in   SYS_WINDOW_PIXEL;
                                DISP_POSITION:   in   SYS_PIXEL;
                                SCROLL_INTRVL:   in   SYS_WINDOW_PIXEL);

--
-- CPM description: Changes the size of a scrollbar.
-- formal parameters
--IN    SCROLLBAR_ID       ID to attached to the scrollbar.
--                         This ID was defined by UWN_DEFINE_SCROLLBAR.
--
--IN    DOC_SIZE           The number of lines in the document buffer.
--
--IN    PIXEL_LENGTH       The number of pixels to be occupied by the
--                         scrollbar.
--IN    SCROLL_INTRVL      The number of pixels the work will be scrolled
```

```
--                          whenever the user selects an arrow button.  Note:
--                          The work will not be scrolled by these utilities
--                          but, this argument is required to calculate
--                          the interactive slidepositioning.
-- end formal parameters;

procedure UWN_CHANGE_WINDOW_LABEL (WINDOW_LABEL:    in   SYS_WINDOW_NAME;
                                   LABEL_POSITION:  in   SYS_TEXT_ALIGNMENT);

--
-- CPM description: UWN_CHANGE_WINDOW_LABEL changes the window label
--                  displayed in the window's top border.
--
-- formal parameters
--IN    WINDOW_LABEL      Textual string to display.
--IN    LABEL_POSITION    The position of the window label in the title
--                        bar to be changed.  A position of NONE will result
--                        in a change to the center window label.
-- end formal parameters;


procedure UWN_CLEAR_WINDOW;

--
-- CPM description: Erases all elements of a defined window.
--
-- formal parameters
-- None
-- end formal parameters;


procedure UWN_CLOSE_WINDOW;
--
-- CPM description:  This procedure closes a window into an icon.
--
-- formal parameters
-- NONE
-- end formal parameters;


procedure UWN_CREATE_EXPOSURE_EVENT (WINDOW_ID:    in   SYS_WINDOW_ELE_ID);
--
-- CPM description:  This procedure creates an exposure event for a
--                   particular window.
--
-- formal parameters
--IN    WINDOW_ID    The ID attached to the window.
-- end formal parameters;


procedure UWN_CREATE_EXPOSURE_EVENT (WINDOW_ID:  in  SYS_WINDOW_ELE_ID;
                                     UL_X:       in  SYS_WINDOW_COLUMN;
                                     UL_Y:       in  SYS_WINDOW_ROW;
                                     EXP_WIDTH:  in  SYS_WINDOW_COLUMN;
                                     EXP_HEIGHT: in  SYS_WINDOW_ROW);
```

```
--
-- CPM description:  This procedure creates an exposure event for a
--                   particular window.
--
-- formal parameters
--IN     WINDOW_ID     The Id of the Window to expose.
--IN     UL_X        = The Upper Left X corner of the area to expose.
--IN     UL_Y        = The Upper Left Y corner of the area to expose.
--IN     EXP_WIDTH   = The Width of the area to Expose.
--IN     EXP_HEIGHT  = The Height of the area to Expose.
-- end formal parameters;


procedure UWN_CREATE_SUBWINDOW (WINDOW_ID:      in   SYS_WINDOW_ELE_ID;
                                MAP_WINDOW:     in   BOOLEAN;
                                PIXEL_COL:      in   SYS_WINDOW_COLUMN;
                                PIXEL_ROW:      in   SYS_WINDOW_ROW;
                                PIXEL_WIDTH:    in   SYS_WINDOW_COLUMN;
                                PIXEL_HEIGHT:   in   SYS_WINDOW_COLUMN;
                                BORDER_WIDTH:   in   SYS_WINDOW_COLUMN;
                                SUBWINDOW_ID:   out  SYS_WINDOW_ELE_ID);
--
-- CPM description: This procedure creates a subwindow to the window
--                  specified by the user. All input selected for the parent
--                  window will be effective for the subwindow also, unless
--                  other input is selected or another menu activated
--                  specifically for this window.
--
-- formal parameters
--IN     WINDOW_ID         The id of the parent window.
--
--IN     MAP_WINDOW        Logical indicating whether window should be mapped.
--
--IN     PIXEL_COL         Column number from within the window where the left
--                         side of the subwindow shall be placed. Column 0 is
--                         at the left of the window.
--
--IN     PIXEL_ROW         Row number from within the window where the top side
--                         of the subwindow shall be placed. Row 0 is at the
--                         top of the window.
--
--IN     PIXEL_WIDTH       The number of pixels to be occupied by the
--                         subwindow's width.
--
--IN     PIXEL_HEIGHT      The number of pixels to be occupied by the
--                         subwindow's height.
--
--IN     BORDER_WIDTH      The width of the border in pixels.  If the border
--                         width is zero, the subwindow will not have a border.
--
--OUT    SUBWINDOW_ID      The id of the subwindow as given by the X window
--                         system.
-- end formal parameters;


procedure UWN_CREATE_WINDOW (WINDOW_ID:       out  SYS_WINDOW_ELE_ID;
```

```
                                WINDOW_LABEL:     in   string;
                                MAP_WINDOW:       in   BOOLEAN;
                                ICON_TYPE:        in   SYS_ICON;
                                ICON_STACK_INDX:  out  SYS_ICON_STACK;
                                ICON_ID:          out  SYS_WINDOW_ELE_ID);

-- CPM description: Creates a basic window skeleton with border, title, icon
--                  and frame popup menu attached.  Only one window per
--                  process.
--
-- formal parameters
--OUT    WINDOW_ID         The id given the window.
--
--IN     WINDOW_LABEL      Textual string to be displayed in the window border.
--
--IN     MAP_WINDOW        Boolean indicating whether window should be mapped
--                         (Made visible upon creation).  If the application
--                         wishes the window to be in iconic form, it should
--                         then call UWN_CLOSE_WINDOW.  Otherwise, when the
--                         application wishes to map the window or make it
--                         visible, it should call UWN_MAP_WINDOW.
--
--IN     ICON_TYPE         Identifies the icon stack that the new window is
--                         assigned to. 0 = Reference Icon
--                                      1 = View C & C Icon
--                                      2 = Process Messages Icon
--                                      3 = Build C & C Icon
--                                      4 = Decision Aids Icon
--                                      5 = Experiment Control Icon
--
--OUT    ICON_STACK_INDX   Position in the Icon stack of the newly created
--                         window (1 - 7);
--
--OUT    ICON_ID           The id given the icon window.
--
-- end formal parameters;


procedure UWN_DEACTIVATE_MENU (MENU_STRUCT_ID:  in  SYS_WINDOW_ELE_ID;
                               MENU_INDEX:      in  SYS_WALKING_CELL);
--
-- CPM description: This routine deactivates an already defined popup menu.
--
-- formal parameters
--IN     MENU_STRUCT_ID    The id of the menu structure given by the
--                         application at the time of the menu definition.
--
--IN     MENU_INDEX        The index into the Start_Array of the submenu to
--                         be activated for a particular window.
--                         If the menu to be activated is not a walking menu,
--                         or is the top level of a walking menu, then this
--                         parameter should be set to NULL.
-- end formal parameters;


procedure UWN_DEFINE_BUTTON (BUTTON_ID:        out  SYS_WINDOW_ELE_ID;
```

```
                                WINDOW_ID:        in   SYS_WINDOW_ELE_ID;
                                ENABLE_FLAG:      in   BOOLEAN;
                                PIXEL_COL:        in   SYS_WINDOW_COLUMN;
                                PIXEL_ROW:        in   SYS_WINDOW_ROW;
                                PIXEL_WIDTH:      in   SYS_WINDOW_COLUMN;
                                PIXEL_HEIGHT:     in   SYS_WINDOW_ROW;
                                BUTTON_TEXT:      in   string);

 --
 -- CPM description: Defines a button on top portion of a window. Once a
 --                  button has been defined, only other buttons may be placed
 --                  beside it.  All other structures must be placed below
 --                  the buttons. These buttons are used mostly for initiating
 --                  a walking menu (see UWN_ACTIVATE_MENU).
 --
 -- formal parameters
 --OUT    BUTTON_ID         ID attached to the defined button. This
 --                         ID is required for all interactions with the button.
 --
 --IN     WINDOW_ID         The ID of the window to attach the button to.
 --
 --IN     ENABLE_FLAG       Logical flag to indicate if the button should be
 --                         backlight when it is selected and the button ID will
 --                         be returned to the application. The disabled mode is
 --                         used to display a walking menu when the button is
 --                         selected.
 --                                    true  = ENABLED
 --                                    false = DISABLED
 --
 --IN     PIXEL_COL         Column number from within the window where the left
 --                         side of the button shall be placed. Column 0 is at
 --                         left of the window.
 --
 --IN     PIXEL_ROW         Row number from within the window where the top side
 --                         of the button shall be placed. Row 0 is at the top
 --                         of the window.
 --
 --IN     PIXEL_WIDTH       The number of columns to be occupied by the button.
 --
 --IN     PIXEL_HEIGHT      The number of rows to be occupied by the button.
 --
 --IN     BUTTON_TEXT       Textual string to display in the button.
 -- end formal parameters;


procedure UWN_DEFINE_BUTTON_MENU (
                MENU_INFORMATION:   in   UWN_BUTTON_MENU_PTR;
                Menu_Window_Id:     out  SYS_WINDOW_ELE_ID;
                Map_Window:         in   BOOLEAN := FALSE;
                Parent_Window:      in   SYS_WINDOW_ELE_ID := SYS_ROOT_WINDOW;
                Parent_Window_X:    in   SYS_WINDOW_COLUMN := 0;
                Parent_Window_Y:    in   SYS_WINDOW_ROW := 0);
--
-- CPM description:   UWN_DEFINE_BUTTON_MENU defines a popup window with a
--                    button menu specified by the application.
--
```

```
-- formal parameters
--IN    MENU_INFORMATION    Record of the button menus to be created and
--                         input gathered from.
--
--OUT   MENU_WINDOW_ID      The ID of the window containing the button menu.
--
--IN    MAP_WINDOW          The logical indicating whether the button menu
--                         window should be mapped upon creation or not.
--                         If it is not, the application can make the
--                         button menu window be visible later via a call
--                         to UWN_MAP_WINDOW.
--
--IN    Parent_Window       The ID of the window to which the button menu
--                         manager window will be a subwindow to.  The
--                         default is the root window thus making the button
--                         menu a popup window.
--
--IN    Parent_Window_X     The pixel column of the parent window where the
--                         button menu window's origin will be placed.  The
--                         default is zero, where the window may be moved
--                         via UWN_MOVE_WINDOW.
--
--IN    Parent_Window_Y     The pixel row of the parent window where the
--                         button menu window's origin will be placed. The
--                         default is zero.
--
-- end formal parameters;


procedure UWN_DEFINE_CHECKBOX (
                EDITOR_ID:        out  SYS_WINDOW_ELE_ID;
                DEST_TYPE:        in   SYS_DESTINATION_TYPE;
                DEST_ID  :        in   SYS_WINDOW_ELE_ID;
                PIXEL_COL:        in   SYS_WINDOW_COLUMN;
                PIXEL_ROW:        in   SYS_WINDOW_ROW;
                NUM_FIELDS:       in   SYS_MENU_BUTTON_INDEX;
                NUM_COLS:         in   SYS_MENU_BUTTON_INDEX;
                LABELS:           in   SYS_MENU_BUTTON_LABEL_PTR;
                STATUS:           in   SYS_MENU_BUTTON_STATUS_PTR;
                SUBPANEL_ID:      in   SYS_WINDOW_ELE_ID := SYS_NULL_SUBPANEL;
                PIXEL_WIDTH:      in   SYS_WINDOW_COLUMN := SYS_NULL_COLUMN;
                PIXEL_HEIGHT:     in   SYS_WINDOW_ROW := SYS_NULL_ROW);
--
-- CPM description: Creates a checkbox button editor.
--
-- formal parameters
--OUT   EDITOR_ID          ID attached to the editor. This
--                         ID is required for all interactions with the editor.
--
--IN    DEST_TYPE         The type of the destination for the editor, where:
--                        SYS_WINDOW_DEST = Window
--                        SYS_PANEL_DEST = Panel
--
--IN    DEST_ID           ID attached to the destination that the editor is
--                        assigned to.  This is set to NULL when the
--                        destination is the RootWindow.
```

```
--
--IN     PIXEL_COL          Column number from within the window where the left
--                          side of the editor shall be placed. Column 0 is at
--                          left of the window.
--
--IN     PIXEL_ROW          Row number from within the window where the top side
--                          of the editor shall be placed. Row 0 is at the top
--                          of the window.
--
--IN     NUM_FIELDS         The total number of checkbox buttons to be in the
--                          editor.
--
--IN     NUM_COLS           The number of columns the checkbox buttons are to be
--                          arranged in.
--
--IN     LABELS             Pointer to the array of label addresses for all
--                          the checkbox buttons.
--
--IN     STATUS             Pointer to the boolean array of statuses for all the
--                          checkbox buttons.
--
--IN     SUBPANEL_ID        ID attached to the subpanel that
--                          the editor is assigned to. If the editor is not
--                          assigned to a subpanel, use a zero which is the
--                          default.
--
--IN     PIXEL_WIDTH        The number of pixel columns wide the checkbox editor
--                          is to be created.  If the width is to be calculated,
--                          use the default value of zero.
--
--IN     PIXEL_HEIGHT       The number of pixel rows high the checkbox editor is
--                          to be created.  If the height is to be calculated,
--                          use the default value of zero.
--
-- end formal parameters;


procedure UWN_DEFINE_EDITOR (EDITOR_ID:          out  SYS_WINDOW_ELE_ID;
                             DEST_TYPE:          in   SYS_DESTINATION_TYPE;
                             DEST_ID   :         in   SYS_WINDOW_ELE_ID;
                             PIXEL_COL:          in   SYS_WINDOW_COLUMN;
                             PIXEL_ROW:          in   SYS_WINDOW_ROW;
                             NUM_COLS:           in   SYS_WINDOW_COLUMN;
                             NUM_ROWS:           in   SYS_WINDOW_ROW;
                             READ_ONLY:          in   BOOLEAN;
                             MAX_BUFFER_SIZE:    in   SYS_PRODUCT_LENGTH;
                             TEXT_BUFFER:        in   SYS_TEXT_PTR;
                             BUFFER_SIZE:        in   SYS_PRODUCT_LENGTH;
                             SUBPANEL_ID:        in   SYS_WINDOW_ELE_ID := 0);

--
-- CPM description: Creates a window full page text editor.
--
-- formal parameters
--OUT   EDITOR_ID          ID attached to the editor. This
--                         ID is required for all interactions with the editor.
```

```
--
--IN     DEST_TYPE        The type of the destination for the editor, where:
--                       SYS_WINDOW_DEST = Window
--                       SYS_PANEL_DEST = Panel
--
--IN     DEST_ID          ID attached to the destination that the editor is
--                       assigned to.  This is set to NULL when the
--                       destination is the RootWindow.
--
--IN     PIXEL_COL         Column number from within the window where the left
--                       side of the editor shall be placed. Column 0 is at
--                       left of the window.
--
--IN     PIXEL_ROW         Row number from within the window where the top side
--                       of the editor shall be placed. Row 0 is at the top
--                       of the window.
--
--IN     NUM_COLS          The number of columns to be occupied by the editor.
--
--IN     NUM_ROWS          The number of rows to be occupied by the editor.
--
--IN     READ_ONLY         Flag indicating if the user has full editing
--                       capabilities or is limited to only scroll and copy
--                       operations.
--                                  true   =  Read only
--                                  false  =  Full edit
--
--IN     MAX_BUFFER_SIZE   Maximum number of pixels that the TEXT_BUFFER
--                       can hold.
--
--IN     TEXT_BUFFER       Buffer of the initial text to display in the editor.
--
--IN     BUFFER_SIZE       The number of pixels in TEXT_BUFFER.
--
--IN     SUBPANEL_ID       ID attached to the subpanel that
--                       the editor is assigned to. If the editor is not
--                       assigned to a subpanel, use a zero.
--
-- end formal parameters;


procedure UWN_DEFINE_NUMBER_FIELD (
                          EDITOR_ID:          out      SYS_WINDOW_ELE_ID;
                          DEST_TYPE:          in       SYS_DESTINATION_TYPE;
                          DEST_ID   :         in       SYS_WINDOW_ELE_ID;
                          PIXEL_COL:          in       SYS_WINDOW_COLUMN;
                          PIXEL_ROW:          in       SYS_WINDOW_ROW;
                          LABEL:              in       STRING;
                          LABEL_POSITION:     in       SYS_LABEL_POSITION;
                          NUMBER_VARIABLE:    in out   STRING;
                          MIN_NUMBER      :   in       STRING;
                          MAX_NUMBER      :   in       STRING;
                          MAX_CHARACTERS:     in       SYS_PRODUCT_LENGTH;
                          SUBPANEL_ID:        in       SYS_WINDOW_ELE_ID := 0);

--
```

```
-- CPM description: Creates a Numeric Field editor.
--                  Note: This function will not cause display of the field
--                        if it is defined in a panel as that is caused by
--                        calling either cwn_end_panel or cwn_end_subpanel.
--
-- formal parameters
--OUT    EDITOR_ID         ID attached to the editor. This
--                         ID is required for all interactions with the editor.
--
--IN     DEST_TYPE         The type of the destination for the editor, where:
--                         SYS_WINDOW_DEST = Window
--                         SYS_PANEL_DEST = Panel
--
--IN     DEST_ID           ID attached to the destination that the editor is
--                         assigned to.  This is set to NULL when the
--                         destination is the RootWindow.
--
--IN     PIXEL_COL          Column number from within the panel where the left
--                         side of the editor shall be placed. Column 0 is at
--                         left of the window.
--
--IN     PIXEL_ROW          Row number from within the panel where the top side
--                         of the editor shall be placed. Row 0 is at the top
--                         of the window.
--
--IN     LABEL              The optional label before the number field.  This
--                         should be set to NULL if no label will be displayed.
--
--IN     LABEL_POSITION     Value specifying whether the optional label should
--                         be placed to the left or the right of the number
--                         field.  The two valid settings for this field are:
--                             0 = Left aligned
--                             1 = Right aligned
--                         If no label is specified, this parameter will
--                         be ignored by the editor.
--
--INOUT NUMBER_VARIABLE    The address of the variable to store the
--                         input number at.  This variable may be
--                         initialized to some number value, which would
--                         be displayed.  This must be a NULL terminated
--                         string.
--
--IN     MIN_NUMBER        The string representing the minimum number
--                         to be allowed as input from the user.  This
--                         string must be MAX_CHARACTERS long with each
--                         digit of the string representing the minimum
--                         value for that digit and the string must be NULL
--                         terminated.
--
--IN     MAX_NUMBER        The string representing the maximum number to be
--                         allowed as input from the user.  This string must
--                         be MAX_CHARACTERS long with each digit of the string
--                         representing the maximum value for that digit and
--                         the string must be NULL terminated.
--
--IN     MAX_CHARACTERS    The maximum number of characters which will
```

```
--                          be allowed to be entered into the field.
--
--IN    SUBPANEL_ID          ID attached to the subpanel that
--                          the editor is assigned to. If the editor is not
--                          assigned to a subpanel, use a zero.
--
-- end formal parameters;


procedure UWN_DEFINE_PANEL (PANEL_ID:   out   SYS_WINDOW_ELE_ID);

--
-- CPM description: Defines a panel within a window. This procedure must be
--                  called before defining any field editors. A panel must
--                  have at least one field editor attached to it.
-- formal parameters
--OUT   PANEL_ID             ID attached to the panel.
--                          This ID is required for all interactions with the
--                          panel.
-- end formal parameters;


procedure UWN_DEFINE_POPUP_MENU (MENU_STRUCT_ID: in  SYS_WINDOW_ELE_ID;
                                 MENU_TITLE:     in   STRING;
                                 START_ARRAY:    in   SYS_POP_UP_START_PTR;
                                 LENGTH_ARRAY:   in   SYS_POP_UP_LENGTH_PTR;
                                 TEXT_ARRAY:     in   SYS_MENU_TEXT_PTR;
                                 CHILD_ARRAY:    in   SYS_POP_UP_CHILD_PTR);

--
-- CPM description: Defines a popup menu which may be a walking menu.
--                  This does not, however, display the menu in
--                  the window.  All arrays are zero origin in index.
--                  The index into Text_Array is used as the menu id.
--
-- formal parameters
--IN      MENU_STRUCT_ID The id given by the application to the popup menu
--                       or entire walking menu structure.
--
--IN      MENU_TITLE     The title of the menu to be displayed at the top
--                       of the menu.  If the menu is a walking menu, then
--                       only the top menu will contain a title.  If the
--                       user doesn't wish the title to be displayed, then
--                       this parameter must be set to NULL.
--
--IN      START_ARRAY    Index into TEXT_ARRAY for the start of each pop-up
--                       menu in the walking menu.
--
--IN      LENGTH_ARRAY   Number of cells in each pop-up menu
--
--IN      TEXT_ARRAY     Text for each cell of each pop-up menu in the
--                       walking menu
--
--IN      CHILD_ARRAY    Pop-up index of the pop-up menu that is the child
--                       of each pop-up menu cell index into START_ARRAY
--                       and LENGTH_ARRAY;
```

```
-- end formal parameters;

procedure UWN_DEFINE_POPUP_WINDOW (WINDOW_ID:       out  SYS_WINDOW_ELE_ID;
                                   MAP_WINDOW:      in   BOOLEAN;
                                   PIXEL_COL:       in   SYS_WINDOW_COLUMN;
                                   PIXEL_ROW:       in   SYS_WINDOW_ROW;
                                   PIXEL_WIDTH:     in   SYS_WINDOW_COLUMN;
                                   PIXEL_HEIGHT:    in   SYS_WINDOW_ROW);

--
-- CPM description: Changes the size of a popup window.
--
-- formal parameters
--OUT    WINDOW_ID         ID attached to the window.
--
--IN     MAP_WINDOW        Boolean logical indicating whether the defined
--                         window should be mapped or not.
--
--IN     PIXEL_COL         Column number from within the display where the left
--                         side of the window shall be placed. Column 0 is at
--                         left of the display.
--
--IN     PIXEL_ROW         Row number from within the display where the top
--                         side of the window shall be placed. Row 0 is at the
--                         top of the display.
--
--IN     PIXEL_WIDTH       The number of columns to be occupied by the window.
--
--IN     PIXEL_HEIGHT      The number of rows to be occupied by the window.
-- end formal parameters;


procedure UWN_DEFINE_PUSHBUTTON (EDITOR_ID:    out  SYS_WINDOW_ELE_ID;
                                 DEST_TYPE:    in   SYS_DESTINATION_TYPE;
                                 DEST_ID  :    in   SYS_WINDOW_ELE_ID;
                                 PIXEL_COL:    in   SYS_WINDOW_COLUMN;
                                 PIXEL_ROW:    in   SYS_WINDOW_ROW;
                                 NUM_FIELDS:   in   SYS_MENU_BUTTON_INDEX;
                                 NUM_COLS:     in   SYS_MENU_BUTTON_INDEX;
                                 LABELS:       in   SYS_MENU_BUTTON_LABEL_PTR;
                                 DEFAULT_BUTTON: in SYS_MENU_BUTTON_VALUES;
                                 SUBPANEL_ID:  in   SYS_WINDOW_ELE_ID := 0);
--
-- CPM description: Creates a pushbutton editor.
--
-- formal parameters
--OUT    EDITOR_ID         ID attached to the editor. This
--                         ID is required for all interactions with the editor.
--
--IN     DEST_TYPE         The type of the destination for the editor, where:
--                         SYS_WINDOW_DEST = Window
--                         SYS_PANEL_DEST = Panel
--
--IN     DEST_ID           ID attached to the destination that the editor is
--                         assigned to.  This is set to NULL when the
```

```
--                          destination is the RootWindow.
--
--IN     PIXEL_COL           Column number from within the window where the left
--                          side of the editor shall be placed. Column 0 is at
--                          left of the window.
--
--IN     PIXEL_ROW           Row number from within the window where the top side
--                          of the editor shall be placed. Row 0 is at the top
--                          of the window.
--
--IN     NUM_FIELDS          The total number of pushbuttons to be in the
--                          editor.
--
--IN     NUM_COLS            The number of columns the pushbuttons are to be
--                          arranged in.
--
--IN     LABELS              Address of the array of label addresses for all the
--                          pushbuttons.
--
--IN     DEFAULT_BUTTON      The index into the pushbutton array of the button to
--                          be drawn "active" or displayed as the default
--                          button.  A value of SYS_NO_DEFAULT_BUTTON will
--                          disable this feature.
--
--IN     SUBPANEL_ID         ID attached to the subpanel that
--                          the editor is assigned to. If the editor is not
--                          assigned to a subpanel, use a zero.
--
-- end formal parameters;


procedure UWN_DEFINE_RADIOBUTTON (EDITOR_ID:  out  SYS_WINDOW_ELE_ID;
                                  DEST_TYPE:       in   SYS_DESTINATION_TYPE;
                                  DEST_ID  :       in   SYS_WINDOW_ELE_ID;
                                  PIXEL_COL:       in   SYS_WINDOW_COLUMN;
                                  PIXEL_ROW:       in   SYS_WINDOW_ROW;
                                  NUM_FIELDS:      in   SYS_MENU_BUTTON_INDEX;
                                  NUM_COLS:        in   SYS_MENU_BUTTON_INDEX;
                                  LABELS:          in   SYS_MENU_BUTTON_LABEL_PTR;
                                  DEFAULT_BUTTON:  in   SYS_MENU_BUTTON_INDEX;
                                  SUBPANEL_ID:     in   SYS_WINDOW_ELE_ID := 0);
--
-- CPM description: Creates a radiobutton editor where only one button is
--                  active at a time.
--
-- formal parameters
--OUT    EDITOR_ID           ID attached to the editor. This
--                           ID is required for all interactions with the editor.
--
--IN     DEST_TYPE          The type of the destination for the editor, where:
--                          SYS_WINDOW_DEST = Window
--                          SYS_PANEL_DEST = Panel
--
--IN     DEST_ID            ID attached to the destination that the editor is
--                          assigned to.  This is set to NULL when the
--                          destination is the RootWindow.
```

```
--
--IN     PIXEL_COL          Column number from within the window where the left
--                         side of the editor shall be placed. Column 0 is at
--                         left of the window.
--
--IN     PIXEL_ROW          Row number from within the window where the top side
--                         of the editor shall be placed. Row 0 is at the top
--                         of the window.
--
--IN     NUM_FIELDS         The total number of radiobuttons to be in the
--                         editor.
--
--IN     NUM_COLS           The number of columns the radiobuttons are to be
--                         arranged in.
--
--IN     LABELS             Address of the array of label addresses for all the
--                         radiobuttons.
--
--IN     DEFAULT_BUTTON     The index into the radiobutton array of the button
--                         to be drawn "active" or displayed as the default
--                         button.
--
--IN     SUBPANEL_ID        ID attached to the subpanel that
--                         the editor is assigned to. If the editor is not
--                         assigned to a subpanel, use a zero.
--
-- end formal parameters;


procedure UWN_DEFINE_SCROLLBAR (SCROLLBAR_ID:    out  SYS_WINDOW_ELE_ID;
                                DEST_TYPE:       in   SYS_DESTINATION_TYPE;
                                DEST_ID  :       in   SYS_WINDOW_ELE_ID;
                                ORIENTATION:     in   SYS_SB_DIRECTION;
                                PIXEL_COL:       in   SYS_WINDOW_COLUMN;
                                PIXEL_ROW:       in   SYS_WINDOW_ROW;
                                PIXEL_WIDTH:     in   SYS_WINDOW_PIXEL;
                                PIXEL_LENGTH:    in   SYS_WINDOW_PIXEL;
                                DOC_SIZE:        in   SYS_PIXEL;
                                DISP_POSITION:   in   SYS_PIXEL;
                                SCROLL_INTRVL:   in   SYS_WINDOW_PIXEL;
                                SUBPANEL_ID:     in   SYS_WINDOW_ELE_ID := 0);

--
-- CPM description: Creates a horizontal or vertical scroll bar in a window.
--                  A scrollbar is always created to fill one character
--                  whether it be vertically or horizontally oriented.
--                  A vertical scrollbar will be one character wide, whereas
--                  a horizontal scrollbar will be one character high.  The
--                  length and document size, therefore, is the number of
--                  pixel rows or columns depending on the orientation.
--
-- formal parameters
--OUT    SCROLLBAR_ID       ID attached to the scrollbar.
--                         This ID is required for all interactions with the
--                         scrollbar.
--
```

```
--IN    DEST_TYPE       The type of the destination for the editor, where:
--                      SYS_WINDOW_DEST = Window
--                      SYS_PANEL_DEST = Panel
--
--IN    DEST_ID         ID attached to the destination that the editor is
--                      assigned to.  This is set to SYS_ROOT_WINDOW when
--                      the destination is the RootWindow.
--
--IN    ORIENTATION     Direction of the scrollbar where it is set to one
--                      of the following:
--                      SYS_SB_DIR_HORZ (Horizontal) or
--                      SYS_SB_DIR_VERT (Vertical)
--
--IN    PIXEL_COL        Column number from within the window where the left
--                      side of the scrollbar shall be placed. Column 0 is
--                      at the left of the window.
--
--IN    PIXEL_ROW       Row number from within the window where the top side
--                      of the scrollbar shall be placed. Row 0 is at the
--                      top of the window.
--
--IN    PIXEL_WIDTH     The number of pixels to be occupied by the
--                      scrollbar's width.
--
--IN    PIXEL_LENGTH    The number of pixels to be occupied by the
--                      scrollbar's length.
--
--IN    DOC_SIZE        The number of lines in the document buffer.
--
--IN    DISP_POSITION   The offset from the beginning of the work surface to
--                      first pixel visible to the user.
--
--IN    SCROLL_INTRVL   The number of pixels the work will be scrolled
--                      whenever the user selects an arrow button.  Note:
--                      The work will not be scrolled by these utilities
--                      but, this argument is required to calculate
--                      the interactive slidepositioning.
--
--IN    SUBPANEL_ID     ID attached to the subpanel that
--                      the editor is assigned to. If the editor is not
--                      assigned to a subpanel, use a zero.
--
-- end formal parameters;


procedure UWN_DEFINE_STATIC_TEXT (STATIC_TEXT_ID:   out  SYS_WINDOW_ELE_ID;
                                  DEST_TYPE:        in   SYS_DESTINATION_TYPE;
                                  DEST_ID  :        in   SYS_WINDOW_ELE_ID;
                                  PIXEL_COL:        in   SYS_WINDOW_COLUMN;
                                  PIXEL_ROW:        in   SYS_WINDOW_ROW;
                                  PIXEL_WIDTH:      in   SYS_WINDOW_COLUMN;
                                  PIXEL_HEIGHT:     in   SYS_WINDOW_ROW;
                                  STATIC_TEXT:      in   SYS_TEXT_PTR;
                                  TEXT_ALIGNMENT:   in   SYS_TEXT_ALIGNMENT;
                                  SUBPANEL_ID:      in   SYS_WINDOW_ELE_ID := 0);
```

```
--
-- CPM description: Creates a static text area in a window. The static text
--                  procedure allows display of product headings that will
--                  not scroll with the product.
--
-- formal parameters
--OUT    STATIC_TEXT_ID    ID attached to the static text
--                         area. This ID is required for all interactions with
--                         the static text area.
--
--IN     DEST_TYPE         The type of the destination for the editor, where:
--                         SYS_WINDOW_DEST = Window
--                         SYS_PANEL_DEST = Panel
--
--IN     DEST_ID           ID attached to the destination that the editor is
--                         assigned to.  This is set to NULL when the
--                         destination is the RootWindow.
--
--IN     PIXEL_COL          Column number from within the window where the left
--                         side of the static text area shall be placed.
--                         Column 0 is at the left of the window.
--
--IN     PIXEL_ROW          Row number from within the window where the top side
--                         of the static text area shall be placed. Row 0 is
--                         at the top of the window.
--
--IN     PIXEL_WIDTH        The number of columns to be occupied by the static
--                         text area.
--
--IN     PIXEL_HEIGHT       The number of rows to be occupied by the static
--                         text area.
--
--IN     STATIC_TEXT        Textual string to display in the button.
--
--IN     TEXT_ALIGNMENT     Alignment of the text within the static text area
--                         (CENTER_ALIGNED, LEFT_ALIGNED, RIGHT_ALIGNED,
--                          NO_ALIGNMENT)
--
--IN     SUBPANEL_ID        ID attached to the subpanel that
--                         the editor is assigned to. If the editor is not
--                         assigned to a subpanel, use a zero.
--
-- end formal parameters;


procedure UWN_DEFINE_STRING_FIELD (
                              EDITOR_ID:          out     SYS_WINDOW_ELE_ID;
                              DEST_TYPE:          in      SYS_DESTINATION_TYPE;
                              DEST_ID  :          in      SYS_WINDOW_ELE_ID;
                              PIXEL_COL:          in      SYS_WINDOW_COLUMN;
                              PIXEL_ROW:          in      SYS_WINDOW_ROW;
                              LABEL:              in      STRING;
                              LABEL_POSITION:     in      SYS_LABEL_POSITION;
                              STRING_VARIABLE:    in out  STRING;
                              MAX_CHARACTERS:     in      SYS_PRODUCT_LENGTH;
                              SUBPANEL_ID:    in   SYS_WINDOW_ELE_ID := 0);
```

```
--
-- CPM description: Creates a String Field editor.
--                  Note: this function will not cause display of the field
--                        as that is caused by calling either cwn_end_panel
--                        or cwn_end_subpanel.
--
-- formal parameters
--OUT   EDITOR_ID        ID attached to the editor. This
--                       ID is required for all interactions with the editor.
--
--IN    DEST_TYPE       The type of the destination for the editor, where:
--                      SYS_WINDOW_DEST = Window
--                      SYS_PANEL_DEST = Panel
--
--IN    DEST_ID         ID attached to the destination that the editor is
--                      assigned to.  This is set to NULL when the
--                      destination is the RootWindow.
--
--IN    PIXEL_COL        Column number from within the panel where the left
--                       side of the editor shall be placed. Column 0 is at
--                       left of the window.
--
--IN    PIXEL_ROW        Row number from within the panel where the top side
--                       of the editor shall be placed. Row 0 is at the top
--                       of the window.
--
--IN    LABEL            The optional label before the string field.  This
--                       should be set to NULL if no label will be displayed.
--
--IN    LABEL_POSITION   Value specifying whether the optional label should
--                       be placed to the left or the right of the number
--                       field.  The two valid settings for this field are:
--                           0 = Left aligned
--                           1 = Right aligned
--                       If no label is specified, this parameter will
--                       be ignored by the editor.
--
--INOUT STRING_VARIABLE  The address of the variable to store the
--                       input string at.  This variable may be
--                       initialized to some string value, which would
--                       be displayed.  This must be a NULL terminated
--                       string.
--
--IN    MAX_CHARACTERS   The maximum number of characters which will
--                       be allowed to be entered into the field.
--
--IN    SUBPANEL_ID      ID attached to the subpanel that
--                       the editor is assigned to. If the editor is not
--                       assigned to a subpanel, use a zero.
--
-- end formal parameters;

procedure UWN_DEFINE_SUBPANEL (SUBPANEL_ID:   out  SYS_WINDOW_ELE_ID;
                               PANEL_ID:      in   SYS_WINDOW_ELE_ID);
```

```
--
-- CPM description: Defines a subpanel within a panel. A subpanel must
--                  have at least one field editor attached to it.
-- formal parameters
--OUT   SUBPANEL_ID      ID attached to the subpanel.
--                       This ID is required for all interactions with the
--                       subpanel.
--IN    PANEL_ID         ID of the panel that the
--                       subpanel is attached to.
-- end formal parameters;


procedure UWN_DELETE_BUTTON (BUTTON_ID   :     in  SYS_WINDOW_ELE_ID);
--
-- CPM description: UWN_DELETE_BUTTON deletes a button that is defined by
--                  UWN_DEFINE_BUTTON.
--
-- formal parameters
--IN    BUTTON_ID        The ID of the button to delete.
-- end formal parameters;


procedure UWN_DELETE_BUTTON_MENU (MENU_WINDOW_ID:   in   SYS_WINDOW_ELE_ID);
--
-- CPM description:  UWN_DELETE_BUTTON_MENU deletes the specified button menu.
--
-- formal parameters
--IN   MENU_WINDOW_ID   The id of the window containing the button menu.
--
-- end formal parameters;


procedure UWN_DELETE_CHECKBOX (CHECKBOX_ID   :   in  SYS_WINDOW_ELE_ID);
--
-- CPM description: UWN_DELETE_CHECKBOX deletes a checkbox editor that is
--                  defined by UWN_DEFINE_CHECKBOX.
--
-- formal parameters
--IN    CHECKBOX_ID      The ID of the checkbox editor to delete.
-- end formal parameters;


procedure UWN_DELETE_EDITOR (EDITOR_ID   :     in   SYS_WINDOW_ELE_ID);
--
-- CPM description: UWN_DELETE_EDITOR deletes an editor that is defined by
--                  UWN_DEFINE_EDITOR.
--
-- formal parameters
--IN    EDITOR_ID        The ID of the editor to delete.
--
-- end formal parameters;


procedure UWN_DELETE_MENU (MENU_ID   :     in  SYS_WINDOW_ELE_ID);
--
-- CPM description: Deletes a walking menu structure.
```

```
--
-- formal parameters
--IN    MENU_ID        The ID of the menu structure to delete.
-- end formal parameters;


procedure UWN_DELETE_NUMBER_FIELD (
                              EDITOR_ID   :     in   SYS_WINDOW_ELE_ID);
--
-- CPM description: Deletes an numeric field editor that
--                  is defined by UWN_DEFINE_NUMBER_FIELD.
--
-- formal parameters
--IN    EDITOR_ID        The ID of the editor to delete.
--
-- end formal parameters;


procedure UWN_DELETE_PANEL (PANEL_ID   :     in  SYS_WINDOW_ELE_ID);
--
-- CPM description: Deletes a panel from a window.
--
-- formal parameters
--IN    PANEL_ID        The ID of the panel to delete.
-- end formal parameters;


procedure UWN_DELETE_POPUP_WINDOW (WINDOW_ID   :    in   SYS_WINDOW_ELE_ID);
--
-- CPM description:   UWN_DELETE_POPUP_WINDOW deletes a popup window that is
--                    defined by UWN_DEFINE_POPUP_WINDOW.
--
-- formal parameters
--IN    WINDOW_ID       The ID of the popup window.
-- end formal parameters;


procedure UWN_DELETE_PUSHBUTTON (PUSHBUTTON_ID   :   in   SYS_WINDOW_ELE_ID);
--
-- CPM description:  UWN_DELETE_PUSHBUTTON deletes a pushbutton editor that
--                   is defined by UWN_DEFINE_PUSHBUTTON.
--
-- formal parameters
--IN    PUSHBUTTON_ID    The ID of the pushbutton editor.
-- end formal parameters;


procedure UWN_DELETE_RADIOBUTTON (RADIOBUTTON_ID  :  in   SYS_WINDOW_ELE_ID);
--
-- CPM description:  UWN_DELETE_RADIOBUTTON deletes a radiobutton editor that
--                   is defined by UWN_DEFINE_RADIOBUTTON.
--
-- formal parameters
--IN    RADIOBUTTON_ID   The ID of the radiobutton editor.
-- end formal parameters;
```

```
procedure UWN_DELETE_SCROLLBAR (SCROLLBAR_ID:      in   SYS_WINDOW_ELE_ID);
--
-- CPM description: UWN_DELETE_SCROLLBAR deletes a scrollbar that is defined
--                   by UWN_DEFINE_SCROLLBAR.
--
-- formal parameters
--IN    SCROLLBAR_ID      The ID of the scrollbar to delete.
--
-- end formal parameters;


procedure UWN_DELETE_STATIC_TEXT (STATIC_ID   :      in   SYS_WINDOW_ELE_ID);
--
-- CPM description: UWN_DELETE_STATIC_TEXT deletes static text that is
--                  defined by UWN_DEFINE_STATIC_TEXT.
--
-- formal parameters
--IN    STATIC_ID         The ID of the static text to delete.
--
-- end formal parameters;


procedure UWN_DELETE_STRING_FIELD (
                              EDITOR_ID   :      in   SYS_WINDOW_ELE_ID);
--
-- CPM description: Deletes an string field editor that
--                  is defined by UWN_DEFINE_STRING_FIELD.
--
-- formal parameters
--IN    EDITOR_ID         The ID of the editor to delete.
--
-- end formal parameters;


procedure UWN_DELETE_SUBPANEL ( SUBPANEL_ID:       in   SYS_WINDOW_ELE_ID);
--
-- CPM description: Deletes a subpanel from a window.
--
-- formal parameters
--IN    SUBPANEL_ID      The ID of the subpanel to delete.
-- end formal parameters;


procedure UWN_DELETE_SUBWINDOW (SUBWINDOW_ID:      in   SYS_WINDOW_ELE_ID);
--
-- CPM description: Deletes a subwindow from the working window.
--
-- formal parameters
--IN    SUBWINDOW_ID      The ID of the subwindow to delete.
-- end formal parameters;


procedure UWN_DISPLAY_SYSTEM_MESSAGE (MESSAGE  :   in   SYS_TEXT_PTR);
--
```

```
-- CPM description: This displays a message in the upper left hand corner of
--                  the display screen.  Unlike cwn_message_box, this routine
--                  is provided mainly for system messages relating the
--                  status or some other information of the system.  The
--                  message is removed via cwn_remove_system_message.
--
-- formal parameters
--IN    MESSAGE     The Message to display.
-- end formal parameters;


procedure UWN_END_PANEL (WINDOW_ID:       in   SYS_WINDOW_ELE_ID;
                         PANEL_ID:        in   SYS_WINDOW_ELE_ID;
                         PIXEL_COL:       in   SYS_WINDOW_COLUMN;
                         PIXEL_ROW:       in   SYS_WINDOW_ROW;
                         PIXEL_WIDTH:     in   SYS_WINDOW_COLUMN;
                         PIXEL_HEIGHT:    in   SYS_WINDOW_ROW);
--
-- CPM description: This procedure completes the panel definition process.
--                  It displays the subpanels and field editors (text
--                  editors, scroll bars, and static text) that are attached
--                  to the panel.
--
-- formal parameters
--IN    WINDOW_ID        ID attached to the window to contain the panel.
--
--IN    PANEL_ID         ID attached to the panel.
--
--IN    PIXEL_COL        Column number from within the window where the left
--                       side of the panel shall be placed. Column 0 is
--                       at the left of the window.
--
--IN    PIXEL_ROW        Row number from within the window where the top side
--                       of the panel shall be placed. Row 0 is at the
--                       top of the window.
--
--IN    PIXEL_WIDTH      The width of the panel in pixels.
--
--IN    PIXEL_HEIGHT     The height of the panel in pixels.
-- end formal parameters;


procedure UWN_END_SUBPANEL (SUBPANEL_ID:    in   SYS_WINDOW_ELE_ID;
                            PIXEL_COL:      in   SYS_WINDOW_COLUMN;
                            PIXEL_ROW:      in   SYS_WINDOW_ROW;
                            PIXEL_WIDTH:    in   SYS_WINDOW_COLUMN;
                            PIXEL_HEIGHT:   in   SYS_WINDOW_ROW);

--
-- CPM description: This procedure completes the subpanel definition process
--                  It displays the field editors (text editors, scroll
--                  bars, and static text) that are attached to the subpanel
--
-- formal parameters
--IN    SUBPANEL_ID      ID attached to the subpanel.
--
```

```
--IN    PIXEL_COL        Column number from within the window where the left
--                       side of the subpanel shall be placed. Column 0 is
--                       at the left of the window.
--
--IN    PIXEL_ROW        Row number from within the window where the top side
--                       of the subpanel shall be placed. Row 0 is at the
--                       top of the window.
--
--IN    PIXEL_WIDTH      The width of the subpanel in pixels.
--
--IN    PIXEL_HEIGHT     The height of the subpanel in pixels.
-- end formal parameters;


procedure UWN_HANDLE_WINDOW_MOVE (WINDOW_ID:      in   SYS_WINDOW_ELE_ID);

--
-- CPM description:  This procedure handles the user interface required
--                   for allowing the user to interactively move a window.
--
-- formal parameters
--IN    WINDOW_ID     The ID attached to the window.
-- end formal parameters;


procedure UWN_HIDE_PANEL (PANEL_ID:       in   SYS_WINDOW_ELE_ID);

--
-- CPM description: This procedure hides a defined panel and disables user
--                  input to any of the panel editors.
--
-- formal parameters
--IN    PANEL_ID         ID attached to the panel to
--                       hide.
-- end formal parameters;


procedure UWN_HIDE_SUBPANEL (SUBPANEL_ID:    in   SYS_WINDOW_ELE_ID);

--
-- CPM description: This procedure hides a defined subpanel and disables user
--                  input to any of the subpanel editors.
--
-- formal parameters
--IN    SUBPANEL_ID      ID attached to the subpanel to
--                       hide.
--
-- end formal parameters;


procedure UWN_INITIALIZE_WINDOW_SYSTEM;
--
-- CPM description: UWN_INITIALIZE_WINDOW_SYSTEM is the initial set-up
--                  procedure for the EDDIC window system. It must be called
--                  before any of the UWN utilities.
--
```

```
-- formal parameters
--    None
-- end formal parameters;


procedure UWN_INPUT (INPUT_TYPE    :   out   SYS_WINDOW_INPUT;
                     WINDOW_ID     :   out   SYS_WINDOW_ELE_ID;
                     INPUT_VALUE   :   out   SYS_WINDOW_VALUE;
                     INPUT_DATA    :   out   SYS_WINDOW_DATA);
--
-- CPM description: Returns user input and internet messages to the
--                  application software.
--
-- formal parameters
--OUT    INPUT_TYPE       Type of input returned from the window system
--
--OUT    WINDOW_ID        The id of the window which received input, if
--                        applicable.  Note, that if the table below has
--                        an "X" under the window_id header for the
--                        input_type, but the window_id equals zero, then
--                        this means that the input took place in the
--                        RootWindow.
--
--OUT    INPUT_VALUE      The value of the input that accompanies the type
--
--OUT    INPUT_DATA       The value of the data that accompanies the type
--                        and input values, if appropriate.
--
--  The following table lists the output returned to the application
--  for its own processing:
--
--                      window-
-- input_type             id       type_code      data
-- ----------           -------    ---------      --------
-- 1 Exit               n/a        n/a            n/a
-- 2 Menu               X          Menu_Id        menu_index
-- 3 Checkbox           X          Editor_Id      Checkbox_index
-- 4 Scrollbar          X          Editor_Id      SlidePosition
-- 5 XrFILE             n/a        fd             n/a
-- 6 ButtonWindow       X          n/a            n/a
-- 7 Mouse Button       X          Button:        window_type:
--   Pressed                          0 = R       1 = window
--                                    1 = M       2 = panel
--                                    2 = L       3 = button
--                                                x, y
-- 8 Mouse Button       X          Button:        window_type:
--   Released                         0 = R       1 = window
--                                    1 = M       2 = panel
--                                    2 = L       3 = button
--                                                x, y
-- 9 Field Traversal    X          Editor_id      editor_type:
--                                                1 = String_field
--                                                2 = Number_field
--                                                type of traversal:
--                                                1 - Next
--                                                2 - Previous
```

```
--                                                    3 - Up
--                                                    4 - Down
--10 Exposure          X          n/a                  x, y, width, height
--11 Open Window      n/a         n/a                  n/a
--12 Window Resized   n/a         n/a                  n/a
--13 Close Window     n/a         n/a                  n/a
--14 XrEDIT_SAVE      X           Editor_Id            bufferCount
--15 XrEDIT_RESET     X           Editor_Id            n/a
--16 Pushbutton       X           Editor_Id            Button_index
--17 Radiobutton      X           Editor_Id            Active_index,
--                                                    Previous_Index
--
-- end formal parameters;


procedure UWN_MAP_WINDOW (WINDOW_ID:   in   SYS_WINDOW_ELE_ID);
--
-- CPM description: Routine to map a window created via UWN_CREATE_WINDOW
--                  whose "map_window" flag was set FALSE.
--
-- formal parameters
--IN    WINDOW_ID   The ID of the window to be mapped.
--
-- end formal parameters;


procedure UWN_MESSAGE_BOX (MESSAGE          :   in    STRING;
                           BUTTONS_ALLOWED  :   in    UWN_BUTTON_ALLOWED;
                           BUTTON_SELECTED  :   out   SYS_WINDOW_ELE_ID;
                           BUTTON_X_PIXEL   :   out   SYS_WINDOW_COLUMN;
                           BUTTON_Y_PIXEL   :   out   SYS_WINDOW_ROW;
                           INPUT_WINDOW_ID  :   out   SYS_WINDOW_ELE_ID);

--
-- CPM description: Displays a message box which the user removes by a click
--                  on the mouse which is allowed by the application. The
--                  message box always appears centered on the display and
--                  the button which activated its disappearance is returned
--                  to the application.
--
-- formal parameters
--IN    MESSAGE          Textual string to display in the message box.
--
--IN    BUTTON_ALLOWED   A logical array indicating which mouse buttons
--                       the application is allowing the user to click
--                       for making the message box go away, where:
--                           [0] = RightButton;
--                           [1] = MiddleButton;
--                           [2] = LeftButton;
--OUT   BUTTON_SELECTED  The number of the selected button (0, 1, or 2);
--OUT   BUTTON_X_PIXEL   The x pixel location where the mouse button was
--                       selected.
--OUT   BUTTON_Y_PIXEL   The y pixel location where the mouse button was
--                       selected.
--OUT   INPUT_WINDOW_ID  The id of the window which received the mouse
--                       button selection input.
```

```
-- end formal parameters;

procedure UWN_MOVE_BUTTON (BUTTON_ID:          in   SYS_WINDOW_ELE_ID;
                           PIXEL_COL:          in   SYS_WINDOW_COLUMN;
                           PIXEL_ROW:          in   SYS_WINDOW_ROW);
--
-- CPM description: Changes the location of a button.
--
-- formal parameters
--IN    BUTTON_ID        ID attached to the button. This
--                       ID is required for all interactions with the button.
--
--IN    PIXEL_COL        Column number from within the window where the left
--                       side of the button shall be placed. Column 0 is at
--                       left of the window.
--
--IN    PIXEL_ROW        Row number from within the window where the top side
--                       of the button shall be placed. Row 0 is at the top
--                       of the window.
-- end formal parameters;

procedure UWN_MOVE_CHECKBOX (CHECKBOX_ID:      in   SYS_WINDOW_ELE_ID;
                             PIXEL_COL:        in   SYS_WINDOW_COLUMN;
                             PIXEL_ROW:        in   SYS_WINDOW_ROW);

--
-- CPM description: Changes the location of a checkbox editor.
--
-- formal parameters
--IN    CHECKBOX_ID      ID attached to the checkbox editor.
--
--IN    PIXEL_COL        Column number from within the window where the left
--                       side of the editor shall be placed. Column 0 is at
--                       left of the window.
--
--IN    PIXEL_ROW        Row number from within the window where the top side
--                       of the editor shall be placed. Row 0 is at the top
--                       of the window.
-- end formal parameters;

procedure UWN_MOVE_EDITOR (EDITOR_ID:          in   SYS_WINDOW_ELE_ID;
                           PIXEL_COL:          in   SYS_WINDOW_COLUMN;
                           PIXEL_ROW:          in   SYS_WINDOW_ROW);
--
-- CPM description: Changes the location of a full page text editor.
--
-- formal parameters
--IN    EDITOR_ID        ID attached to the editor. This
--                       ID is required for all interactions with the editor.
--
--IN    PIXEL_COL        Column number from within the window where the left
--                       side of the editor shall be placed. Column 0 is at
--                       left of the window.
```

```
--
--IN    PIXEL_ROW          Row number from within the window where the top side
--                        of the editor shall be placed. Row 0 is at the top
--                        of the window.
-- end formal parameters;


procedure UWN_MOVE_NUMBER_FIELD (
                            EDITOR_ID:          in   SYS_WINDOW_ELE_ID;
                            PIXEL_COL:          in   SYS_WINDOW_COLUMN;
                            PIXEL_ROW:          in   SYS_WINDOW_ROW);
--
-- CPM description: Changes the location of a numeric field editor.
--
-- formal parameters
--IN    EDITOR_ID          ID attached to the editor. This
--                        ID is required for all interactions with the editor.
--
--IN    PIXEL_COL          Column number from within the window where the left
--                        side of the editor shall be placed. Column 0 is at
--                        left of the window.
--
--IN    PIXEL_ROW          Row number from within the window where the top side
--                        of the editor shall be placed. Row 0 is at the top
--                        of the window.
-- end formal parameters;


procedure UWN_MOVE_PANEL (PANEL_ID:           in   SYS_WINDOW_ELE_ID;
                          PIXEL_COL:          in   SYS_WINDOW_COLUMN;
                          PIXEL_ROW:          in   SYS_WINDOW_ROW);
--
-- CPM description: Changes the location of a panel.
--
-- formal parameters
--IN    PANEL_ID           ID attached to the panel to move.
--
--IN    PIXEL_COL          Column number from within the window where the left
--                        side of the panel shall be placed. Column 0 is at
--                        left of the window.
--
--IN    PIXEL_ROW          Row number from within the window where the top side
--                        of the panel shall be placed. Row 0 is at the top
--                        of the window.
-- end formal parameters;


procedure UWN_MOVE_POPUP_WINDOW (WINDOW_ID:      in   SYS_WINDOW_ELE_ID;
                                 PIXEL_COL:      in   SYS_WINDOW_COLUMN;
                                 PIXEL_ROW:      in   SYS_WINDOW_ROW);
--
-- CPM description: Changes the location of a popup window.
--
-- formal parameters
--IN    WINDOW_ID          ID attached to the popup window to move.
--
```

```
--IN     PIXEL_COL          Column number from within the display where the left
--                          side of the window shall be placed. Column 0 is at
--                          left of the display.
--
--IN     PIXEL_ROW          Row number from within the display where the top
--                          side of the window shall be placed. Row 0 is at the
--                          top of the display.
-- end formal parameters;


procedure UWN_MOVE_PUSHBUTTON (PUSHBUTTON_ID:      in   SYS_WINDOW_ELE_ID;
                               PIXEL_COL:          in   SYS_WINDOW_COLUMN;
                               PIXEL_ROW:          in   SYS_WINDOW_ROW);
--
-- CPM description: Changes the location of a pushbutton editor.
--
-- formal parameters
--IN     PUSHBUTTON_ID      ID attached to the pushbutton editor to move.
--
--IN     PIXEL_COL          Column number from within the window where the left
--                          side of the editor shall be placed. Column 0 is at
--                          left of the window.
--
--IN     PIXEL_ROW          Row number from within the window where the top side
--                          of the editor shall be placed. Row 0 is at the top
--                          of the window.
-- end formal parameters;


procedure UWN_MOVE_RADIOBUTTON (RADIOBUTTON_ID:    in   SYS_WINDOW_ELE_ID;
                                PIXEL_COL:         in   SYS_WINDOW_COLUMN;
                                PIXEL_ROW:         in   SYS_WINDOW_ROW);
--
-- CPM description: Changes the location of a radiobutton editor.
--
-- formal parameters
--IN     RADIOBUTTON_ID     ID attached to the radiobutton editor to move.
--
--IN     PIXEL_COL          Column number from within the window where the left
--                          side of the editor shall be placed. Column 0 is at
--                          left of the window.
--
--IN     PIXEL_ROW          Row number from within the window where the top side
--                          of the editor shall be placed. Row 0 is at the top
--                          of the window.
-- end formal parameters;


procedure UWN_MOVE_SCROLLBAR (
                              SCROLLBAR_ID:      in   SYS_WINDOW_ELE_ID;
                              PIXEL_COL:         in   SYS_WINDOW_COLUMN;
                              PIXEL_ROW:         in   SYS_WINDOW_ROW);
--
-- CPM description: Changes the location of a scrollbar.
--
-- formal parameters
```

```
--IN    SCROLLBAR_ID      ID attached to the scrollbar.
--                        This ID is required for all interactions with the
--                        scrollbar.
--
--IN    PIXEL_COL         Column number from within the window where the left
--                        side of the scrollbar shall be placed. Column 0 is
--                        at left of the window.
--
--IN    PIXEL_ROW         Row number from within the window where the top side
--                        of the scrollbar shall be placed. Row 0 is at the
--                        top of the panel.
-- end formal parameters;


procedure UWN_MOVE_STATIC_TEXT (
                            TEXT_ID:          in   SYS_WINDOW_ELE_ID;
                            PIXEL_COL:        in   SYS_WINDOW_COLUMN;
                            PIXEL_ROW:        in   SYS_WINDOW_ROW);
--
-- CPM description: Changes the location of static text.
--
-- formal parameters
--IN    EDITOR_ID         ID attached to the text. This
--                        ID is required for all interactions with the text.
--
--IN    PIXEL_COL         Column number from within the window where the left
--                        side of the text shall be placed. Column 0 is at
--                        left of the window.
--
--IN    PIXEL_ROW         Row number from within the window where the top side
--                        of the text shall be placed. Row 0 is at the top
--                        of the window.
-- end formal parameters;


procedure UWN_MOVE_STRING_FIELD (
                            EDITOR_ID:        in   SYS_WINDOW_ELE_ID;
                            PIXEL_COL:        in   SYS_WINDOW_COLUMN;
                            PIXEL_ROW:        in   SYS_WINDOW_ROW);
--
-- CPM description: Changes the location of a string field editor.
--
-- formal parameters
--IN    EDITOR_ID         ID attached to the editor. This
--                        ID is required for all interactions with the editor.
--
--IN    PIXEL_COL         Column number from within the window where the left
--                        side of the editor shall be placed. Column 0 is at
--                        left of the window.
--
--IN    PIXEL_ROW         Row number from within the window where the top side
--                        of the editor shall be placed. Row 0 is at the top
--                        of the window.
-- end formal parameters;
```

```
procedure UWN_MOVE_SUBWINDOW (SUBWINDOW_ID:      in   SYS_WINDOW_ELE_ID;
                              PIXEL_COL:         in   SYS_WINDOW_COLUMN;
                              PIXEL_ROW:         in   SYS_WINDOW_ROW);
--
-- CPM description: Changes the location of a subwindow.
--
-- formal parameters
--IN     SUBWINDOW_ID      ID attached to the subwindow to move.
--
--IN     PIXEL_COL         Column number from within the window where the left
--                         side of the subwindow shall be placed. Column 0 is
--                         at left of the window.
--
--IN     PIXEL_ROW         Row number from within the window where the top side
--                         of the subwindow shall be placed. Row 0 is at the
--                         top of the window.
-- end formal parameters;


procedure UWN_MOVE_WINDOW (WINDOW_ID:       in   SYS_WINDOW_ELE_ID;
                           PIXEL_COL:       in   SYS_WINDOW_COLUMN;
                           PIXEL_ROW:       in   SYS_WINDOW_ROW);
--
-- CPM description: Changes the location of a window.
--
-- formal parameters
--IN     WINDOW_ID         ID attached to the window to move.
--
--IN     PIXEL_COL         Column number where the left side of the window
--                         shall be placed.
--
--IN     PIXEL_ROW         Row number where the top side of the window
--                         shall be placed.
-- end formal parameters;


procedure UWN_OPEN_ICON;
--
-- CPM description: Opens the window from the existing icon.
--
-- formal parameters
--NONE
-- end formal parameters;


procedure UWN_POST_MENU     (MENU_STRUCT_ID:  in  SYS_WINDOW_ELE_ID;
                             MENU_INDEX:      in  SYS_WALKING_CELL;
                             WINDOW_TYPE:     in  SYS_WINDOW_TYPE;
                             WINDOW_ID:       in  SYS_WINDOW_ELE_ID;
                             PIXEL_X:         in  SYS_WINDOW_COLUMN;
                             PIXEL_Y:         in  SYS_WINDOW_ROW);
--
-- CPM description: This routine activates and posts an already defined
--                  popup menu at a specified location for either:
--                     a. A defined window,
--                     b. a displayed panel (via cwn_end_panel),
```

```
--                      c. or, a defined button (via cwn_define_button).
--
-- formal parameters
--IN     MENU_STRUCT_ID    The id of the menu structure given by the
--                        application at the time of the menu definition.
--
--IN     MENU_INDEX        The index into the Text_Array of the submenu to
--                        be activated for a particular window, if applicable.
--                        If the menu to be activated is not a walking menu,
--                        or is the top level of a walking menu, then this
--                        parameter should be set to NULL.
--
--IN     WINDOW_TYPE       The type of window the menu will be activated for,
--                        where:
--                        SYS_WINDOW          = a defined window
--                        SYS_DISPLAY_PANEL   = a displayed panel
--                        SYS_DEFINED_BUTTON  = defined button
--
--IN     WINDOW_ID         The id given by the application at the time of the
--                        window type's creation where:
--                        If window_type is SYS_WINDOW and window_id is 0,
--                        then the menu will be activated for the RootWindow
--                        or (Display).  Otherwise, the menu will be activated
--                        for the matching window_id.
--                        If window_type = SYS_DISPLAY_PANEL, the id should
--                        be the panel id.
--                        If window_type = SYS_DEFINED_BUTTON, the id should
--                        be the button id.
--
--IN     PIXEL_X           The X pixel coordinate for posting the menu.
--
--IN     PIXEL_Y           The Y pixel coordinate for posting the menu.
-- end formal parameters;


procedure UWN_QUERY_CHECKBOX_RECTS (CHECKBOX_ID :   in   SYS_WINDOW_ELE_ID;
                    CHECKBOX_RECTS:   in out   UWN_RECTANGLE_ARRAY_PTR);
--
-- CPM description:  Returns the rectangular descriptions of the individual
--                   checkboxes.  Note: these descriptions do not include
--                   the labels in the widths and this routine cannot be
--                   called before the panel containing the checkbox instance
--                   has been ended via UWN_END_PANEL.
-- formal parameters
--IN       CHECKBOX_ID      ID attached to the editor.
--
--IN OUT   CHECKBOX_RECTS   The array of rectangle descriptions.
-- end formal parameters;


procedure UWN_QUERY_CHECKBOX_SIZE (CHECKBOX_ID:      in   SYS_WINDOW_ELE_ID;
                                   PIXEL_COL:        out  SYS_WINDOW_COLUMN;
                                   PIXEL_ROW:        out  SYS_WINDOW_ROW);
--
-- CPM description: Returns the number of pixel columns and rows that
--                  a checkbox editor occupies.
```

```
--
-- formal parameters
--IN    CHECKBOX_ID     ID attached to the editor.
--
--OUT   PIXEL_COL       Number of pixel columns in the editor.
--
--OUT   PIXEL_ROW       Number of pixel rows in the editor.
-- end formal parameters;


procedure UWN_QUERY_DISPLAY_SIZE (WIDTH :   out  SYS_WINDOW_COLUMN;
                                  HEIGHT:   out  SYS_WINDOW_ROW);
--
-- CPM description: Returns the number of pixel columns and rows that
--                  are in the Display screen.
--
-- formal parameters
--OUT   WIDTH         Number of pixel columns in the Display screen.
--
--OUT   HEIGHT        Number of pixel rows in the Display screen.
-- end formal parameters;


procedure UWN_QUERY_EDITOR_SIZE (EDITOR_ID:         in   SYS_WINDOW_ELE_ID;
                                 PIXEL_COL:         out  SYS_WINDOW_COLUMN;
                                 PIXEL_ROW:         out  SYS_WINDOW_ROW);
--
-- CPM description: Returns the number of pixel columns and rows that
--                  an editor occupies.
--
-- formal parameters
--IN    EDITOR_ID       ID attached to the editor.
--
--OUT   PIXEL_COL       Number of pixel columns in the editor.
--
--OUT   PIXEL_ROW       Number of pixel rows in the editor.
-- end formal parameters;


procedure UWN_QUERY_FONT_SIZE (PIXEL_COL:         out  SYS_WINDOW_COLUMN;
                               PIXEL_ROW:         out  SYS_WINDOW_ROW);
--
-- CPM description: Returns the number of pixel columns and rows that
--                  a font occupies.
--
-- formal parameters
--OUT   PIXEL_COL       Number of pixel columns in the font.
--
--OUT   PIXEL_ROW       Number of pixel rows in the font.
-- end formal parameters;


procedure UWN_QUERY_NUMBER_FIELD_SIZE (
                                 EDITOR_ID:         in   SYS_WINDOW_ELE_ID;
                                 PIXEL_COL:         out  SYS_WINDOW_COLUMN;
                                 PIXEL_ROW:         out  SYS_WINDOW_ROW);
```

```
--
-- CPM description: Returns the number of pixel columns and rows that
--                  an numeric field editor occupies.
--
-- formal parameters
--IN    EDITOR_ID        ID attached to the editor.
--
--OUT   PIXEL_COL        Number of pixel columns in the editor.
--
--OUT   PIXEL_ROW        Number of pixel rows in the editor.
-- end formal parameters;


procedure UWN_QUERY_PANEL_SIZE (PANEL_ID:          in   SYS_WINDOW_ELE_ID;
                                PIXEL_COL:         out  SYS_WINDOW_COLUMN;
                                PIXEL_ROW:         out  SYS_WINDOW_ROW);
--
-- CPM description: Returns the number of pixel columns and rows that
--                  a panel requires. The size is determined by using the
--                  locations and sizes of the editors that are attached
--                  to the panel.
--
-- formal parameters
--IN    PANEL_ID         ID attached to the panel.
--
--OUT   PIXEL_COL        Number of pixel columns in the window.
--
--OUT   PIXEL_ROW        Number of pixel rows in the window.
-- end formal parameters;


procedure UWN_QUERY_PUSHBUTTON_RECTS (PUSHBUTTON_ID:   in  SYS_WINDOW_ELE_ID;
                    PUSHBUTTON_RECTS: in out   UWN_RECTANGLE_ARRAY_PTR);
--
-- CPM description:  Returns the rectangular descriptions of the individual
--                   pushbuttons.  Note: these descriptions do not include
--                   the labels in the widths and this routine cannot be
--                   called before the panel containing the pushbutton
--                   instance has been ended via UWN_END_PANEL.
-- formal parameters
--IN       PUSHBUTTON_ID     ID attached to the editor.
--
--IN OUT   PUSHBUTTON_RECTS   The array of rectangle descriptions.
-- end formal parameters;


procedure UWN_QUERY_PUSHBUTTON_SIZE (PUSHBUTTON_ID:   in   SYS_WINDOW_ELE_ID;
                                     PIXEL_COL:       out  SYS_WINDOW_COLUMN;
                                     PIXEL_ROW:       out  SYS_WINDOW_ROW);
--
-- CPM description: Returns the number of pixel columns and rows that
--                  a pushbutton editor occupies.
--
-- formal parameters
--IN    PUSHBUTTON_ID   ID attached to the editor.
--
```

```
--OUT   PIXEL_COL       Number of pixel columns in the editor.
--
--OUT   PIXEL_ROW       Number of pixel rows in the editor.
-- end formal parameters;


 procedure UWN_QUERY_RADIOBUTTON_RECTS (
                      RADIOBUTTON_ID:   in   SYS_WINDOW_ELE_ID;
                      RADIOBUTTON_RECTS: in out   UWN_RECTANGLE_ARRAY_PTR);
--
-- CPM description:  Returns the rectangular descriptions of the individual
--                   radiobuttons.  Note: these descriptions do not include
--                   the labels in the widths and this routine cannot be
--                   called before the panel containing the radiobutton
--                   instance has been ended via UWN_END_PANEL.
-- formal parameters
--IN       RADIOBUTTON_ID     ID attached to the editor.
--
--IN OUT   RADIOBUTTON_RECTS   The array of rectangle descriptions.
--
-- end formal parameters;


procedure UWN_QUERY_RADIOBUTTON_SIZE (
                                    RADIOBUTTON_ID:    in   SYS_WINDOW_ELE_ID;
                                    PIXEL_COL:         out  SYS_WINDOW_COLUMN;
                                    PIXEL_ROW:         out  SYS_WINDOW_ROW);
--
-- CPM description: Returns the number of pixel columns and rows that
--                  a radiobutton editor occupies.
--
-- formal parameters
--IN    RADIOBUTTON_ID  ID attached to the editor.
--
--OUT   PIXEL_COL       Number of pixel columns in the editor.
--
--OUT   PIXEL_ROW       Number of pixel rows in the editor.
-- end formal parameters;


procedure UWN_QUERY_SCROLLBAR_SIZE (
                                   SCROLLBAR_ID:      in   SYS_WINDOW_ELE_ID;
                                   PIXEL_COL:         out  SYS_WINDOW_COLUMN;
                                   PIXEL_ROW:         out  SYS_WINDOW_ROW);
--
-- CPM description: Returns the number of pixel columns and rows that
--                  a scrollbar occupies.
--
-- formal parameters
--IN    SCROLLBAR_ID     ID attached to the scrollbar.
--
--OUT   PIXEL_COL        Number of pixel columns in the scrollbar.
--
--OUT   PIXEL_ROW        Number of pixel rows in the scrollbar.
-- end formal parameters;
```

```
procedure UWN_QUERY_STRING_FIELD_SIZE (
                                    EDITOR_ID:          in   SYS_WINDOW_ELE_ID;
                                    PIXEL_COL:          out  SYS_WINDOW_COLUMN;
                                    PIXEL_ROW:          out  SYS_WINDOW_ROW);
--
-- CPM description: Returns the number of pixel columns and rows that
--                  an string field editor occupies.
--
-- formal parameters
--IN    EDITOR_ID         ID attached to the editor.
--
--OUT   PIXEL_COL         Number of pixel columns in the editor.
--
--OUT   PIXEL_ROW         Number of pixel rows in the editor.
-- end formal parameters;


procedure UWN_QUERY_SUBPANEL_SIZE (SUBPANEL_ID:       in   SYS_WINDOW_ELE_ID;
                                    PIXEL_COL:          out  SYS_WINDOW_COLUMN;
                                    PIXEL_ROW:          out  SYS_WINDOW_ROW);
--
-- CPM description: Returns the number of pixel columns and rows that
--                  a subpanel requires. The size is determined by using the
--                  locations and sizes of the editors that are attached
--                  to the subpanel.
--
-- formal parameters
--IN    SUBPANEL_ID       ID attached to the subpanel.
--
--OUT   PIXEL_COL         Number of pixel columns in the window.
--
--OUT   PIXEL_ROW         Number of pixel rows in the window.
-- end formal parameters;


procedure UWN_QUERY_WINDOW_SIZE (WINDOW_ID:    in   SYS_WINDOW_ELE_ID;
                                    PIXEL_X:      out  SYS_WINDOW_COLUMN;
                                    PIXEL_Y:      out  SYS_WINDOW_ROW;
                                    PIXEL_COL:    out  SYS_WINDOW_COLUMN;
                                    PIXEL_ROW:    out  SYS_WINDOW_ROW);
--
-- CPM description: Returns the x and y display coordinates of the upper left
--                  corner of the window and the number of pixel columns and
--                  rows that will fit in a window. If buttons have been
--                  created in a window, it is advisable to query for
--                  window size before creating other window structures.
--
-- formal parameters
--IN    WINDOW_ID         The id of the window whose size is being queried.
--
--OUT   PIXEL_X           X screen coordinate of window origin.
--
--OUT   PIXEL_Y           Y screen coordinate of window origin.
--
--OUT   PIXEL_COL         Number of pixel columns in the window.
```

```
--
--OUT   PIXEL_ROW        Number of pixel rows in the window.
-- end formal parameters;


procedure UWN_REMOVE_INPUT_SOCKET (SOCKET_ID:  in   SYS_CLIENT);

--
-- CPM description: UWN_REMOVE_INPUT_SOCKET deletes a socket id to be
--                  watched by UWN_INPUT.
--
-- formal parameters
--IN    SOCKET_ID         ID of the socket to stop watching for input.
-- end formal parameters;


procedure UWN_REMOVE_SYSTEM_MESSAGE;
--
-- CPM description: This routine removes any system message displayed via
--                  cwn_display_system_message.  This should be called
--                  before another system message is displayed.
--
-- formal parameters
--   None
-- end formal parameters;


procedure UWN_RESIZE_CHECKBOX (CHECKBOX_ID:     in   SYS_WINDOW_ELE_ID;
                               PIXEL_COL:       in   SYS_WINDOW_COLUMN;
                               PIXEL_ROW:       in   SYS_WINDOW_ROW;
                               PIXEL_WIDTH:     in   SYS_WINDOW_COLUMN;
                               PIXEL_HEIGHT:    in   SYS_WINDOW_ROW);
--
-- CPM description: Changes the size of a checkbox button editor.
--
-- formal parameters
--IN    CHECKBOX_ID       ID of the editor.
--
--IN    PIXEL_COL         Column number from within the window where the left
--                        side of the editor shall be placed. Column 0 is at
--                        left of the window.
--
--IN    PIXEL_ROW         Row number from within the window where the top side
--                        of the editor shall be placed. Row 0 is at the top
--                        of the window.
--
--IN    PIXEL_WIDTH       The number of columns to be occupied by the editor.
--
--IN    PIXEL_HEIGHT      The number of rows to be occupied by the editor.
-- end formal parameters;


procedure UWN_RESIZE_EDITOR (EDITOR_ID:         in   SYS_WINDOW_ELE_ID;
                             PIXEL_COL:         in   SYS_WINDOW_COLUMN;
                             PIXEL_ROW:         in   SYS_WINDOW_ROW;
                             PIXEL_WIDTH:       in   SYS_WINDOW_COLUMN;
```

```
                                PIXEL_HEIGHT:       in   SYS_WINDOW_ROW);
--
-- CPM description: Changes the size of a window full page text editor.
--
-- formal parameters
--IN     EDITOR_ID          ID of the editor.
--
--IN     PIXEL_COL          Column number from within the window where the left
--                          side of the editor shall be placed. Column 0 is at
--                          left of the window.
--
--IN     PIXEL_ROW          Row number from within the window where the top side
--                          of the editor shall be placed. Row 0 is at the top
--                          of the window.
--
--IN     PIXEL_WIDTH        The number of columns to be occupied by the editor.
--
--IN     PIXEL_HEIGHT       The number of rows to be occupied by the editor.
-- end formal parameters;


procedure UWN_RESIZE_NUMBER_FIELD (
                                EDITOR_ID:          in   SYS_WINDOW_ELE_ID;
                                PIXEL_COL:          in   SYS_WINDOW_COLUMN;
                                PIXEL_ROW:          in   SYS_WINDOW_ROW;
                                PIXEL_WIDTH:        in   SYS_WINDOW_COLUMN;
                                PIXEL_HEIGHT:       in   SYS_WINDOW_ROW);
--
-- CPM description: Changes the size of a numeric field editor.
--
-- formal parameters
--IN     EDITOR_ID          ID of the editor.
--
--IN     PIXEL_COL          Column number from within the window where the left
--                          side of the editor shall be placed. Column 0 is at
--                          left of the window.
--
--IN     PIXEL_ROW          Row number from within the window where the top side
--                          of the editor shall be placed. Row 0 is at the top
--                          of the window.
--
--IN     PIXEL_WIDTH        The number of columns to be occupied by the editor.
--
--IN     PIXEL_HEIGHT       The number of rows to be occupied by the editor.
-- end formal parameters;


procedure UWN_RESIZE_PANEL (PANEL_ID:           in   SYS_WINDOW_ELE_ID;
                            PIXEL_COL:          in   SYS_WINDOW_COLUMN;
                            PIXEL_ROW:          in   SYS_WINDOW_ROW;
                            PIXEL_WIDTH:        in   SYS_WINDOW_COLUMN;
                            PIXEL_HEIGHT:       in   SYS_WINDOW_ROW);
--
-- CPM description: Changes the size of a window panel.
--
-- formal parameters
```

```
--IN     PANEL_ID          ID attached to the panel.
--
--IN     PIXEL_COL         Column number from within the window where the left
--                         side of the panel shall be placed. Column 0 is at
--                         left of the window.
--
--IN     PIXEL_ROW         Row number from within the window where the top side
--                         of the panel shall be placed. Row 0 is at the top
--                         of the window.
--
--IN     PIXEL_WIDTH       The number of columns to be occupied by the panel.
--
--IN     PIXEL_HEIGHT      The number of rows to be occupied by the panel.
-- end formal parameters;


procedure UWN_RESIZE_PUSHBUTTON (
                              PUSHBUTTON_ID:      in   SYS_WINDOW_ELE_ID;
                              PIXEL_COL:          in   SYS_WINDOW_COLUMN;
                              PIXEL_ROW:          in   SYS_WINDOW_ROW;
                              PIXEL_WIDTH:        in   SYS_WINDOW_COLUMN;
                              PIXEL_HEIGHT:       in   SYS_WINDOW_ROW);
--
-- CPM description: Changes the size of a pushbutton editor.
--
-- formal parameters
--IN     PUSHBUTTON_ID     ID of the pushbutton editor.
--
--IN     PIXEL_COL         Column number from within the window where the left
--                         side of the editor shall be placed. Column 0 is at
--                         left of the window.
--
--IN     PIXEL_ROW         Row number from within the window where the top side
--                         of the editor shall be placed. Row 0 is at the top
--                         of the window.
--
--IN     PIXEL_WIDTH       The number of columns to be occupied by the editor.
--
--IN     PIXEL_HEIGHT      The number of rows to be occupied by the editor.
-- end formal parameters;


procedure UWN_RESIZE_RADIOBUTTON (
                              RADIOBUTTON_ID:     in   SYS_WINDOW_ELE_ID;
                              PIXEL_COL:          in   SYS_WINDOW_COLUMN;
                              PIXEL_ROW:          in   SYS_WINDOW_ROW;
                              PIXEL_WIDTH:        in   SYS_WINDOW_COLUMN;
                              PIXEL_HEIGHT:       in   SYS_WINDOW_ROW);
--
-- CPM description: Changes the size of a radiobutton editor.
--
-- formal parameters
--IN     RADIOBUTTON_ID    ID of the radiobutton editor.
--
--IN     PIXEL_COL         Column number from within the window where the left
--                         side of the editor shall be placed. Column 0 is at
```

```
--                          left of the window.
--
--IN    PIXEL_ROW            Row number from within the window where the top side
--                          of the editor shall be placed. Row 0 is at the top
--                          of the window.
--
--IN    PIXEL_WIDTH          The number of columns to be occupied by the editor.
--
--IN    PIXEL_HEIGHT         The number of rows to be occupied by the editor.
-- end formal parameters;


procedure UWN_RESIZE_STRING_FIELD (
                                EDITOR_ID:          in   SYS_WINDOW_ELE_ID;
                                PIXEL_COL:          in   SYS_WINDOW_COLUMN;
                                PIXEL_ROW:          in   SYS_WINDOW_ROW;
                                PIXEL_WIDTH:        in   SYS_WINDOW_COLUMN;
                                PIXEL_HEIGHT:       in   SYS_WINDOW_ROW);
--
-- CPM description: Changes the size of a string field editor.
--
-- formal parameters
--IN    EDITOR_ID            ID of the editor.
--
--IN    PIXEL_COL            Column number from within the window where the left
--                          side of the editor shall be placed. Column 0 is at
--                          left of the window.
--
--IN    PIXEL_ROW            Row number from within the window where the top side
--                          of the editor shall be placed. Row 0 is at the top
--                          of the window.
--
--IN    PIXEL_WIDTH          The number of columns to be occupied by the editor.
--
--IN    PIXEL_HEIGHT         The number of rows to be occupied by the editor.
-- end formal parameters;


procedure UWN_RESIZE_WINDOW (WINDOW_ID:         in   SYS_WINDOW_ELE_ID;
                             PIXEL_COL:         in   SYS_WINDOW_COLUMN;
                             PIXEL_ROW:         in   SYS_WINDOW_ROW;
                             PIXEL_WIDTH:       in   SYS_WINDOW_COLUMN;
                             PIXEL_HEIGHT:      in   SYS_WINDOW_ROW);
--
-- CPM description: Changes the size of a window.
--
-- formal parameters
--IN    WINDOW_ID            ID attached to the window.
--
--IN    PIXEL_COL            Column number where the left side of the window
--                          shall be placed.
--
--IN    PIXEL_ROW            Row number where the top side of the window shall
--                          be placed.
--
--IN    PIXEL_WIDTH          The number of columns to be occupied by the
```

```
--                          window.
--
--IN     PIXEL_HEIGHT        The number of rows to be occupied by the window.
-- end formal parameters;


procedure UWN_SELECT_INPUT (WINDOW_TYPE:      in   SYS_WINDOW_TYPE;
                            WINDOW_ID:        in   SYS_WINDOW_ELE_ID;
                            MOUSE_BUTTONS:    in   UWN_BUTTON_ACTION;
                            EXPOSURE:         in   BOOLEAN);
--
-- CPM description: This function allows the user to select exposure events
--                  and various mouse inputs for a particular window.
--                  Each call for the same window overrides any previous
--                  call.  Only the input selected will be returned to the
--                  application, however, the application must be aware that
--                  if the input occurs within any editor or is an
--                  input handled by either the  menu or panel managers,
--                  then the application will not be notified of the input.
--
-- formal parameters
--IN     WINDOW_TYPE         The type of window for which the input is being
--                           selected for, where:
--                           SYS_WINDOW         = a defined window
--                           SYS_DISPLAY_PANEL  = a displayed panel
--                           SYS_DEFINED_BUTTON = defined button
--
--IN     WINDOW_ID           The id given by the application at the time of the
--                           window type's creation where:
--                           If window_type is SYS_WINDOW and window_id is 0,
--                           then the selection will be for the RootWindow
--                           or (Display).  Otherwise, the selection will be
--                           for the matching window_id.
--                           If window_type = SYS_DISPLAY_PANEL, the id should
--                           be the panel id.
--                           If window_type = SYS_DEFINED_BUTTON, the id should
--                           be the button id.
--
--IN     MOUSE_BUTTONS       Array of logicals indicating selection of mouse
--                           button operations whose input the application
--                           wishes to be notified of, where:
--                                  true  = select
--                                  false = do not select
--                            [0] = Right Button Down
--                            [1] = Middle Button Down
--                            [2] = Left Button Down
--                            [3] = Right Button Up
--                            [4] = Middle Button Up
--                            [5] = Left Button Up
--
--IN     EXPOSURE            Logical indicating whether the application wishes
--                           to be notified of exposure events to the working
--                           window, where:
--                                0 = Do not notify of exposure events
--                                1 = Notify of exposure events
```

```
--                          NOTE:  This logical applies only to windows as
--                          currently exposure events to a panel or button are
--                          handled internally.
-- end formal parameters;


procedure UWN_SHOW_PANEL (PANEL_ID:       in   SYS_WINDOW_ELE_ID);
--
-- CPM description: This procedure displays a panel that has been hidden by
--                  UWN_HIDE_PANEL and enables user input to any of the
--                  panel editors.
--
-- formal parameters
--IN    PANEL_ID          ID attached to the panel to
--                         show.
-- end formal parameters;


procedure UWN_SHOW_SUBPANEL (SUBPANEL_ID:    in   SYS_WINDOW_ELE_ID);
--
-- CPM description: This procedure displays a subpanel that has been hidden
--                  by UWN_HIDE_SUBPANEL and enables user input to any of the
--                  subpanel editors.
-- formal parameters
--IN    SUBPANEL_ID       ID attached to the subpanel to
--                         show.
-- end formal parameters;


procedure UWN_TERMINATE_WINDOW;
--
-- CPM description: This procedure terminates the window system. It must be
--                  called to free the slot in the icon stack assigned to the
--                  window when it was created.
-- formal parameters
--      None
-- end formal parameters;


procedure UWN_TOGGLE_BUTTON (BUTTON_ID:      in   SYS_WINDOW_ELE_ID;
                             BUTTON_LABEL:   in   STRING);
--
-- CPM description: This procedure toggles the state of a button and
--                  optionally relabels it.
-- formal parameters
--IN    BUTTON_ID         ID attached to the button to
--                         toggle.
--
--IN    BUTTON_LABEL      An optional new label for the button.  If this is
--                         set to NULL, then the original label will remain.
-- end formal parameters;


procedure UWN_UNMAP_WINDOW (WINDOW_ID:   in   SYS_WINDOW_ELE_ID);
--
-- CPM description: Routine to unmap a created window.  Any child window
```

```
--                  will no longer be visible until another map call is
--                  made on the parent via UWN_MAP_WINDOW.
--
-- formal parameters
--IN    WINDOW_ID   The ID of the window to be unmapped.
--
-- end formal parameters;


procedure UWN_UPDATE_PANEL (PANEL_ID:   in   SYS_WINDOW_ELE_ID);
--
-- CPM description: Causes a panel to update its structures with additions
--                  or deletions of editors.
--
-- formal parameters
--IN    PANEL_ID          ID attached to the panel.
--                        This ID is required for all interactions with the
--                        panel.
-- end formal parameters;


procedure UWN_USER_INPUT_FIELD (
                  Field_Type :          in       SYS_FIELD_TYPE;
                  Input_String :        in out   STRING;
                  Max_String_Size :     in       POSITIVE;
                  Opt_Label :           in       STRING;
                  X_Pixel :             in       SYS_WINDOW_COLUMN;
                  Y_Pixel :             in       SYS_WINDOW_ROW);
--
-- CPM description: This puts up an editing field for user input of
--                  alphanumeric or numeric strings anywhere within the
--                  display screen.
--
-- formal parameters
--IN    Field_Type       The type of field to be defined and used:
--                       SYS_STRING_FIELD
--                       SYS_NUMBER_FIELD
--
--INOUT Input_String     The variable which will receive the user input.
--                       This variable may be initialized to some value, which
--                       would be displayed.  This must be a NULL terminated
--                       string.
--
--IN    Max_String_Size  The maximum string size allowed for input. The
--                       field will be defined according to this size.
--
--IN    Opt_Label        The optional label (prompt or string) which the
--                       application wishes to be displayed on the left side
--                       of the input field.
--
--IN    X_Pixel          The x screen pixel where the upper left corner of
--                       the field will be placed.
--
--IN    Y_Pixel          The y screen pixel of the display where the upper
--                       left corner of the input field will be placed.
--
```

```
    -- end formal parameters;

    procedure UWN_WINDOW_TERMINATE (WINDOW_ID        :   in   SYS_WINDOW_ELE_ID;
                                    TERM_FLAG        :   out  BOOLEAN);
    --
    --cpm description: UWN_WINDOW_TERMINATE displays the window terminate prompt
    --                 and determines if the left button was clicked to confirm
    --                 the terminate action
    --
    -- formal parameters:
    -- IN   WINDOW_ID    The ID of the window to be terminated.
    --
    -- OUT  TERM_FLAG    The Boolean flag indicating whether the terminate action
    --                   took place.
    --
    -- end formal parameters;

end UWN_WINDOW_SYSTEM;
```

APPENDIX B - Ada PROGRAM SPECIFICATIONS

This appendix contains the package specifications that are included as part of the EDDIC programs. Most of the package specifications exist to reduce the size of the program's code and to make the program more modularized. As a general rule these specifications are designed for use by only the program that they are a part of; however, they might also be handy for other purposes.

The following EDDIC programs contain subordinate package specifications:

CDB - Command and Control (C2) product data base manager.

HLP - Help window display manager.

SCL - Station control manager

SDB - Situation data base manager

WTD - Tool window display manager.

CDB Program Package Specifications

The following package specification is included in the C2 product data base manager program:

CDB_GENERATE_PRODUCT

```
--cpc package specification name: CDB_GENERATE_PRODUCT
--
--cpc description: The CDB_GENERATE_PRODUCT CPC contains the procedures
--                 to generate the command and control reports containing
--                 situation data.
--
--cpc design notes:
--
--cpc package author: Bruce Packard
--                    Science Applications International Corporation
--                    424 Delaware, Suite C3
--                    Leavenworth, KS 66048
--

with SYSTEM_PACKAGE;        use SYSTEM_PACKAGE;
with SDB_SITUATION_DB;      use SDB_SITUATION_DB;

package CDB_GENERATE_PRODUCT is

   procedure CDB_BLUEFOR_AMMUNITION (
                UNIT_ID              : in     SDB_BLUEFOR_UNIT_ID;
                DATE_TIME            : in     SYS_DATE_TIME;
                OPPLAN               : in     SYS_OPPLAN;
                SITUATION_SOCK       : in out SYS_CLIENT;
                CHAR_COUNT           : out    SYS_PRODUCT_LENGTH;
                REPORT_TEXT          : in     SYS_TEXT_PTR);
   --
   -- CPM description: This procedure generates a ammunition strength report
   --                  for an BLUEFOR unit.
   --
   -- formal parameters
   --IN         UNIT_ID            - The ID of the unit to generate an
   --                                ammunition strength report for.
   --
   --IN         DATE_TIME          - Date-time group of requested report
   --
   --IN         OPPLAN             - Operational Plan of requested report
   --
   --IN         SITUATION_SOCK     - The socket number of the situation data
   --                                router.
   --OUT        CHAR_COUNT         - The number of characters in the report.
   --
   --OUT        REPORT_TEXT        - The ASCII text of the report.
   --
   -- end formal parameters;

   procedure CDB_BLUEFOR_EQUIPMENT (
                UNIT_ID              : in     SDB_BLUEFOR_UNIT_ID;
                DATE_TIME            : in     SYS_DATE_TIME;
```

```
                OPPLAN                  : in     SYS_OPPLAN;
                SITUATION_SOCK          : in out SYS_CLIENT;
                CHAR_COUNT              : out    SYS_PRODUCT_LENGTH;
                REPORT_TEXT             : in     SYS_TEXT_PTR);
--
-- CPM description: This procedure generates a equipment strength report
--                  for an BLUEFOR unit.
--
-- formal parameters
--IN          UNIT_ID                 - The ID of the unit to generate an
--                                      equipment strength report for.
--
--IN          DATE_TIME               - Date-time group of requested report
--
--IN          OPPLAN                  - Operational Plan of requested report
--
--IN          SITUATION_SOCK          - The socket number of the situation data
--                                      router.
--OUT         CHAR_COUNT              - The number of characters in the report.
--
--OUT         REPORT_TEXT             - The ASCII text of the report.
--
-- end formal parameters;

procedure CDB_BLUEFOR_FUEL (
                UNIT_ID                 : in     SDB_BLUEFOR_UNIT_ID;
                DATE_TIME               : in     SYS_DATE_TIME;
                OPPLAN                  : in     SYS_OPPLAN;
                SITUATION_SOCK          : in out SYS_CLIENT;
                CHAR_COUNT              : out    SYS_PRODUCT_LENGTH;
                REPORT_TEXT             : in     SYS_TEXT_PTR);
--
-- CPM description: This procedure generates a fuel strength report
--                  for an BLUEFOR unit.
--
-- formal parameters
--IN          UNIT_ID                 - The ID of the unit to generate a fuel
--                                      strength report for.
--
--IN          DATE_TIME               - Date-time group of requested report
--
--IN          OPPLAN                  - Operational Plan of requested report
--
--IN          SITUATION_SOCK          - The socket number of the situation data
--                                      router.
--OUT         CHAR_COUNT              - The number of characters in the report.
--
--OUT         REPORT_TEXT             - The ASCII text of the report.
--
-- end formal parameters;

procedure CDB_BLUEFOR_PERSONNEL (
                UNIT_ID                 : in     SDB_BLUEFOR_UNIT_ID;
                DATE_TIME               : in     SYS_DATE_TIME;
                OPPLAN                  : in     SYS_OPPLAN;
                SITUATION_SOCK          : in out SYS_CLIENT;
```

```
                CHAR_COUNT              : out    SYS_PRODUCT_LENGTH;
                REPORT_TEXT             : in     SYS_TEXT_PTR);
--
-- CPM description: This procedure generates a personnel strength report
--                  for an BLUEFOR unit.
--
-- formal parameters
--IN          UNIT_ID              - The ID of the unit to generate a
--                                   personnel strength report for.
--
--IN          DATE_TIME            - Date-time group of requested report
--
--IN          OPPLAN               - Operational Plan of requested report
--
--IN          SITUATION_SOCK       - The socket number of the situation data
--                                   router.
--OUT         CHAR_COUNT           - The number of characters in the report.
--
--OUT         REPORT_TEXT          - The ASCII text of the report.
--
-- end formal parameters;

procedure CDB_BLUEFOR_TASK_ORG (
                UNIT_ID                 : in     SDB_BLUEFOR_UNIT_ID;
                DATE_TIME               : in     SYS_DATE_TIME;
                OPPLAN                  : in     SYS_OPPLAN;
                SITUATION_SOCK          : in out SYS_CLIENT;
                CHAR_COUNT              : out    SYS_PRODUCT_LENGTH;
                REPORT_TEXT             : in     SYS_TEXT_PTR);
--
-- CPM description: This procedure generates a task organization report
--                  for a BLUEFOR unit.
--
-- formal parameters
--IN          UNIT_ID              - The ID of the unit to generate a task
--                                   organization report for.
--IN          DATE_TIME            - Date-time group of requested report
--
--IN          OPPLAN               - Operational Plan of requested report
--
--IN          SITUATION_SOCK       - The socket number of the situation data
--                                   router.
--OUT         CHAR_COUNT           - The number of characters in the report.
--
--OUT         REPORT_TEXT          - The ASCII text of the report.
--
-- end formal parameters;

procedure CDB_OPFOR_COMMITTED (
                UNIT_ID                 : in     SDB_OPFOR_UNIT_ID;
                DATE_TIME               : in     SYS_DATE_TIME;
                OPPLAN                  : in     SYS_OPPLAN;
                SITUATION_SOCK          : in out SYS_CLIENT;
                CHAR_COUNT              : out    SYS_PRODUCT_LENGTH;
                REPORT_TEXT             : in     SYS_TEXT_PTR);
--
```

```
-- CPM description: This procedure generates a OPFOR committed report
--                  for an OPFOR unit.
--
-- formal parameters
--IN          UNIT_ID              - The ID of the unit to generate a OPFOR
--                                   committed report for.
--
--IN          DATE_TIME            - Date-time group of requested report
--
--IN          OPPLAN               - Operational Plan of requested report
--
--IN          SITUATION_SOCK       - The socket number of the situation data
--                                   router.
--OUT         CHAR_COUNT           - The number of characters in the report.
--
--OUT         REPORT_TEXT          - The ASCII text of the report.
--
-- end formal parameters;

procedure CDB_OPFOR_EQUIPMENT (
              UNIT_ID                : in     SDB_OPFOR_UNIT_ID;
              DATE_TIME              : in     SYS_DATE_TIME;
              OPPLAN                 : in     SYS_OPPLAN;
              SITUATION_SOCK         : in out SYS_CLIENT;
              CHAR_COUNT             : out    SYS_PRODUCT_LENGTH;
              REPORT_TEXT            : in     SYS_TEXT_PTR);
--
-- CPM description: This procedure generates a equipment strength report
--                  for an OPFOR unit.
--
-- formal parameters
--IN          UNIT_ID              - The ID of the unit to generate a OPFOR
--                                   equipment strength report for.
--
--IN          DATE_TIME            - Date-time group of requested report
--
--IN          OPPLAN               - Operational Plan of requested report
--
--IN          SITUATION_SOCK       - The socket number of the situation data
--                                   router.
--OUT         CHAR_COUNT           - The number of characters in the report.
--
--OUT         REPORT_TEXT          - The ASCII text of the report.
--
-- end formal parameters;

procedure CDB_OPFOR_REINFORCED (
              UNIT_ID                : in     SDB_OPFOR_UNIT_ID;
              DATE_TIME              : in     SYS_DATE_TIME;
              OPPLAN                 : in     SYS_OPPLAN;
              SITUATION_SOCK         : in out SYS_CLIENT;
              CHAR_COUNT             : out    SYS_PRODUCT_LENGTH;
              REPORT_TEXT            : in     SYS_TEXT_PTR);
--
-- CPM description: This procedure generates a OPFOR reinforcement report
--                  for an OPFOR unit.
```

```
    --
    -- formal parameters
    --IN          UNIT_ID              - The ID of the unit to generate a OPFOR
    --                                   reinforcement report for.
    --
    --IN          DATE_TIME            - Date-time group of requested report
    --
    --IN          OPPLAN               - Operational Plan of requested report
    --
    --IN          SITUATION_SOCK       - The socket number of the situation data
    --                                   router.
    --OUT         CHAR_COUNT           - The number of characters in the report.
    --
    --OUT         REPORT_TEXT          - The ASCII text of the report.
    --
    -- end formal parameters;

    procedure CDB_OPFOR_TASK_ORG (
                  UNIT_ID                : in     SDB_OPFOR_UNIT_ID;
                  DATE_TIME              : in     SYS_DATE_TIME;
                  OPPLAN                 : in     SYS_OPPLAN;
                  SITUATION_SOCK         : in out SYS_CLIENT;
                  CHAR_COUNT             : out    SYS_PRODUCT_LENGTH;
                  REPORT_TEXT            : in     SYS_TEXT_PTR);
    --
    -- CPM description: This procedure generates a task organization report
    --                  for an OPFOR unit.
    --
    -- formal parameters
    --IN          UNIT_ID              - The ID of the unit to generate a task
    --                                   organization report for.
    --
    --IN          DATE_TIME            - Date-time group of requested report
    --
    --IN          OPPLAN               - Operational Plan of requested report
    --
    --IN          SITUATION_SOCK       - The socket number of the situation data
    --                                   router.
    --OUT         CHAR_COUNT           - The number of characters in the report.
    --
    --OUT         REPORT_TEXT          - The ASCII text of the report.
    --
    -- end formal parameters;

end CDB_GENERATE_PRODUCT;
```

HLP Program Package Specifications

The following package specification is included in the Help Window manager program:

HLP_HELP_REPORT

```
--cpm procedure name: HLP_HELP_REPORT
--
--cpm description: HLP_HELP_REPORT displays a EDDIC Help Report Window.
--
--cpm design notes:
--
--cpm procedure author: Bruce Packard
--                      Science Applications International Corporation
--                      424 Delaware, Suite C3
--                      Leavenworth, KS 66048
--
with SYSTEM_PACKAGE;                use SYSTEM_PACKAGE;
with HDB_HELP_DB;                   use HDB_HELP_DB;

package HLP_HELP_REPORT is

   task type HLP_HELP_TASK is
      entry INITIALIZE    (PRODUCT          : in  SYS_TEXT_PTR;
                           LENGTH           : in  SYS_PRODUCT_LENGTH;
                           COLOR_FONT       : in  SYS_WINDOW_ELE_ID;
                           FONT_WIDTH       : in  SYS_WINDOW_COLUMN;
                           FONT_HEIGHT      : in  SYS_WINDOW_ROW;
                           FONT_MASK        : in  SYS_COLOR_MASK;
                           WINDOW           : out SYS_WINDOW_ELE_ID);
      --
      -- CPM description:  This entry point creates a popup window to display
      --                   a EDDIC Help report in and gets the report from the
      --                   Help DB manager.
      --
      -- formal parameters
      --IN    PRODUCT      Textual Report to Display.
      --
      --IN    LENGTH       The number of characters in the report.
      --
      --IN    COLOR_FONT   The ID of the color font that was downloaded by the
      --                   calling process.
      --
      --IN    FONT_WIDTH   The width of the color font.
      --
      --IN    FONT_HEIGHT  The height of the color font.
      --
      --IN    FONT_MASK    The mask to use with the color font.
      --
      --OUT   WINDOW       The ID of the newly created popup window.
      --
      -- end formal parameters;

      entry PROCESS_INPUT (NEW_WINDOW_INPUT  : in  SYS_WINDOW_INPUT;
                           NEW_WINDOW_VALUE  : in  SYS_WINDOW_VALUE;
                           NEW_WINDOW_DATA   : in  SYS_WINDOW_DATA;
                           WINDOW_TERMINATED : out BOOLEAN);

      --
      -- CPM description:  This entry point processes and input that has
      --                   happended for the popup window created by INITIALIZE.
      --                   This entry point should be called for all input
```

```
        --                      from UWN that matches the window ID from INITIALIZE.
        --                      The WINDOW_TERMINATED flag is set to true if the
        --                      selected action causes the deletion of the popup
        --                      window.
        --
        -- formal parameters
        --IN   NEW_WINDOW_INPUT  Input type (See UWN_WINDOW_SYSTEM for a
        --                       complete description).
        --
        --IN   NEW_WINDOW_VALUE  Input value (See UWN_WINDOW_SYSTEM for a
        --                       complete description).
        --
        --IN   NEW_WINDOW_DATA   Input data (See UWN_WINDOW_SYSTEM for a
        --                       complete description).
        --
        --OUT  WINDOW_TERMINATED Window Termination flag
        --                       true  = Window was terminated
        --                       false = Window was not terminated.
        --
        -- end formal parameters;

        entry TERMINATE_TASK;
        --
        -- CPM description:  This entry point terminates popup status window.
        --

    end;

end HLP_HELP_REPORT;
```

SCL Program Package Specifications

The following package specification is included in the station control manager program:

LUT_MANAGER

```
--cpc package specification name: LUT_MANAGER
--
--cpc description: LUT_MANAGER contains the low level color lookup utilities.
--
--cpc design notes:
--
--cpc package author: Bruce Packard
--                    Science Applications International Corporation
--                    424 Delaware, Suite C3
--                    Leavenworth, KS 66048
--
with SYSTEM_PACKAGE;              use SYSTEM_PACKAGE;

package LUT_MANAGER is

   procedure LUT_LOAD_GRID_CONT_COLOR;
   --
   -- CPM description: Loads the grid and contour colors into the color lookup
   --                  table.
   --
   -- formal parameters
   --   None
   --

   procedure LUT_LOAD_GEN_COLOR;
   --
   -- CPM description: Loads the general colors into the color lookup table.
   --
   -- formal parameters
   --   None
   --

   procedure LUT_INIT_LOOK_UP_TABLE (
                     HILITE_FILE   :     in    STRING;
                     UNHILITE_FILE :     in    STRING);
   --
   -- CPM description: Initializes the color lookup table arrays.
   --
   -- formal parameters
   --IN    HILITE_FILE       The name of the file containing the map highlight
   --                        lookup table file names.
   --
   --IN    UNHILITE_FILE     The name of the file containing the map unhighlight
   --                        lookup table file names.
   --

   procedure LUT_LOAD_HYDRO_COLOR (
                     HILITE_FLAG   :     in    BOOLEAN);
   --
   -- CPM description: Loads the hydrography colors into the color lookup table.
   --
   -- formal parameters
   --IN    HILITE_FLAG       Flag indicating if the hydrography is being
   --                        highlighted or returned to the normal color.
   --                        (True = Highlight;  False = Single color)
   --
```

```
procedure LUT_LOAD_BACK_COLOR;
--
-- CPM description: Loads the map colors into the color lookup table.
--
-- formal parameters
--      None

procedure LUT_LOAD_MISC_COLOR (
                  HILITE_FLAG   :     in    BOOLEAN);
--
-- CPM description: Loads the miscellaneous feature colors into the color
--                  lookup table.
--
-- formal parameters
--IN    HILITE_FLAG       Flag indicating if the misc. features are being
--                        highlighted or returned to the normal color.
--                        (True = Highlight;  False = Single color)
--

procedure LUT_LOAD_OVERLAY_COLOR;
--
-- CPM description: Loads overlay colors into the color lookup table.
--
-- formal parameters
--      None
--

procedure LUT_LOAD_ROAD_COLOR (
                  HILITE_FLAG   :     in    BOOLEAN);
--
-- CPM description: Loads the road colors into the color lookup table.
--
-- formal parameters
--IN    HILITE_FLAG       Flag indicating if the roads are being
--                        highlighted or returned to the normal color.
--                        (True = Highlight;  False = Single color)
--

procedure LUT_LOAD_URBAN_COLOR (
                  HILITE_FLAG   :     in    BOOLEAN);
--
-- CPM description: Loads the urban area colors into the color lookup table.
--
-- formal parameters
--IN    HILITE_FLAG       Flag indicating if the urban areas are being
--                        highlighted or returned to the normal color.
--                        (True = Highlight;  False = Single color)
--

procedure LUT_READ_HILITE_LUT;
--
-- CPM description: Reads the colors for the highlighted digital map. It
--                  uses the current map background type to determine which
--                  file to read
--
```

```
    -- formal parameters
    --      None
    --

    procedure LUT_READ_OVERLAY_LUT (
                      FILE_NAME        :       in     STRING);
    --
    -- CPM description: Reads the colors for the overlay planes.
    --
    -- formal parameters
    --IN    FILE_NAME          The name of the overlay color lookup table file.
    --

    procedure LUT_READ_UNHILITE_LUT;
    --
    -- CPM description: Reads the colors for the unhighlighted digital map. It
    --                  uses the current map background type to determine which
    --                  file to read
    --
    -- formal parameters
    --      None
    --

end LUT_MANAGER;
```

SDB Program Package Specifications

The following package specifications are included in the situation data base manager program:

```
SDB_INPUT_OUTPUT
SDB_PACKAGE
SDB_SEND_DATA
SDB_UPDATE_DB
```

```
--cpc package specification name: SDB_INPUT_OUTPUT
--
--cpc description: This package contains the disk input/output utilities for
--                 SDB_SITUATION_DB_MANAGER
--
--cpc design notes:
--
--cpc package author: Bruce Packard
--                    Science Applications International Corporation
--                    424 Delaware, Suite C3
--                    Leavenworth, KS 66048
--

with SYSTEM_PACKAGE;   use SYSTEM_PACKAGE;
with SDB_SITUATION_DB; use SDB_SITUATION_DB;

package SDB_INPUT_OUTPUT is

   procedure SDB_OPEN_SITUATION_DB;
   --
   --cpm description: SDB_OPEN_SITUATION_DB opens the data bases that contain
   --                 the scenario situation data
   -- formal parameters
   --   None
   --

   procedure SDB_READ_INDEX_FILES;
   --
   --cpm description: SDB_READ_INDEX_FILES reads the index files for the
   --                 situation data bases
   -- formal parameters
   --   None
   --

   procedure SDB_WRITE_INDEX_FILES;
   --
   --cpm description: SDB_WRITE_INDEX_FILES writes the index files to disk.
   --
   -- formal parameters
   --   None
   --

   procedure SDB_FIND_CNTRL_MSR
                 (CM_ID            : in    SDB_CONTROL_MEASURE_ID;
                  TIME             : in    SYS_DATE_TIME;
                  OPPLAN           : in    SYS_OPPLAN;
                  INDEX            : out   SDB_CNTRL_MSR_PTR;
                  FOUND_FLAG       : out   BOOLEAN);
   --
   --cpm description: Finds a control measure record for a control measure.
   --
   -- formal parameters
   --IN    CM_ID          The ID of the control measure to find.
   --
   --IN    TIME           Date/Time group for data selection.
   --
```

```
--IN    OPLLAN          OPLAN ID for data selection.
--
--OUT   INDEX           Index into the data base for the control measure. If
--                      the control measure is not found, the INDEX points to
--                      place to insert a new record.
--
--OUT   FOUND_FLAG      Logical flag to indicate if a record was found.
--                      (True = Record found; False = Record not found)
--

procedure SDB_FIND_CNTRL_MSR_PNT
               (CM_ID             : in    SDB_CONTROL_MEASURE_ID;
                TIME              : in    SYS_DATE_TIME;
                OPPLAN            : in    SYS_OPPLAN;
                INDEX             : out   SDB_CNTRL_POINT_PTR;
                FOUND_FLAG        : out   BOOLEAN);
--
--cpm description: Finds a point control measure record for a point control
--                 measure.
--
-- formal parameters
--IN    CM_ID           The ID of the point control measure to find.
--
--IN    TIME            Date/Time group for data selection.
--
--IN    OPLLAN          OPLAN ID for data selection.
--
--OUT   INDEX           Index into the data base for the point control
--                      measure. If the control measure is not found, the
--                      INDEX points to place to insert a new record.
--
--OUT   FOUND_FLAG      Logical flag to indicate if a record was found.
--                      (True = Record found; False = Record not found)
--

procedure SDB_FIND_OBSTACLE
              (OBS_ID            : in    SDB_OBSTACLE_ID;
               TIME              : in    SYS_DATE_TIME;
               OPPLAN            : in    SYS_OPPLAN;
               INDEX             : out   SDB_OBST_PTR;
               FOUND_FLAG        : out   BOOLEAN);
--
--cpm description: Finds an obstacle record for a specified time.
--
-- formal parameters
--IN    OBS_ID          The ID of the obstacle to find.
--
--IN    TIME            Date/Time group for data selection.
--
--IN    OPLLAN          OPLAN ID for data selection.
--
--OUT   INDEX           Index into the data base for the obstacle. If the
--                      obstacle is not found, the INDEX points to place
--                      to insert a new record.
--
--OUT   FOUND_FLAG      Logical flag to indicate if a record was found.
```

```
--                      (True = Record found; False = Record not found)
--

procedure SDB_FIND_BLUEFOR_AUTH_AMMO
                (UNIT_ID          : in    SDB_BLUEFOR_UNIT_ID;
                 TIME             : in    SYS_DATE_TIME;
                 OPPLAN           : in    SYS_OPPLAN;
                 INDEX            : out   SDB_BLUE_AM_AUTH_PTR;
                 FOUND_FLAG       : out   BOOLEAN);
--
--cpm description: Finds the authorized ammunition record for a unit.
--
-- formal parameters
--IN     UNIT_ID         The ID of the unit that owns the ammunition.
--
--IN     TIME            Date/Time group for data selection.
--
--IN     OPLLAN          OPLAN ID for data selection.
--
--OUT    INDEX           Index into the data base for the authorized ammo
--                       record. If the authorized ammo record is not found,
--                       the INDEX points to place to insert a new record.
--
--OUT    FOUND_FLAG      Logical flag to indicate if a record was found.
--                       (True = Record found; False = Record not found)
--

procedure SDB_FIND_BLUEFOR_CURR_AMMO
                (UNIT_ID          : in    SDB_BLUEFOR_UNIT_ID;
                 AMMO_ID          : in    SDB_BLUEFOR_AMMO_ID;
                 TIME             : in    SYS_DATE_TIME;
                 OPPLAN           : in    SYS_OPPLAN;
                 INDEX            : out   SDB_BLUE_AM_CURR_PTR;
                 FOUND_FLAG       : out   BOOLEAN);
--
--cpm description: Finds the on-hand ammunition record for a unit and ammo
--                 type.
--
-- formal parameters
--IN     UNIT_ID         The ID of the unit that owns the ammunition.
--
--IN     AMMO_ID         The ID of the ammunition to find.
--
--IN     TIME            Date/Time group for data selection.
--
--IN     OPLLAN          OPLAN ID for data selection.
--
--OUT    INDEX           Index into the data base for the on-hand ammo
--                       record. If the on-hand ammo record is not found,
--                       the INDEX points to place to insert a new record.
--
--OUT    FOUND_FLAG      Logical flag to indicate if a record was found.
--                       (True = Record found; False = Record not found)
--

procedure SDB_FIND_BLUEFOR_AUTH_EQUIP
```

```
                (UNIT_ID          : in    SDB_BLUEFOR_UNIT_ID;
                 TIME             : in    SYS_DATE_TIME;
                 OPPLAN           : in    SYS_OPPLAN;
                 INDEX            : out   SDB_BLUE_EQ_AUTH_PTR;
                 FOUND_FLAG       : out   BOOLEAN);
--
--cpm description: Finds the authorized equipment record for a unit.
--
-- formal parameters
--IN     UNIT_ID         The ID of the unit that owns the equipment.
--
--IN     TIME            Date/Time group for data selection.
--
--IN     OPLLAN          OPLAN ID for data selection.
--
--OUT    INDEX           Index into the data base for the authorized equipment
--                       record. If the authorized equip record is not found,
--                       the INDEX points to place to insert a new record.
--
--OUT    FOUND_FLAG      Logical flag to indicate if a record was found.
--                       (True = Record found; False = Record not found)
--

procedure SDB_FIND_BLUEFOR_CURR_EQUIP
                (UNIT_ID          : in    SDB_BLUEFOR_UNIT_ID;
                 EQUIP_ID         : in    SDB_BLUEFOR_EQUIP_ID;
                 TIME             : in    SYS_DATE_TIME;
                 OPPLAN           : in    SYS_OPPLAN;
                 INDEX            : out   SDB_BLUE_EQ_CURR_PTR;
                 FOUND_FLAG       : out   BOOLEAN);
--
--cpm description: Finds the operational equipment record for a unit and
--                 equip type.
--
-- formal parameters
--IN     UNIT_ID         The ID of the unit that owns the equipment.
--
--IN     EQUIP_ID        The ID of the equipment to find.
--
--IN     TIME            Date/Time group for data selection.
--
--IN     OPLLAN          OPLAN ID for data selection.
--
--OUT    INDEX           Index into the data base for the operational equipment
--                       record. If the operational equip record is not found,
--                       the INDEX points to place to insert a new record.
--
--OUT    FOUND_FLAG      Logical flag to indicate if a record was found.
--                       (True = Record found; False = Record not found)
--

procedure SDB_FIND_BLUEFOR_PERS
                (UNIT_ID          : in    SDB_BLUEFOR_UNIT_ID;
                 TIME             : in    SYS_DATE_TIME;
                 OPPLAN           : in    SYS_OPPLAN;
                 INDEX            : out   SDB_BLUE_PERS_PTR;
```

```
                FOUND_FLAG        : out   BOOLEAN);
--
--cpm description: Finds the personnel record for a unit.
--
-- formal parameters
--IN    UNIT_ID          The ID of the unit that owns the equipment.
--
--IN    TIME             Date/Time group for data selection.
--
--IN    OPLLAN           OPLAN ID for data selection.
--
--OUT   INDEX            Index into the data base for the personnel record. If
--                       the personnel record is not found, the INDEX points to
--                       the place to insert a new record.
--
--OUT   FOUND_FLAG       Logical flag to indicate if a record was found.
--                       (True = Record found; False = Record not found)
--

procedure SDB_FIND_BLUEFOR_FUEL
               (UNIT_ID           : in    SDB_BLUEFOR_UNIT_ID;
                TIME              : in    SYS_DATE_TIME;
                OPPLAN            : in    SYS_OPPLAN;
                INDEX             : out   SDB_BLUE_FUEL_PTR;
                FOUND_FLAG        : out   BOOLEAN);
--
--cpm description: Finds the fuel record for a unit.
--
-- formal parameters
--IN    UNIT_ID          The ID of the unit that owns the fuel.
--
--IN    TIME             Date/Time group for data selection.
--
--IN    OPLLAN           OPLAN ID for data selection.
--
--OUT   INDEX            Index into the data base for the fuel record. If
--                       the fuel record is not found, the INDEX points to
--                       the place to insert a new record.
--
--OUT   FOUND_FLAG       Logical flag to indicate if a record was found.
--                       (True = Record found; False = Record not found)
--

procedure SDB_FIND_BLUEFOR_STATUS
               (UNIT_ID           : in    SDB_BLUEFOR_UNIT_ID;
                TIME              : in    SYS_DATE_TIME;
                OPPLAN            : in    SYS_OPPLAN;
                INDEX             : out   SDB_BLUE_STAT_PTR;
                FOUND_FLAG        : out   BOOLEAN);
--
--cpm description: Finds the status record for a unit.
--
-- formal parameters
--IN    UNIT_ID          The ID of the unit to find.
--
--IN    TIME             Date/Time group for data selection.
```

```
--
--IN     OPLLAN          OPLAN ID for data selection.
--
--OUT    INDEX           Index into the data base for the unit status record.
--                       If the unit status record is not found, the INDEX
--                       points to the place to insert a new record.
--
--OUT    FOUND_FLAG      Logical flag to indicate if a record was found.
--                       (True = Record found; False = Record not found)
--

procedure SDB_FIND_BLUEFOR_LOCATION
                (UNIT_ID          : in    SDB_BLUEFOR_UNIT_ID;
                 TIME             : in    SYS_DATE_TIME;
                 OPPLAN           : in    SYS_OPPLAN;
                 INDEX            : out   SDB_BLUE_ULOC_PTR;
                 FOUND_FLAG       : out   BOOLEAN);
--
--cpm description: Finds the unit location record for a unit.
--
-- formal parameters
--IN     UNIT_ID         The ID of the unit to find the location of.
--
--IN     TIME            Date/Time group for data selection.
--
--IN     OPLLAN          OPLAN ID for data selection.
--
--OUT    INDEX           Index into the data base for the unit location record.
--                       If the unit location record is not found, the INDEX
--                       points to the place to insert a new record.
--
--OUT    FOUND_FLAG      Logical flag to indicate if a record was found.
--                       (True = Record found; False = Record not found)
--

procedure SDB_FIND_OPFOR_AUTH_EQUIP
                (UNIT_ID          : in    SDB_OPFOR_UNIT_ID;
                 TIME             : in    SYS_DATE_TIME;
                 OPPLAN           : in    SYS_OPPLAN;
                 INDEX            : out   SDB_OPFOR_EQ_AUTH_PTR;
                 FOUND_FLAG       : out   BOOLEAN);
--
--cpm description: Finds the authorized equipment record for a unit.
--
-- formal parameters
--IN     UNIT_ID         The ID of the unit that owns the equipment.
--
--IN     TIME            Date/Time group for data selection.
--
--IN     OPLLAN          OPLAN ID for data selection.
--
--OUT    INDEX           Index into the data base for the authorized equipment
--                       record. If the authorized equip record is not found,
--                       the INDEX points to place to insert a new record.
--
--OUT    FOUND_FLAG      Logical flag to indicate if a record was found.
```

```
--                      (True = Record found; False = Record not found)
--

procedure SDB_FIND_OPFOR_CURR_EQUIP
                (UNIT_ID          : in    SDB_OPFOR_UNIT_ID;
                 EQUIP_ID         : in    SDB_OPFOR_EQUIP_ID;
                 TIME             : in    SYS_DATE_TIME;
                 OPPLAN           : in    SYS_OPPLAN;
                 INDEX            : out   SDB_OPFOR_EQ_CURR_PTR;
                 FOUND_FLAG       : out   BOOLEAN);
--
--cpm description: Finds the operational equipment record for a unit and
--                 equip type.
--
-- formal parameters
--IN    UNIT_ID         The ID of the unit that owns the equipment.
--
--IN    EQUIP_ID        The ID of the equipment to find.
--
--IN    TIME            Date/Time group for data selection.
--
--IN    OPLLAN          OPLAN ID for data selection.
--
--OUT   INDEX           Index into the data base for the operational equipment
--                      record. If the operational equip record is not found,
--                      the INDEX points to place to insert a new record.
--
--OUT   FOUND_FLAG      Logical flag to indicate if a record was found.
--                      (True = Record found; False = Record not found)
--

procedure SDB_FIND_OPFOR_STATUS
                (UNIT_ID          : in    SDB_OPFOR_UNIT_ID;
                 TIME             : in    SYS_DATE_TIME;
                 OPPLAN           : in    SYS_OPPLAN;
                 INDEX            : out   SDB_OPFOR_STAT_PTR;
                 FOUND_FLAG       : out   BOOLEAN);
--
--cpm description: Finds the status record for a unit.
--
-- formal parameters
--IN    UNIT_ID         The ID of the unit to find.
--
--IN    TIME            Date/Time group for data selection.
--
--IN    OPLLAN          OPLAN ID for data selection.
--
--OUT   INDEX           Index into the data base for the unit status record.
--                      If the unit status record is not found, the INDEX
--                      points to the place to insert a new record.
--
--OUT   FOUND_FLAG      Logical flag to indicate if a record was found.
--                      (True = Record found; False = Record not found)
--

procedure SDB_FIND_OPFOR_LOCATION
```

```
                (UNIT_ID          : in    SDB_OPFOR_UNIT_ID;
                 TIME             : in    SYS_DATE_TIME;
                 OPPLAN           : in    SYS_OPPLAN;
                 INDEX            : out   SDB_OPFOR_ULOC_PTR;
                 FOUND_FLAG       : out   BOOLEAN);
--
--cpm description: Reads the unit location record for a unit.
--
-- formal parameters
--IN     UNIT_ID         The ID of the unit to find the location of.
--
--IN     TIME            Date/Time group for data selection.
--
--IN     OPLLAN          OPLAN ID for data selection.
--
--OUT    INDEX           Index into the data base for the unit location record.
--                       If the unit location record is not found, the INDEX
--                       points to the place to insert a new record.
--
--OUT    FOUND_FLAG      Logical flag to indicate if a record was found.
--                       (True = Record found; False = Record not found)
--

procedure SDB_WRITE_CNTRL_MSR
                (INDEX           : in    SDB_CNTRL_MSR_PTR;
                 CM_REC          : in    SDB_CONTROL_MEASURE_REC;
                 ADD_FLAG        : in    boolean);
--
--cpm description: Writes a control measure record for a specific time.
--
-- formal parameters
--IN     INDEX         Index into the data base where the record is to be
--                     inserted.
--
--IN     CM_REC        Description of the control measure to write.
--
--IN     ADD_FLAG      Logical flag to indicate if the record should be
--                     added or updated.
--                           True = Add the record;
--                           False = replace the data currently in the
--                                   data base for this control measure
--

procedure SDB_WRITE_CNTRL_MSR_PNT
                (INDEX           : in    SDB_CNTRL_POINT_PTR;
                 CM_REC          : in    SDB_CNTRL_MSR_POINT_REC;
                 ADD_FLAG        : in    boolean);
--
--cpm description: Writes a point control measure record for a specific time.
--
-- formal parameters
--IN     INDEX         Index into the data base where the record is to be
--                     inserted.
--
--IN     CM_REC        Description of the point control measure to write.
--
```

```
--IN    ADD_FLAG     Logical flag to indicate if the record should be
--                   added or updated.
--                         True = Add the record;
--                         False = replace the data currently in the
--                                 data base for this point control measure
--

procedure SDB_WRITE_OBSTACLE
                (INDEX            : in    SDB_OBST_PTR;
                 OBS_REC          : in    SDB_OBSTACLE_REC;
                 ADD_FLAG         : in    boolean);
--
--cpm description: Writes an obstacle record for a specific time.
--
-- formal parameters
--IN    INDEX        Index into the data base where the record is to be
--                   inserted.
--
--IN    OBS_REC      Description of the obstacle to write.
--
--IN    ADD_FLAG     Logical flag to indicate if the record should be
--                   added or updated.
--                         True = Add the record;
--                         False = replace the data currently in the
--                                 data base for this obstacle
--

procedure SDB_WRITE_BLUEFOR_AUTH_AMMO
               (INDEX             : in    SDB_BLUE_AM_AUTH_PTR;
                AUTH_REC          : in    SDB_AMMO_AUTH_LIST;
                ADD_FLAG          : in    boolean);
--
--cpm description: Writes the authorized ammunition record for a unit.
--
-- formal parameters
--IN    INDEX        Index into the data base where the record is to be
--                   inserted.
--
--IN    AUTH_REC     Description of the authorized ammunition to write.
--
--IN    ADD_FLAG     Logical flag to indicate if the record should be
--                   added or updated.
--                         True = Add the record;
--                         False = replace the data currently in the
--                                 data base for this authorize ammunition
--

procedure SDB_WRITE_BLUEFOR_CURR_AMMO
               (INDEX             : in    SDB_BLUE_AM_CURR_PTR;
                CURR_REC          : in    SDB_BLUEFOR_AMMO_QTY;
                ADD_FLAG          : in    boolean);
--
--cpm description: Writes the on-hand ammunition for a unit and ammo type.
--
-- formal parameters
--IN    INDEX        Index into the data base where the record is to be
```

```
--                      inserted.
--
--IN     CURR_REC        Description of the on-hand ammunition to write.
--
--IN     ADD_FLAG        Logical flag to indicate if the record should be
--                       added or updated.
--                             True = Add the record;
--                             False = replace the data currently in the
--                                     data base for this on-hand ammunition
--

procedure SDB_WRITE_BLUEFOR_AUTH_EQUIP
                (INDEX              : in    SDB_BLUE_EQ_AUTH_PTR;
                 AUTH_REC           : in    SDB_EQUIP_AUTH_LIST;
                 ADD_FLAG           : in    boolean);
--
--cpm description: Writes the authorized equipment for a unit.
--
-- formal parameters
--IN     INDEX           Index into the data base where the record is to be
--                       inserted.
--
--IN     AUTH_REC        Description of the authorized equipment to write.
--
--IN     ADD_FLAG        Logical flag to indicate if the record should be
--                       added or updated.
--                             True = Add the record;
--                             False = replace the data currently in the
--                                     data base for this authorized equipment
--

procedure SDB_WRITE_BLUEFOR_CURR_EQUIP
               (INDEX               : in    SDB_BLUE_EQ_CURR_PTR;
                CURR_REC            : in    SDB_BLUEFOR_EQUIP_QTY;
                ADD_FLAG            : in    boolean);
--
--cpm description: Writes the operational equipment for a unit and equip
--                 type.
--
-- formal parameters
--IN     INDEX           Index into the data base where the record is to be
--                       inserted.
--
--IN     CURR_REC        Description of the operational equipment to write.
--
--IN     ADD_FLAG        Logical flag to indicate if the record should be
--                       added or updated.
--                             True = Add the record;
--                             False = replace the data currently in the
--                                     data base for this operational equipment
--

procedure SDB_WRITE_BLUEFOR_PERS
               (INDEX               : in    SDB_BLUE_PERS_PTR;
                PERS_REC            : in    SDB_PERSONNEL;
                ADD_FLAG            : in    boolean);
```

```
--
--cpm description: Writes the personnel record for a unit.
--
-- formal parameters
--IN     INDEX         Index into the data base where the record is to be
--                     inserted.
--
--IN     PERS_REC      Description of the personnel record to write.
--
--IN     ADD_FLAG      Logical flag to indicate if the record should be
--                     added or updated.
--                           True = Add the record;
--                           False = replace the personnel data currently in the
--                                   data base for this unit
--

procedure SDB_WRITE_BLUEFOR_FUEL
                (INDEX             : in    SDB_BLUE_FUEL_PTR;
                 FUEL_REC          : in    SDB_FUELS;
                 ADD_FLAG          : in    boolean);
--
--cpm description: Writes the fuel record for a unit.
--
-- formal parameters
--IN     INDEX         Index into the data base where the record is to be
--                     inserted.
--
--IN     FUEL_REC      Description of the fuel record to write.
--
--IN     ADD_FLAG      Logical flag to indicate if the record should be
--                     added or updated.
--                           True = Add the record;
--                           False = replace the fuel data currently in the
--                                   data base for this unit
--

procedure SDB_WRITE_BLUEFOR_STATUS
                (INDEX             : in    SDB_BLUE_STAT_PTR;
                 STATUS_REC        : in    SDB_BLUE_UNIT_STATUS;
                 ADD_FLAG          : in    boolean);
--
--cpm description: Writes the status record for a unit.
--
-- formal parameters
--IN     INDEX         Index into the data base where the record is to be
--                     inserted.
--
--IN     STATUS_REC    Description of the unit status record to write.
--
--IN     ADD_FLAG      Logical flag to indicate if the record should be
--                     added or updated.
--                           True = Add the record;
--                           False = replace the status data currently in the
--                                   data base for this unit
--
```

```
procedure SDB_WRITE_BLUEFOR_LOCATION
                (INDEX              : in    SDB_BLUE_ULOC_PTR;
                 LOCATION_REC       : in    SDB_UNIT_LOCATION;
                 ADD_FLAG           : in    boolean);
--
--cpm description: Writes the unit location record for a unit.
--
--
-- formal parameters
--IN    INDEX         Index into the data base where the record is to be
--                    inserted.
--
--IN    LOCATION_REC  Description of the unit location record to write.
--
--IN    ADD_FLAG      Logical flag to indicate if the record should be
--                    added or updated.
--                          True = Add the record;
--                          False = replace the location data currently in the
--                                  data base for this unit

procedure SDB_WRITE_OPFOR_AUTH_EQUIP
               (INDEX             : in    SDB_OPFOR_EQ_AUTH_PTR;
                AUTH_REC          : in    SDB_EQUIP_AUTH_LIST;
                ADD_FLAG          : in    boolean);
--
--cpm description: Writes the authorized equipment for a unit.
--
-- formal parameters
--IN    INDEX         Index into the data base where the record is to be
--                    inserted.
--
--IN    AUTH_REC      Description of the authorized equipment to write.
--
--IN    ADD_FLAG      Logical flag to indicate if the record should be
--                    added or updated.
--                          True = Add the record;
--                          False = replace the data currently in the
--                                  data base for this authorized equipment
--

procedure SDB_WRITE_OPFOR_CURR_EQUIP
               (INDEX             : in    SDB_OPFOR_EQ_CURR_PTR;
                CURR_REC          : in    SDB_OPFOR_EQUIP_QTY;
                ADD_FLAG          : in    boolean);
--
--cpm description: Writes the operational equipment for a unit and equip
--                 type.
--
-- formal parameters
--IN    INDEX         Index into the data base where the record is to be
--                    inserted.
--
--IN    CURR_REC      Description of the operational equipment to write.
--
--IN    ADD_FLAG      Logical flag to indicate if the record should be
--                    added or updated.
```

```
--                          True = Add the record;
--                          False = replace the data currently in the
--                                  data base for this operational equipment
--

procedure SDB_WRITE_OPFOR_STATUS
                (INDEX              : in    SDB_OPFOR_STAT_PTR;
                 STATUS_REC         : in    SDB_OPFOR_UNIT_STATUS;
                 ADD_FLAG           : in    boolean);
--
--cpm description: Writes the status record for a unit.
--
-- formal parameters
--IN    INDEX         Index into the data base where the record is to be
--                    inserted.
--
--IN    STATUS_REC    Description of the unit status record to write.
--
--IN    ADD_FLAG      Logical flag to indicate if the record should be
--                    added or updated.
--                          True = Add the record;
--                          False = replace the status data currently in the
--                                  data base for this unit
--

procedure SDB_WRITE_OPFOR_LOCATION
                (INDEX              : in    SDB_OPFOR_ULOC_PTR;
                 LOCATION_REC       : in    SDB_UNIT_LOCATION;
                 ADD_FLAG           : in    boolean);
--
--cpm description: Writes the unit location record for a unit.
--
-- formal parameters
--IN    INDEX         Index into the data base where the record is to be
--                    inserted.
--
--IN    LOCATION_REC  Description of the unit location record to write.
--
--IN    ADD_FLAG      Logical flag to indicate if the record should be
--                    added or updated.
--                          True = Add the record;
--                          False = replace the location data currently in    --
--                           the data base for this unit
--

procedure SDB_READ_CNTRL_MSR
                (INDEX              : in    SDB_CNTRL_MSR_PTR;
                 CM_REC             : out   SDB_CONTROL_MEASURE_REC);
--
--cpm description: Reads a control measure record.
--
-- formal parameters
--IN    INDEX         Index into the data base of the record to be read.
--
--OUT   CM_REC        Control measure record read.
--
```

```
procedure SDB_READ_CNTRL_MSR_LIST
                (TIME              : in     SYS_DATE_TIME;
                 OPPLAN            : in    SYS_OPPLAN;
                 CM_LIST           : in out SDB_ALL_CNTRL_MSR);
--
--cpm description: Reads the control measure list for a specific time.
--
-- formal parameters
--IN    TIME           Date/Time group for data selection.
--
--IN    OPLLAN         OPLAN ID for data selection.
--
--OUT   CM_LIST        List of control measures for specified time.
--

procedure SDB_READ_CNTRL_MSR_PNT
               (INDEX             : in    SDB_CNTRL_POINT_PTR;
                CM_REC            : out   SDB_CNTRL_MSR_POINT_REC);
--
--cpm description: Reads a point control measure record.
--
-- formal parameters
--IN    INDEX        Index into the data base of tne record to be read.
--
--OUT   CM_REC       Point control measure record read.
--

procedure SDB_READ_CNTRL_MSR_PNT_LIST
               (TIME              : in     SYS_DATE_TIME;
                OPPLAN            : in     SYS_OPPLAN;
                CM_LIST           : in out SDB_ALL_CNTRL_POINT);
--
--cpm description: Reads the control measure list for a specific time.
--
-- formal parameters
--IN    TIME           Date/Time group for data selection.
--
--IN    OPLLAN         OPLAN ID for data selection.
--
--OUT   CM_LIST        List of point control measures for specified time.
--

procedure SDB_READ_OBSTACLE
               (INDEX             : in    SDB_OBST_PTR,
                OBS_REC           : out   SDB_OBSTACLE_REC);
--
--cpm description: Reads an obstacle record.
--
-- formal parameters
--IN    INDEX        Index into the data base of the record to be read.
--
--OUT   OBS_REC      Obstacle record read.
--

procedure SDB_READ_OBSTACLE_LIST
```

```
                (TIME              : in     SYS_DATE_TIME;
                 OPPLAN            : in    SYS_OPPLAN;
                 OBS_LIST          : in out SDB_ALL_OBSTACLE);
--
--cpm description: Reads the obstacle list for a specific time.
--
-- formal parameters
--IN    TIME           Date/Time group for data selection.
--
--IN    OPLLAN         OPLAN ID for data selection.
--
--OUT   OBS_LIST       List of obstacles for specified time.
--

procedure SDB_READ_BLUEFOR_AUTH_AMMO
               (INDEX             : in    SDB_BLUE_AM_AUTH_PTR;
                AUTH_REC          : out   SDB_AMMO_AUTH_LIST);
--
--cpm description: Reads the authorized ammunition for a unit.
--
-- formal parameters
--IN    INDEX        Index into the data base of the record to be read.
--
--OUT   AUTH_REC     Authorized ammunition record read.
--

procedure SDB_READ_BLUEFOR_CURR_AMMO
               (INDEX             : in    SDB_BLUE_AM_CURR_PTR;
                CURR_REC          : out   SDB_BLUEFOR_AMMO_QTY);
--
--cpm description: Reads a on-hand ammunition record.
--
-- formal parameters
--IN    INDEX        Index into the data base of the record to be read.
--
--OUT   CURR_REC     On-hand ammunition record read.
--

procedure SDB_READ_BLUEFOR_CURR_AMMO_LIST
              (UNIT_ID           : in     SDB_BLUEFOR_UNIT_ID;
               TIME              : in     SYS_DATE_TIME;
               OPPLAN            : in    SYS_OPPLAN;
               AUTH_REC          : in     SDB_AMMO_AUTH_LIST;
               CURR_LIST         : in out SDB_AMMO_ON_HAND_REC);
--
--cpm description: Reads a on-hand list of ammunition for a unit for a
--                 specified time and OPLAN.
--
-- formal parameters
--IN    UNIT_ID        ID of the unit that owns the ammunition.
--
--IN    TIME           Date/Time group for data selection.
--
--IN    OPLLAN         OPLAN ID for data selection.
--
--IN    AUTH_REC       Description of the authorized ammunition for this
```

```
--                      unit.
--
--OUT   CURR_LIST       List of on-hand ammunition for specified time.
--


procedure SDB_READ_BLUEFOR_AUTH_EQUIP
                (INDEX              : in    SDB_BLUE_EQ_AUTH_PTR;
                 AUTH_REC           : out   SDB_EQUIP_AUTH_LIST);
--
--cpm description: Reads the authorized equipment for a unit.
--
-- formal parameters
--IN    INDEX          Index into the data base of the record to be read.
--
--OUT   AUTH_REC       Authorized equipment record read.
--

procedure SDB_READ_BLUEFOR_CURR_EQUIP
                (INDEX              : in    SDB_BLUE_EQ_CURR_PTR;
                 CURR_REC           : out   SDB_BLUEFOR_EQUIP_QTY);
--
--cpm description: Reads a operational equipment record.
--
-- formal parameters
--IN    INDEX          Index into the data base of the record to be read.
--
--OUT   CURR_REC       Operational equipment record read.
--

procedure SDB_READ_BLUEFOR_CURR_EQUIP_LIST
               (UNIT_ID             : in     SDB_BLUEFOR_UNIT_ID;
                TIME                : in     SYS_DATE_TIME;
                OPPLAN              : in     SYS_OPPLAN;
                AUTH_REC            : in     SDB_EQUIP_AUTH_LIST;
                CURR_LIST           : in out SDB_EQUIP_OPER_REC);
--
--cpm description: Reads a operational list of equipment for a unit.
--
-- formal parameters
--IN    UNIT_ID         ID of the unit that owns the ammunition.
--
--IN    TIME            Date/Time group for data selection.
--
--IN    OPLLAN          OPLAN ID for data selection.
--
--IN    AUTH_REC        Description of the authorized equipment for this
--                      unit.
--
--OUT   CURR_LIST       List of operational equipment for specified time.
--

procedure SDB_READ_BLUEFOR_PERS
                (INDEX              : in    SDB_BLUE_PERS_PTR;
                 PERS_REC           : out   SDB_PERSONNEL);
--
```

```
--cpm description: Reads the personnel record for a unit.
--
-- formal parameters
--IN    INDEX        Index into the data base of the record to be read.
--
--OUT   PERS_REC     Personnel record read.
--

procedure SDB_READ_BLUEFOR_FUEL
               (INDEX            : in    SDB_BLUE_FUEL_PTR;
                FUEL_REC         : out   SDB_FUELS);
--
--cpm description: Reads the fuel record for a unit.
--
-- formal parameters
--IN    INDEX        Index into the data base of the record to be read.
--
--OUT   FUEL_REC     Fuel record read.
--

procedure SDB_READ_BLUEFOR_STATUS
               (INDEX            : in    SDB_BLUE_STAT_PTR;
                STATUS_REC       : out   SDB_BLUE_UNIT_STATUS);
--
--cpm description: Reads the status record for a unit.
--
-- formal parameters
--IN    INDEX        Index into the data base of the record to be read.
--
--OUT   STATUS_REC   Unit Status record read.
--

procedure SDB_READ_BLUEFOR_LOCATION
               (INDEX            : in    SDB_BLUE_ULOC_PTR;
                LOCATION_REC     : out   SDB_UNIT_LOCATION);
--
--cpm description: Reads the unit location record for a unit.
--
-- formal parameters
--IN    INDEX        Index into the data base of the record to be read.
--
--OUT   LOCATION_REC Unit location record read.
--

procedure SDB_READ_OPFOR_AUTH_EQUIP
               (INDEX            : in    SDB_OPFOR_EQ_AUTH_PTR;
                AUTH_REC         : out   SDB_EQUIP_AUTH_LIST);
--
--cpm description: Reads the authorized equipment for a unit.
--
-- formal parameters
--IN    INDEX        Index into the data base of the record to be read.
--
--OUT   AUTH_REC     Authorized equipment record read.
--
```

```
procedure SDB_READ_OPFOR_CURR_EQUIP
                (INDEX              : in    SDB_OPFOR_EQ_CURR_PTR;
                 CURR_REC           : out   SDB_OPFOR_EQUIP_QTY);
--
--cpm description: Reads a operational equipment record.
--
-- formal parameters
--IN     INDEX         Index into the data base of the record to be read.
--
--OUT    CURR_REC      Operational equipment record read.
--

procedure SDB_READ_OPFOR_CURR_EQUIP_LIST
                (UNIT_ID            : in     SDB_OPFOR_UNIT_ID;
                 TIME               : in     SYS_DATE_TIME;
                 OPPLAN             : in     SYS_OPPLAN;
                 AUTH_REC           : in     SDB_EQUIP_AUTH_LIST;
                 CURR_LIST          : in out SDB_EQUIP_OPER_REC);
--
--cpm description: Reads a operational list of equipment for a unit.
--
-- formal parameters
--IN     UNIT_ID         ID of the unit that owns the ammunition.
--
--IN     TIME            Date/Time group for data selection.
--
--IN     OPLLAN          OPLAN ID for data selection.
--
--IN     AUTH_REC        Description of the authorized equipment for this
--                       unit.
--
--OUT    CURR_LIST       List of operational equipment for specified time.
--

procedure SDB_READ_OPFOR_STATUS
                (INDEX              : in    SDB_OPFOR_STAT_PTR;
                 STATUS_REC         : out   SDB_OPFOR_UNIT_STATUS);
--
--cpm description: Reads the status record for a unit.
--
-- formal parameters
--IN     INDEX         Index into the data base of the record to be read.
--
--OUT    STATUS_REC    Unit Status record read.
--

procedure SDB_READ_OPFOR_LOCATION
                (INDEX              : in    SDB_OPFOR_ULOC_PTR;
                 LOCATION_REC       : out   SDB_UNIT_LOCATION);
--
--cpm description: Reads the unit location record for a unit.
--
-- formal parameters
--IN     INDEX         Index into the data base of the record to be read.
--
--OUT    LOCATION_REC Unit location record read.
```

```
    --

    procedure SDB_CLOSE_SITUATION_DB;
    --
    --cpm description: Closes all the situation data bases.
    --
    -- formal parameters
    --   None
    --

end SDB_INPUT_OUTPUT;
```

```
--cpc package specification name: SDB_PACKAGE
--
--cpc description: This package contains objects for SDB_SITUATION_DB_MANAGER
--
--cpc design notes:
--
--cpc package author: Bruce Packard
--                    Science Applications International Corporation
--                    424 Delaware, Suite C3
--                    Leavenworth, KS 66048
--

with SYSTEM_PACKAGE;        use SYSTEM_PACKAGE;
with MSG_MESSAGE;           use MSG_MESSAGE;
with SDB_SITUATION_DB;      use SDB_SITUATION_DB;

package SDB_PACKAGE is

   -- Situation Data Update Messages
   SDB_AMMO_MESSAGE            :   MSG_MESSAGE_POINT := new
                                   MSG_VAR_MESSAGES (MSG_AMMO_UPDATE);
   SDB_EQUIP_MESSAGE           :   MSG_MESSAGE_POINT := new
                                   MSG_VAR_MESSAGES (MSG_EQUIP_UPDATE);
   SDB_PERS_MESSAGE            :   MSG_MESSAGE_POINT := new
                                   MSG_VAR_MESSAGES (MSG_PERS_UPDATE);
   SDB_FUEL_MESSAGE            :   MSG_MESSAGE_POINT := new
                                   MSG_VAR_MESSAGES (MSG_FUEL_UPDATE);
   SDB_LOC_MESSAGE             :   MSG_MESSAGE_POINT := new
                                   MSG_VAR_MESSAGES (MSG_LOC_UPDATE);
   SDB_ACT_MESSAGE             :   MSG_MESSAGE_POINT := new
                                   MSG_VAR_MESSAGES (MSG_ACTIVITY_UPDATE);
   SDB_MISS_MESSAGE            :   MSG_MESSAGE_POINT := new
                                   MSG_VAR_MESSAGES (MSG_MISSION_UPDATE);

   -- Message type and length
   MESSAGE_TYPE                :  MSG_MESSAGES;
   MESSAGE_LENGTH              :  MSG_MESSAGE_LEN;
   MESSAGE_OVERHEAD            :  MSG_MESSAGE_LEN    := 9;

   -- Situation Request Messages
   SDB_REQUEST             :  MSG_MESSAGE_POINT := new
                              MSG_VAR_MESSAGES (MSG_SD_REQUEST);
   SDB_STOP                :  MSG_MESSAGE_POINT := new
                              MSG_VAR_MESSAGES (MSG_STOP);
   SDB_CONNECT             :  MSG_MESSAGE_POINT := new
                              MSG_VAR_MESSAGES (MSG_CONNECT);

   -- Socket IDs for the situation data message router
   SIT_ROUTER_SOCK_NUM         :  SYS_CLIENT  := 0;

   -- Error code returned from communications utilities
   ERROR_CODE                  :  SYS_ERROR   := 0;

   -- File name passed in via the unix setenv command
   SDB_FILE_NAME      :  string (SYS_ENV_STRING);
   SDB_ENV_NAME       :  string (SYS_ENV_STRING);
```

```
    SDB_NULL_STRING    :  string (SYS_ENV_STRING);

    -- Change data base or add record flag
    SDB_CHANGE         :  boolean    := false;
    SDB_ADD            :  boolean    := true;

    -- Operational Planning Chaining list
    SDB_OPPLAN_COUNT   :  SYS_OPPLAN;
    SDB_OPPLAN_BASE    :  array (SYS_OPPLAN) of SYS_OPPLAN;
    SDB_OPPLAN_DATE    :  array (SYS_OPPLAN) of SYS_DATE_TIME;

    -- Last used ID for control measures and obstacles
    SDB_LAST_CNTRL_MSR:  SDB_CONTROL_MEASURE_ID;
    SDB_LAST_CNTRL_PNT:  SDB_CONTROL_MEASURE_ID;
    SDB_LAST_OBSTACLE :  SDB_OBSTACLE_ID;

    -- Date time definitions
    START_DATE_TIME    :  SYS_DATE_TIME;
    SYSTEM_START_MIN   :  SYS_MINUTE_TOTAL;

end SDB_PACKAGE;
```

```
--cpc package specification name: SDB_SEND_DATA
--
--cpc description: This package describes the procedures to send data to
--                  requesting processes
--
--cpc design notes:
--
--cpc package author: Bruce Packard
--                    Science Applications International Corporation
--                    424 Delaware, Suite C3
--                    Leavenworth, KS 66048
--

package SDB_SEND_DATA is

   procedure SDB_SEND_CNTRL_MSR;
   --
   --cpm description: Sends a list of the control measures that are effective
   --                 during the requested time.
   --
   -- formal parameters
   --   None
   --

   procedure SDB_SEND_CNTRL_MSR_PNT;
   --
   --cpm description: Sends a list of the point control measures that are
   --                 effective during the requested time.
   --
   -- formal parameters
   --   None
   --

  procedure SDB_SEND_OBSTACLE;
  --
  --cpm description: Sends a list of the obstacles that are effective
  --                 during the requested time.
  --
  -- formal parameters
  --   None
  --

  procedure SDB_SEND_BLUEFOR_AUTH_AMMO;
  --
  --cpm description: Sends a list of the authorized ammunition for a BLUEFOR
  --                 unit.
  --
  -- formal parameters
  --   None
  --

  procedure SDB_SEND_BLUEFOR_CURR_AMMO;
  --
  --cpm description: Sends a list of the on-hand ammunition for a BLUEFOR
  --                 unit.
  --
```

```
-- formal parameters
--   None
--

procedure SDB_SEND_BLUEFOR_EQUIP_AUTH;
--
--cpm description: Sends a list of the authorized equipment for a BLUEFOR
--                 unit.
--
-- formal parameters
--   None
--

procedure SDB_SEND_BLUEFOR_EQUIP_CURR;
--
--cpm description: Sends a list of the operational equipment for a BLUEFOR
--                 unit.
--
-- formal parameters
--   None
--

procedure SDB_SEND_BLUEFOR_PERS;
--
--cpm description: Sends the personnel strength for a BLUEFOR unit.
--
-- formal parameters
--   None
--

procedure SDB_SEND_BLUEFOR_FUEL;
--
--cpm description: Sends the fuel status for a BLUEFOR unit.
--
-- formal parameters
--   None
--

procedure SDB_SEND_BLUEFOR_STATUS;
--
--cpm description: Sends the unit status for a BLUEFOR unit.
--
-- formal parameters
--   None
--

procedure SDB_SEND_BLUEFOR_LOCATION;
--
--cpm description: Sends the unit location for a BLUEFOR unit.
--
-- formal parameters
--   None
--

procedure SDB_SEND_BLUEFOR_TASK_ORG;
--
```

```
--cpm description: Sends the task organization for a BLUEFOR unit.
--
-- formal parameters
--   None
--

procedure SDB_SEND_BLUEFOR_ALL_LOCATIONS;
--
--cpm description: Sends the all the BLUEFOR unit locations.
--
-- formal parameters
--   None
--

procedure SDB_SEND_OPFOR_EQUIP_AUTH;
--
--cpm description: Sends a list of the operational equipment for a OPFOR
--                 unit.
--
-- formal parameters
--   None
--

procedure SDB_SEND_OPFOR_EQUIP_CURR;
--
--cpm description: Sends a list of the operational equipment for a OPFOR
--                 unit.
--
-- formal parameters
--   None
--

procedure SDB_SEND_OPFOR_STATUS;
--
--cpm description: Sends the unit status for a OPFOR unit.
--
-- formal parameters
--   None
--

procedure SDB_SEND_OPFOR_LOCATION;
--
--cpm description: Sends the unit location for a OPFOR unit.
--
-- formal parameters
--   None
--

procedure SDB_SEND_OPFOR_TASK_ORG;
--
--cpm description: Sends the task organization for a OPFOR unit.
--
-- formal parameters
--   None
--
```

```
   procedure SDB_SEND_OPFOR_ALL_LOCATIONS;
   --
   --cpm description: Sends the all the OPFOR unit locations.
   --
   -- formal parameters
   --   None
   --

end SDB_SEND_DATA;
```

```
--cpc package specification name: SDB_UPDATE_DB
--
--cpc description: This package contains the situation data base update
--                 utilities for SDB_SITUATION_DB_MANAGER
--
--cpc design notes:
--
--cpc package author: Bruce Packard
--                    Science Applications International Corporation
--                    424 Delaware, Suite C3
--                    Leavenworth, KS 66048
--

with SYSTEM_PACKAGE; use SYSTEM_PACKAGE;
with SDB_SITUATION_DB; use SDB_SITUATION_DB;

package SDB_UPDATE_DB is

   procedure SDB_UPDATE_BLUEFOR_AMMO (UNIT_ID   :  in SDB_BLUEFOR_UNIT_ID;
                                      ADD_FLAG  :  in boolean);
   --
   --cpm description: Updates the on-hand amount of ammunition assigned to a
   --                 BLUEFOR unit.
   --
   -- formal parameters
   --IN    UNIT_ID      ID of the unit to update the on-hand ammunition level.
   --
   --IN    ADD_FLAG     Logical flag to indicate if the record should be
   --                   added or updated.
   --                         True = Add the record;
   --                         False = replace the data currently in the
   --                                 data base for this unit
   --

   procedure SDB_UPDATE_BLUEFOR_AMMO_AUTH (UNIT_ID   :  in SDB_BLUEFOR_UNIT_ID;
                                           AMMO_NAME :  in STRING;
                                           ADD_FLAG  :  in boolean);
   --
   --cpm description: Updates the authorized amount of ammunition assigned to
--                   a BLUEFOR unit.
   --
   -- formal parameters
   --IN    UNIT_ID      ID of the unit to update the authorized ammunition
   --                   level.
   --
   --IN    AMMO_NAME    Name of the ammunition type that is being updated.
   --
   --IN    ADD_FLAG     Logical flag to indicate if the record should be
   --                   added or updated.
   --                         True = Add the record;
   --                         False = replace the data currently in the
   --                                 data base for this unit
   --

  procedure SDB_UPDATE_BLUEFOR_EQUIP (UNIT_ID    :   in SDB_BLUEFOR_UNIT_ID;
                                      ADD_FLAG   :   in boolean);
```

```
   --
   --cpm description: Updates the operational amount of equipment assigned to
--                  a BLUEFOR unit.
   --
   -- formal parameters
   --IN    UNIT_ID       ID of the unit to update the operational equipment
   --                    level.
   --
   --IN    ADD_FLAG      Logical flag to indicate if the record should be
   --                    added or updated.
   --                          True = Add the record;
   --                          False = replace the data currently in the
   --                                  data base for this unit
   --

   procedure SDB_UPDATE_BLUEFOR_EQUIP_AUTH (UNIT_ID  :   in SDB_BLUEFOR_UNIT_ID;
                                            EQUIP_NAME :  in STRING;
                                            ADD_FLAG   :  in boolean);
   --
   --cpm description: Updates the authorized amount of equipment assigned to a
   --                 BLUEFOR unit.
   --
   -- formal parameters
   --IN    UNIT_ID       ID of the unit to update the operational equipment
   --                    level.
   --
   --IN    EQUIP_NAME    Name of the equipment type that is being updated.
   --
   --IN    ADD_FLAG      Logical flag to indicate if the record should be
   --                    added or updated.
   --                          True = Add the record;
   --                          False = replace the data currently in the
   --                                  data base for this unit
   --

   procedure SDB_UPDATE_BLUEFOR_PERS (UNIT_ID    :  in  SDB_BLUEFOR_UNIT_ID;
                                      ADD_FLAG   :  in  boolean);
   --
   --cpm description: Updates the number of personnel assigned to a
   --                 BLUEFOR unit.
   --
   -- formal parameters
   --IN    UNIT_ID       ID of the unit to update the current personnel level.
   --
   --IN    ADD_FLAG      Logical flag to indicate if the record should be
   --                    added or updated.
   --                          True = Add the record;
   --                          False = replace the data currently in the
   --                                  data base for this unit
   --

   procedure SDB_UPDATE_BLUEFOR_PERS_AUTH (UNIT_ID   :  in  SDB_BLUEFOR_UNIT_ID;
                                           ADD_FLAG   :  in  boolean);
   --
   --cpm description: Updates the number of personnel assigned to a
   --                 BLUEFOR unit.
```

```
--
-- formal parameters
--IN    UNIT_ID      ID of the unit to update the authorized personnel level.
--
--IN    ADD_FLAG     Logical flag to indicate if the record should be
--                   added or updated.
--                         True = Add the record;
--                         False = replace the data currently in the
--                                 data base for this control measure
--

procedure SDB_UPDATE_BLUEFOR_FUEL (UNIT_ID  :  in  SDB_BLUEFOR_UNIT_ID;
                                   ADD_FLAG :  in  boolean);
--
--cpm description: Updates the amount of fuel assigned to a
--                BLUEFOR unit.
--
-- formal parameters
--IN    UNIT_ID      ID of the unit to update the current fuel level.
--
--IN    ADD_FLAG     Logical flag to indicate if the record should be
--                   added or updated.
--                         True = Add the record;
--                         False = replace the data currently in the
--                                 data base for this unit
--

procedure SDB_UPDATE_BLUEFOR_FUEL_AUTH (UNIT_ID  :  in  SDB_BLUEFOR_UNIT_ID;
                                        ADD_FLAG :  in  boolean);
--
--cpm description: Updates the amount of fuel authorized for a
--                BLUEFOR unit.
--
-- formal parameters
--IN    UNIT_ID      ID of the unit to update the authorized fuel level.
--
--IN    ADD_FLAG     Logical flag to indicate if the record should be
--                   added or updated.
--                         True = Add the record;
--                         False = replace the data currently in the
--                                 data base for this unit
--

procedure SDB_UPDATE_BLUEFOR_LOCATION (ADD_FLAG  :  in boolean);
--
--cpm description: Updates the location of a BLUEFOR unit.
--
-- formal parameters
--IN    ADD_FLAG     Logical flag to indicate if the record should be
--                   added or updated.
--                         True = Add the record;
--                         False = replace the data currently in the
--                                 data base for this unit
--
```

```
procedure SDB_UPDATE_BLUEFOR_TASK_ORG (ADD_FLAG  :  in boolean;
                                       START_TIME:  in SYS_DATE_TIME);
--
--cpm description: Updates the task organization of a BLUEFOR unit.
--
-- formal parameters
--IN    ADD_FLAG     Logical flag to indicate if the record should be
--                   added or updated.
--                         True = Add the record;
--                         False = replace the data currently in the
--                                 data base for this unit
--
--IN    START_TIME   Scenario start time
--

procedure SDB_UPDATE_BLUEFOR_ACTIVITY (ADD_FLAG  :  in boolean);
--
--cpm description: Updates the activity of a BLUEFOR unit.
--
-- formal parameters
--IN    ADD_FLAG     Logical flag to indicate if the record should be
--                   added or updated.
--                         True = Add the record;
--                         False = replace the data currently in the
--                                 data base for this unit
--

procedure SDB_UPDATE_BLUEFOR_MISSION (ADD_FLAG  :  in boolean);
--
--cpm description: Updates the mission of a BLUEFOR unit.
--
-- formal parameters
--IN    ADD_FLAG     Logical flag to indicate if the record should be
--                   added or updated.
--                         True = Add the record;
--                         False = replace the data currently in the
--                                 data base for this unit
--

procedure SDB_UPDATE_OPFOR_EQUIP (UNIT_ID  :   in SDB_BLUEFOR_UNIT_ID;
                                  ADD_FLAG :   in boolean);
--
--cpm description: Updates the operational amount of equipment assigned to a
--                 OPFOR unit.
--
-- formal parameters
--IN    UNIT_ID      ID of the unit to update the operational equipment
--                   level.
--
--IN    ADD_FLAG     Logical flag to indicate if the record should be
--                   added or updated.
--                         True = Add the record;
--                         False = replace the data currently in the
--                                 data base for this unit
--
```

```
procedure SDB_UPDATE_OPFOR_EQUIP_AUTH (UNIT_ID   :   in SDB_BLUEFOR_UNIT_ID;
                                       EQUIP_NAME :   in STRING;
                                       ADD_FLAG   :   in boolean);
--
--cpm description: Updates the authorized amount of equipment assigned to a
--                 OPFOR unit.
--
-- formal parameters
--IN    UNIT_ID      ID of the unit to update the operational equipment
--                   level.
--
--IN    EQUIP_NAME   Name of the equipment type that is being updated.
--
--IN    ADD_FLAG     Logical flag to indicate if the record should be
--                   added or updated.
--                         True = Add the record;
--                         False = replace the data currently in the
--                                 data base for this unit
--

procedure SDB_UPDATE_OPFOR_LOCATION (ADD_FLAG  :  in boolean);
--
--cpm description: Updates the location of a OPFOR unit.
--
-- formal parameters
--IN    ADD_FLAG     Logical flag to indicate if the record should be
--                   added or updated.
--                         True = Add the record;
--                         False = replace the data currently in the
--                                 data base for this unit
--

procedure SDB_UPDATE_OPFOR_TASK_ORG (ADD_FLAG  :  in boolean;
                                     START_TIME:  in SYS_DATE_TIME);
--
--cpm description: Updates the task organization of a OPFOR unit.
--
-- formal parameters
--IN    ADD_FLAG     Logical flag to indicate if the record should be
--                   added or updated.
--                         True = Add the record;
--                         False = replace the data currently in the
--                                 data base for this unit
--
--IN    START_TIME   Scenario start time
--


procedure SDB_UPDATE_OPFOR_ACTIVITY (ADD_FLAG  :  in boolean);
--
--cpm description: Updates the activity of a OPFOR unit.
--
-- formal parameters
--IN    ADD_FLAG     Logical flag to indicate if the record should be
--                   added or updated.
--                         True = Add the record;
```

```
--                          False = replace the data currently in the
--                                  data base for this unit
--

procedure SDB_UPDATE_OPFOR_MISSION (ADD_FLAG  :  in boolean);
--
--cpm description: Updates the mission of a OPFOR unit.
--
-- formal parameters
--IN    ADD_FLAG     Logical flag to indicate if the record should be
--                   added or updated.
--                         True = Add the record;
--                         False = replace the data currently in the
--                                 data base for this unit
--

procedure SDB_UPDATE_OPFOR_REINF (ADD_FLAG  :  in boolean;
                                  START_TIME:  in SYS_DATE_TIME);
--
--cpm description: Updates the reinforcing time of a OPFOR unit.
--
-- formal parameters
--IN    ADD_FLAG     Logical flag to indicate if the record should be
--                   added or updated.
--                         True = Add the record;
--                         False = replace the data currently in the
--                                 data base for this unit
--
--IN    START_TIME   Scenario start time
--

procedure SDB_UPDATE_OPFOR_STRENGTH (ADD_FLAG  :  in boolean;
                                     START_TIME:  in SYS_DATE_TIME);
--
--cpm description: Updates the percent strength of a OPFOR unit.
--
-- formal parameters
--IN    ADD_FLAG     Logical flag to indicate if the record should be
--                   added or updated.
--                         True = Add the record;
--                         False = replace the data currently in the
--                                 data base for this unit
--
--IN    START_TIME   Scenario start time
--

procedure SDB_ADD_CNTRL_MSR (ADD_FLAG  :  in boolean;
                             START_TIME:  in SYS_DATE_TIME);
--
--cpm description: Adds a control measure to the situation data base.
--
-- formal parameters
--IN    ADD_FLAG     Logical flag to indicate if the record should be
--                   added or updated.
--                         True = Add the record;
--                         False = replace the data currently in the
```

```
--                                    data base for this control measure
--
--IN    START_TIME    Scenario start time
--

procedure SDB_UPDATE_CNTRL_MSR_EFF (ADD_FLAG   :  in boolean;
                                    START_TIME:  in SYS_DATE_TIME);
--
--cpm description: Updates the effective time of a control measure.
--
-- formal parameters
--IN    ADD_FLAG     Logical flag to indicate if the record should be
--                   added or updated.
--                         True = Add the record;
--                         False = replace the data currently in the
--                                 data base for this control measure
--
--IN    START_TIME    Scenario start time
--

procedure SDB_UPDATE_CNTRL_MSR_LOC (ADD_FLAG   :  in boolean;
                                    START_TIME:  in SYS_DATE_TIME);
--
--cpm description: Updates the location of a control measure.
--
-- formal parameters
--IN    ADD_FLAG     Logical flag to indicate if the record should be
--                   added or updated.
--                         True = Add the record;
--                         False = replace the data currently in the
--                                 data base for this control measure
--
--IN    START_TIME    Scenario start time
--

procedure SDB_UPDATE_CNTRL_MSR_STAT (ADD_FLAG   :  in boolean;
                                     START_TIME:  in SYS_DATE_TIME);
--
--cpm description: Updates the status of a control measure.
--
-- formal parameters
--IN    ADD_FLAG     Logical flag to indicate if the record should be
--                   added or updated.
--                         True = Add the record;
--                         False = replace the data currently in the
--                                 data base for this control measure
--
--IN    START_TIME    Scenario start time
--

procedure SDB_DELETE_CNTRL_MSR (ADD_FLAG   :  in boolean;
                                START_TIME:  in SYS_DATE_TIME);
--
--cpm description: Deletes a control measure.
--
-- formal parameters
```

```
--IN    ADD_FLAG     Logical flag to indicate if the record should be
--                   added or updated.
--                         True = Add the record;
--                         False = replace the data currently in the
--                                 data base for this control measure
--
--IN    START_TIME   Scenario start time
--

procedure SDB_ADD_CNTRL_MSR_PNT (ADD_FLAG  :  in boolean;
                                 START_TIME:  in SYS_DATE_TIME);
--
--cpm description: Adds a point control measure to the situation data base.
--
-- formal parameters
--IN    ADD_FLAG     Logical flag to indicate if the record should be
--                   added or updated.
--                         True = Add the record;
--                         False = replace the data currently in the
--                                 data base for this control measure
--
--IN    START_TIME   Scenario start time
--

procedure SDB_ADD_OBSTACLE (ADD_FLAG  :  in boolean;
                            START_TIME:  in SYS_DATE_TIME);
--
--cpm description: Adds a obstacle to the situation data base.
--
-- formal parameters
--IN    ADD_FLAG     Logical flag to indicate if the record should be
--                   added or updated.
--                         True = Add the record;
--                         False = replace the data currently in the
--                                 data base for this obstacle
--
--IN    START_TIME   Scenario start time
--

procedure SDB_UPDATE_OBSTACLE_EFF (ADD_FLAG  :  in boolean;
                                   START_TIME:  in SYS_DATE_TIME);
--
--cpm description: Updates the effective time of an obstacle.
--
-- formal parameters
--IN    ADD_FLAG     Logical flag to indicate if the record should be
--                   added or updated.
--                         True = Add the record;
--                         False = replace the data currently in the
--                                 data base for this obstacle
--
--IN    START_TIME   Scenario start time
--

procedure SDB_UPDATE_OBSTACLE_LOC (ADD_FLAG  :  in boolean;
                                   START_TIME:  in SYS_DATE_TIME);
```

```
    --
    --cpm description: Updates the location of an obstacle.
    --
    -- formal parameters
    --IN    ADD_FLAG     Logical flag to indicate if the record should be
    --                   added or updated.
    --                         True = Add the record;
    --                         False = replace the data currently in the
    --                                 data base for this obstacle
    --
    --IN    START_TIME   Scenario start time
    --

    procedure SDB_UPDATE_OBSTACLE_STAT (ADD_FLAG  :  in boolean;
                                        START_TIME:  in SYS_DATE_TIME);
    --
    --cpm description: Updates the status of an obstacle.
    --
    -- formal parameters
    --IN    ADD_FLAG     Logical flag to indicate if the record should be
    --                   added or updated.
    --                         True = Add the record;
    --                         False = replace the data currently in the
    --                                 data base for this obstacle
    --
    --IN    START_TIME   Scenario start time
    --

    procedure SDB_DELETE_OBSTACLE (ADD_FLAG  :  in boolean;
                                   START_TIME:  in SYS_DATE_TIME);
    --
    --cpm description: Deletes an obstacle.
    --
    -- formal parameters
    --IN    ADD_FLAG     Logical flag to indicate if the record should be
    --                   added or updated.
    --                         True = Add the record;
    --                         False = replace the data currently in the
    --                                 data base for this obstacle
    --
    --IN    START_TIME   Scenario start time
    --

end SDB_UPDATE_DB;
```

WTD Program Package Specifications

The following package specifications are included in the tool window display manager program:

```
CALC_CALCULATOR
TOT_EDITOR
```

```
--cpc package specification name: CALC_CALCULATOR
--
--cpc description: Calculator for the tool window
--
--cpc design notes:
--
--cpc program author: Bruce Packard
--                    Science Applications International Corporation
--                    424 Delaware, Suite C3
--                    Leavenworth, KS 66048
--
with SYSTEM_PACKAGE;                use SYSTEM_PACKAGE;

package CALC_CALCULATOR is

   procedure CALC_INITIALIZE (WINDOW_ID  : in SYS_WINDOW_ELE_ID);
   --
   -- CPM description: This procedure displays the calculator.
   --
   -- formal parameters
   -- IN          TOT_WIN_ID  - The Id number of the Task Organization Tool
   --                           parent Window.
   --

   procedure CALC_PROCESS_INPUT (INPUT_TYPE       : in  SYS_WINDOW_INPUT;
                                 INPUT_WINDOW_ID  : in  SYS_WINDOW_ELE_ID;
                                 INPUT_VALUE      : in  SYS_WINDOW_VALUE;
                                 INPUT_DATA       : in  SYS_WINDOW_DATA;
                                 FINISHED_PROCESSING : out BOOLEAN);
   --
   -- CPM description: This procedure processes all input on the calculator.
   --
   -- formal parameters
   --IN    INPUT_TYPE       - The Type of Input (See UWN_WINDOW_SYSTEM for a
   --                         complete description).
   --IN    INPUT_WINDOW_ID  - The Id of the Window the Input took place in (See
   --                         UWN_WINDOW_SYSTEM for a complete description).
   --IN    INPUT_VALUE      - The Value of the Input (See UWN_WINDOW_SYSTEM
   --                         for a complete description).
   --IN    INPUT_DATA       - The Input Data (See UWN_WINDOW_SYSTEM for a
   --                         complete description).
   --OUT   FINISHED_PROCESSING - A flag telling the calling process if all the
   --                            Processing is Finished for this input.
   --                          = True  - Processing is finished - don't do
   --                                    anything else.
   --                          = False - Processing is not finished - finish it
   --                                    yourself.
   --

   procedure CALC_TERMINATE;
   --
   -- CPM description: This procedure terminates the calculator tool.
   --
   -- formal parameters
   --          None.
   --
```

```
end CALC_CALCULATOR;
```

```
--CPC package specification name:
--     TOT_EDITOR
--
--CPC description:
--     TOT_EDITOR CPC is the Task Organization Tool, written in the "Ada"
--     programming language, which defines the variables and variable types
--     needed to Edit the task organization unit structure.
--
--CPC design notes:
--     1.)  This package can raise the following exceptions:
--          SYS_TSB_EXCEPTION.
--
--CPC package author:
--     Richard T. Zarse      13 Oct 1988
--     Science Applications International Corporation (SAIC)
--     424 Delaware, Suite C-3
--     Leavenworth, KS  66048   (913) 651-7925
--
--
with SYSTEM_PACKAGE;     use SYSTEM_PACKAGE;

package TOT_EDITOR is

-- ##############################################################################
   procedure TOT_INITIALIZE (TOT_WIN_ID       : in   SYS_WINDOW_ELE_ID;
                             PROCESS_ID       : in   SYS_EDDIC_PROCESSES;
                             HEADR_BTNS_WIDTH : out  SYS_WINDOW_COLUMN);
   --
   --CPM description:
   --     This module, as part of the Task Organization Tool, performs all of
   --     the Initialization needed for the TOT editor.
   --
   --CPM design notes:
   --     1.)  This module is called once, up front, each time the tool is
   --     invoked.
   --     2.)  The parent window to have been created by the application, before
   --     this module is called.
   --
   --formal parameters
   --IN      TOT_WIN_ID       - The Id of the Task Organization Tool parent
   --                           Window.
   --IN      PROCESS_ID       - The Id of the calling process.
   --OUT     HEADR_BTNS_WIDTH - The combined Width of all of the Header Buttons
   --                           displayed in the tool.
   --end formal parameters;
   --

-- ##############################################################################
   procedure TOT_PROCESS_INPUT (INPUT_TYPE          : in   SYS_WINDOW_INPUT;
                                INPUT_WINDOW_ID     : in   SYS_WINDOW_ELE_ID;
                                INPUT_VALUE         : in   SYS_WINDOW_VALUE;
                                INPUT_DATA          : in   SYS_WINDOW_DATA;
                                FINISHED_PROCESSING : out  BOOLEAN);
   --
   --CPM description:
   --     This module, as part of the Task Organization Tool, Processes any
```

```
    --    Input event that has happened in/to the TOT editor.
    --
    --CPM design notes:
    --    1.)  This module does not contain the call to UWN_INPUT, the
    --    application does it so it can have greater control.  Because of this,
    --    it is possible to receive input which has nothing to do with TOT.
    --    2.)  This module is called every time an event is received by the
    --    application.
    --
    --formal parameters
    --IN      INPUT_TYPE           - The Type of Input.
    --IN      INPUT_WINDOW_ID      - The Id of the Window the Input took place in.
    --IN      INPUT_VALUE          - The Value of the Input.
    --IN      INPUT_DATA           - The Input Data.
    --      (See UWN_WINDOW_SYSTEM for a complete description of all 4 of these).
    --OUT     FINISHED_PROCESSING - A flag telling the calling process if all the
    --                              Processing is Finished for this input.
    --                             = True  - Processing is finished - don't do
    --                                       anything else.
    --                             = False - Processing is not finished - finish
    --                                       it yourself.
    --end formal parameters;
    --

-- ##############################################################################
   procedure TOT_TERMINATE;
   --
   --CPM description:
   --    This module, as part of the Task Organization Tool, performs the
   --    shutdown functions needed to Terminate the TOT editor.
   --
   --CPM design notes:
   --    1.)  This module is called once, at the end.
   --    2.)  This module is expects the application to terminate the parent
   --    window.
   --
   --formal parameters
   --       None.
   --end formal parameters;
   --

end TOT_EDITOR;
```

APPENDIX C - C BINDING SPECIFICATIONS

The appendix contains the package specifications for binding Ada to equivalent C routines in C libraries. The following package specifications are included in this appendix:

CIN_INTERNET_COMMUNICATIONS
CIW_IMAGE_WINDOW
CUX_UTIL
CWN_WINDOW_SYSTEM

```
--cpc package specification name:
--    CIN_INTERNET_COMMUNICATIONS
--
--cpc description:
--    CIN_INTERNET_COMMUNICATIONS CPC is a set of Utility communications
--    primitives, written in the "C" programming language, which allow
--    processes to communicate with each other using an InterNet protocol.
--    These primitives work both within one processor and over an ethernet
--    network.  This specification is what allows Ada to call, or bind,
--    these C modules.
--
--cpc design notes:
--    1.)  None.
--
--cpc package author:
--    Bruce J. Packard
--    Science Applications International Corporation (SAIC)
--    424 Delaware, Suite C-3
--    Leavenworth, KS  66048   (913) 651-7925
--

with SYSTEM;  use SYSTEM;

package CIN_INTERNET_COMMUNICATIONS is

-- ##############################################################################
   procedure CIN_CLIENT_CONNECT_SERVER (HOST_ID:       in  ADDRESS;
                                        SERVICE_ID:    in  ADDRESS;
                                        MSTR_SOCK_NUM: in  ADDRESS);
   --
   --CPM description:
   --    This module allows a Client (user process) to connect to the
   --    InterNet master (Server) socket, returning the master socket number.
   --
   --CPM design notes:
   --    1.)  None.
   --
   --formal parameters
   --IN     HOST_ID       - A string which the environment equates to the
   --                       name (Id) of the Host (server) machine.
   --IN     SERVICE_ID    - A string which the environment equates to the
   --                       Service Id (INET port number).
   --OUT    MSTR_SOCK_NUM - A pointer to the server (Master) Socket Number.
   --end formal parameters;

-- ##############################################################################
   procedure CIN_CLOSE_SOCKET (CSN_INDEX:       in  ADDRESS;
                               CLIENT_SOCK_NUM: in  ADDRESS;
                               CLIENT_DISP_NUM: in  ADDRESS;
                               NUM_CLIENTS:     in  ADDRESS);
   --
   --CPM description:
   --    This module Closes the specified Internet client Socket and removse
   --    it from the list of client sockets.
   --
   --CPM design notes:
```

```
    --
    --formal parameters
    --IN     CSN_INDEX       - The  array Index of the Client Socket Number
    --                         being closed.
    --I/O    CLIENT_SOCK_NUM - The list of Client Socket Numbers.
    --I/O    CLIENT_DISP_NUM - The list of Client Display Numbers.  This is
    --                         machine number of the corresponding client
    --                         socket number.
    --I/O    NUM_CLIENTS     - The pointer to the actual Number of Client
    --                         sockets currently in the system.
    --end formal parameters;

-- ##############################################################################
   procedure CIN_ESTABLISH_SERVER (HOST_ID       : in  ADDRESS;
                                   SERVICE_ID    : in  ADDRESS;
                                   MSTR_SOCK_NUM : in  ADDRESS);
    --
    --CPM description:
    --    This module sets up and opens an InterNet Server returning the
    --    master socket number.
    --
    --CPM design notes:
    --    1.)  None.
    --
    --formal parameters
    --IN     HOST_ID       - A string which the environment equates to the
    --                       name (Id) of the Host (server) machine.
    --IN     SERVICE_ID    - A string which the environment equates to the
    --                       Service Id (INET port number).
    --OUT    MSTR_SOCK_NUM - A pointer to the server (Master) Socket Number.
    --end formal parameters;

-- ##############################################################################
   procedure CIN_FLUSH_MSG (SOCK_NUM    : in  ADDRESS;
                            FLUSH_LEN   : in  ADDRESS;
                            FLUSH_ERROR : in  ADDRESS);
    --
    --CPM description:
    --    This module Flushes a Message from the InterNet buffer system.
    --
    --CPM design notes:
    --    1.)  None.
    --
    --formal parameters
    --IN     SOCK_NUM    - The Socket Number to read from.
    --OUT    FLUSH_LEN   - The length of the message flushed if it worked, and
    --                      the error number if the flush failed.
    --OUT    FLUSH_ERROR - A pointer to a logical flag which tells if there
    --                      was an Error while flushing.
    --                    = TRUE  - There was an error trying to flush.
    --                    = FALSE - There were no errors in the flush.
    --end formal parameters;

-- ##############################################################################
   procedure CIN_RECV_MSG (PEEK_FLAG     : in  ADDRESS;
                           FROM_SOCK_NUM : in  ADDRESS;
```

```
                              MSG_LEN       : in  ADDRESS;
                              MSG           : in  ADDRESS;
                              ERROR_CODE    : in  ADDRESS);
   --
   --CPM description:
   --    This module sneaks a peek at, or Receives a Message which is being
   --    buffered in the InterNet system.
   --
   --CPM design notes:
   --    1.)  None.
   --
   --formal parameters
   --IN     PEEK_FLAG     - A Flag which tells this module whether to actually
   --                       receive the message or just "peek" at the first
   --                       "msg_len" bytes.
   --                     = TRUE  - just peek at the message.
   --                     = FALSE - read the entire message.
   --IN     FROM_SOCK_NUM - The Socket Number to read From.
   --I/O    MSG_LEN       - The number of bytes to read, or peek at, on the way
   --                       in and the number of bytes received, or the error
   --                       number if the received failed, on the way out.
   --OUT    MSG           - The Message received.
   --OUT    ERROR_CODE    - A pointer to a logical flag which tells if an
   --                       Error Code was encountered on the received.
   --                     = TRUE  - There was an error trying to receive.
   --                     = FALSE - There were no errors in the receive.
   --end formal parameters;

-- ##############################################################################
   procedure CIN_SEND_MSG (TO_SOCK_NUM : in  ADDRESS;
                           MSG         : in  ADDRESS;
                           MSG_LEN     : in  ADDRESS;
                           ERROR_CODE  : in  ADDRESS);
   --
   --CPM description:
   --    This module Sends a Message across the InterNet system.
   --
   --CPM design notes:
   --    1.)  None.
   --
   --formal parameters
   --IN     TO_SOCK_NUM - The Socket Number to write To.
   --IN     MSG         - The Message to write.
   --I/O    MSG_LEN     - The number of bytes to write on the way in and the
   --                     number of bytes written, or the error number if the
   --                     received failed, on the way out.
   --OUT    ERROR_CODE  - A pointer to a logical flag which tells if an
   --                     Error Code was encountered on the send.
   --                   = TRUE  - There was an error trying to send.
   --                   = FALSE - There were no errors in the send.
   --end formal parameters;

-- ##############################################################################
   procedure CIN_SERVER_CONNECT_CLIENT (MSTR_SOCK_NUM   : in  ADDRESS;
                                        MAX_CLIENTS     : in  ADDRESS;
                                        NUM_CLIENTS     : in  ADDRESS;
```

```
                                        CLIENT_SOCK_NUM : in  ADDRESS;
                                        CLIENT_DISP_NUM : in  ADDRESS);
    --
    --CPM description:
    --    This module allows the Server to Connect (accept) a client socket,
    --    returning the socket number.
    --
    --CPM design notes:
    --    1.)  None.
    --
    --formal parameters
    --IN     MSTR_SOCK_NUM    - The server (Master) Socket Number.
    --IN     MAX_CLIENTS      - The Maximum number of Clients allowed in the
    --                          system.
    --I/O    NUM_CLIENTS      - A pointer to the actual Number of Client sockets
    --                          currently in the system.
    --OUT    CLIENT_SOCK_NUM - The list of Client Socket Numbers.
    --OUT    CLIENT_DISP_NUM - The list of Display Numbers for each Client,
    --                          related to the corresponding "client_sock_num".
    --end formal parameters;

-- ##############################################################################
   procedure CIN_SERVER_WAIT (MSTR_SOCK_NUM    : in  ADDRESS;
                              NUM_CLIENTS      : in  ADDRESS;
                              CLIENT_SOCK_NUM  : in  ADDRESS;
                              CALLING_SOCK_NUM : in  ADDRESS;
                              SOCKET_INDEX     : in  ADDRESS);
   --
   --CPM description:
   --    This module causes the Server program to Wait for a response from
   --    one of the clients on the InterNet.
   --
   --CPM design notes:
   --    1.)  None.
   --
   --formal parameters
   --IN     MSTR_SOCK_NUM    - The server (Master) Socket Number.
   --IN     NUM_CLIENTS      - The actual Number of Client sockets currently
   --                          in the system.
   --IN     CLIENT_SOCK_NUM  - The list of Client Socket Numbers.
   --OUT    CALLING_SOCK_NUM - A pointer to the Number of the Socket who just
   --                          called the server.
   --OUT    SOCKET_INDEX     - A pointer to the Client Socket Number array
   --                          Index, for the client who just called.
   --end formal parameters;


private

   -- Communications utilities implemented in C

   pragma INTERFACE (C, CIN_CLIENT_CONNECT_SERVER);
   pragma INTERFACE (C, CIN_CLOSE_SOCKET);
   pragma INTERFACE (C, CIN_ESTABLISH_SERVER);
   pragma INTERFACE (C, CIN_FLUSH_MSG);
   pragma INTERFACE (C, CIN_RECV_MSG);
```

```
   pragma INTERFACE (C, CIN_SEND_MSG);
   pragma INTERFACE (C, CIN_SERVER_CONNECT_CLIENT);
   pragma INTERFACE (C, CIN_SERVER_WAIT);

end CIN_INTERNET_COMMUNICATIONS;
```

```
--CPC package specification name:
--     CIW_IMAGE_WINDOW
--
--CPC description:
--     CIW_IMAGE_WINDOW CPC is a set of color graphics primitives, written in
--     the "C" programming language, which allow programs to perform color
--     imaging functions within X windows.  This specification is what allows
--     Ada to call, or bind, these C modules.
--
--CPC design notes:
--     1.)  None.
--
--CPC package author:
--     Bruce J. Packard
--     Science Applications International Corporation (SAIC)
--     424 Delaware, Suite C-3
--     Leavenworth, KS  66048   (913) 651-7925
--
with SYSTEM;                          use SYSTEM;
with SYSTEM_PACKAGE;                  use SYSTEM_PACKAGE;

package CIW_IMAGE_WINDOW is

-- ##############################################################################
   procedure CIW_CREATE_PIXMAP (SIZE_X    : in  ADDRESS;
                                SIZE_Y    : in  ADDRESS;
                                BIT_IMAGE : in  ADDRESS;
                                COLOR     : in  ADDRESS;
                                PIXMAP_ID : in  ADDRESS);
   --
   --CPM description:
   --     This module Creates a Pixmap out of bitmapped data.
   --
   --CPM design notes:
   --     1.)  The bit image must be in memory order (Bits 0 - 15) for each 16
   --     bit word.
   --     2.)  The pixmap is displayed and erased with CIW_DISPLAY_BIT_IMAGE.
   --     3.)  The pixmap must be removed from memory with CIW_FREE_PIXMAP, when
   --     the pixel image is no longer required (see CIW_FREE_PIXMAP).
   --
   --formal parameters
   --IN      SIZE_X    - The Size of the image in the X direction.
   --IN      SIZE_Y    - The Size of the image in the Y direction.
   --IN      BIT_IMAGE - The Bit Image to transform.  The image is organized in
   --                    rows from the top to the bottom.  Each row contains
   --                    "SIZE_X" bits and there are "SIZE_Y" rows in the image.
   --IN      COLOR     - The index into the color lookup table for the Color
   --                    assigned to the on bits in this pixmap.
   --OUT     PIXMAP_ID - The Id assigned to this Pixmap.  This id is required
   --                    for displaying and freeing the pixmap.
   --end formal parameters;
   --

-- ##############################################################################
   procedure CIW_DISPLAY_BIT_IMAGE (WINDOW_ID       : in  ADDRESS;
                                    SUB_ADD_FLAG    : in  ADDRESS;
```

```
                                    DISPLAY_FUNTION : in  ADDRESS;
                                    PIXEL_UL_X      : in  ADDRESS;
                                    PIXEL_UL_Y      : in  ADDRESS;
                                    SIZE_X          : in  ADDRESS;
                                    SIZE_Y          : in  ADDRESS;
                                    PIXMAP_ID       : in  ADDRESS;
                                    PLANE_MASK      : in  ADDRESS);
    --
    --CPM description:
    --    This module Displays or erases a Bit Image (pixmap) in the
    --    specified planes.
    --
    --CPM design notes:
    --    1.)  The pixmap is created by CIW_CREATE_PIXMAP.
    --
    --formal parameters
    --IN     WINDOW_ID       - The Id of the Window to display the image in.  It
    --                         can be obtained by calling UWM_QUERY_WINDOW_ID.
    --IN     SUB_ADD_FLAG    - Image Subtraction or Addition Flag.  During
    --                         subtraction, the bits set in the raster image
    --                         shall be subtracted out of the selected planes.
    --                         During addition, the bits set in the raster image
    --                         shall be added into the selected planes.
    --                       = 0 - Subtract the image.
    --                       = 1 - Add the image.
    --IN     DISPLAY_FUNTION - The means of adding/subtracting the image to the
    --                         displayed image (and, or, copy...).
    --IN     PIXEL_UL_X      - The window X coordinate of the Upper Left corner
    --                         of the image.
    --IN     PIXEL_UL_Y      - The window Y coordinate of the Upper Left corner
    --                         of the image.
    --IN     SIZE_X          - The Size of the image in the X direction.
    --IN     SIZE_Y          - The Size of the image in the Y direction.
    --IN     PIXMAP_ID       - The Pixmap Id returned from CIW_CREATE_PIXMAP.
    --IN     PLANE_MASK      - A bit map representation of the Planes to be
    --                         affected by the image.  Value can be obtained from
    --                         "CIW_PLANE_MASK".
    --end formal parameters;
    --

-- ##############################################################################
    procedure CIW_DISPLAY_CIRCLE (WINDOW_ID    : in  ADDRESS;
                                  SUB_ADD_FLAG : in  ADDRESS;
                                  CENTER_X     : in  ADDRESS;
                                  CENTER_Y     : in  ADDRESS;
                                  RADIUS       : in  ADDRESS;
                                  COLOR        : in  ADDRESS;
                                  PLANE_MASK   : in  ADDRESS);
    --
    --CPM description:
    --    This module Displays or erases a Circle in the specified planes.
    --
    --CPM design notes:
    --    1.)  None.
    --
    --formal parameters
```

```
--IN     WINDOW_ID     - The Id of the Window to display the circle in.  It
--                       can be obtained by calling UWM_QUERY_WINDOW_ID.
--IN     SUB_ADD_FLAG - Image Subtraction or Addition Flag.  During
--                       subtraction, the bits set in the raster image shall
--                       be subtracted out of the selected planes.  During
--                       addition, the bits set in the raster image shall be
--                       added into the selected planes.
--                     = 0 - Subtract the circle.
--                     = 1 - Add the cirlce.
--IN     CENTER_X      - The window X coordinate of the Center of the circle.
--IN     CENTER_Y      - The window Y coordinate of the Center of the circle.
--IN     RADIUS        - The Radius of the circle, in pixels.
--IN     COLOR         - The index into the color lookup table for the Color
--                       of the circle.
--IN     PLANE_MASK    - A bit map representation of the Planes to be
--                       affected by the circle.  Value can be obtained from
--                       "CIW_PLANE_MASK".
--end formal parameters;
--

-- ##############################################################################
   procedure CIW_DISPLAY_IMAGE (WINDOW_ID       : in  ADDRESS;
                                BITS_DEEP       : in  ADDRESS;
                                SUB_ADD_FLAG    : in  ADDRESS;
                                DISPLAY_FUNTION : in  ADDRESS;
                                PIXEL_UL_X      : in  ADDRESS;
                                PIXEL_UL_Y      : in  ADDRESS;
                                SIZE_X          : in  ADDRESS;
                                SIZE_Y          : in  ADDRESS;
                                IMAGE           : in  ADDRESS;
                                PLANE_MASK      : in  ADDRESS);
   --
   --CPM description:
   --    This module Displays or erases a raster image in the specified planes.
   --
   --CPM design notes:
   --    1.)  Image depths (BITS_DEEP) of 1 should use CIW_DISPLAY_BIT_IMAGE.
   --    2.)  The only image depth (BITS_DEEP) currently supported is 8.
   --
   --formal parameters
   --IN     WINDOW_ID       - The Id of the Window to display the image in.  It
   --                         can be obtained by calling UWM_QUERY_WINDOW_ID.
   --IN     BITS_DEEP       - The Depth of each pixel value in the raster image.
   --                       = 8 - Byte image.
   --IN     SUB_ADD_FLAG    - Image Subtraction or Addition Flag.  During
   --                         subtraction, the bits set in the raster image
   --                         shall be subtracted out of the selected planes.
   --                         During addition, the bits set in the raster image
   --                         shall be added into the selected planes.
   --                       = 0 - Subtract the image.
   --                       = 1 - Add the image.
   --IN     DISPLAY_FUNTION - The means of adding/subtracting the image to the
   --                         displayed image (and, or, copy...).
   --IN     PIXEL_UL_X      - The window X coordinate of the Upper Left corner
   --                         of the image.
   --IN     PIXEL_UL_Y      - The window Y coordinate of the Upper Left corner
```

```
--                              of the image.
--IN     SIZE_X           - The size of the image in the X direction.
--IN     SIZE_Y           - The size of the image in the Y direction.
--IN     IMAGE            - The raster Image to display/erase.  The image is
--                          organized in rows from the top to the bottom.
--                          Each row contains "SIZE_X" elements and there are
--                          "SIZE_Y" rows in the image.  Each element of the
--                          image occupies "BITS_DEEP" bits.
--IN     PLANE_MASK       - A bit map representation of the Planes to be
--                          affected by the image.  Value can be obtained from
--                          "CIW_PLANE_MASK".
--end formal parameters;
--

-- ##############################################################################
   procedure CIW_DISPLAY_LINE (WINDOW_ID     : in  ADDRESS;
                               SUB_ADD_FLAG  : in  ADDRESS;
                               LINE_START_X  : in  ADDRESS;
                               LINE_START_Y  : in  ADDRESS;
                               LINE_END_X    : in  ADDRESS;
                               LINE_END_Y    : in  ADDRESS;
                               COLOR         : in  ADDRESS;
                               PLANE_MASK    : in  ADDRESS);
   --
   --CPM description:
   --    This module Displays or erases a Line in the specified planes.
   --
   --CPM design notes:
   --    1.)  None.
   --
   --formal parameters
   --IN     WINDOW_ID     - The Id of the Window to display the line in.  It
   --                       can be obtained by calling UWM_QUERY_WINDOW_ID.
   --IN     SUB_ADD_FLAG  - Image Subtraction or Addition Flag.  During
   --                       subtraction, the bits set in the raster image shall
   --                       be subtracted out of the selected planes.  During
   --                       addition, the bits set in the raster image shall be
   --                       added into the selected planes.
   --                     = 0 - Subtract the line.
   --                     = 1 - Add the line.
   --IN     LINE_START_X  - The window X coordinate of the Start of the Line.
   --IN     LINE_START_Y  - The window Y coordinate of the Start of the Line.
   --IN     LINE_END_X    - The window X coordinate of the End of the Line.
   --IN     LINE_END_Y    - The window Y coordinate of the End of the Line.
   --IN     COLOR         - The index into the color lookup table for the Color
   --                       of the line.
   --IN     PLANE_MASK    - A bit map representation of the Planes to be
   --                       affected by the line.  Value can be obtained from
   --                       "CIW_PLANE_MASK".
   --end formal parameters;
   --

-- ##############################################################################
   procedure CIW_DISPLAY_LINES (WINDOW_ID     : in  ADDRESS;
                                SUB_ADD_FLAG  : in  ADDRESS;
                                X_POINTS      : in  ADDRESS;
```

```
                                Y_POINTS      : in  ADDRESS;
                                NUMBER_POINTS : in  ADDRESS;
                                COLOR         : in  ADDRESS;
                                PLANE_MASK    : in  ADDRESS);
    --
    --CPM description:
    --    This module Displays or erases contiguous Line segments in the
    --    specified planes.
    --
    --CPM design notes:
    --    1.)  This module will draw single or multiple line segments.
    --
    --formal parameters
    --IN     WINDOW_ID     - The Id of the Window to display the lines in.  It
    --                      can be obtained by calling UWM_QUERY_WINDOW_ID.
    --IN     SUB_ADD_FLAG  - Image Subtraction or Addition Flag.  During
    --                      subtraction, the bits set in the raster image shall
    --                      be subtracted out of the selected planes.  During
    --                      addition, the bits set in the raster image shall be
    --                      added into the selected planes.
    --                     = 0 - Subtract the lines.
    --                     = 1 - Add the lines.
    --IN     X_POINTS      - The list of window X coordinate Points in the
    --                      contiguous line segments.
    --IN     Y_POINTS      - The list of window Y-coordinate Points in the
    --                      contiguous line segments.
    --IN     NUMBER_POINTS - The Number of Points in the list.  This will
    --                      produce (number_points - 1) line segments.
    --                   >= 2 and fit in a 32 bit integer.
    --IN     COLOR         - The index into the color lookup table for the Color
    --                      of the lines.
    --IN     PLANE_MASK    - A bit map representation of the Planes to be
    --                      affected by the line.  Value can be obtained from
    --                      "CIW_PLANE_MASK".
    --end formal parameters;
    --

-- ##########################################################################
   procedure CIW_DISPLAY_SYMBOL (WINDOW_ID     : in  ADDRESS;
                                 FONT_ID       : in  ADDRESS;
                                 SUB_ADD_FLAG  : in  ADDRESS;
                                 PIXEL_COLUMN  : in  ADDRESS;
                                 PIXEL_ROW     : in  ADDRESS;
                                 SYMBOL_VALUE  : in  ADDRESS;
                                 COLOR         : in  ADDRESS;
                                 PLANE_MASK    : in  ADDRESS);
    --
    --CPM description:
    --    This module Displays or erases a font Symbol in the specified planes.
    --
    --CPM design notes:
    --    1.)  The font must be initialized with CIW_INIT_FONT before an element
    --    can be displayed.
    --
    --formal parameters
    --IN     WINDOW_ID     - The Id of the Window to display the symbol in.  It
```

```
--                              can be obtained by calling UWM_QUERY_WINDOW_ID.
--IN     FONT_ID          - The Id of the symbol Font.  Value is output from
--                              "CIW_INIT_FONT".
--IN     SUB_ADD_FLAG     - Image Subtraction or Addition Flag.  During
--                              subtraction, the bits set in the raster image shall
--                              be subtracted out of the selected planes.  During
--                              addition, the bits set in the raster image shall be
--                              added into the selected planes.
--                            = 0 - Subtract the symbol.
--                            = 1 - Add the symbol.
--IN     PIXEL_COLUMN     - The Pixel Column of the upper left corner of the
--                              symbol.
--IN     PIXEL_ROW        - The Pixel Row of the upper left corner of the
--                              symbol.
--IN     SYMBOL_VALUE     - The integer Value of the Symbol to be displayed.
--IN     COLOR            - The index into the color lookup table for the Color
--                              of the symbol.
--IN     PLANE_MASK       - A bit map representation of the Planes to be
--                              affected by the symbol.  Value can be obtained from
--                              "CIW_PLANE_MASK".
--end formal parameters;
--

-- ##############################################################################
   procedure CIW_DISPLAY_TEXT  WINDOW_ID     : in  ADDRESS;
                               FONT_ID       : in  ADDRESS;
                               SUB_ADD_FLAG  : in  ADDRESS;
                               PIXEL_COLUMN  : in  ADDRESS;
                               PIXEL_ROW     : in  ADDRESS;
                               TEXT_STRING   : in  ADDRESS;
                               COLOR         : in  ADDRESS;
                               PLANE_MASK    : in  ADDRESS);
   --
   --CPM description:
   --    This module Displays or erases a Text string in the specified planes.
   --
   --CPM design notes:
   --    1.)  The font must be initialized with CIW_INIT_FONT before a string
   --    can be displayed.
   --
   --formal parameters
   --IN     WINDOW_ID        - The Id of the Window to display the text string in.
   --                            It can be obtained by calling UWM_QUERY_WINDOW_ID.
   --IN     FONT_ID          - The Id of the text Font.  Value is output from
   --                            "CIW_INIT_FONT".
   --IN     SUB_ADD_FLAG     - Image Subtraction or Addition Flag.  During
   --                            subtraction, the bits set in the raster image shall
   --                            be subtracted out of the selected planes.  During
   --                            addition, the bits set in the raster image shall be
   --                            added into the selected planes.
   --                          = 0 - Subtract the text.
   --                          = 1 - Add the text.
   --IN     PIXEL_COLUMN     - The Pixel Column of the upper left corner of the
   --                            text.
   --IN     PIXEL_ROW        - The Pixel Row of the upper left corner of the text.
   --IN     TEXT_STRING      - The String of Text to be displayed.
```

```
    --IN     COLOR          - The index into the color lookup table for the Color
    --                        of the text string.
    --IN     PLANE_MASK     - A bit map representation of the Planes to be
    --                        affected by the text string.  Value can be obtained
    --                        from "CIW_PLANE_MASK".
    --end formal parameters;
    --

-- ##############################################################################
   procedure CIW_ERASE_PLANES (WINDOW_ID  : in  ADDRESS;
                               PIXEL_UL_X : in  ADDRESS;
                               PIXEL_UL_Y : in  ADDRESS;
                               SIZE_X     : in  ADDRESS;
                               SIZE_Y     : in  ADDRESS;
                               PLANE_MASK : in  ADDRESS);
    --
    --CPM description:
    --    This module Erases everything in a given rectangular image out of the
    --    specified Planes.
    --
    --CPM design notes:
    --    1.)  None.
    --
    --formal parameters
    --IN     WINDOW_ID  - The Id of the Window to erase the planes in.  It can
    --                    be obtained by calling UWM_QUERY_WINDOW_ID.
    --IN     PIXEL_UL_X - The window X coordinate of the Upper Left corner of
    --                    the image.
    --IN     PIXEL_UL_Y - The window Y coordinate of the Upper Left corner of
    --                    the image.
    --IN     SIZE_X     - The Size of the image in the X direction.
    --IN     SIZE_Y     - The Size of the image in the Y direction.
    --IN     PLANE_MASK - A bit map representation of the Planes to be affected
    --                    by the image.  Value can be obtained from
    --                    "CIW_PLANE_MASK".
    --end formal parameters;
    --

-- ##############################################################################
   procedure CIW_FLUSH_BUFFER;
    --
    --CPM description:
    --    This module Flushes the graphics command Buffer.
    --
    --CPM design notes:
    --    1.)  X Windows buffers its commands and flushes that buffer after
    --    certain commands or when the buffer is full.  Therefore this module
    --    only needs to be called when a previous command must be seen
    --    immediately.
    --
    --formal parameters
    --       None
    --end formal parameters;
    --

-- ##############################################################################
```

```
    procedure CIW_FREE_PIXMAP (PIXMAP_ID : in  ADDRESS);
    --
    --CPM description:
    --    This module Frees up the memory allocated to a Pixmap back in
    --    CIW_CREATE_PIXMAP.
    --
    --CPM design notes:
    --    1.)  In EDDIC the contours pixmaps should be freed after each block is
    --    displayed, but the unit symbology pixmaps can be defined once and left
    --    for the duration of the run.
    --
    --formal parameters
    --IN     PIXMAP_ID - The Pixmap Id returned from CIW_CREATE_PIXMAP.
    --end formal parameters;
    --

-- ##############################################################################
   procedure CIW_INIT_FONT (FONT_NAME   : in  ADDRESS;
                            FONT_ID     : in  ADDRESS;
                            FONT_HEIGHT : in  ADDRESS;
                            FONT_WIDTH  : in  ADDRESS);
   --
   --CPM description:
   --    This module Initializes a specified Font.
   --
   --CPM design notes:
   --    1.)  Fonts are only initialized once.
   --    2.)  It is legal to have multiple fonts in a single process.
   --
   --formal parameters
   --IN     FONT_NAME   - The string containing the Font's directory and Name.
   --OUT    FONT_ID     - The Id of the Font as returned by the X system.
   --OUT    FONT_HEIGHT - The Height, in pixels, of a Font character.
   --OUT    FONT_WIDTH  - The Width, in pixels, of a Font character.
   --end formal parameters;
   --

-- ##############################################################################
   procedure CIW_INIT_LOOKUP_TABLE (MAX_PLANES : in  ADDRESS);
   --
   --CPM description:
   --    This module Initializes (allocates space for) the color Lookup Table.
   --
   --CPM design notes:
   --    1.)  The lookup table is only initialized once.
   --
   --formal parameters
   --IN     MAX_PLANES - The Maximum number of color Planes currently allowed
   --                   in the system.
   --end formal parameters;
   --

-- ##############################################################################
   procedure CIW_LOAD_LOOKUP_TABLE (LUT_INDEX    : in  ADDRESS;
                                    RED_INTENS   : in  ADDRESS;
                                    GREEN_INTENS : in  ADDRESS;
```

```
                                        BLUE_INTENS  : in  ADDRESS);
    --
    --CPM description:
    --    This module Loads color values into the color Lookup Table.
    --
    --CPM design notes:
    --    1.)  The display is not altered by calling this module; the display
    --    is altered by calling CIW_STORE_LOOKUP_TABLE.
    --
    --formal parameters
    --IN    LUT_INDEX    - The Index into the Lookup Table to load.  Zero is
    --                     the first cell in the lookup table.
    --IN    RED_INTENS   - The Intensity for Red.
    --IN    GREEN_INTENS - The Intensity for Green.
    --IN    BLUE_INTENS  - The Intensity for Blue.
    --end formal parameters;
    --

-- ##############################################################################
    procedure CIW_MOVE_IMAGE (WINDOW_ID      : in  ADDRESS;
                              OLD_PIXEL_UL_X : in  ADDRESS;
                              OLD_PIXEL_UL_Y : in  ADDRESS;
                              NEW_PIXEL_UL_X : in  ADDRESS;
                              NEW_PIXEL_UL_Y : in  ADDRESS;
                              SIZE_X         : in  ADDRESS;
                              SIZE_Y         : in  ADDRESS);
    --
    --CPM description:
    --    This module Moves a raster Image from one location in a window to
    --    another location within the same window.
    --
    --CPM design notes:
    --    1.)  None.
    --
    --formal parameters
    --IN     WINDOW_ID      - The Id of the Window the image is in.  It can be
    --                        obtained by calling UWM_QUERY_WINDOW_ID.
    --IN     OLD_PIXEL_UL_X - The window X coordinate of the Upper Left corner
    --                        of the source image.
    --IN     OLD_PIXEL_UL_Y - The window Y coordinate of the Upper Left corner
    --                        of the source image.
    --IN     NEW_PIXEL_UL_X - The window X coordinate of the Upper Left corner
    --                        of the destination image.
    --IN     NEW_PIXEL_UL_Y   The window Y coordinate of the Upper Left corner
    --                        of the destination image.
    --IN     SIZE_X         - The size of the image in the X direction.
    --IN     SIZE_Y         - The size of the image in the Y direction.
    --end formal parameters;
    --

-- ##############################################################################
    procedure CIW_PLANE_MASK (START_PLANE : in  ADDRESS;
                              END_PLANE   : in  ADDRESS;
                              PLANE_MASK  : in  ADDRESS);
    --
    --CPM description:
```

```
--     This module calculates a bit map representation (Mask) of the Planes
--      requested by the user for later use.
--
--CPM design notes:
--     1.)  None.
--
--formal parameters
--IN      START_PLANE - The Plane number of the lowest plane to be affected
--                      by the image.  Bit 1 of the raster image shall be
--                      loaded into this plane.  Plane numbers start at 1.
--IN      END_PLANE   - The Plane number of the highest plane to be affected
--                      by the image.  Image bits that are greater than
--                      (end_plane - start_plane + 1) shall be ignored.
--OUT     PLANE_MASK  - A bit map representation of the Planes which the
--                      user would like to affect in a future window call.
--end formal parameters;
--

-- ##############################################################################
   procedure CIW_RUBBERBAND_LINE (WINDOW_ID    : in  ADDRESS;
                                  FROM_POINT_X : in  ADDRESS;
                                  FROM_POINT_Y : in  ADDRESS;
                                  COLOR        : in  ADDRESS;
                                  PLANE_MASK   : in  ADDRESS;
                                  END_POINT_X  : in  ADDRESS;
                                  END_POINT_Y  : in  ADDRESS);
   --
   --CPM description:
   --     This module draws a Rubberband Line in the specified window from
   --     the specified point to the cursor and returns the end point selected
   --     by the user.
   --
   --CPM design notes:
   --     1.)  If the user moves the cursor outside the window and selects the
   --     point, the end point coordinates are the lines window boundry crossing.
   --     2.)  If the user moves the cursor outside the window and selects the
   --     point, the rubberband line is not drawn upon return.
   --
   --formal parameters
   --IN      WINDOW_ID    - The Id of the Window the line is in.  It can be
   --                       obtained by calling UWM_QUERY_WINDOW_ID.
   --IN      FROM_POINT_X - The window X coordinate of the Point the lines
   --                       rubberbanding emanates From.
   --IN      FROM_POINT_Y - The window Y coordinate of the Point the lines
   --                       rubberbanding emanates From.
   --IN      COLOR        - The index into the color lookup table for the Color
   --                       of the line.
   --IN      PLANE_MASK   - A bit map representation of the Planes to be
   --                       affected by the line.  Value can be obtained from
   --                       "CIW_PLANE_MASK".
   --OUT     END_POINT_X  - The window X coordinate of the lines End Point as
   --                       selected by the user.
   --OUT     END_POINT_Y  - The window Y coordinate of the lines End Point as
   --                       selected by the user.
   --end formal parameters;
```

```
-- ##############################################################################
   procedure CIW_STORE_LOOKUP_TABLE;
   --
   --CPM description:
   --    This module Stores the color Lookup Table.
   --
   --CPM design notes:
   --    1.)  Calling this module alters the display provided some of the
   --    values were changed with CIW_LOAD_LOOKUP_TABLE.
   --
   --formal parameters
   --       None.
   --end formal parameters;
   --

-- ##############################################################################
   procedure CUX_16BIT_SWAP (NUMBER_16BIT : in  INTEGER;
                             BIT_IMAGE    : in  ADDRESS);
   --
   --CPM description:
   --    This module Swaps the Bits of 16 bit words, an order X windows happens
   --    to prefer.
   --
   --CPM design notes:
   --    1.)  bit 0 -> bit 15;   bit 15 -> bit 0
   --         bit 1 -> bit 14;   bit 14 -> bit 1 ...
   --
   --formal parameters
   --IN     NUMBER_16BIT - The number of 16 bit words in the image.
   --I/O    BIT_IMAGE    - Buffer containing the bit image.
   --end formal parameters;
   --


private

   pragma INTERFACE (C, CIW_CREATE_PIXMAP);
   pragma INTERFACE (C, CIW_DISPLAY_BIT_IMAGE);
   pragma INTERFACE (C, CIW_DISPLAY_CIRCLE);
   pragma INTERFACE (C, CIW_DISPLAY_IMAGE);
   pragma INTERFACE (C, CIW_DISPLAY_LINE);
   pragma INTERFACE (C, CIW_DISPLAY_LINES);
   pragma INTERFACE (C, CIW_DISPLAY_SYMBOL);
   pragma INTERFACE (C, CIW_DISPLAY_TEXT);
   pragma INTERFACE (C, CIW_ERASE_PLANES);
   pragma INTERFACE (C, CIW_FLUSH_BUFFER);
   pragma INTERFACE (C, CIW_FREE_PIXMAP);
   pragma INTERFACE (C, CIW_INIT_FONT);
   pragma INTERFACE (C, CIW_INIT_LOOKUP_TABLE);
   pragma INTERFACE (C, CIW_MOVE_IMAGE);
   pragma INTERFACE (C, CIW_LOAD_LOOKUP_TABLE);
   pragma INTERFACE (C, CIW_PLANE_MASK);
   pragma INTERFACE (C, CIW_RUBBERBAND_LINE);
   pragma INTERFACE (C, CIW_STORE_LOOKUP_TABLE);
   pragma INTERFACE (C, CUX_16BIT_SWAP);
```

```
end CIW_IMAGE_WINDOW;
```

```
--CPC package specification name:
--     CUX_UTIL
--
--CPC description:
--     CUX_UTIL CPC is a set of Utility primitives, written in the "C"
--     programming language, which allow programs to access UNIX operating
--     system commands.  This specification is what allows Ada to call, or bind,
--     these C modules.
--
--CPC design notes:
--     1.)  None.
--
--CPC package author:
--     Bruce J. Packard
--     Science Applications International Corporation (SAIC)
--     424 Delaware, Suite C-3
--     Leavenworth, KS  66048   (913) 651-7925
--

with SYSTEM;           use SYSTEM;
with SYSTEM_PACKAGE;   use SYSTEM_PACKAGE;

package CUX_UTIL is

-- ##############################################################################
   procedure CUX_BINARY_READ (FILE_DESC     : in  ADDRESS;
                              OFFSET        : in  ADDRESS;
                              RECORD_LENGTH : in  ADDRESS;
                              FORMAT        : in  ADDRESS;
                              BUFFER        : in  ADDRESS);
   --
   --CPM description:
   --     This module performs a binary (unformatted) read on a specific record
   --     of the specified file, which was opened by CUX_OPEN_FILE.
   --
   --CPM design notes:
   --     1.)  None.
   --
   --formal parameters
   --IN      FILE_DESC     - A pointer to the file descriptor returned from
   --                        CUX_OPEN_FILE.
   --IN      OFFSET        - The offset from the beginning of the file (Starts
   --                        at one).  For fixed length record files the offset
   --                        units are records.  For variable length record
   --                        files the offset units are bytes.
   --IN      RECORD_LENGTH - Number of bytes in this record to be read.
   --IN      FORMAT        - File format.
   --                      = 0 - Fixed length records.
   --                      = 1 - Variable length records.
   --OUT     BUFFER        - Pointer to the Buffer that was read.
   --end formal parameters;

-- ##############################################################################
   procedure CUX_BINARY_WRITE (FILE_DESC     : in  ADDRESS;
                               OFFSET        : in  ADDRESS;
                               RECORD_LENGTH : in  ADDRESS;
```

```
                                FORMAT          : in  ADDRESS;
                                BUFFER          : in  ADDRESS);
    --
    --CPM description:
    --    This module performs a binary (unformatted) write on a specific record
    --    of the specified file, which was opened by CUX_OPEN_FILE.
    --
    --CPM design notes:
    --    1.)  None.
    --
    --formal parameters
    --IN     FILE_DESC     - A pointer to the file descriptor returned from
    --                       CUX_OPEN_FILE.
    --IN     OFFSET        - The offset from the beginning of the file (Starts
    --                       at one).  For fixed length record files the offset
    --                       units are records.  For variable length record
    --                       files the offset units are bytes.
    --IN     RECORD_LENGTH - Number of bytes in this record to be written.
    --IN     FORMAT        - File format.
    --                     = 0 - Fixed length records.
    --                     = 1 - Variable length records.
    --IN     BUFFER        - Pointer to the Buffer to write to.
    --end formal parameters;

-- ##############################################################################
   procedure CUX_CLOSE_FILE (FILE_DESC : in  ADDRESS);
   --
   --CPM description:
   --    This module closes a file opened by CUX_OPEN_FILE.
   --
   --CPM design notes:
   --    1.)  None.
   --
   --formal parameters
   --IN     FILE_DESC - A pointer to the file descriptor returned from
   --                   CUX_OPEN_FILE.
   --end formal parameters;

-- ##############################################################################
   procedure CUX_GETENV (ENV_STRING    : in  ADDRESS;
                         RESULT_STRING : in  ADDRESS);
   --
   --CPM description:
   --    This module searches the Unix Environment list and returns (Gets) the
   --    evaluated, requested string.
   --
   --CPM design notes:
   --    1.)  None.
   --
   --formal parameters
   --IN     ENV_STRING    - The string that was created by a setenv.
   --OUT    RESULT_STRING - The evaluated Environment String.
   --end formal parameters;

-- ##############################################################################
   procedure CUX_OPEN_FILE (FILE_NAME      : in  ADDRESS;
```

```
                              FILE_OPERATION : in  ADDRESS;
                              FILE_DESC      : in  ADDRESS);
   --
   --CPM description:
   --    This module opens a file for the performing of binary reads and writes.
   --
   --CPM design notes:
   --    1.)  None.
   --
   --formal parameters
   --IN     FILE_NAME       - The name of the file to be opened.
   --IN     FILE_OPERATION  - A flag that tells which Mode to Open the file.
   --                       = 0 - Read only.
   --                       = 1 - Read, write, and create if needed.
   --                       = 2 - Append.
   --OUT    FILE_DESC       - File descriptor assigned to the open file.
   --end formal parameters;

-- ##############################################################################
   procedure CUX_SETENV (ENV_STRING   : in  ADDRESS;
                         VALUE_STRING : in  ADDRESS);
   --
   --CPM description:
   --    This module sets a Unix Environment variable to the requested string.
   --
   --CPM design notes:
   --    1.)  None.
   --
   --formal parameters
   --IN     ENV_STRING    - The environment variable string name.
   --IN     VALUE_STRING  - The value to set the environment variable to.
   --end formal parameters;

-- ##############################################################################
   procedure CUX_SYSTEM (CMD_STRING : in  ADDRESS);
   --
   --CPM description:
   --    This module executes a Unix System call.
   --
   --CPM design notes:
   --    1.)  None.
   --
   --formal parameters
   --IN     CMD_STRING  - Command string to execute in the UNIX environment.
   --end formal parameters;

-- ##############################################################################
   procedure CUX_WAIT (SECONDS_TO_WAIT : in  ADDRESS;
                       SECONDS_WAITED  : in  ADDRESS);
   --
   --CPM description:
   --    This module suspends a process for a specified period of time.
   --
   --CPM design notes:
   --    1.)  None.
   --
```

```
   --formal parameters
   --IN     SECONDS_TO_WAIT  - The number of seconds to suspend the process.
   --OUT    SECONDS_WAITED   - The number of seconds actually suspend.
   --end formal parameters;

private

   pragma INTERFACE (C, CUX_BINARY_READ);
   pragma INTERFACE (C, CUX_BINARY_WRITE);
   pragma INTERFACE (C, CUX_CLOSE_FILE);
   pragma INTERFACE (C, CUX_GETENV);
   pragma INTERFACE (C, CUX_OPEN_FILE);
   pragma INTERFACE (C, CUX_SETENV);
   pragma INTERFACE (C, CUX_SYSTEM);
   pragma INTERFACE (C, CUX_WAIT);

end CUX_UTIL;
```

```
--cpc package specification name: CWN_WINDOW_SYSTEM
--
--cpc description: CWN_WINDOW_SYSTEM is the C version of the EDDIC window
--                 utilities using the X-window protocol. --
--cpc design notes:
--
--cpc package author: Bruce Packard
--                    Laura McClanahan
--                    Science Applications International Corporation
--                    424 Delaware, Suite C3
--                    Leavenworth, KS 66048
--
with SYSTEM;            use SYSTEM;
with SYSTEM_PACKAGE;    use SYSTEM_PACKAGE;

package CWN_WINDOW_SYSTEM is

   procedure CWN_ACTIVATE_EDITOR (EDITOR_ID:   in   ADDRESS);
   --
   -- CPM description:  This routine activates an existing editor.  It is
   --                   provided basically for traversing from a string field
   --                   or numeric field to an editor.
   --
   -- formal parameters
   --IN   EDITOR_ID   The id of the editor to activate.
   --
   -- end formal parameters;


   procedure CWN_ACTIVATE_MENU (MENU_STRUCT_ID:  in  ADDRESS;
                                MENU_INDEX:      in  ADDRESS;
                                WINDOW_TYPE:     in  ADDRESS;
                                WINDOW_ID:       in  ADDRESS);
   --
   -- CPM description: This routine activates an already defined popup menu for
   --                  either:
   --                     a. A defined window,
   --                     b. a displayed panel (via cwn_end_panel),
   --                     c. or, a defined button (via cwn_define_button).
   --                  It also specifies the mode for posting the menu.
   -- formal parameters
   --IN    MENU_STRUCT_ID   The id of the menu structure given at the time
   --                       of the menu definition.
   --
   --IN    MENU_INDEX       The index into the Text_Array of the submenu to
   --                       be activated for a particular window, if applicable.
   --                       If the menu to be activated is not a walking menu,
   --                       or is the top level of a walking menu, then this
   --                       parameter should be set to NULL.
   --
   --IN    WINDOW_TYPE      The type of window the menu will be activated for,
   --                       where:
   --                       SYS_WINDOW         = a defined window
   --                       SYS_DISPLAY_PANEL  = a displayed panel
   --                       SYS_DEFINED_BUTTON = defined button
   --
```

```
--IN    WINDOW_ID          The id given at the time of the window type's
--                         creation where:
--                         If window_type is SYS_WINDOW and window_id is 0,
--                         then the menu will be activated for the RootWindow
--                         or (Display).  Otherwise, the menu will be activated
--                         for the matching window_id.
--                         If window_type = SYS_DISPLAY_PANEL, the id should
--                         be the panel id.
--                         If window_type = SYS_DEFINED_BUTTON, the id should
--                         be the button id.
--
-- end formal parameters;


procedure CWN_ACTIVATE_NUMBER_FIELD (
                              NUMBER_FIELD_ID: in  ADDRESS);
--
-- CPM description: This routine activates an existing number field.  It is
--                  provided basically for traversing from one number field
--                  to another.
--
-- formal parameters
--IN    NUMBER_FIELD_ID  The id of the numeric field to activate.
--
--end formal parameters;


procedure CWN_ACTIVATE_STRING_FIELD (
                              STRING_FIELD_ID: in  ADDRESS);
--
-- CPM description: This routine activates an existing string field.  It is
--                  provided basically for traversing from one string field
--                  to another.
--
-- formal parameters
--IN    STRING_FIELD_ID  The id of the string field to activate.
--
--end formal parameters;


procedure CWN_ADD_INPUT_SOCKET (SOCKET_ID:  in   ADDRESS);
--
-- CPM description: CWN_ADD_INPUT_SOCKET adds a socket id to be watched by
--                  CWN_INPUT.  When a message is received on this socket,
--                  CWN_INPUT returns type SYS_INPUT_MESSAGE along with the
--                  socket ID. The applications software is responsible for
--                  reading the message.
--
-- formal parameters
--IN    SOCKET_ID        ID of the socket to watch for input.
-- end formal parameters;


procedure CWN_CHANGE_BUTTON_LABEL (BUTTON_ID:       in   ADDRESS;
                                   BUTTON_TEXT:     in   ADDRESS);
--
```

```
-- CPM description: CWN_CHANGE_BUTTON_LABEL changes the text displayed inside
--                  a button created with CWN_DEFINE_BUTTON.
--
-- formal parameters
--IN    BUTTON_ID        ID attached to the button.
--
--IN    BUTTON_TEXT      Textual string to display in the button.
-- end formal parameters;


procedure CWN_CHANGE_CHECKBOX_STATES (Checkbox_ID:   in   ADDRESS;
                                      Num_Fields:    in   ADDRESS;
                                      Start_Index:   in   ADDRESS;
                                      Status_Array:  in   ADDRESS;
                                      State_Flag:    in   ADDRESS);
--
-- CPM description:  CWN_CHANGE_CHECKBOX_STATES changes one or more
--                   checkbox states according to the input state flag.
--
-- formal parameters
--IN   Checkbox_ID       The ID attached to the checkbox editor.
--
--IN   Num_Fields        The number of checkbox(es) states to be changed.
--
--IN   Start_Index       The correlating index of the checkbox which the
--                       start of the array to the order the items were
--                       originally created; the first element is always
--                       zero.
--
--IN   Status_Array      The array of current status of the checkboxes to
--                       be changed.
--
--IN   State_Flag        The flag indicating the state all the checkboxes
--                       are to match.
-- end formal parameters;

procedure CWN_CHANGE_EDITOR_TEXT (EDITOR_ID:   in   ADDRESS;
                                  MAX_BUFFER_SIZE:   in   ADDRESS;
                                  TEXT_BUFFER:       in   ADDRESS;
                                  BUFFER_SIZE:       in   ADDRESS);
--
-- CPM description: Changes the text buffer used   window full page
--                  text editor.
--
-- formal parameters
--IN    EDITOR_ID        ID attached to the editor.
--
--IN    MAX_BUFFER_SIZE  Maximum number of pixels that the TEXT_BUFFER
--                       can hold.
--
--IN    TEXT_BUFFER      Buffer of the initial text to display in the editor.
--
--IN    BUFFER_SIZE      The number of pixels in TEXT_BUFFER.
-- end formal parameters;
```

```
procedure CWN_CHANGE_ICON_LABEL (ICON_LABEL:       in   ADDRESS);
--
-- CPM description: CWN_CHANGE_ICON_LABEL changes the icon label displayed
--                  in the window's icon.
--
-- formal parameters
--IN    ICON_LABEL        Textual string to display in the icon.
--                        Note: This must be a null terminated string of
--                        7 characters in length.
-- end formal parameters;


procedure CWN_CHANGE_SCROLLBAR (SCROLLBAR_ID:   in   ADDRESS;
                                DOC_SIZE:       in   ADDRESS;
                                PIXEL_LENGTH:   in   ADDRESS;
                                DISP_POSITION:  in   ADDRESS;
                                SCROLL_INTRVL:  in   ADDRESS);
--
-- CPM description: Changes the size of a scrollbar.
-- formal parameters
--IN    SCROLLBAR_ID      ID to attached to the scrollbar.
--                        This ID was defined by CWN_DEFINE_SCROLLBAR.
--
--IN    DOC_SIZE          The number of lines in the document buffer.
--
--IN    PIXEL_LENGTH      The number of pixels to be occupied
--                        scrollbar.
--IN    PIXEL_LENGTH      The number of pixels to be occupied
--                        scrollbar.
--IN    SCROLL_INTRVL     The number of pixels the work will be scrolled
--                        whenever the user selects an arrow button.  Note:
--                        The work will not be scrolled  se utilities
--                        but, this argument is required to calculate
--                        the interactive slidepositioning.
-- end formal parameters;


procedure CWN_CHANGE_WINDOW_LABEL (WINDOW_LABEL:       in   ADDRESS;
                                   LABEL_POSITION:     in   ADDRESS);
--
-- CPM description: CWN_CHANGE_WINDOW_LABEL changes the window label
--                  displayed in the window's top border.
--
-- formal parameters
--IN    WINDOW_LABEL        Textual string to display.
--                         Note: This must be a null terminated string of
--                         as described in SYS_WINDOW_NAME.
--IN    LABEL_POSITION      The position of the window title bar this
--                          label is changing as described in SYS_TEXT_
--                          ALIGNMENT.  An alignment of NONE will result
--                          in a change of the CENTER label.
-- end formal parameters;


procedure CWN_CLEAR_WINDOW;
--
```

```
-- CPM description: Erases all element inside a defined window.
--
-- formal parameters
--  None
-- end formal parameters;


procedure CWN_CLOSE_WINDOW;
--
-- CPM description:  Closes a window into an icon.
--
-- formal parameters
--  None
-- end formal parameters;


procedure CWN_CREATE_EXPOSURE_EVENT (WINDOW_ID:   in   ADDRESS);
--
-- CPM description:  This procedure creates an exposure event for a
--                   particular window.
--
-- formal parameters
--IN    WINDOW_ID    The ID attached to the window.
-- end formal parameters;


procedure CWN_CREATE_SUBWINDOW (WINDOW_ID:      in   ADDRESS;
                                MAP_WINDOW:     in   ADDRESS;
                                PIXEL_COL:      in   ADDRESS;
                                PIXEL_ROW:      in   ADDRESS;
                                PIXEL_WIDTH:    in   ADDRESS;
                                PIXEL_LENGTH:   in   ADDRESS;
                                BORDER_WIDTH:   in   ADDRESS;
                                SUBWINDOW_ID:   in   ADDRESS);
--
-- CPM description: This procedure creates a subwindow to the window
--                  specified by the user.  All input selected for the
--                  parent window and any menu activated for the parent
--                  window will be effective for the subwindow also, unless
--                  other input is selected or another menu activated
--                  specifically for this window.
--
-- formal parameters
--IN    WINDOW_ID         The id of the parent window.
--
--IN    MAP_WINDOW        Boolean indicating whether window should be mapped.
--
--IN    PIXEL_COL         Column number from within the window where the left
--                        side of the subwindow shall be placed. Column 0 is
--                        at the left of the window.
--
--IN    PIXEL_ROW         Row number from within the window where the top side
--                        of the subwindow shall be placed. Row 0 is at the
--                        top of the window.
--
--IN    PIXEL_WIDTH       The number of pixels to be occupied
```

```
--                          subwindow's width.
--
--IN    PIXEL_LENGTH         The number of pixels to be occupied
--                          subwindow's length.
--
--IN    BORDER_WIDTH         The width of the border in pixels.  If the border
--                          width is zero, the subwindow will not have a border.
--
--OUT   SUBWINDOW_ID         The id of the subwindow as given by the X window
--                          system.
-- end formal parameters;


procedure CWN_CREATE_WINDOW (WINDOW_ID:       in   ADDRESS;
                             WINDOW_LABEL:    in   ADDRESS;
                             MAP_WINDOW:      in   ADDRESS;
                             ICON_TYPE:       in   ADDRESS;
                             ICON_STACK_INDX: in   ADDRESS;
                             ICON_ID:         in   ADDRESS);
--
-- CPM description: Creates a basic window skeleton with border, title, icon
--                  and frame popup menu attached.  Only one window per
--                  process.
--
-- formal parameters
--OUT   WINDOW_ID          The id given the window.
--
--IN    WINDOW_LABEL       Textual string to be displayed in the window border.
--
--IN    MAP_WINDOW         Boolean indicating whether window should be mapped
--                         (Made visible upon creation).
--
--IN    ICON_TYPE          Identifies the icon stack that the new window is
--                         assigned to. 0 = Reference Icon
--                                      1 = View C & C Icon
--                                      2 = Process Messages Icon
--                                      3 = Build C & C Icon
--                                      4 = Decision Aids Icon
--                                      5 = Experiment Control Icon
--
--OUT   ICON_STACK_INDX    Position in the Icon stack of the newly created
--                         window (1 - 7);
--
--OUT   ICON_ID            The id of the icon window.
--
-- end formal parameters;


procedure CWN_DEACTIVATE_MENU (MENU_STRUCT_ID:  in  ADDRESS;
                               MENU_INDEX:      in  ADDRESS);
--
-- CPM description: This routine deactivates an already defined popup menu.
--
-- formal parameters
--IN    MENU_STRUCT_ID     The id of the menu structure given by the
--                         application at the time of the menu definition.
```

```
--
--IN     MENU_INDEX        The index into the Start_Array of the submenu to
--                         be deactivated for a particular window.
--                         If the menu to be activated is not a walking menu,
--                         or is the top level of a walking menu, then this
--                         parameter should be set to NULL.
--
-- end formal parameters;


procedure CWN_DEFINE_BUTTON (BUTTON_ID:       in   ADDRESS;
                             WINDOW_ID:       in   ADDRESS;
                             ENABLE_FLAG:     in   ADDRESS;
                             PIXEL_COL:       in   ADDRESS;
                             PIXEL_ROW:       in   ADDRESS;
                             PIXEL_WIDTH:     in   ADDRESS;
                             PIXEL_HEIGHT:    in   ADDRESS;
                             BUTTON_TEXT:     in   ADDRESS);
--
-- CPM description: Defines a button on top portion of a window. Once a
--                  button has been defined, only other buttons may be placed
--                  beside it.  All other structures must be placed below
--                  the buttons. These buttons are used mostly for initiating
--                  a walking menu (see CWN_ACTIVATE_MENU).
--
-- formal parameters
--OUT    BUTTON_ID         The ID attached to the defined button. This
--                         ID is required for all interactions with the button.
--
--IN     WINDOW_ID         The ID of the window to attach the button to.
--
--IN     ENABLE_FLAG       Logical flag to indicate if the button should be
--                         backlight when it is selected and the button ID will
--                         be returned to the application. The disabled mode is
--                         used to display a walking menu when the button is
--                         selected.
--                                     true  = ENABLED
--                                     false = DISABLED
--
--IN     PIXEL_COL         Column number from within the window where the left
--                         side of the button shall be placed. Column 0 is at
--                         left of the window.
--
--IN     PIXEL_ROW         Row number from within the window where the top side
--                         of the button shall be placed. Row 0 is at the top
--                         of the window.
--
--IN     PIXEL_WIDTH       The number of columns to be occupied   button.
--
--IN     PIXEL_HEIGHT      The number of rows to be occupied   button.
--
--IN     BUTTON_TEXT       Textual string to display in the button.
-- end formal parameters;


procedure CWN_DEFINE_CHECKBOX (
```

```
      EDITOR_ID:        in   ADDRESS;
             DEST_TYPE:        in   ADDRESS;
             DEST_ID:          in   ADDRESS;
             PIXEL_COL:        in   ADDRESS;
             PIXEL_ROW:        in   ADDRESS;
             NUM_FIELDS:       in   ADDRESS;
             NUM_COLS:         in   ADDRESS;
             LABELS:           in   ADDRESS;
             LABEL_LENGTH:     in   INTEGER;
             STATUS:           in   ADDRESS;
             SUBPANEL_ID:      in   ADDRESS := SYS_NULL_SUBPANEL'ADDRESS;
             PIXEL_WIDTH:      in   ADDRESS := SYS_NULL_COLUMN'ADDRESS;
             PIXEL_HEIGHT:     in   ADDRESS := SYS_NULL_ROW'ADDRESS);
--
-- CPM description: Creates a checkbox button editor.
--
-- formal parameters
--OUT  EDITOR_ID        Address of variable to hold ID attached to the
--                      editor. This ID is required for all interactions
--                      with the editor.
--
--IN    DEST_TYPE       The type of the destination for the editor, where:
--                      SYS_WINDOW_DEST = Window
--                      SYS_PANEL_DEST = Panel
--
--IN    DEST_ID         ID attached to the destination that the editor is
--                      assigned to.  This is set to NULL when the
--                      destination is the RootWindow.
--
--IN    PIXEL_COL       Column number from within the window where the left
--                      side of the editor shall be placed. Column 0 is at
--                      left of the window.
--
--IN    PIXEL_ROW       Row number from within the window where the top side
--                      of the editor shall be placed. Row 0 is at the top
--                      of the window.
--
--IN    NUM_FIELDS      The total number of checkbox buttons to be in the
--                      editor.
--
--IN    NUM_COLS        The number of columns the checkbox buttons are to be
--                      arranged in.
--
--IN    LABELS          Pointer to the array of labels for all the checkbox
--                      buttons.
--
--IN    LABEL_LENGTH    The maximum length of the labels.
--
--IN    STATUS          Pointer to the boolean array of statuses for all the
--                      checkbox buttons.
--
--IN    SUBPANEL_ID     ID attached to the subpanel that
--                      the editor is assigned to. If the editor is not
--                      assigned to a subpanel, use a zero which is the
--                      default.
--
```

```
--IN    PIXEL_WIDTH       The number of pixel columns wide the checkbox is to
--                        be created.  If the width is to be calculated, use
--                        the default value of zero.
--
--IN    PIXEL_HEIGHT      The number of pixel rows wide the checkbox is to be
--                        created.  If the height is to be calculated, use the
--                        default value of zero.
--
-- end formal parameters;


procedure CWN_DEFINE_EDITOR (EDITOR_ID:          in   ADDRESS;
              DEST_TYPE:          in   ADDRESS;
              DEST_ID:            in   ADDRESS;
              PIXEL_COL:          in   ADDRESS;
              PIXEL_ROW:          in   ADDRESS;
              PIXEL_WIDTH:        in   ADDRESS;
              PIXEL_HEIGHT:       in   ADDRESS;
              READ_ONLY:          in   ADDRESS;
              MAX_BUFFER_SIZE:    in   ADDRESS;
              TEXT_BUFFER:        in   ADDRESS;
              BUFFER_SIZE:        in   ADDRESS;
              SUBPANEL_ID:        in   ADDRESS := SYS_NULL_SUBPANEL'ADDRESS);
--
-- CPM description: Creates a window full page text editor.
--
-- formal parameters
--OUT   EDITOR_ID         Address of variable to hold ID attached to the
--                        editor. This ID is required for all interactions
--                        with the editor.
--
--IN    DEST_TYPE         The type of the destination for the editor, where:
--                        SYS_WINDOW_DEST = Window
--                        SYS_PANEL_DEST = Panel
--
--IN    DEST_ID           ID attached to the destination that the editor is
--                        assigned to.  This is set to NULL when the
--                        destination is the RootWindow.
--
--IN    PIXEL_COL         Column number from within the window where the left
--                        side of the editor shall be placed. Column 0 is at
--                        left of the window.
--
--IN    PIXEL_ROW         Row number from within the window where the top side
--                        of the editor shall be placed. Row 0 is at the top
--                        of the window.
--
--IN    PIXEL_WIDTH       The number of columns to be occupied   editor.
--
--IN    PIXEL_HEIGHT      The number of rows to be occupied   editor.
--
--IN    READ_ONLY         Flag indicating if the user has full editing
--                        capabilities or is limited to only scroll and copy
--                        operations.
--                                  true   =  Read only
--                                  false  =  Full edit
```

```
--
--IN    MAX_BUFFER_SIZE  Maximum number of pixels that the TEXT_BUFFER
--                      can hold.
--
--IN    TEXT_BUFFER      Buffer of the initial text to display in the editor.
--
--IN    BUFFER_SIZE      The number of pixels in TEXT_BUFFER.
--
--IN    SUBPANEL_ID      ID attached to the subpanel that
--                      the editor is assigned to. If the editor is not
--                      assigned to a subpanel, use a zero.
--
-- end formal parameters;


procedure CWN_DEFINE_NUMBER_FIELD (
                EDITOR_ID:          in   ADDRESS;
                DEST_TYPE:          in   ADDRESS;
                DEST_ID:            in   ADDRESS;
                PIXEL_COL:          in   ADDRESS;
                PIXEL_ROW:          in   ADDRESS;
                LABEL:              in   ADDRESS;
                LABEL_POSITION:     in   ADDRESS;
                NUMBER_VARIABLE:    in   ADDRESS;
                MIN_NUMBER:         in   ADDRESS;
                MAX_NUMBER:         in   ADDRESS;
                MAX_CHARACTERS:     in   ADDRESS;
                SUBPANEL_ID:        in   ADDRESS := SYS_NULL_SUBPANEL'ADDRESS);
--
-- CPM description: Creates a Numeric Field editor.
--                  Note: This function will not cause display of the field
--                        that is defined in a panel as that is caused by
--                        calling either cwn_end_panel or cwn_end_subpanel.
--
-- formal parameters
--OUT   EDITOR_ID        Address of variable to hold ID attached to the
--                       editor. This ID is required for all interactions
--                       with the editor.
--
--IN    DEST_TYPE       The type of the destination for the editor, where:
--                      SYS_WINDOW_DEST = Window
--                      SYS_PANEL_DEST = Panel
--
--IN    DEST_ID         ID attached to the destination that the editor is
--                      assigned to.  This is set to NULL when the
--                      destination is the RootWindow.
--
--IN    PIXEL_COL        Column number from within the panel where the left
--                       side of the editor shall be placed. Column 0 is at
--                       left of the window.
--
--IN    PIXEL_ROW        Row number from within the panel where the top side
--                       of the editor shall be placed. Row 0 is at the top
--                       of the window.
--
--IN    LABEL            The optional label before the number field.  This
```

```
--                          should be set to NULL if no label will be displayed.
--
--IN    LABEL_POSITION       Value specifying whether the optional label should
--                          be placed to the left or the right of the number
--                          field.  The two valid settings for this field are:
--                              0 = Left aligned
--                              1 = Right aligned
--                          If no label is specified, this parameter will
--                          be ignored   editor.
--
--INOUT NUMBER_VARIABLE     The address of the variable to store the
--                          input number at.  This variable may be
--                          initialized to some number value, which would
--                          be displayed.  This must be a NULL terminated
--                          string.
--
--IN    MIN_NUMBER          The string representing the minimum number
--                          to be allowed as input from the user.  This
--                          string must be MAX_CHARACTERS long with each
--                          digit of the string representing the minimum
--                          value for that digit and the string must be NULL
--                          terminated.
--
--IN    MAX_NUMBER          The string representing the maximum number to be
--                          allowed as input from the user.  This string must
--                          be MAX_CHARACTERS long with each digit of the string
--                          representing the maximum value for that digit and
--                          the string must be NULL terminated.
--
--IN    MAX_CHARACTERS      The maximum number of pixels which will
--                          be allowed to be entered into the field.
--
--IN    SUBPANEL_ID         ID attached to the subpanel that
--                          the editor is assigned to. If the editor is not
--                          assigned to a subpanel, use a zero.
--
-- end formal parameters;


procedure CWN_DEFINE_PANEL (PANEL_ID:   in   ADDRESS);
--
-- CPM description: Defines a panel within a window. This procedure must be
--                  called before defining any field editors. A panel must
--                  have at least one field editor attached to it.
-- formal parameters
--OUT   PANEL_ID          Address of variable to hold ID attached to the
--                        panel.  This ID is required for all interactions
--                        with the panel.
-- end formal parameters;


procedure CWN_DEFINE_POPUP_MENU (MENU_STRUCT_ID: in   ADDRESS;
                                 MENU_TITLE:     in   ADDRESS;
                                 START_ARRAY:    in   ADDRESS;
                                 LENGTH_ARRAY:   in   ADDRESS;
                                 TEXT_ARRAY:     in   ADDRESS;
```

```
                                        CHILD_ARRAY:    in   ADDRESS);
--
-- CPM description: Defines a popup menu which may be a walking menu up to 4
--                  levels deep. This does not, however, display the menu in
--                  the window.  Only one popup per window allowed. All
--                  arrays are zero origin in index.  The index into
--                  Text_Array is used as the menu id.
--
-- formal parameters
--IN      MENU_STRUCT_ID The id given by the application to the popup menu
--                      or entire walking menu structure.
--
--IN      MENU_TITLE     The title of the menu to be displayed at the top
--                      of the menu.  If the menu is a walking menu, then
--                      only the top menu will contain a title.  If the
--                      user doesn't wish the title to be displayed, then
--                      this parameter must be set to NULL.
--
--IN      START_ARRAY    Index into TEXT_ARRAY for the start of each pop-up
--                      menu in the walking menu.
--
--IN      LENGTH_ARRAY   Number of cells in each pop-up menu
--
--IN      TEXT_ARRAY     Text for each cell of each pop-up menu in the
--                      walking menu
--
--IN      CHILD_ARRAY    Pop-up index of the pop-up menu that is the child
--                      of each pop-up menu cell index into START_ARRAY
--                      and LENGTH_ARRAY;
-- end formal parameters;


procedure CWN_DEFINE_POPUP_WINDOW (WINDOW_ID:      in   ADDRESS;
                                   MAP_WINDOW:     in   ADDRESS;
                                   PIXEL_COL:      in   ADDRESS;
                                   PIXEL_ROW:      in   ADDRESS;
                                   PIXEL_WIDTH:    in   ADDRESS;
                                   PIXEL_HEIGHT:   in   ADDRESS);
--
-- CPM description: Changes the size of a popup window.
--
-- formal parameters
--OUT   WINDOW_ID        Address of variable to hold ID attached to the
--                      window.
--
--IN    MAP_WINDOW       Boolean logical indicating whether defined window
--                      should be mapped or not.
--
--IN    PIXEL_COL        Column number from within the display where the left
--                      side of the window shall be placed. Column 0 is at
--                      left of the display.
--
--IN    PIXEL_ROW        Row number from within the display where the top side
--                      of the window shall be placed. Row 0 is at the top
--                      of the display.
--
```

```
--IN    PIXEL_WIDTH      The number of columns to be occupied by the window.
--
--IN    PIXEL_HEIGHT     The number of rows to be occupied by the window.
-- end formal parameters;


procedure CWN_DEFINE_PUSHBUTTON (PUSHBUTTON_ID:  in   ADDRESS;
              DEST_TYPE:      in   ADDRESS;
              DEST_ID:        in   ADDRESS;
              PIXEL_COL:      in   ADDRESS;
              PIXEL_ROW:      in   ADDRESS;
              NUM_FIELDS:     in   ADDRESS;
              NUM_COLS:       in   ADDRESS;
              LABELS:         in   ADDRESS;
              LABEL_LENGTH:   in   INTEGER;
              DEFAULT_BUTTON: in   ADDRESS;
              SUBPANEL_ID:    in   ADDRESS := SYS_NULL_SUBPANEL'ADDRESS);
--
-- CPM description: Creates a pushbutton editor.
--
-- formal parameters
--OUT   EDITOR_ID        Address of variable to hold ID attached to the
--                       editor. This ID is required for all interactions
--                       with the editor.
--
--IN    DEST_TYPE       The type of the destination for the editor, where:
--                      SYS_WINDOW_DEST = Window
--                      SYS_PANEL_DEST = Panel
--
--IN    DEST_ID         ID attached to the destination that the editor is
--                      assigned to.  This is set to NULL when the
--                      destination is the RootWindow.
--
--IN    PIXEL_COL        Column number from within the window where the left
--                       side of the editor shall be placed. Column 0 is at
--                       left of the window.
--
--IN    PIXEL_ROW        Row number from within the window where the top side
--                       of the editor shall be placed. Row 0 is at the top
--                       of the window.
--
--IN    NUM_FIELDS       The total number of pushbuttons to be in the
--                       editor.
--
--IN    NUM_COLS         The number of columns the pushbuttons are to be
--                       arranged in.
--
--IN    LABELS           Pointer to the array of labels for all the
--                       pushbuttons.
--
--IN    LABEL_LENGTH     The maximum length of the labels.
--
--IN    DEFAULT_BUTTON   The index into the pushbutton array of the button to
--                       be drawn "active" or displayed as the default
--                       button.  A value of SYS_NO_DEFAULT_PUSHBUTTON will
--                       disable this feature.
```

```
--
--IN    SUBPANEL_ID       ID attached to the subpanel that
--                       the editor is assigned to. If the editor is not
--                       assigned to a subpanel, use a zero.
--
-- end formal parameters;


procedure CWN_DEFINE_RADIOBUTTON(RADIOBUTTON_ID: in   ADDRESS;
                DEST_TYPE:      in   ADDRESS;
                DEST_ID:        in   ADDRESS;
                PIXEL_COL:      in   ADDRESS;
                PIXEL_ROW:      in   ADDRESS;
                NUM_FIELDS:     in   ADDRESS;
                NUM_COLS:       in   ADDRESS;
                LABELS:         in   ADDRESS;
                LABEL_LENGTH:   in   INTEGER;
                DEFAULT_BUTTON: in   ADDRESS;
                SUBPANEL_ID:    in   ADDRESS := SYS_NULL_SUBPANEL'ADDRESS);
--
-- CPM description: Creates a radiobutton editor where only one button is
--                  active at a time.
--
-- formal parameters
--OUT   EDITOR_ID         Address of variable to hold ID attached to the
--                       editor. This ID is required for all interactions
--                       with the editor.
--
--IN    DEST_TYPE        The type of the destination for the editor, where:
--                       SYS_WINDOW_DEST = Window
--                       SYS_PANEL_DEST = Panel
--
--IN    DEST_ID          ID attached to the destination that the editor is
--                       assigned to.  This is set to NULL when the
--                       destination is the RootWindow.
--
--IN    PIXEL_COL         Column number from within the window where the left
--                       side of the editor shall be placed. Column 0 is at
--                       left of the window.
--
--IN    PIXEL_ROW         Row number from within the window where the top side
--                       of the editor shall be placed. Row 0 is at the top
--                       of the window.
--
--IN    NUM_FIELDS       The total number of radiobuttons to be in the
--                       editor.
--
--IN    NUM_COLS         The number of columns the radiobuttons are to be
--                       arranged in.
--
--IN    LABELS           Pointer to the array of labels for all the
--                       radiobuttons.
--
--IN    LABEL_LENGTH     The maximum length of the labels.
--
--IN    DEFAULT_BUTTON   The index into the radiobutton array of the button
```

```
--                          to be drawn "active" or displayed as the default
--                          button.
--
--IN    SUBPANEL_ID          ID attached to the subpanel that
--                          the editor is assigned to. If the editor is not
--                          assigned to a subpanel, use a zero.
--
-- end formal parameters;


procedure CWN_DEFINE_SCROLLBAR (SCROLLBAR_ID:   in   ADDRESS;
                DEST_TYPE:      in   ADDRESS;
                DEST_ID:        in   ADDRESS;
                ORIENTATION:    in   ADDRESS;
                PIXEL_COL:      in   ADDRESS;
                PIXEL_ROW:      in   ADDRESS;
                PIXEL_WIDTH:    in   ADDRESS;
                PIXEL_LENGTH:   in   ADDRESS;
                DOC_SIZE:       in   ADDRESS;
                DISP_POSITION:  in   ADDRESS;
                SCROLL_INTRVL:  in   ADDRESS;
                SUBPANEL_ID:    in   ADDRESS := SYS_NULL_SUBPANEL'ADDRESS);
--
-- CPM description: Creates a horizontal or vertical scroll bar in a window.
--
-- formal parameters
--OUT   SCROLLBAR_ID      ID attached to the scrollbar.
--                        This ID is required for all interactions with the
--                        scrollbar.
--
--IN    DEST_TYPE        The type of the destination for the editor, where:
--                       SYS_WINDOW_DEST = Window
--                       SYS_PANEL_DEST = Panel
--
--IN    DEST_ID          ID attached to the destination that the editor is
--                       assigned to.  This is set to NULL when the
--                       destination is the RootWindow.
--
--IN    ORIENTATION       Direction of the scrollbar (Horizontal or Vertical)
--
--IN    PIXEL_COL         Column number from within the panel where the left
--                        side of the scrollbar shall be placed. Column 0 is
--                        at the left of the window.
--
--IN    PIXEL_ROW         Row number from within the panel where the top side
--                        of the scrollbar shall be placed. Row 0 is at the
--                        top of the window.
--
--IN    PIXEL_WIDTH       The number of pixels to be occupied
--                        scrollbar's width.
--
--IN    PIXEL_LENGTH      The number of pixels to be occupied
--                        scrollbar's length.
--
--IN    DOC_SIZE          The number of lines in the document buffer.
--
```

```
--IN    DISP_POSITION     The offset from the beginning of the work surface to
--                        first pixel visible to the user.
--
--IN    SCROLL_INTRVL     The number of pixels the work will be scrolled
--                        whenever the user selects an arrow button.  Note:
--                        The work will not be scrolled  se utilities
--                        but, this argument is required to calculate
--                        the interactive slidepositioning.
--
--IN    SUBPANEL_ID       ID attached to the subpanel that
--                        the editor is assigned to. If the editor is not
--                        assigned to a subpanel, use a zero.
--
-- end formal parameters;


procedure CWN_DEFINE_STATIC_TEXT (STATIC_TEXT_ID:   in   ADDRESS;
                DEST_TYPE:         in   ADDRESS;
                DEST_ID:           in   ADDRESS;
                PIXEL_COL:         in   ADDRESS;
                PIXEL_ROW:         in   ADDRESS;
                PIXEL_WIDTH:       in   ADDRESS;
                PIXEL_HEIGHT:      in   ADDRESS;
                STATIC_TEXT:       in   ADDRESS;
                TEXT_ALIGNMENT:    in   ADDRESS;
                SUBPANEL_ID:       in   ADDRESS := SYS_NULL_SUBPANEL'ADDRESS);
--
-- CPM description: Creates a static text area in a window. The static text
--                  procedure allows display of product headings that will
--                  not scroll with the product.
--
-- formal parameters
--OUT   STATIC_TEXT_ID    ID attached to the static text area.
--                        This ID is required for all interactions with
--                        the static text area.
--
--IN    DEST_TYPE        The type of the destination for the editor, where:
--                       SYS_WINDOW_DEST = Window
--                       SYS_PANEL_DEST = Panel
--
--IN    DEST_ID          ID attached to the destination that the editor is
--                       assigned to.  This is set to NULL when the
--                       destination is the RootWindow.
--
--IN    PIXEL_COL         Column number from within the window where the left
--                        side of the static text area shall be placed.
--                        Column 0 is at the left of the window.
--
--IN    PIXEL_ROW         Row number from within the window where the top side
--                        of the static text area shall be placed. Row 0 is
--                        at the top of the window.
--
--IN    PIXEL_WIDTH       The number of columns to be occupied   static
--                        text area.
--
--IN    PIXEL_HEIGHT      The number of rows to be occupied   static
```

```
--                          text area.
--
--IN     STATIC_TEXT         Textual string to display in the button.
--
--IN     TEXT_ALIGNMENT      Alignment of the text within the static text area
--                           (CENTER_ALIGNED, LEFT_ALIGNED, RIGHT_ALIGNED,
--                            NO_ALIGNMENT)
--
--IN     SUBPANEL_ID         ID attached to the subpanel that
--                           the editor is assigned to. If the editor is not
--                           assigned to a subpanel, use a zero.
--
-- end formal parameters;


procedure CWN_DEFINE_STRING_FIELD (
              EDITOR_ID:          in   ADDRESS;
              DEST_TYPE:          in   ADDRESS;
              DEST_ID:            in   ADDRESS;
              PIXEL_COL:          in   ADDRESS;
              PIXEL_ROW:          in   ADDRESS;
              LABEL:              in   ADDRESS;
              LABEL_POSITION:     in   ADDRESS;
              STRING_VARIABLE:    in   ADDRESS;
              MAX_CHARACTERS:     in   ADDRESS;
              SUBPANEL_ID:       in   ADDRESS := SYS_NULL_SUBPANEL'ADDRESS);
--
-- CPM description: Creates a String Field editor.
--                  Note: this function will not cause display of the field
--                        as that is caused by calling either cwn_end_panel
--                        or cwn_end_subpanel.
--
-- formal parameters
--OUT   EDITOR_ID           ID attached to the editor. This ID is
--                          required for all interactions with the editor.
--
--IN     DEST_TYPE          The type of the destination for the editor, where:
--                          SYS_WINDOW_DEST = Window
--                          SYS_PANEL_DEST= Panel
--
--IN     DEST_ID            ID attached to the destination that the editor is
--                          assigned to.  This is set to NULL when the
--                          destination is the RootWindow.
--
--IN     PIXEL_COL           Column number from within the panel where the left
--                          side of the editor shall be placed. Column 0 is at
--                          left of the window.
--
--IN     PIXEL_ROW          Row number from within the panel where the top side
--                          of the editor shall be placed. Row 0 is at the top
--                          of the window.
--
--IN     LABEL              The optional label before the string field.  This
--                          should be set to NULL if no label will be displayed.
--
--IN     LABEL_POSITION     Value specifying whether the optional label should
```

```
--                          be placed to the left or the right of the number
--                          field.  The two valid settings for this field are:
--                              0 = Left aligned
--                              1 = Right aligned
--                          If no label is specified, this parameter will
--                          be ignored   editor.
--IN     STRING_VARIABLE     The address of the variable to store the
--                          input string at.  This variable may be
--                          initialized to some string value, which would
--                          be displayed.  This must be a NULL terminated
--                          string.
--IN     MAX_CHARACTERS      The maximum number of characters which will
--                          be allowed to be entered into the field.
--
--IN     SUBPANEL_ID         ID attached to the subpanel that
--                          the editor is assigned to. If the editor is not
--                          assigned to a subpanel, use a zero.
--
-- end formal parameters;


procedure CWN_DEFINE_SUBPANEL (SUBPANEL_ID:    in   ADDRESS;
                               PANEL_ID:       in   ADDRESS);
--
-- CPM description: Defines a subpanel within a panel. A subpanel must
--                  have at least one field editor attached to it.
-- formal parameters
--OUT    SUBPANEL_ID         ID attached to the subpanel.
--                          This ID is required for all interactions with the
--                          subpanel.
--IN     PANEL_ID            ID of the panel that the
--                          subpanel is attached to.
-- end formal parameters;


procedure CWN_DELETE_BUTTON (BUTTON_ID    :     in  ADDRESS);
--
-- CPM description: CWN_DELETE_BUTTON deletes a button that is defined by
--                  CWN_DEFINE_BUTTON.
--
-- formal parameters
--IN     BUTTON_ID           The ID of the button to delete.
-- end formal parameters;

procedure CWN_DELETE_CHECKBOX (CHECKBOX_ID    :   in  ADDRESS);
--
-- CPM description: CWN_DELETE_CHECKBOX deletes a checkbox editor that is
--                  defined by CWN_DEFINE_CHECKBOX.
--
-- formal parameters
--IN     CHECKBOX_ID         The ID of the checkbox editor to delete.
-- end formal parameters;


procedure CWN_DELETE_EDITOR (EDITOR_ID    :     in   ADDRESS);
--
```

```
-- CPM description: CWN_DELETE_EDITOR deletes an editor that is defined by
--                  CWN_DEFINE_EDITOR.
--
-- formal parameters
--IN    EDITOR_ID        The ID of the editor to delete.
--
-- end formal parameters;


procedure CWN_DELETE_MENU (MENU_ID    :     in  ADDRESS);
--
-- CPM description: CWN_DELETE_EDITOR deletes a walking menu structure.
--
-- formal parameters
--IN    MENU_ID        The ID of the menu structure to delete.
-- end formal parameters;


procedure CWN_DELETE_NUMBER_FIELD (
                              EDITOR_ID    :     in  ADDRESS);
--
-- CPM description: Deletes an numeric field editor that
--                  is defined by CWN_DEFINE_NUMBER_FIELD.
--
-- formal parameters
--IN    EDITOR_ID        The ID of the editor to delete.
--
-- end formal parameters;


procedure CWN_DELETE_PANEL (PANEL_ID    :     in  ADDRESS);
--
-- CPM description: Deletes a panel from a window.
--
-- formal parameters
--IN    PANEL_ID        The ID of the panel to delete.
-- end formal parameters;


procedure CWN_DELETE_POPUP_WINDOW (WINDOW_ID    :    in   ADDRESS);
--
-- CPM description:   CWN_DELETE_POPUP_WINDOW deletes a popup window that is
--                    defined by CWN_DEFINE_POPUP_WINDOW.
--
-- formal parameters
--IN    WINDOW_ID       The ID of the popup window.
-- end formal parameters;


procedure CWN_DELETE_PUSHBUTTON (PUSHBUTTON_ID    :   in   ADDRESS);
--
-- CPM description:  CWN_DELETE_PUSHBUTTON deletes a pushbutton editor that
--                   is defined by CWN_DEFINE_PUSHBUTTON.
--
-- formal parameters
--IN    PUSHBUTTON_ID    The ID of the pushbutton editor.
```

```
-- end formal parameters;


procedure CWN_DELETE_RADIOBUTTON (RADIOBUTTON_ID   :   in   ADDRESS);
--
-- CPM description:  CWN_DELETE_RADIOBUTTON deletes a radiobutton editor that
--                   is defined by CWN_DEFINE_RADIOBUTTON.
--
-- formal parameters
--IN    RADIOBUTTON_ID   The ID of the radiobutton editor.
-- end formal parameters;


procedure CWN_DELETE_SCROLLBAR (SCROLLBAR_ID   :     in  ADDRESS);
--
-- CPM description: CWN_DELETE_SCROLLBAR deletes a scrollbar that is defined
--                   by CWN_DEFINE_SCROLLBAR.
--
-- formal parameters
--IN    SCROLLBAR_ID     The ID of the scrollbar to delete.
--
-- end formal parameters;


procedure CWN_DELETE_STATIC_TEXT (STATIC_ID   :     in  ADDRESS);
--
-- CPM description: CWN_DELETE_STATIC_TEXT deletes static text that is
--                  defined by CWN_DEFINE_STATIC_TEXT.
--
-- formal parameters
--IN    STATIC_ID     The ID of the static text to delete.
--
-- end formal parameters;


procedure CWN_DELETE_STRING_FIELD (
                              EDITOR_ID   :     in  ADDRESS);
--
-- CPM description: Deletes an string field editor that
--                  is defined by CWN_DEFINE_STRING_FIELD.
--
-- formal parameters
--IN    EDITOR_ID        The ID of the editor to delete.
--
-- end formal parameters;


procedure CWN_DELETE_SUBPANEL (SUBPANEL_ID:      in   ADDRESS);
--
-- CPM description: Deletes a subpanel from a window.
--
-- formal parameters
--IN    SUBPANEL_ID     The ID of the subpanel to delete.
-- end formal parameters;
```

```
procedure CWN_END_PANEL (WINDOW_ID:        in   ADDRESS;
                         PANEL_ID:         in   ADDRESS;
                         PIXEL_COL:        in   ADDRESS;
                         PIXEL_ROW:        in   ADDRESS;
                         PIXEL_WIDTH:      in   ADDRESS;
                         PIXEL_HEIGHT:     in   ADDRESS);
--
-- CPM description: This procedure completes the panel definition process.
--                  It displays the subpanels and field editors (text
--                  editors, scroll bars, and static text) that are attached
--                  to the panel.
--
-- formal parameters
--IN    WINDOW_ID         ID attached to the window to contain the panel.
--
--IN    PANEL_ID          ID attached to the panel.
--
--IN    PIXEL_COL         Column number from within the window where the left
--                        side of the panel shall be placed. Column 0 is
--                        at the left of the window.
--
--IN    PIXEL_ROW         Row number from within the window where the top side
--                        of the panel shall be placed. Row 0 is at the
--                        top of the window.
--
--IN    PIXEL_WIDTH       The width of the panel in pixels.
--
--IN    PIXEL_HEIGHT      The height of the panel in pixels.
-- end formal parameters;


procedure CWN_DELETE_SUBWINDOW (SUBWINDOW_ID:      in   ADDRESS);
--
-- CPM description: Deletes a subwindow from the working window.
--
-- formal parameters
--IN    SUBWINDOW_ID     The ID of the subWINDOW to delete.
-- end formal parameters;


procedure CWN_DISPLAY_SYSTEM_MESSAGE (MESSAGE  :   in   ADDRESS);
--
-- CPM description: This displays a message in the upper left hand corner of
--                  the display screen.  Unlike cwn_message_box, this routine
--                  is provided mainly for system messages relating the
--                  status or some other information of the system.  The
--                  message is removed via cwn_remove_system_message.
--
-- formal parameters
--IN    MESSAGE     The Message to display.
-- end formal parameters;


procedure CWN_END_SUBPANEL (SUBPANEL_ID:    in   ADDRESS;
                            PIXEL_COL:      in   ADDRESS;
                            PIXEL_ROW:      in   ADDRESS;
```

```
                                PIXEL_WIDTH:    in   ADDRESS;
                                PIXEL_HEIGHT:   in   ADDRESS);
--
-- CPM description: This procedure completes the subpanel definition process
--                  It displays the field editors (text editors, scroll
--                  bars, and static text) that are attached to the subpanel
--
-- formal parameters
--IN    SUBPANEL_ID       ID attached to the subpanel.
--
--IN    PIXEL_COL         Column number from within the window where the left
--                        side of the subpanel shall be placed. Column 0 is
--                        at the left of the window.
--
--IN    PIXEL_ROW         Row number from within the window where the top side
--                        of the subpanel shall be placed. Row 0 is at the
--                        top of the window.
--
--IN    PIXEL_WIDTH       The width of the subpanel in pixels.
--
--IN    PIXEL_HEIGHT      The height of the subpanel in pixels.
-- end formal parameters;


procedure CWN_HANDLE_WINDOW_MOVE (WINDOW_ID:      in   SYS_WINDOW_ELE_ID;
                                  MESSAGE:        in   INTEGER := 0;
                                  DATA:           in   INTEGER := 0);
--
-- CPM description:  This procedure handles the user interface required
--                   for allowing the user to interactively move a window.
--
-- formal parameters
--IN    WINDOW_ID     The ID attached to the window.
--IN    MESSAGE       Currently not applicable from ADA.
--IN    DATA          Currently not applicable from ADA.
-- end formal parameters;


procedure CWN_HIDE_PANEL (PANEL_ID:       in   ADDRESS);
--
-- CPM description: This procedure hides a defined panel and disables user
--                  input to any of the panel editors.
--
-- formal parameters
--IN    PANEL_ID         ID attached to the panel to
--                       hide.
-- end formal parameters;


procedure CWN_HIDE_SUBPANEL (SUBPANEL_ID:    in   ADDRESS);
--
-- CPM description: This procedure hides a defined subpanel and disables user
--                  input to any of the subpanel editors.
--
-- formal parameters
--IN    SUBPANEL_ID      ID attached to the subpanel to
```

```
--                        hide.
--
-- end formal parameters;


procedure CWN_INITIALIZE_WINDOW_SYSTEM;
--
-- CPM description: CWN_INITIALIZE_WINDOW_SYSTEM is the initial set-up
--                  procedure for the EDDIC window system. It must be called
--                  before any of the CWN utilities.
--
-- formal parameters
--    None
-- end formal parameters;


procedure CWN_INPUT (INPUT_TYPE   :   in   ADDRESS;
                     WINDOW_ID    :   in   ADDRESS;
                     INPUT_VALUE  :   in   ADDRESS;
                     INPUT_DATA   :   in   ADDRESS);
--
-- CPM description: Returns user input and internet messages to the
--                  application software.
--
-- formal parameters
--OUT   INPUT_TYPE       Type of input returned from the window system
--
--OUT   WINDOW_ID        The id of the window which received input, if
--                       applicable.  Note, that if the table below has
--                       an "X" under the window_id header for the
--                       input_type, but the window_id equals zero, then
--                       this means that the input took place in the
--                       RootWindow.
--
--OUT   INPUT_VALUE      The value of the input that accompanies the type
--
--OUT   INPUT_DATA       The data that accompanies the type and value, if
--                       appropriate.
--
--  The following table lists the output returned to the application
--  for its own processing:
--
--                     window-
-- input_type            id       type_code     data
-- ----------          -------    ---------     --------
-- 1 Exit              n/a        n/a           n/a
-- 2 Menu              n/a        Menu_Id       menu_index
-- 3 Checkbox          X          Editor_Id     Checkbox_index
-- 4 Scrollbar         X          Editor_Id     SlidePosition
-- 5 XrFILE            n/a        fd            n/a
-- 6 ButtonWindow      X          n/a           n/a
-- 7 Mouse Button      X          Button:       window_type:
--   Pressed                       0 = R        1 = button
--                                 1 = M        2 = panel
--                                 2 = L        3 = window
--                                              x, y
```

```
-- 8 Mouse Button      X          Button:       window_type:
--   Released                      0 = R         1 = button
--                                 1 = M         2 = panel
--                                 2 = L         3 = window
--                                               x, y
-- 9 Field Traversal   X          Editor_id     editor_type:
--                                              1 = String_field
--                                              2 = Number_field
--                                              type of traversal:
--                                              1 - Next
--                                              2 - Previous
--                                              3 - Up
--                                              4 - Down
--10 Exposure          X           n/a          x, y, width, height
--11 Open Window       n/a         n/a          n/a
--12 Window Resized    n/a         n/a          n/a
--13 Close Window      n/a         n/a          n/a
--14 XrEEDIT_SAVE      X           Editor_Id    bufferCount
--15 XrEEDIT_RESET     X           Editor_Id    n/a
--16 Pushbutton        X           Editor_Id    Button_index
--17 Radiobutton       X           Editor_Id    Active_index,
--                                              Previous_Index
--
-- end formal parameters;


procedure CWN_MAP_WINDOW (WINDOW_ID:   in   ADDRESS);
--
-- CPM description: Routine to map a window created via cwn_create_window
--                  whose "map_window" flag was set FALSE.
--
-- formal parameters
--IN   WINDOW_ID    The id of the window to be mapped.
--
-- end formal parameters;


procedure CWN_MESSAGE_BOX (MESSAGE          :   in    ADDRESS;
                           BUTTONS_ALLOWED  :   in    ADDRESS;
                           BUTTON_SELECTED  :   in    ADDRESS;
                           BUTTON_X_PIXEL   :   in    ADDRESS;
                           BUTTON_Y_PIXEL   :   in    ADDRESS;
                           INPUT_WINDOW_ID  :   in    ADDRESS);
--
-- CPM description: Displays a message box which the user removes by a click
--                  on the mouse which is allowed   application. The
--                  message box always appears centered on the display and
--                  the button which activated its disappearance is returned
--                  to the application.
--
-- formal parameters
--IN    MESSAGE          Textual string to display in the message box.
--
--IN    BUTTON_ALLOWED   A logical array indicating which mouse buttons
--                       the application is allowing the user to click
--                       for making the message box go away, where:
```

```
--                              [0] = RightButton;
--                              [1] = MiddleButton;
--                              [2] = LeftButton;
--OUT    BUTTON_SELECTED  The number of the selected button (0, 1, or 2);
--
--OUT    BUTTON_X_PIXEL   The x pixel location where the mouse button was
--                        selected.
--
--OUT    BUTTON_Y_PIXEL   The y pixel location where the mouse button was
--                        selected.
--
--OUT    INPUT_WINDOW_ID  The id of the window which received the mouse
--                        button selection input.
-- end formal parameters;


procedure CWN_MOVE_BUTTON (BUTTON_ID:          in   ADDRESS;
                           PIXEL_COL:          in   ADDRESS;
                           PIXEL_ROW:          in   ADDRESS);
--
-- CPM description: Changes the location of a button.
--
-- formal parameters
--IN     BUTTON_ID        ID to attach to the button. This
--                        ID is required for all interactions with the button.
--
--IN     PIXEL_COL        Column number from within the window where the left
--                        side of the button shall be placed. Column 0 is at
--                        left of the window.
--
--IN     PIXEL_ROW        Row number from within the window where the top side
--                        of the button shall be placed. Row 0 is at the top
--                        of the window.
-- end formal parameters;


procedure CWN_MOVE_CHECKBOX (CHECKBOX_ID:      in   ADDRESS;
                             PIXEL_COL:        in   ADDRESS;
                             PIXEL_ROW:        in   ADDRESS);
--
-- CPM description: Changes the location of a checkbox editor.
--
-- formal parameters
--IN     CHECKBOX_ID      ID attached to the checkbox editor.
--
--IN     PIXEL_COL        Column number from within the window where the left
--                        side of the editor shall be placed. Column 0 is at
--                        left of the window.
--
--IN     PIXEL_ROW        Row number from within the window where the top side
--                        of the editor shall be placed. Row 0 is at the top
--                        of the window.
-- end formal parameters;


procedure CWN_MOVE_EDITOR (EDITOR_ID:          in   ADDRESS;
```

```
                              PIXEL_COL:          in   ADDRESS;
                              PIXEL_ROW:          in   ADDRESS);
--
-- CPM description: Changes the location of a full page text editor.
--
-- formal parameters
--IN     EDITOR_ID          ID to attach to the editor. This
--                          ID is required for all interactions with the editor.
--
--IN     PIXEL_COL          Column number from within the window where the left
--                          side of the editor shall be placed. Column 0 is at
--                          left of the window.
--
--IN     PIXEL_ROW          Row number from within the window where the top side
--                          of the editor shall be placed. Row 0 is at the top
--                          of the window.
-- end formal parameters;


procedure CWN_MOVE_NUMBER_FIELD (
                              EDITOR_ID:          in   ADDRESS;
                              PIXEL_COL:          in   ADDRESS;
                              PIXEL_ROW:          in   ADDRESS);

--
-- CPM description: Changes the location of a numeric field editor.
--
-- formal parameters
--IN     EDITOR_ID          ID to attach to the editor. This
--                          ID is required for all interactions with the editor.
--
--IN     PIXEL_COL          Column number from within the window where the left
--                          side of the editor shall be placed. Column 0 is at
--                          left of the window.
--
--IN     PIXEL_ROW          Row number from within the window where the top side
--                          of the editor shall be placed. Row 0 is at the top
--                          of the window.
-- end formal parameters;


procedure CWN_MOVE_PANEL (PANEL_ID:           in   ADDRESS;
                          PIXEL_COL:          in   ADDRESS;
                          PIXEL_ROW:          in   ADDRESS);
--
-- CPM description: Changes the location of a panel.
--
-- formal parameters
--IN     PANEL_ID           ID attached to the panel to
--                          move.
--
--IN     PIXEL_COL          Column number from within the window where the left
--                          side of the panel shall be placed. Column 0 is at
--                          left of the window.
--
--IN     PIXEL_ROW          Row number from within the window where the top side
```

```
--                          of the panel shall be placed. Row 0 is at the top
--                          of the window.
-- end formal parameters;


procedure CWN_MOVE_POPUP_WINDOW (WINDOW_ID:      in   ADDRESS;
                                 PIXEL_COL:      in   ADDRESS;
                                 PIXEL_ROW:      in   ADDRESS);
--
-- CPM description: Changes the location of a popup window.
--
-- formal parameters
--IN    WINDOW_ID          ID attached to the popup window to move.
--
--IN    PIXEL_COL          Column number from within the display where the left
--                         side of the window shall be placed. Column 0 is at
--                         left of the display.
--
--IN    PIXEL_ROW          Row number from within the display where the top side
--                         of the window shall be placed. Row 0 is at the top
--                         of the display.
-- end formal parameters;


procedure CWN_MOVE_PUSHBUTTON (PUSHBUTTON_ID:    in   ADDRESS;
                               PIXEL_COL:        in   ADDRESS;
                               PIXEL_ROW:        in   ADDRESS);
--
-- CPM description: Changes the location of a pushbutton editor.
--
-- formal parameters
--IN    PUSHBUTTON_ID      ID attached to the pushbutton editor to move.
--
--IN    PIXEL_COL          Column number from within the window where the left
--                         side of the editor shall be placed. Column 0 is at
--                         left of the window.
--
--IN    PIXEL_ROW          Row number from within the window where the top side
--                         of the editor shall be placed. Row 0 is at the top
--                         of the window.
-- end formal parameters;


procedure CWN_MOVE_RADIOBUTTON (RADIOBUTTON_ID:    in   ADDRESS;
                                PIXEL_COL:         in   ADDRESS;
                                PIXEL_ROW:         in   ADDRESS);
--
-- CPM description: Changes the location of a radiobutton editor.
--
-- formal parameters
--IN    RADIOBUTTON_ID     ID attached to the radiobutton editor to move.
--
--IN    PIXEL_COL          Column number from within the window where the left
--                         side of the editor shall be placed. Column 0 is at
--                         left of the window.
```

```
--
--IN    PIXEL_ROW          Row number from within the window where the top side
--                        of the editor shall be placed. Row 0 is at the top
--                        of the window.
-- end formal parameters;


procedure CWN_MOVE_SCROLLBAR (
                            SCROLLBAR_ID:      in   ADDRESS;
                            PIXEL_COL:         in   ADDRESS;
                            PIXEL_ROW:         in   ADDRESS);
--
-- CPM description: Changes the location of a scrollbar.
--
-- formal parameters
--IN    SCROLLBAR_ID       ID to attach to the scrollbar.
--                        This ID is required for all interactions with the
--                        scrollbar.
--
--IN    PIXEL_COL          Column number from within the window where the left
--                        side of the scrollbar shall be placed. Column 0 is
--                        at left of the window.
--
--IN    PIXEL_ROW          Row number from within the window where the top side
--                        of the scrollbar shall be placed. Row 0 is at the
--                        top of the panel.
-- end formal parameters;


procedure CWN_MOVE_STATIC_TEXT (
                            TEXT_ID:           in   ADDRESS;
                            PIXEL_COL:         in   ADDRESS;
                            PIXEL_ROW:         in   ADDRESS);
--
-- CPM description: Changes the location of static text.
--
-- formal parameters
--IN    EDITOR_ID          ID to attach to the text. This
--                        ID is required for all interactions with the text.
--
--IN    PIXEL_COL          Column number from within the window where the left
--                        side of the text shall be placed. Column 0 is at
--                        left of the window.
--
--IN    PIXEL_ROW          Row number from within the window where the top
--                        side of the text shall be placed. Row 0 is at the
--                        top of the window.
-- end formal parameters;


procedure CWN_MOVE_STRING_FIELD (
                            EDITOR_ID:         in   ADDRESS;
                            PIXEL_COL:         in   ADDRESS;
                            PIXEL_ROW:         in   ADDRESS);
--
-- CPM description: Changes the location of a string field editor.
```

```
--
-- formal parameters
--IN     EDITOR_ID         ID to attach to the editor. This
--                         ID is required for all interactions with the editor.
--
--IN     PIXEL_COL         Column number from within the window where the left
--                         side of the editor shall be placed. Column 0 is at
--                         left of the window.
--
--IN     PIXEL_ROW         Row number from within the window where the top side
--                         of the editor shall be placed. Row 0 is at the top
--                         of the window.
-- end formal parameters;


procedure CWN_MOVE_SUBWINDOW (SUBWINDOW_ID:      in   ADDRESS;
                              PIXEL_COL:         in   ADDRESS;
                              PIXEL_ROW:         in   ADDRESS);
--
-- CPM description: Changes the location of a subwindow.
--
-- formal parameters
--IN     SUBWINDOW_ID      ID attached to the subwindow to move.
--
--IN     PIXEL_COL         Column number from within the window where the left
--                         side of the subwindow shall be placed. Column 0 is
--                         at left of the window.
--
--IN     PIXEL_ROW         Row number from within the window where the top side
--                         of the subwindow shall be placed. Row 0 is at the
--                         top of the window.
-- end formal parameters;


procedure CWN_MOVE_WINDOW (WINDOW_ID:      in   ADDRESS;
                           PIXEL_COL:      in   ADDRESS;
                           PIXEL_ROW:      in   ADDRESS);
--
-- CPM description: Changes the location of a window.
--
-- formal parameters
--IN     WINDOW_ID         ID attached to the window to move.
--
--IN     PIXEL_COL         Column number where the left side of the window
--                         shall be placed.
--
--IN     PIXEL_ROW         Row number where the top side of the window
--                         shall be placed.
-- end formal parameters;


procedure CWN_OPEN_ICON;
--
-- CPM description:  Opens the window from an existing icon.
--
-- formal parameters
```

```
-- NONE
-- end formal parameters;

procedure CWN_POST_MENU      (MENU_STRUCT_ID:  in  ADDRESS;
                              MENU_INDEX:      in  ADDRESS;
                              WINDOW_TYPE:     in  ADDRESS;
                              WINDOW_ID:       in  ADDRESS;
                              PIXEL_X:         in  ADDRESS;
                              PIXEL_Y:         in  ADDRESS);
--
-- CPM description: This routine activates and posts an already defined
--                  popup menu at a specified location for either:
--                     a. A defined window,
--                     b. a displayed panel (via cwn_end_panel),
--                     c. or, a defined button (via cwn_define_button).
--
-- formal parameters
--IN     MENU_STRUCT_ID    The id of the menu structure given
--                         application at the time of the menu definition.
--
--IN     MENU_INDEX        The index into the Text_Array of the submenu to
--                         be activated for a particular window, if applicable.
--                         If the menu to be activated is not a walking menu,
--                         or is the top level of a walking menu, then this
--                         parameter should be set to NULL.
--
--IN     WINDOW_TYPE       The type of window the menu will be activated for,
--                         where:
--                         SYS_WINDOW         = a defined window
--                         SYS_DISPLAY_PANEL  = a displayed panel
--                         SYS_DEFINED_BUTTON = defined button
--
--IN     WINDOW_ID         The id given   application at the time of the
--                         window type's creation where:
--                         If window_type is SYS_WINDOW and window_id is 0,
--                         then the menu will be activated for the RootWindow
--                         or (Display).  Otherwise, the menu will be activated
--                         for the matching window_id.
--                         If window_type = SYS_DISPLAY_PANEL, the id should
--                         be the panel id.
--                         If window_type = SYS_DEFINED_BUTTON, the id should
--                         be the button id.
--
--IN     PIXEL_X           The X pixel coordinate for posting the menu.
--
--IN     PIXEL_Y           The Y pixel coordinate for posting the menu.
--end formal parameters;

procedure CWN_QUERY_CHECKBOX_RECTS (CHECKBOX_ID    :   in   ADDRESS;
                                    CHECKBOX_RECTS :   in   ADDRESS);
--
-- CPM description:  Returns the rectangular descriptions of the individual
--                   checkboxes.  Note: these descriptions do not include
--                   the labels in the widths and this routine cannot be
--                   called before the panel containing the checkbox instance
```

```
--                    has been ended via CWN_END_PANEL.
-- formal parameters
--IN    CHECKBOX_ID      ID attached to the editor.
--
--IN    CHECKBOX_RECTS   The address of the array of rectangle descriptions.
-- end formal parameters;


procedure CWN_QUERY_CHECKBOX_SIZE (CHECKBOX_ID:     in   ADDRESS;
                                   PIXEL_COL:       in   ADDRESS;
                                   PIXEL_ROW:       in   ADDRESS);
--
-- CPM description: Returns the number of pixel columns and rows that
--                  a checkbox editor occupies.
--
-- formal parameters
--IN    CHECKBOX_ID      ID attached to the editor.
--
--IN    PIXEL_COL        Address of variable to hold number of pixel columns
--                       in the editor.
--
--IN    PIXEL_ROW        Address of variable to hold number of pixel rows in
--                      the editor.
-- end formal parameters;


procedure CWN_QUERY_DISPLAY_SIZE (WIDTH :   in   ADDRESS;
                                  HEIGHT:   in   ADDRESS);
--
-- CPM description: Returns the number of pixel columns and rows that
--                  are in the Display screen.
--
-- formal parameters
--OUT   WIDTH         Number of pixel columns in the Display screen.
--
--OUT   HEIGHT        Number of pixel rows in the Display screen.
-- end formal parameters;


procedure CWN_QUERY_EDITOR_SIZE (EDITOR_ID:         in   ADDRESS;
                                 PIXEL_COL:         in   ADDRESS;
                                 PIXEL_ROW:         in   ADDRESS);
--
-- CPM description: Returns the number of pixel columns and rows that
--                  an editor occupies.
--
-- formal parameters
--IN    EDITOR_ID        ID to attach to the editor.
--
--OUT   PIXEL_COL        Number of pixel columns in the editor.
--
--OUT   PIXEL_ROW        Number of pixel rows in the editor.
-- end formal parameters;


procedure CWN_QUERY_FONT_SIZE (PIXEL_COL:          in   ADDRESS;
```

```
                                PIXEL_ROW:          in   ADDRESS);
--
-- CPM description: Returns the number of pixel columns and rows that
--                  a font occupies.
--
-- formal parameters
--OUT   PIXEL_COL        Number of pixel columns in the font.
--
--OUT   PIXEL_ROW        Number of pixel rows in the font.
-- end formal parameters;


procedure CWN_QUERY_NUMBER_FIELD_SIZE (
                                EDITOR_ID:          in   ADDRESS;
                                PIXEL_COL:          in   ADDRESS;
                                PIXEL_ROW:          in   ADDRESS);
--
-- CPM description: Returns the number of pixel columns and rows that
--                  an numeric field editor occupies.
--
-- formal parameters
--IN    EDITOR_ID         ID to attach to the editor.
--
--OUT   PIXEL_COL         Number of pixel columns in the editor.
--
--OUT   PIXEL_ROW         Number of pixel rows in the editor.
-- end formal parameters;


procedure CWN_QUERY_PANEL_ORIGIN (PANEL_ID:          in   ADDRESS;
                                PIXEL_COL:          in   ADDRESS;
                                PIXEL_ROW:          in   ADDRESS);
--
-- CPM description: Returns the pixel column and row that designates the
--                  origin of a panel.
--
-- formal parameters
--IN    PANEL_ID          ID to attach to the panel.
--
--OUT   PIXEL_COL         Pixel column of the origin in the window.
--
--OUT   PIXEL_ROW         Pixel row of the origin in the window.
-- end formal parameters;


procedure CWN_QUERY_PANEL_SIZE (PANEL_ID:          in   ADDRESS;
                                PIXEL_COL:          in   ADDRESS;
                                PIXEL_ROW:          in   ADDRESS);
--
-- CPM description: Returns the number of pixel columns and rows that
--                  a panel requires. The size is determined by using the
--                  locations and sizes of the editors that are attached
--                  to the panel.
--
-- formal parameters
--IN    PANEL_ID          ID to attach to the panel.
```

```
--
--OUT   PIXEL_COL       Number of pixel columns in the window.
--
--OUT   PIXEL_ROW       Number of pixel rows in the window.
-- end formal parameters;


procedure CWN_QUERY_PUSHBUTTON_RECTS (PUSHBUTTON_ID:     in  ADDRESS;
                                      PUSHBUTTON_RECTS : in  ADDRESS);
--
-- CPM description:  Returns the rectangular descriptions of the individual
--                   pushbuttons.  Note: these descriptions do not include
--                   the labels in the widths and this routine cannot be
--                   called before the panel containing the pushbutton
--                   instance has been ended via CWN_END_PANEL.
-- formal parameters
--IN    PUSHBUTTON_ID     ID attached to the editor.
--
--INOUT PUSHBUTTON_RECTS  The address of the array of rectangle
--                        descriptions.
-- end formal parameters;


procedure CWN_QUERY_PUSHBUTTON_SIZE (PUSHBUTTON_ID:   in   ADDRESS;
                                     PIXEL_COL:       in   ADDRESS;
                                     PIXEL_ROW:       in   ADDRESS);
--
-- CPM description: Returns the number of pixel columns and rows that
--                  a pushbutton editor occupies.
--
-- formal parameters
--IN    PUSHBUTTON_ID   ID attached to the editor.
--
--OUT   PIXEL_COL       Address of variable to hold number of pixel columns
--                      in the editor.
--
--OUT   PIXEL_ROW       Address of variable to hold number of pixel rows in
--                      the editor.
-- end formal parameters;


 procedure CWN_QUERY_RADIOBUTTON_RECTS (RADIOBUTTON_ID :    in  ADDRESS;
                                        RADIOBUTTON_RECTS : in  ADDRESS);
--
-- CPM description:  Returns the rectangular descriptions of the individual
--                   radiobuttons.  Note: these descriptions do not include
--                   the labels in the widths and this routine cannot be
--                   called before the panel containing the radiobutton
--                   instance has been ended via CWN_END_PANEL.
-- formal parameters
--IN    RADIOBUTTON_ID     ID attached to the editor.
--
--INOUT RADIOBUTTON_RECTS   The address of the array of rectangle
--                          descriptions.
-- end formal parameters;
```

```
procedure CWN_QUERY_RADIOBUTTON_SIZE (
                                    RADIOBUTTON_ID:     in   ADDRESS;
                                    PIXEL_COL:          in   ADDRESS;
                                    PIXEL_ROW:          in   ADDRESS);
--
-- CPM description: Returns the number of pixel columns and rows that
--                  a radiobutton editor occupies.
--
-- formal parameters
--IN    RADIOBUTTON_ID  ID attached to the editor.
--
--OUT   PIXEL_COL       Address of variable to hold number of pixel columns
--                      in the editor.
--
--OUT   PIXEL_ROW       Address of variable to hold number of pixel rows in
--                      the editor.
-- end formal parameters;


procedure CWN_QUERY_SCROLLBAR_SIZE (
                                  SCROLLBAR_ID:       in   ADDRESS;
                                  PIXEL_COL:          in   ADDRESS;
                                  PIXEL_ROW:          in   ADDRESS);
--
-- CPM description: Returns the number of pixel columns and rows that
--                  a scrollbar occupies.
--
-- formal parameters
--IN    SCROLLBAR_ID     ID to attach to the scrollbar.
--
--OUT   PIXEL_COL        Number of pixel columns in the scrollbar.
--
--OUT   PIXEL_ROW        Number of pixel rows in the scrollbar.
-- end formal parameters;


procedure CWN_QUERY_STRING_FIELD_SIZE (
                                  EDITOR_ID:          in   ADDRESS;
                                  PIXEL_COL:          in   ADDRESS;
                                  PIXEL_ROW:          in   ADDRESS);
--
-- CPM description: Returns the number of pixel columns and rows that
--                  an string field editor occupies.
--
-- formal parameters
--IN    EDITOR_ID        ID to attach to the editor.
--
--OUT   PIXEL_COL        Number of pixel columns in the editor.
--
--OUT   PIXEL_ROW        Number of pixel rows in the editor.
-- end formal parameters;


procedure CWN_QUERY_SUBPANEL_SIZE (SUBPANEL_ID:       in   ADDRESS;
                                   PIXEL_COL:         in   ADDRESS;
```

```
                                        PIXEL_ROW:          in   ADDRESS);
--
-- CPM description: Returns the number of pixel columns and rows that
--                  a subpanel requires. The size is determined by using the
--                  locations and sizes of the editors that are attached
--                  to the subpanel.
--
-- formal parameters
--IN    SUBPANEL_ID     ID to attach to the subpanel.
--
--OUT   PIXEL_COL       Number of pixel columns in the window.
--
--OUT   PIXEL_ROW       Number of pixel rows in the window.
-- end formal parameters;


procedure CWN_QUERY_WINDOW_SIZE (WINDOW_ID:   in   ADDRESS;
                                 PIXEL_X:     in   ADDRESS;
                                 PIXEL_Y:     in   ADDRESS;
                                 PIXEL_COL:   in   ADDRESS;
                                 PIXEL_ROW:   in   ADDRESS);
--
-- CPM description: Returns the x and y display coordinates of the upper
--                  left corner of the window and the number of pixel
--                  columns and rows that will fit in a window. If buttons
--                  have been created in a window, it is advisable to query
--                  for window size before creating other window structures.
--
-- formal parameters
--IN    WINDOW_ID       The id of the window whose size is being queried.
--
--OUT   PIXEL_X         X screen coordinate of window origin.
--
--OUT   PIXEL_Y         Y screen coordinate of window origin.
--
--OUT   PIXEL_COL       Number of pixel columns in the window.
--
--OUT   PIXEL_ROW       Number of pixel rows in the window.
-- end formal parameters;


procedure CWN_REMOVE_INPUT_SOCKET (SOCKET_ID:  in   ADDRESS);
--
-- CPM description: CWN_REMOVE_INPUT_SOCKET deletes a socket id to be
--                  watched by CWN_INPUT.
--
-- formal parameters
--IN    SOCKET_ID       ID of the socket to stop watching for input.
-- end formal parameters;


procedure CWN_REMOVE_SYSTEM_MESSAGE;
--
-- CPM description: This routine removes any system message displayed via
--                  cwn_display_system_message.  This should be called
--                  before another system message is displayed.
```

```
--
-- formal parameters
--   None
-- end formal parameters;


procedure CWN_RESIZE_CHECKBOX (CHECKBOX_ID:     in   ADDRESS;
                               PIXEL_COL:       in   ADDRESS;
                               PIXEL_ROW:       in   ADDRESS;
                               PIXEL_WIDTH:     in   ADDRESS;
                               PIXEL_HEIGHT:    in   ADDRESS);
--
-- CPM description: Changes the size of a checkbox button editor.
--
-- formal parameters
--IN    CHECKBOX_ID      ID of the editor.
--
--IN    PIXEL_COL        Column number from within the window where the left
--                       side of the editor shall be placed. Column 0 is at
--                       left of the window.
--
--IN    PIXEL_ROW        Row number from within the window where the top side
--                       of the editor shall be placed. Row 0 is at the top
--                       of the window.
--
--IN    PIXEL_WIDTH      The number of columns to be occupied by the editor.
--
--IN    PIXEL_HEIGHT     The number of rows to be occupied by the editor.
-- end formal parameters;


procedure CWN_RESIZE_EDITOR (EDITOR_ID:         in   ADDRESS;
                             PIXEL_COL:         in   ADDRESS;
                             PIXEL_ROW:         in   ADDRESS;
                             PIXEL_WIDTH:       in   ADDRESS;
                             PIXEL_HEIGHT:      in   ADDRESS);
--
-- CPM description: Changes the size of a window full page text editor.
--
-- formal parameters
--IN    EDITOR_ID        ID of the editor.
--
--IN    PIXEL_COL        Column number from within the window where the left
--                       side of the editor shall be placed. Column 0 is at
--                       left of the window.
--
--IN    PIXEL_ROW        Row number from within the window where the top side
--                       of the editor shall be placed. Row 0 is at the top
--                       of the window.
--
--IN    PIXEL_WIDTH      The number of columns to be occupied   editor.
--
--IN    PIXEL_HEIGHT     The number of rows to be occupied   editor.
-- end formal parameters;
```

```
procedure CWN_RESIZE_NUMBER_FIELD (
                                EDITOR_ID:          in   ADDRESS;
                                PIXEL_COL:          in   ADDRESS;
                                PIXEL_ROW:          in   ADDRESS;
                                PIXEL_WIDTH:        in   ADDRESS;
                                PIXEL_HEIGHT:       in   ADDRESS);
--
-- CPM description: Changes the size of a numeric field editor.
--
-- formal parameters
--IN    EDITOR_ID          ID of the editor.
--
--IN    PIXEL_COL          Column number from within the window where the left
--                         side of the editor shall be placed. Column 0 is at
--                         left of the window.
--
--IN    PIXEL_ROW          Row number from within the window where the top side
--                         of the editor shall be placed. Row 0 is at the top
--                         of the window.
--
--IN    PIXEL_WIDTH        The number of columns to be occupied   editor.
--
--IN    PIXEL_HEIGHT       The number of rows to be occupied   editor.
-- end formal parameters;


procedure CWN_RESIZE_PANEL (PANEL_ID:           in   ADDRESS;
                            PIXEL_COL:          in   ADDRESS;
                            PIXEL_ROW:          in   ADDRESS;
                            PIXEL_WIDTH:        in   ADDRESS;
                            PIXEL_HEIGHT:       in   ADDRESS);
--
-- CPM description: Changes the size of a window panel.
--
-- formal parameters
--IN    PANEL_ID           ID attached to the panel.
--
--IN    PIXEL_COL          Column number from within the window where the left
--                         side of the panel shall be placed. Column 0 is at
--                         left of the window.
--
--IN    PIXEL_ROW          Row number from within the window where the top side
--                         of the panel shall be placed. Row 0 is at the top
--                         of the window.
--
--IN    PIXEL_WIDTH        The number of columns to be occupied   panel.
--
--IN    PIXEL_HEIGHT       The number of rows to be occupied   panel.
-- end formal parameters;


procedure CWN_RESIZE_PUSHBUTTON (
                            PUSHBUTTON_ID:      in   ADDRESS;
                            PIXEL_COL:          in   ADDRESS;
                            PIXEL_ROW:          in   ADDRESS;
                            PIXEL_WIDTH:        in   ADDRESS;
```

```
                                PIXEL_HEIGHT:       in   ADDRESS);
   --
   -- CPM description: Changes the size of a pushbutton editor.
   --
   -- formal parameters
   --IN     PUSHBUTTON_ID     ID of the pushbutton editor.
   --
   --IN     PIXEL_COL         Column number from within the window where the left
   --                         side of the editor shall be placed. Column 0 is at
   --                         left of the window.
   --
   --IN     PIXEL_ROW         Row number from within the window where the top side
--                          of the editor shall be placed. Row 0 is at the top
   --                         of the window.
   --
   --IN     PIXEL_WIDTH       The number of columns to be occupied by the editor.
   --
   --IN     PIXEL_HEIGHT      The number of rows to be occupied by the editor.
   -- end formal parameters;


   procedure CWN_RESIZE_RADIOBUTTON (
                                RADIOBUTTON_ID:     in   ADDRESS;
                                PIXEL_COL:          in   ADDRESS;
                                PIXEL_ROW:          in   ADDRESS;
                                PIXEL_WIDTH:        in   ADDRESS;
                                PIXEL_HEIGHT:       in   ADDRESS);
   --
   -- CPM description: Changes the size of a radiobutton editor.
   --
   -- formal parameters
   --IN     RADIOBUTTON_ID    ID of the radiobutton editor.
   --
   --IN     PIXEL_COL         Column number from within the window where the left
   --                         side of the editor shall be placed. Column 0 is at
   --                         left of the window.
   --
   --IN     PIXEL_ROW         Row number from within the window where the top side
   --                         of the editor shall be placed. Row 0 is at the top
   --                         of the window.
   --
   --IN     PIXEL_WIDTH       The number of columns to be occupied by the editor.
   --
   --IN     PIXEL_HEIGHT      The number of rows to be occupied by the editor.
   -- end formal parameters;


  procedure CWN_RESIZE_STRING_FIELD (
                                EDITOR_ID:          in   ADDRESS;
                                PIXEL_COL:          in   ADDRESS;
                                PIXEL_ROW:          in   ADDRESS;
                                PIXEL_WIDTH:        in   ADDRESS;
                                PIXEL_HEIGHT:       in   ADDRESS);
  --
  -- CPM description: Changes the size of a string field editor.
  --
```

```
-- formal parameters
--IN     EDITOR_ID          ID of the editor.
--
--IN     PIXEL_COL          Column number from within the window where the left
--                          side of the editor shall be placed. Column 0 is at
--                          left of the window.
--
--IN     PIXEL_ROW          Row number from within the window where the top side
--                          of the editor shall be placed. Row 0 is at the top
--                          of the window.
--
--IN     PIXEL_WIDTH        The number of cclumns to be occupied   editor.
--
--IN     PIXEL_HEIGHT       The number of rows to be occupied   editor.
-- end formal parameters;


procedure CWN_RESIZE_WINDOW (WINDOW_ID:      in   ADDRESS;
                             PIXEL_COL:         in   ADDRESS;
                             PIXEL_ROW:         in   ADDRESS;
                             PIXEL_WIDTH:       in   ADDRESS;
                             PIXEL_HEIGHT:      in   ADDRESS);
--
-- CPM description: Changes the size of a window.
--
-- formal parameters
--IN     WINDOW_ID          ID attached to the window.
--
--IN     PIXEL_COL          Column number where the left side of the window
--                          shall be placed.
--
--IN     PIXEL_ROW          Row number where the top side of the window shall
--                          be placed.
--
--IN     PIXEL_WIDTH        The number of columns to be occupied by the
--                          window.
--
--IN     PIXEL_HEIGHT       The number of rows to be occupied by the window.
-- end formal parameters;


procedure CWN_SELECT_INPUT (WINDOW_TYPE:     in   ADDRESS;
                            WINDOW_ID:       in   ADDRESS;
                            MOUSE_BUTTONS:   in   ADDRESS;
                            EXPOSURE:        in   ADDRESS);
--
-- CPM description: This function allows the user to select various mouse
--                  inputs for a particular window and/or exposure events if
--                  the window is the working window.  Each call for the
--                  same window overrides any previous call.  Only the input
--                  selected will be returned to the application, however,
--                  the application must be aware that if the input occurs
--                  within any editor or is an input handled by either the
--                  menu or panel managers, then the application will not be
--                  notified of the input.
--
```

```
-- formal parameters
--IN     WINDOW_TYPE       The type of window for which the input is being
--                         selected for, where:
--                         SYS_WINDOW         = a defined window
--                         SYS_DISPLAY_PANEL  = a displayed panel
--                         SYS_DEFINED_BUTTON = defined button
--
--IN     WINDOW_ID         The id given at the time of the window type's
--                         creation where:
--                         If window_type is SYS_WINDOW and window_id is 0,
--                         then the menu will be activated for the RootWindow
--                         or (Display).  Otherwise, the menu will be activated
--                         for the matching window_id.
--                         If window_type = SYS_DISPLAY_PANEL, the id should
--                         be the panel id.
--                         If window_type = SYS_DEFINED_BUTTON, the id should
--                         be the button id.
--
--IN     MOUSE_BUTTONS     Array of logicals indicating selection of mouse
--                         button operations whose input the application
--                         wishes to be notified of, where:
--                                1 = select
--                                0 = do not select
--                          [0] = Right Button Down
--                          [1] = Middle Button Down
--                          [2] = Left Button Down
--                          [3] = Right Button Up
--                          [4] = Middle Button Up
--                          [5] = Left Button Up
--
--IN     EXPOSURE          Logical indicating whether the application wishes
--                         to be notified of exposure events to the working
--                         window, where:
--                              0 = Do not notify of exposure events
--                              1 = Notify of exposure events
--                         NOTE:  This parameter is valid only for the working
--                         window and is ignored for any other window type.
--
-- end formal parameters;


procedure CWN_SHOW_PANEL (PANEL_ID:       in   ADDRESS);
--
-- CPM description: This procedure displays a panel that has been hidden by
--                  CWN_HIDE_PANEL and enables user input to any of the
--                  panel editors.
--
-- formal parameters
--IN     PANEL_ID          ID attached to the panel to
--                         show.
-- end formal parameters;


procedure CWN_SHOW_SUBPANEL (SUBPANEL_ID:    in   ADDRESS);
--
-- CPM description: This procedure displays a subpanel that has been hidden
```

```
--                  by CWN_HIDE_SUBPANEL and enables user input to any of the
--                  subpanel editors.
-- formal parameters
--IN    SUBPANEL_ID      ID attached to the subpanel to
--                       show.
--
-- end formal parameters;


procedure CWN_SIZE_CHECKBOX (
                          NUM_FIELDS:      in   ADDRESS;
                          NUM_COLS:        in   ADDRESS;
                          LABELS:          in   ADDRESS;
                          LABEL_LENGTH:    in   INTEGER;
                          PIXEL_WIDTH:     in   ADDRESS;
                          PIXEL_HEIGHT:    in   ADDRESS);
--
-- CPM description:  Sizes a checkbox button editor.
--
-- formal parameters
--IN    NUM_FIELDS       The total number of checkbox buttons to be in the
--                       editor.
--
--IN    NUM_COLS         The number of columns the checkbox buttons are to be
--                       arranged in.
--
--IN    LABELS           Pointer to the array of label addresses for all
--                       the checkbox buttons.
--
--IN    LABEL_LENGTH     The maximum length of the labels.
--
--OUT   PIXEL_WIDTH      The number of pixels needed to define the width of
--                       the checkbox editor as specified.
--
--OUT   PIXEL_HEIGHT     The number of pixels needed to define the height of
--                       the checkbox editor as specified.
--
-- end formal parameters;


procedure CWN_SIZE_EDITOR (
                          NUM_COLS:          in   ADDRESS;
                          NUM_ROWS:          in   ADDRESS;
                          EDITOR_WIDTH:      in   ADDRESS;
                          EDITOR_HEIGHT:     in   ADDRESS);
--
-- CPM description:  Sizes a full page text editor.
--
-- formal parameters
--IN    NUM_COLS         The number of columns to be occupied by the editor.
--
--IN    NUM_ROWS         The number of rows to be occupied by the editor.
--
--OUT   EDITOR_WIDTH     The width in pixels required to hold the specified
--                       editor.
--
```

```
--OUT   EDITOR_HEIGHT     The height in pixels of the rectangle required to
--                        hold the specified editor.
--
-- end formal parameters;


procedure CWN_SIZE_NUMBER_FIELD (
                          LABEL:             in       ADDRESS;
                          MAX_CHARACTERS:    in       ADDRESS;
                          PIXEL_WIDTH:       in       ADDRESS;
                          PIXEL_HEIGHT:      in       ADDRESS);
--
-- CPM description:  Returns the size of a Numeric Field editor.
--
-- formal parameters
--IN    LABEL              The optional label before the number field.  This
--                         should be set to NULL if no label will be displayed.
--
--IN    MAX_CHARACTERS     The maximum number of characters which will
--                         be allowed to be entered into the field.
--
--OUT   PIXEL_WIDTH        The width in pixels required to hold the specified
--                         editor.
--
--OUT   PIXEL_HEIGHT       The height in pixels of the rectangle required to
--                         hold the specified editor.
--
-- end formal parameters;


procedure CWN_SIZE_PUSHBUTTON (
                                NUM_FIELDS:      in   ADDRESS;
                                NUM_COLS:        in   ADDRESS;
                                LABELS:          in   ADDRESS;
                                Label_Length:    in   INTEGER;
                                DEFAULT_BUTTON:  in   ADDRESS;
                                PIXEL_WIDTH:     in   ADDRESS;
                                PIXEL_HEIGHT:    in   ADDRESS);
--
-- CPM description: Sizes a Pushbutton editor.
--
-- formal parameters
--IN    NUM_FIELDS         The total number of pushbuttons to be in the
--                         editor.
--
--IN    NUM_COLS           The number of columns the pushbuttons are to be
--                         arranged in.
--
--IN    LABELS             Address of the array of label addresses for all the
--                         pushbuttons.
--
--IN    LABEL_LENGTH       The maximum length of the labels.
--
--IN    DEFAULT_BUTTON     The index into the pushbutton array of the button to
--                         be drawn "active" or displayed as the default
--                         button.  A value of SYS_NO_DEFAULT_BUTTON will
```

```
--                          disable this feature.
--
--OUT    PIXEL_WIDTH         The width in pixels required to hold the specified
--                          editor.
--
--OUT    PIXEL_HEIGHT        The height in pixels of the rectangle required to
--                          hold the specified editor.
--
-- end formal parameters;


procedure CWN_SIZE_RADIOBUTTON (
                                NUM_FIELDS:     in   ADDRESS;
                                NUM_COLS:       in   ADDRESS;
                                LABELS:         in   ADDRESS;
                                PIXEL_WIDTH:    in   ADDRESS;
                                PIXEL_HEIGHT:   in   ADDRESS);
--
-- CPM description:  Sizes a Radiobutton editor.
--
-- formal parameters
--IN     NUM_FIELDS          The total number of radiobuttons to be in the
--                          editor.
--
--IN     NUM_COLS            The number of columns the radiobuttons are to be
--                          arranged in.
--
--IN     LABELS              Address of the array of label addresses for all the
--                          radiobuttons.
--
--OUT    PIXEL_WIDTH         The width in pixels required to hold the specified
--                          editor.
--
--OUT    PIXEL_HEIGHT        The height in pixels of the rectangle required to
--                          hold the specified editor.
--
-- end formal parameters;


procedure CWN_SIZE_STATIC_TEXT (
                              STATIC_TEXT:      in   ADDRESS;
                              TEXT_ALIGNMENT:   in   ADDRESS;
                              PIXEL_WIDTH:      in   ADDRESS;
                              PIXEL_HEIGHT:     in   ADDRESS);
--
-- CPM description:  Sizes a static text editor.
--
-- formal parameters
--IN     STATIC_TEXT         Textual string to display in the button.
--
--IN     TEXT_ALIGNMENT      Alignment of the text within the static text area
--                          (CENTER_ALIGNED, LEFT_ALIGNED, RIGHT_ALIGNED,
--                           NO_ALIGNMENT)
--
--OUT    PIXEL_WIDTH         The width in pixels required to hold the specified
--                          editor.
```

```
--
--OUT   PIXEL_HEIGHT      The height in pixels of the rectangle required to
--                        hold the specified editor.
--
-- end formal parameters;


procedure CWN_SIZE_STRING_FIELD (
                              LABEL:           in      ADDRESS;
                              MAX_CHARACTERS:  in      ADDRESS;
                              PIXEL_WIDTH:     in      ADDRESS;
                              PIXEL_HEIGHT:    in      ADDRESS);
--
-- CPM description: Sizes a String Field editor.
--
-- formal parameters
--IN    LABEL              The optional label before the string field.  This
--                         should be set to NULL if no label will be displayed.
--
--IN    MAX_CHARACTERS     The maximum number of characters which will
--                         be allowed to be entered into the field.
--
--OUT   PIXEL_WIDTH        The width in pixels required to hold the specified
--                         editor.
--
--OUT   PIXEL_HEIGHT       The height in pixels of the rectangle required to
--                         hold the specified editor.
--
-- end formal parameters;


procedure CWN_TERMINATE_WINDOW;
--
-- CPM description: This procedure terminates the window system. It must be
--                  called to free the slot in the icon stack assigned to the
--                  window when it was created.
-- formal parameters
--      None
-- end formal parameters;


procedure CWN_TOGGLE_BUTTON (BUTTON_ID:      in   ADDRESS;
                             BUTTON_LABEL:   in   ADDRESS);
--
-- CPM description: This procedure toggles the state of a button and
--                  optionally relabels it.
-- formal parameters
--IN    BUTTON_ID          ID attached to the button to
--                         toggle.
--
--IN    BUTTON_LABEL       An optional new label for the button.  If this is
--                         set to NULL, then the original label will remain.
-- end formal parameters;


procedure CWN_UNMAP_WINDOW (WINDOW_ID:   in   ADDRESS);
```

```
--
-- CPM description: Routine to unmap a created window.  Any child
--                  window will no longer be visible until another map
--                  call is made on the parent via CWN_MAP_WINDOW.
--
-- formal parameters
--IN   WINDOW_ID    The id of the window to be unmapped.
--
-- end formal parameters;


procedure CWN_UPDATE_PANEL (PANEL_ID:   in   ADDRESS);
--
-- CPM description: Causes a panel to update its structures with additions
--                  or deletions of editors.
--
-- formal parameters
--IN    PANEL_ID          ID to attach to the panel.
--                        This ID is required for all interactions with the
--                        panel.
-- end formal parameters;


procedure CWN_USER_INPUT_FIELD (Field_Type :         in   ADDRESS;
                                Input_String :       in   ADDRESS;
                                Max_String_Size :    in   ADDRESS;
                                Opt_Label :          in   ADDRESS;
                                X_Pixel :            in   ADDRESS;
                                Y_Pixel :            in   ADDRESS);
--
-- CPM description: This puts up an editing field for user input of
--                  alphanumeric or numeric strings anywhere within the
--                  display screen.
--
-- formal parameters
--IN    Field_Type       The type of field to be defined and used:
--                       SYS_STRING_FIELD
--                       SYS_NUMBER_FIELD
--
--IN    Input_String     The address of the variable which will
--                       receive the user input.  This variable may be
--                       initialized to some value, which would
--                       be displayed.  This must be a NULL terminated
--                       string.
--
--IN    Max_String_Size  The maximum string size allowed for input. The
--                       field will be defined according to this size.
--
--IN    Opt_Label        The optional label (prompt or string) which the
--                       application wishes to be displayed on the left side
--                       of the input field.
--
--IN    X_Pixel          The x screen pixel where the upper left corner of
--                       the field will be placed.
--
--IN    Y_Pixel          The y screen pixel of the display where the upper
```

```
   --                          left corner of the input field will be placed.
   --
   -- end formal parameters;

private

   pragma INTERFACE (C, CWN_ACTIVATE_EDITOR);
   pragma INTERFACE (C, CWN_ACTIVATE_MENU);
   pragma INTERFACE (C, CWN_ACTIVATE_NUMBER_FIELD);
   pragma INTERFACE (C, CWN_ACTIVATE_STRING_FIELD);
   pragma INTERFACE (C, CWN_ADD_INPUT_SOCKET);
   pragma INTERFACE (C, CWN_CHANGE_BUTTON_LABEL);
   pragma INTERFACE (C, CWN_CHANGE_CHECKBOX_STATES);
   pragma INTERFACE (C, CWN_CHANGE_EDITOR_TEXT);
   pragma INTERFACE (C, CWN_CHANGE_ICON_LABEL);
   pragma INTERFACE (C, CWN_CHANGE_SCROLLBAR);
   pragma INTERFACE (C, CWN_CHANGE_WINDOW_LABEL);
   pragma INTERFACE (C, CWN_CLEAR_WINDOW);
   pragma INTERFACE (C, CWN_CLOSE_WINDOW);
   pragma INTERFACE (C, CWN_CREATE_EXPOSURE_EVENT);
   pragma INTERFACE (C, CWN_CREATE_SUBWINDOW);
   pragma INTERFACE (C, CWN_CREATE_WINDOW);
   pragma INTERFACE (C, CWN_DEACTIVATE_MENU);
   pragma INTERFACE (C, CWN_DEFINE_BUTTON);
   pragma INTERFACE (C, CWN_DEFINE_CHECKBOX);
   pragma INTERFACE (C, CWN_DEFINE_EDITOR);
   pragma INTERFACE (C, CWN_DEFINE_NUMBER_FIELD);
   pragma INTERFACE (C, CWN_DEFINE_PANEL);
   pragma INTERFACE (C, CWN_DEFINE_POPUP_MENU);
   pragma INTERFACE (C, CWN_DEFINE_POPUP_WINDOW);
   pragma INTERFACE (C, CWN_DEFINE_PUSHBUTTON);
   pragma INTERFACE (C, CWN_DEFINE_RADIOBUTTON);
   pragma INTERFACE (C, CWN_DEFINE_SCROLLBAR);
   pragma INTERFACE (C, CWN_DEFINE_STATIC_TEXT);
   pragma INTERFACE (C, CWN_DEFINE_STRING_FIELD);
   pragma INTERFACE (C, CWN_DEFINE_SUBPANEL);
   pragma INTERFACE (C, CWN_DELETE_BUTTON);
   pragma INTERFACE (C, CWN_DELETE_CHECKBOX);
   pragma INTERFACE (C, CWN_DELETE_EDITOR);
   pragma INTERFACE (C, CWN_DELETE_MENU);
   pragma INTERFACE (C, CWN_DELETE_NUMBER_FIELD);
   pragma INTERFACE (C, CWN_DELETE_PANEL);
   pragma INTERFACE (C, CWN_DELETE_POPUP_WINDOW);
   pragma INTERFACE (C, CWN_DELETE_PUSHBUTTON);
   pragma INTERFACE (C, CWN_DELETE_RADIOBUTTON);
   pragma INTERFACE (C, CWN_DELETE_SCROLLBAR);
   pragma INTERFACE (C, CWN_DELETE_STATIC_TEXT);
   pragma INTERFACE (C, CWN_DELETE_STRING_FIELD);
   pragma INTERFACE (C, CWN_DELETE_SUBPANEL);
   pragma INTERFACE (C, CWN_DELETE_SUBWINDOW);
   pragma INTERFACE (C, CWN_DISPLAY_SYSTEM_MESSAGE);
   pragma INTERFACE (C, CWN_END_PANEL);
   pragma INTERFACE (C, CWN_END_SUBPANEL);
   pragma INTERFACE (C, CWN_HANDLE_WINDOW_MOVE);
   pragma INTERFACE (C, CWN_HIDE_PANEL);
```

```
pragma INTERFACE (C, CWN_HIDE_SUBPANEL);
pragma INTERFACE (C, CWN_INITIALIZE_WINDOW_SYSTEM);
pragma INTERFACE (C, CWN_INPUT);
pragma INTERFACE (C, CWN_MAP_WINDOW);
pragma INTERFACE (C, CWN_MESSAGE_BOX);
pragma INTERFACE (C, CWN_MOVE_BUTTON);
pragma INTERFACE (C, CWN_MOVE_CHECKBOX);
pragma INTERFACE (C, CWN_MOVE_EDITOR);
pragma INTERFACE (C, CWN_MOVE_NUMBER_FIELD);
pragma INTERFACE (C, CWN_MOVE_PANEL);
pragma INTERFACE (C, CWN_MOVE_POPUP_WINDOW);
pragma INTERFACE (C, CWN_MOVE_PUSHBUTTON);
pragma INTERFACE (C, CWN_MOVE_RADIOBUTTON);
pragma INTERFACE (C, CWN_MOVE_SCROLLBAR);
pragma INTERFACE (C, CWN_MOVE_STATIC_TEXT);
pragma INTERFACE (C, CWN_MOVE_STRING_FIELD);
pragma INTERFACE (C, CWN_MOVE_SUBWINDOW);
pragma INTERFACE (C, CWN_MOVE_WINDOW);
pragma INTERFACE (C, CWN_OPEN_ICON);
pragma INTERFACE (C, CWN_POST_MENU);
pragma INTERFACE (C, CWN_QUERY_CHECKBOX_RECTS);
pragma INTERFACE (C, CWN_QUERY_CHECKBOX_SIZE);
pragma INTERFACE (C, CWN_QUERY_DISPLAY_SIZE);
pragma INTERFACE (C, CWN_QUERY_EDITOR_SIZE);
pragma INTERFACE (C, CWN_QUERY_FONT_SIZE);
pragma INTERFACE (C, CWN_QUERY_NUMBER_FIELD_SIZE);
pragma INTERFACE (C, CWN_QUERY_PANEL_ORIGIN);
pragma INTERFACE (C, CWN_QUERY_PANEL_SIZE);
pragma INTERFACE (C, CWN_QUERY_PUSHBUTTON_RECTS);
pragma INTERFACE (C, CWN_QUERY_PUSHBUTTON_SIZE);
pragma INTERFACE (C, CWN_QUERY_RADIOBUTTON_RECTS);
pragma INTERFACE (C, CWN_QUERY_RADIOBUTTON_SIZE);
pragma INTERFACE (C, CWN_QUERY_SCROLLBAR_SIZE);
pragma INTERFACE (C, CWN_QUERY_STRING_FIELD_SIZE);
pragma INTERFACE (C, CWN_QUERY_SUBPANEL_SIZE);
pragma INTERFACE (C, CWN_QUERY_WINDOW_SIZE);
pragma INTERFACE (C, CWN_REMOVE_INPUT_SOCKET);
pragma INTERFACE (C, CWN_RESIZE_CHECKBOX);
pragma INTERFACE (C, CWN_RESIZE_EDITOR);
pragma INTERFACE (C, CWN_REMOVE_SYSTEM_MESSAGE);
pragma INTERFACE (C, CWN_RESIZE_NUMBER_FIELD);
pragma INTERFACE (C, CWN_RESIZE_PANEL);
pragma INTERFACE (C, CWN_RESIZE_PUSHBUTTON);
pragma INTERFACE (C, CWN_RESIZE_RADIOBUTTON);
pragma INTERFACE (C, CWN_RESIZE_STRING_FIELD);
pragma INTERFACE (C, CWN_RESIZE_WINDOW);
pragma INTERFACE (C, CWN_SELECT_INPUT);
pragma INTERFACE (C, CWN_SHOW_PANEL);
pragma INTERFACE (C, CWN_SHOW_SUBPANEL);
pragma INTERFACE (C, CWN_SIZE_CHECKBOX);
pragma INTERFACE (C, CWN_SIZE_EDITOR);
pragma INTERFACE (C, CWN_SIZE_NUMBER_FIELD);
pragma INTERFACE (C, CWN_SIZE_PUSHBUTTON);
pragma INTERFACE (C, CWN_SIZE_RADIOBUTTON);
pragma INTERFACE (C, CWN_SIZE_STATIC_TEXT);
pragma INTERFACE (C, CWN_SIZE_STRING_FIELD);
```

```
   pragma INTERFACE (C, CWN_TERMINATE_WINDOW);
   pragma INTERFACE (C, CWN_TOGGLE_BUTTON);
   pragma INTERFACE (C, CWN_UNMAP_WINDOW);
   pragma INTERFACE (C, CWN_UPDATE_PANEL);
   pragma INTERFACE (C, CWN_USER_INPUT_FIELD);

end CWN_WINDOW_SYSTEM;
```

APPENDIX D - EDDIC DATA BASES

This appendix describes the format of the EDDIC Sun-based data bases. Table D-1 lists the Sun-based data bases. This appendix also includes the record layouts for the data bases.

Table D-1. EDDIC Sun-Based Data Bases

Data Base Name	Description
BLUEFOR_AMMO_SOURCE	Initial Ammunition levels for BLUEFOR units. (ASCII format).
BLUEFOR_AMMO_TRACK	List of ammunition types to include in the graphical unit status report (ASCII format).
BLUEFOR_ASSET_UNIT	List of BLUEFOR units that have initial levels of assets assigned to them (ASCII format).
BLUEFOR_AUTH_AMMO_INDEX	Index file for the BLUEFOR authorized ammunition levels data base (Ada format).
BLUEFOR_AUTH_AMMO	BLUEFOR authorized ammunition levels (Ada format).
BLUEFOR_AUTH_EQUIP_INDEX	Index file for the BLUEFOR authorized equipment levels data base (Ada format).
BLUEFOR_AUTH_EQUIP	BLUEFOR authorized equipment levels (Ada format).
BLUEFOR_CM_EDIT_MENU	Description of the walking menu to display when a BLUEFOR control measure is selected on the tactical map in a window with edit capability (ASCII format).
BLUEFOR_CM_VIEW_MENU	Description of the walking menu to display when a BLUEFOR control measure is selected on the tactical map in a window with view only capability (ASCII format).
BLUEFOR_CURR_AMMO	BLUEFOR current ammunition levels (Ada format).
BLUEFOR_CURR_AMMO_INDEX	Index file for the BLUEFOR current ammunition levels data base (Ada format).
BLUEFOR_CURR_EQUIP_INDEX	Index file for the BLUEFOR current equipment levels data base (Ada format).
BLUEFOR_CURR_EQUIP	BLUEFOR current equipment levels (Ada format).

BLUEFOR_EQUIP_SOURCE	Initial equipment levels for BLUEFOR units (ASCII format).
BLUEFOR_EQUIP_TRACK	List of BLUEFOR equipment types to include in the graphical unit status report (ASCII format).
BLUEFOR_FUEL	BLUEFOR authorized and current fuel levels (Ada format).
BLUEFOR_FUEL_INDEX	Index file for the BLUEFOR fuel level data base (Ada format).
BLUEFOR_FUEL_SOURCE	Initial fuel levels for BLUEFOR units (ASCII format).
BLUEFOR_OBS_EDIT_MENU	Description of the walking menu to display when a BLUEFOR obstacle is selected on the tactical map in a window with view only capability (ASCII format).
BLUEFOR_OBS_VIEW_MENU	Description of the walking menu to display when a BLUEFOR obstacle is selected on the tactical map in a window with edit capability (ASCII format).
BLUEFOR_ORGANIC_TASK_ORG	Organic task organization for the BLUEFOR units (ASCII format).
BLUEFOR_PERSONNEL	BLUEFOR authorized and current personnel levels (Ada format).
BLUEFOR_PERSONNEL_INDEX	Index file for the BLUEFOR personnel level data base (Ada format).
BLUEFOR_PERSONNEL_SOURCE	Initial personnel levels for BLUEFOR units (ASCII format).
BLUEFOR_TASK_ORG_SOURCE	Initial task organization and status for the BLUEFOR units (ASCII format).
BLUEFOR_UNIT_CONVERT	Data base to convert BLUEFOR unit names to unit numbers (Ada format).
BLUEFOR_UNIT_LOC_INDEX	Index file for the BLUEFOR unit location data base (Ada format).
BLUEFOR_UNIT_LOC_SOURCE	Initial unit locations for the BLUEFOR units (ASCII format).
BLUEFOR_UNIT_LOC	BLUEFOR unit location data base (Ada format).

BLUEFOR_UNIT_EDIT_MENU	Description of the walking menu to display when a BLUEFOR unit is selected on a tactical map in a window with edit capability (ASCII format).
BLUEFOR_UNIT_NAME	List of the BLUEFOR unit names. This file is used to assign names to the unit transactions in the situation recorded data (ASCII format).
BLUEFOR_UNIT_STATUS	BLUEFOR unit status (Ada format).
BLUEFOR_UNIT_STATUS_INDEX	Index file for the BLUEFOR unit status data base (Ada format).
BLUEFOR_UNIT_VIEW_MENU	Description of the walking menu to display when a BLUEFOR unit is selected on the tactical map in a window with view only capability (ASCII format).
C2_PRODUCT	Command and control product data base. Includes the products in the view situation, build and view message windows (Ada format).
C2_PRODUCT_DESC	Command and control product description data base. This data base indicates which record from the C2_PRODUCT data base to use for a product (Ada format).
C2_PRODUCT_HEADER	Command and control report headers. The report headers only applies to those products in the view situation window (Ada format).
C2_PRODUCT_NAME	List of the command and control product names. This file is used to assign names to the command and control transactions in the C2 product recorded data (ASCII format).
C2_PRODUCT_RECORD	Command and control data recording transactions (Ada format).
C2_PRODUCT_SOURCE	Description of the command and control products to include in the view situation and build windows (ASCII format).
CNTRL_MSR_POINT	Point control measures (Ada format).
CNTRL_MSR_POINT_INDEX	Index file for the point control measure data base (Ada format).
CNTRL_MSR_POINT_NAME	List of the point control measure names. This file is used to assign names to the point control measure transactions in the situation recorded data (ASCII format).

CONTOUR_1TO160	Map contour image file for the 1:160,000 map scale (Binary format).
CONTOUR_1TO400	Map contour image file for the 1:400,000 map scale (Binary format).
CONTOUR_1TO80	Map contour image file for the 1:80,000 map scale (Binary format).
CONTOUR_1TO800	Map contour image file for the 1:800,000 map scale (Binary format).
CONTOUR_DESC	Description of the contour files to include in the tactical map system (ASCII format).
CONTOUR_DESC_1TO160	Description of the 1:160,000 contour image file (ASCII format).
CONTOUR_DESC_1TO400	Description of the 1:400,000 contour image file (ASCII format).
CONTOUR_DESC_1TO80	Description of the 1:80,000 contour image file (ASCII format).
CONTOUR_DESC_1TO800	Description of the 1:800,000 contour image file (ASCII format).
CONTROL_MEASURE	Control measures (Ada format).
CONTROL_MEASURE_NAME	List of the control measure names. This file is used to assign names to the control measure transactions in the situation recorded data (ASCII format).
CONTROL_MEASURE_SOURCE	Initial control measures (ASCII format).
CONTROL_MEASURE_INDEX	Index file for the control measure data base (Ada format).
ELEVATION_1TO400	Elevation file for the 1:400,000 map scale (Binary format).
ELEVATION_DESC_1TO400	Description of the 1:400,000 elevation file (ASCII format).
ELEV_BAND_1TO160	Elevation band image file for the 1:160,000 map scale (Binary format).
ELEV_BAND_1TO400	Elevation band image file for the 1:400,000 map scale (Binary format).
ELEV_BAND_1TO80	Elevation band image file for the 1:80,000 map scale (Binary format).
ELEV_BAND_1TO800	Elevation band image file for the 1:800,000 map scale (Binary format).

ELEV_BAND_DESC_1TO160	Description of the 1:160,000 elevation band image file (ASCII format).
ELEV_BAND_DESC_1TO400	Description of the 1:400,000 elevation band image file (ASCII format).
ELEV_BAND_DESC_1TO80	Description of the 1:80,000 elevation band image file (ASCII format).
ELEV_BAND_DESC_1TO800	Description of the 1:800,000 elevation band image file (ASCII format).
EXP_CONTROL_MENU	Description of the experiment control product walking menu. This file is created from the product names in the experiment control source file (ASCII format).
EXP_CONTROL_NAME	List of the experiment control product names. This file is used to assign names to the experiment control transactions in the experiment control recorded data (ASCII format).
EXP_CONTROL_PARTICIPANT	List of participants that the experimenter can send experiment control messages to (Ada format).
EXP_CONTROL_PRODUCT	Experiment control products (Ada format).
EXP_CONTROL_PROD_DESC	Experiment control product description data base. This data base indicates which record from the experiment control data base to use for a product (Ada format).
EXP_CONTROL_RECORD	Experiment control data recording transactions (Ada format).
EXP_CONTROL_SOURCE	Description of the products to include in the experiment control window (ASCII format).
FORM_DESCRIPTION	Description and layout of EDDIC form.
G2_BUILD_MENU	Description of the build product walking menu for the G2 workstation. This file is created from the command and control product source file (ASCII format).
G2_REFERENCE_MENU	Description of the reference product walking menu for the G2 workstation. This file is created from the reference product source file (ASCII format).

G2_VIEW_C2_MENU	Description of the view situation product walking menu for the G2 workstation. This file is created from the command and control product source file (ASCII format).
G3_BUILD_MENU	Description of the build product walking menu for the G3 workstation. This file is created from the command and control product source file (ASCII format).
G3_REFERENCE_MENU	Description of the reference product walking menu for the G3 workstation. This file is created from the reference product source file (ASCII format).
G3_VIEW_C2_MENU	Description of the view situation product walking menu for the G3 workstation. This file is created from the command and control product source file (ASCII format).
G4_BUILD_MENU	Description of the build product walking menu for the G4 workstation. This file is created from the command and control product source file (ASCII format).
G4_REFERENCE_MENU	Description of the reference product walking menu for the G4 workstation. This file is created from the reference product source file (ASCII format).
G4_VIEW_C2_MENU	Description of the view situation product walking menu for the G3 workstation. This file is created from the command and control product source file (ASCII format).
HELP_MENU	Description of the help product walking menu for the G3 workstation. This file is created from the help product source file (ASCII format).
HELP_NAME	List of the help product names. This file is used to assign names to the help transactions in the reference recorded data (ASCII format).
HELP_PROD_DESC	Help product description data base. This data base indicates which record from the help product data base to use for a product (Ada format).
HELP_PRODUCT	Help products (Ada format).
HELP_SOURCE	Description of the products to include in the help window (ASCII format).

ICON_STACK_DB	Icon stack status data base. Indicates which stack positions are used and which ones are free (C format).
LUT_HILITE_DESC	Description of the color lookup table files to use when features are hilighted (ASCII format).
LUT_HILITE_MAP_ON	Color lookup table to use when a map background (elevation band, shaded relief, or vegetation) is displayed and map features are hilighted (ASCII format).
LUT_HILITE_MAP_OFF	Color lookup table to use when a map with a null background is displayed and map features are hilighted (ASCII format).
LUT_OVERLAY	Color lookup table for the overlay planes (ASCII format).
LUT_UNHILITE_DESC	Description of the color lookup table files to use when features are not hilighted (ASCII format).
LUT_UNHILITE_MAP_ON	Color lookup table to use when a map background (elevation band, shaded relief, or vegetation) is displayed and map features are not hilighted (ASCII format).
LUT_UNHILITE_MAP_OFF	Color lookup table to use when a map with a null background is displayed and map features are not hilighted (ASCII format).
MAP_BUILD_MENU	Description of the map options walking menu for the build window (ASCII format).
MAP_DESC	Description of the map image files to include in the tactical map system (ASCII format).
MAP_LEGEND	Description of what to display in the map legend (ASCII format).
MAP_MESSAGE_MENU	Description of the map options walking menu for the view message window (ASCII format).
MAP_VIEW_C2_MENU	description of the map options walking menu for the view situation window (ASCII format).
MESSAGE_LOG	Log of all the messages sent (Ada format).
OBSTACLE	Obstacles (Ada format).
OBSTACLE_INDEX	Index for the obstacle data base (Ada format).

OBSTACLE_SOURCE	Initial obstacles (ASCII format).
OPFOR_AUTH_EQUIP	OPFOR authorized equipment levels (Ada format).
OPFOR_AUTH_EQUIP_INDEX	Index file for the OPFOR authorized equipment levels data base (Ada format).
OPFOR_CM_EDIT_MENU	Description of the walking menu to display when a OPFOR control measure is selected on the tactical map in a window with edit capability (ASCII format).
OPFOR_CM_VIEW_MENU	Description of the walking menu to display when a OPFOR control measure is selected on the tactical map in a window with view only capability (ASCII format).
OPFOR_CURR_EQUIP_INDEX	Index file for the OPFOR current equipment levels data base (Ada format).
OPFOR_CURR_EQUIP	OPFOR current equipment levels (Ada format).
OPFOR_EQUIP_NAME	List of the OPFOR equipment names. This file is used to assign names to the equipment types in the situation data base (ASCII format).
OPFOR_EQUIP_SOURCE	Initial equipment levels for OPFOR units (ASCII format).
OPFOR_OBS_EDIT_MENU	Description of the walking menu to display when a OPFOR obstacle is selected on the tactical map in a window with view only capability (ASCII format).
OPFOR_OBS_VIEW_MENU	Description of the walking menu to display when a OPFOR obstacle is selected on the tactical map in a window with edit capability (ASCII format).
OPFOR_ORGANIC_TASK_ORG	Organic task organization for the OPFOR units (ASCII format).
OPFOR_REINFORCE_TIME	Initial reinforcing times for OPFOR units (ASCII format).
OPFOR_TASK_ORG_SOURCE	Initial task organization for the OPFOR units (ASCII format).
OPFOR_UNIT_CONVERT	Data base to convert OPFOR unit names to unit numbers (Ada format).

OPFOR_UNIT_EDIT_MENU	Description of the walking menu to display when a OPFOR unit is selected on a tactical map in a window with edit capability (ASCII format).
OPFOR_UNIT_LOC	OPFOR unit location data base (Ada format).
OPFOR_UNIT_LOC_INDEX	Index file for the OPFOR unit location data base (Ada format).
OPFOR_UNIT_LOC_SOURCE	Initial unit locations for the OPFOR units (ASCII format).
OPFOR_UNIT_NAME	List of the OPFOR unit names. This file is used to assign names to the unit transactions in the situation recorded data (ASCII format).
OPFOR_UNIT_STATUS_INDEX	Index file for the OPFOR unit status data base (Ada format).
OPFOR_UNIT_STATUS	OPFOR unit status (Ada format).
OPFOR_UNIT_STATUS_SOURCE	Initial status of the OPFOR units (ASCII format).
OPFOR_UNIT_VIEW_MENU	Description of the walking menu to display when a OPFOR unit is selected on the tactical map in a window with view only capability (ASCII format).
OPLAN_LIST	List of existing Operational plans in the system (Ada format).
OPLAN_LIST_SOURCE	Operational plans to initially have in the system (ASCII format).
PRODUCT_HARDCOPY	ASCII output file of the products printed by CDB_HARDCOPY.
REFERENCE_HEADER	Reference report headers (Ada format).
REFERENCE_NAME	List of the reference product names. This file is used to assign names to the reference transactions in the reference recorded data (ASCII format).
REFERENCE_PROD_DESC	Reference product description data base. This data base indicates which records from the reference product data base to use for a product (Ada format).
REFERENCE_PRODUCT	Reference product data base (Ada format).

REFERENCE_RECORD	Reference data recording transactions (Ada format).
REFERENCE_SOURCE	Description of the reference products to include in the view reference window (ASCII format).
ROOT_WINDOW_MENU	Description of the walking menu to display in the root window. The root window is any part of the screen where a window or button is not displayed (ASCII format).
SCREEN_DUMP_IMAGE	Bitmap image of a screen of a Sun workstation (Bitmap format).
SEND_PARTICIPANT_SOURCE	List of the participants that messages can be sent to (ASCII format).
SHAD_RELF_1TO160	Shaded relief image file for the 1:160,000 map scale (Binary format).
SHAD_RELF_1TO400	Shaded relief image file for the 1:400,000 map scale (Binary format).
SHAD_RELF_1TO80	Shaded relief image file for the 1:80,000 map scale (Binary format).
SHAD_RELF_1TO800	Shaded relief image file for the 1:800,000 map scale (Binary format).
SHAD_RELF_DESC_1TO160	Description of the 1:160,000 shaded relief image file (ASCII format).
SHAD_RELF_DESC_1TO400	Description of the 1:400,000 shaded relief image file (ASCII format).
SHAD_RELF_DESC_1TO80	Description of the 1:80,000 shaded relief image file (ASCII format).
SHAD_RELF_DESC_1TO800	Description of the 1:800,000 shaded relief image file (ASCII format).
SITUATION_RECORD	Situation data recording transactions (Ada format).
TASK_ORG_TOOL_MENU	Description of the walking menu to display as a popup menu for the task organization tool (ASCII format).
TASK_ORG_TOP_UNIT_MENU	Description of the walking menu to display when the top unit button is selected in the task organization tool (ASCII format).

TASK_ORG_UNIT_MENU	Description of the walking menu to display as a popup menu when a unit is selected in the task organization tool (ASCII format).
TASK_ORG_UNIT_TYPE_MENU	Description of the multiple selection menu to display when the unit type button is selected in the task organization tool (ASCII format).
TOOL_MENU	Description of the walking menu defining the tools available in the tool window (ASCII format).
TRAN_ACTIVITY	Unit activity update recorded transactions (ASCII format).
TRAN_AMMUNITION	Unit ammunition update recorded transactions (ASCII format).
TRAN_BLUEFOR_TASK_ORG	BLUEFOR task organization update recorded transactions (ASCII format).
TRAN_C2_REQUEST	Request for command and control product recorded transactions (ASCII format).
TRAN_C2_WINDOW	View situation, build, and view message window manipulation recorded transactions (ASCII format).
TRAN_CNTRL_MSR_DEL	Control measure delete recorded transactions (ASCII format).
TRAN_CNTRL_MSR_EFF_TIME	Control measure effective time update recorded transactions (ASCII format).
TRAN_CNTRL_MSR_LOC	Control measure location update recorded transactions (ASCII format).
TRAN_CNTRL_MSR_STAT	Control measure status update recorded transactions (ASCII format).
TRAN_CONTROL_REQUEST	Request for experiment control product recorded transactions (ASCII format).
TRAN_CONTROL_WINDOW	Tool and experiment control window manipulation recorded transactions (ASCII format).
TRAN_EQUIPMENT	Unit equipment update recorded transactions (ASCII format).
TRAN_FUEL	Unit fuel update recorded transactions (ASCII format).
TRAN_LOOKUP_TABLE	Color lookup table update recorded transactions (ASCII format).

TRAN_MAP	Tactical map control recorded transactions (ASCII format).
TRAN_NEW_C2	New command and control product recorded transactions (ASCII format).
TRAN_NEW_CNTRL_MSR	New control measure recorded transactions (ASCII format).
TRAN_NEW_OBSTACLE	New obstacle recorded transactions (ASCII format).
TRAN_OBSTACLE_DEL	Obstacle delete recorded transactions (ASCII format).
TRAN_OBSTACLE_EFF_TIME	Obstacle effective time update recorded transactions (ASCII format).
TRAN_OBSTACLE_LOC	Obstacle location update recorded transactions (ASCII format).
TRAN_OBSTACLE_STATUS	Obstacle status update recorded transactions (ASCII format).
TRAN_OPFOR_REINFORCE	OPFOR unit reinforcing time update recorded transactions (ASCII format).
TRAN_OPFOR_STRENGTH	OPFOR unit strength update recorded transactions (ASCII format).
TRAN_OPFOR_TASK_ORG	OPFOR task organization update recorded transactions (ASCII format).
TRAN_PERSONNEL	Unit personnel update recorded transactions (ASCII format).
TRAN_REF_REQUEST	Request for reference product recorded transaction (ASCII format).
TRAN_REF_WINDOW	View reference window manipulation recorded transactions (ASCII format).
TRAN_SITUATION_REQUEST	Request for situation data recorded transactions (ASCII format).
TRAN_SITUATION_WINDOW	Window manipulation recorded transactions (ASCII format).
TRAN_UNIT_MISSION	Unit mission update recorded transactions (ASCII format).
TRAN_UNIT_LOCATION	Unit location update recorded transactions (ASCII format).
VEGETATION_1TO160	Vegetation image file for the 1:160,000 map scale (Binary format).

VEGETATION_1TO400	Vegetation image file for the 1:400,000 map scale (Binary format).
VEGETATION_1TO80	Vegetation image file for the 1:80,000 map scale (Binary format).
VEGETATION_1TO800	Vegetation image file for the 1:800,000 map scale (Binary format).
VEGETATION_DESC_1TO160	Description of the 1:160,000 vegetation image file (ASCII format).
VEGETATION_DESC_1TO400	Description of the 1:400,000 vegetation image file (ASCII format).
VEGETATION_DESC_1TO80	Description of the 1:80,000 vegetation image file (ASCII format).
VEGETATION_DESC_1TO800	Description of the 1:800,000 vegetation image file (ASCII format).

The following section describes the record layout of the EDDIC Sun-based data bases.

DATA BASE: BLUEFOR_AMMO_SOURCE

TYPE: VARIABLE ASCII

Description
Initial Ammunition levels for BLUEFOR units.

Column	Field Name	Type	Width	Dec
Record 1 (Unit)				
14	Unit Name	Character	12	
35	Number of Ammunition Types	Numeric	2	0
Record 2 (Ammo)				
15	Ammunition Name	Character	12	
30	Authorized Amount	Numeric	6	0
40	On-Hand Amount	Numeric	6	0

Note: The Ammo records must immediately follow the Unit record. The number of Ammo records must equal the number of ammunition types in the Unit record. A date/time record is used to assign a date/time to the ammunition data. Format: *DDHHMM MON starting in column 1. (Example: *021800 SEP). The date/time record is followed by two comment records.

DATA BASE: BLUEFOR_AMMO_TRACK

TYPE: FIXED ASCII

Description
List of ammunition types to include in the graphical unit status report.

Column	Field Name	Type	Width	Dec
1	Ammunition Name	Character	12	
15	Track (True or False)	Boolean	5	

Note: The first record on this file is a comment.

DATA BASE: BLUEFOR_ASSET_UNIT

TYPE: FIXED ASCII

Description
List of BLUEFOR units that have initial levels of assets assigned to them.

Column	Field Name	Type	Width	Dec
3	Unit Name	Character	12	

Note: A date/time record is used to assign a date/time to the units. Format: *DDHHMM MON starting in column 1. (Example: *021800 SEP)

DATA BASE: BLUEFOR_AUTH_AMMO_INDEX

TYPE: Ada

Description
Index file for the BLUEFOR authorized ammunition levels data base.

```
type SDB_BLUEFOR_AMMO_PTR is
   record
      SDB_UNIT_ID       :       SDB_BLUEFOR_UNIT_ID;
      SDB_TIME          :       SYS_DATE_TIME;
      SDB_OPPLAN        :       SYS_OPPLAN;
      SDB_RECORD        :       SYS_DB_SIZE;
   end record;
```

DATA BASE: BLUEFOR_AUTH_AMMO

TYPE: Ada

Description
BLUEFOR authorized ammunition levels.

```
type SDB_AMMO_REC is
   record
```

```
        SDB_ID              :       SDB_AMMUNITION;
        SDB_NAME            :       string (SDB_AMMO_NAME_LEN);
        SDB_BASIC_LOAD      :       SYS_QUANTITY;
        SDB_KEY_ITEM        :       BOOLEAN;
     end record;

  type SDB_AMMO_ARRAY is array (SDB_AMMUNITION) of
                                  SDB_AMMO_REC;

  type SDB_AMMO_AUTH_LIST is
     record
        SDB_UNIT_ID         :       SDB_UNIT;
        SDB_TIME            :       SYS_DATE_TIME;
        SDB_OPPLAN          :       SYS_OPPLAN;
        SDB_COUNT           :       SDB_AMMUNITION;
        SDB_LIST            :       SDB_AMMO_ARRAY;
     end record;
```

DATA BASE: BLUEFOR_AUTH_EQUIP_INDEX

TYPE: Ada

Description

Index file for the BLUEFOR authorized equipment levels data base.

```
  type SDB_BLUEFOR_EQUIP_PTR is
     record
        SDB_UNIT_ID         :       SDB_BLUEFOR_UNIT_ID;
        SDB_TIME            :       SYS_DATE_TIME;
        SDB_OPPLAN          :       SYS_OPPLAN;
        SDB_RECORD          :       SYS_DB_SIZE;
     end record;
```

DATA BASE: BLUEFOR_AUTH_EQUIP

TYPE: Ada

Description

BLUEFOR authorized equipment levels.

```
  type SDB_EQUIP_REC is
     record
        SDB_ID              :       SDB_EQUIPMENT;
        SDB_NAME            :       string (SDB_EQUIP_NAME_LEN);
        SDB_AUTHORIZED      :       SYS_QUANTITY;
        SDB_CATEGORY        :       SDB_EQUIP_CATEGORY;
     end record;

  type SDB_EQUIP_ARRAY is array (SDB_EQUIPMENT) of
                                  SDB_EQUIP_REC;

  type SDB_EQUIP_AUTH_LIST is
     record
        SDB_UNIT_ID         :       SDB_UNIT;
```

```
    SDB_TIME            :       SYS_DATE_TIME;
    SDB_OPPLAN          :       SYS_OPPLAN;
    SDB_COUNT           :       SDB_EQUIPMENT;
    SDB_LIST            :       SDB_EQUIP_ARRAY;
end record;
```

DATA BASE: BLUEFOR_CM_EDIT_MENU

TYPE: FIXED ASCII

Description
Description of the walking menu to display when a BLUEFOR control measure is selected on the tactical map in a window with edit capability.

Column	Field Name	Type	Width	Dec
3	Menu Option Title	Character	20	
35	Control Measure Option (SYS_CM_OPTION)	Character	16	

Note: Submenu Menu Option Titles must be indented 1 character from their parent and must appear immediately after the parent. Using a question mark "?" as the Menu Option Title will cause a blank item in the menu. The first record of this file is a comment.

DATA BASE: BLUEFOR_CM_VIEW_MENU

TYPE: FIXED ASCII

Description
Description of the walking menu to display when a BLUEFOR control measure is selected on the tactical map in a window with view only capability.

Column	Field Name	Type	Width	Dec
3	Menu Option Title	Character	20	
35	Control Measure Option (SYS_CM_OPTION)	Character	16	

Note: Submenu Menu Option Titles must be indented 1 character from their parent and must appear immediately after the parent. Using a question mark "?" as the Menu Option Title will cause a blank item in the menu. The first record of this file is a comment.

DATA BASE: BLUEFOR_CURR_AMMO

TYPE: Ada

Description
BLUEFOR current ammunition levels.

```
type SDB_BLUEFOR_AMMO_QTY is
   record
      SDB_UNIT_ID       :       SDB_BLUEFOR_UNIT_ID;
      SDB_AMMO_ID       :       SDB_BLUEFOR_AMMO_ID;
      SDB_TIME          :       SYS_DATE_TIME;
      SDB_OPPLAN        :       SYS_OPPLAN;
      SDB_ON_HAND       :      SYS_QUANTITY;
   end record;
```

DATA BASE: BLUEFOR_CURR_AMMO_INDEX

TYPE: Ada

Description
Index file for the BLUEFOR current ammunition levels data base.

```
type SDB_BLUEFOR_AMMO_QTY_PTR is
   record
      SDB_UNIT_ID        :       SDB_BLUEFOR_UNIT_ID;
      SDB_AMMO_ID        :       SDB_BLUEFOR_AMMO_ID;
      SDB_TIME           :       SYS_DATE_TIME;
      SDB_OPPLAN         :       SYS_OPPLAN;
      SDB_RECORD         :       SYS_DB_SIZE;
   end record;
```

DATA BASE: BLUEFOR_CURR_EQUIP_INDEX

TYPE: Ada

Description
Index file for the BLUEFOR current equipment levels data base.

```
type SDB_BLUEFOR_EQUIP_QTY_PTR is
   record
      SDB_UNIT_ID        :       SDB_BLUEFOR_UNIT_ID;
      SDB_EQUIP_ID       :       SDB_BLUEFOR_EQUIP_ID;
      SDB_TIME           :       SYS_DATE_TIME;
      SDB_OPPLAN         :       SYS_OPPLAN;
      SDB_RECORD         :       SYS_DB_SIZE;
   end record;
```

DATA BASE: BLUEFOR_CURR_EQUIP

TYPE: Ada

Description
BLUEFOR current equipment levels.

```
type SDB_EQUIP_OPER_LIST is  array (SDB_EQUIPMENT)
                                   of SYS_QUANTITY;

type SDB_EQUIP_OPER_REC is
   record
      SDB_UNIT_ID          :  SDB_UNIT;
      SDB_TIME             :  SYS_DATE_TIME;
      SDB_SIDE             :  SDB_SIDE_TYPE;
      SDB_NUMBER_TYPES     :  SDB_EQUIPMENT;
      SDB_LIST             :  SDB_EQUIP_OPER_LIST;
end record;
```

DATA BASE: BLUEFOR_EQUIP_SOURCE

TYPE: VARIABLE ASCII

Description
Initial equipment levels for BLUEFOR units.

Column	Field Name	Type	Width	Dec
Record 1 (Unit)				
14	Unit Name	Character	12	
35	Number of Equipment Types	Numeric	2	0
Record 2 (Equip)				
15	Equipment Name	Character	12	
30	Authorized Amount	Numeric	5	0
40	Operational Amount	Numeric	5	0

Note: The Equip records must immediately follow the Unit record. The number of equip records must equal the number of equipment types in the Unit record. A date/time record is used to assign a date/time to the ammunition data. Format: *DDHHMM MON starting in column 1. (Example: *021800 SEP). The date/time record is followed by two comment records.

DATA BASE: BLUEFOR_EQUIP_TRACK

TYPE: FIXED ASCII

Description
List of BLUEFOR equipment types to include in the graphical unit status report.

Column	Field Name	Type	Width	Dec
1	Equipment Name	Character	12	
15	Equipment Category C3_SYSTEM PACING_ITEM SUPPORT_SYSTEM OTHER_ITEM	Character	14	

Note: The first record on this file is a comment.

DATA BASE: BLUEFOR_FUEL

TYPE: Ada

Description
BLUEFOR authorized and current fuel levels.

```
type SDB_FUELS is
   record
      SDB_UNIT_ID        :       SDB_BLUEFOR_UNIT_ID;
      SDB_TIME           :       SYS_DATE_TIME;
      SDB_OPPLAN         :       SYS_OPPLAN;
      SDB_MOGAS_REQ      :       SYS_QUANTITY  range 0..999999;
      SDB_MOGAS_ON_HAND  :       SYS_QUANTITY  range 0..999999;
      SDB_AVGAS_REQ      :       SYS_QUANTITY  range 0..999999;
      SDB_AVGAS_ON_HAND  :       SYS_QUANTITY  range 0..999999;
      SDB_DIESEL_REQ     :       SYS_QUANTITY  range 0..999999;
      SDB_DIESEL_ON_HAND:        SYS_QUANTITY  range 0..999999;
   end record;
```

DATA BASE: BLUEFOR_FUEL_INDEX

TYPE: Ada

Description
Index file for the BLUEFOR fuel level data base.

```
type SDB_BLUEFOR_FUEL_PTR is
   record
      SDB_UNIT_ID        :       SDB_BLUEFOR_UNIT_ID;
      SDB_TIME           :       SYS_DATE_TIME;
      SDB_OPPLAN         :       SYS_OPPLAN;
      SDB_RECORD         :       SYS_DB_SIZE;
   end record;
```

DATA BASE: BLUEFOR_FUEL_SOURCE

TYPE: FIXED ASCII

Description
Initial fuel levels for BLUEFOR units.

Column	Field Name	Type	Width	Dec
7	Unit Name	Character	12	
22	Authorized Diesel	Numeric	6	0
30	Current Diesel	Numeric	6	0
39	Authorized MOGAS	Numeric	6	0
47	Current MOGAS	Numeric	6	0
56	Authorized AVGAS	Numeric	6	0
64	Current AVGAS	Numeric	6	0

Note: A date/time record is used to assign a date/time to the fuel data. Format: *DDHHMM MON starting in column 1. (Example: *021800 SEP). The date/time record is followed by two comment records.

DATA BASE: BLUEFOR_OBS_EDIT_MENU

TYPE: FIXED ASCII

Description
Description of the walking menu to display when a BLUEFOR obstacle is selected on the tactical map in a window with view only capability.

Column	Field Name	Type	Width	Dec
3	Menu Option Title	Character	20	
35	Obstacle Option (SYS_OBS_OPTION)	Character	15	

Note: Submenu Menu Option Titles must be indented 1 character from their parent and must appear immediately after the parent. Using a question mark "?" as the Menu Option Title will cause a blank item in the menu. The first record of this file is a comment.

DATA BASE: BLUEFOR_OBS_VIEW_MENU

TYPE: FIXED ASCII

Description
Description of the walking menu to display when a BLUEFOR obstacle is selected on the tactical map in a window with edit capability.

Column	Field Name	Type	Width	Dec
3	Menu Option Title	Character	20	
35	Obstacle Option (SYS_OBS_OPTION)	Character	15	

Note: Submenu Menu Option Titles must be indented 1 character from their parent and must appear immediately after the parent. Using a question mark "?" as the Menu Option Title will cause a blank item in the menu. The first record of this file is a comment.

DATA BASE: BLUEFOR_ORGANIC_TASK_ORG

TYPE: FIXED ASCII

Description
Organic task organization for the BLUEFOR units.

Column	Field Name	Type	Width	Dec
1	Unit Name	Character	12	
31	"TOP" if the unit is to be included in the Top Unit menu in the task organization tool	Character	3	

Note: Subordinate unit names must be indented 2 spaces from their parent units name.

DATA BASE: BLUEFOR_PERSONNEL

TYPE: Ada

Description
BLUEFOR authorized and current personnel levels.

```
type SDB_PERSONNEL is
   record
      SDB_UNIT_ID        :    SDB_BLUEFOR_UNIT_ID;
      SDB_TIME           :    SYS_DATE_TIME;
      SDB_OPPLAN         :    SYS_OPPLAN;
      SDB_OFFICERS_AUTH  :    SYS_QUANTITY   range 0..9999;
      SDB_OFFICERS_CURR  :    SYS_QUANTITY   range 0..9999;
      SDB_ENLISTED_AUTH  :    SYS_QUANTITY   range 0..999999;
      SDB_ENLISTED_CURR  :    SYS_QUANTITY   range 0..999999;
   end record;
```

DATA BASE: BLUEFOR_PERSONNEL_INDEX

TYPE: Ada

Description
Index file for the BLUEFOR personnel level data base.

```
type SDB_BLUEFOR_PERS_PTR is
   record
      SDB_UNIT_ID       :       SDB_BLUEFOR_UNIT_ID;
      SDB_TIME          :       SYS_DATE_TIME;
      SDB_OPPLAN        :       SYS_OPPLAN;
      SDB_RECORD        :       SYS_DB_SIZE;
   end record;
```

DATA BASE: BLUEFOR_PERSONNEL_SOURCE

TYPE: FIXED ASCII

Description
Initial personnel levels for BLUEFOR units.

Column	Field Name	Type	Width	Dec
14	Unit Name	Character	12	
34	Authorized Officers	Numeric	4	0
46	Authorized Enlisted	Numeric	4	0
59	Current Officers	Numeric	4	0
71	Current Enlisted	Numeric	4	0

Note: A date/time record is used to assign a date/time to the personnel data. Format: *DDHHMM MON starting in column 1. (Example: *021800 SEP). The date/time record is followed by two comment records.

DATA BASE: BLUEFOR_TASK_ORG_SOURCE

TYPE: VARIABLE ASCII

Description
Initial task organization and status for the BLUEFOR units.

Column	Field Name	Type	Width	Dec
Record 1 (Echelon Count)				
1	Echelon Count	Numeric	3	0
Record 2 (Echelon Name)				
1	Echelon Name	Character	20	

Record 3 (Unit)

1	Unit Name	Character	12
25	Echelon	Character	6
34	Unit Type	Character	6
42	Battle Function	Character	6
51	Activity	Character	6
61	Mission	Character	6
70	Relationship	Character	6

Note: The Echelon Name records must appear directly after the Echelon Count Record. Each subsequent echelon record must be indented 2 spaces from the previous one. Subordinate unit names in the Unit record must be indented 2 spaces from their parent units name. A date/time record is used to assign a date/time to the task organization data. Format: *DDHHMM MON starting in column 1. (Example: *021800 SEP). The date/time record is followed by one comment record and the Echelon Count and Echelon Name records.

DATA BASE: BLUEFOR_UNIT_CONVERT

TYPE: Ada

Description
Data base to convert BLUEFOR unit names to unit numbers.

```
type BLUE_ORGANIC_UNIT is
   record
      OLD_ID          :       SDB_BLUEFOR_UNIT_ID;
      NEW_ID          :       SDB_BLUEFOR_UNIT_ID;
      NAME            :       string (SDB_UNIT_NAME_LEN);
end record;
```

DATA BASE: BLUEFOR_UNIT_LOC_INDEX

TYPE: Ada

Description
Index file for the BLUEFOR unit location data base.

```
type SDB_BLUEFOR_LOCATION_PTR is
   record
      SDB_UNIT_ID       :       SDB_BLUEFOR_UNIT_ID;
      SDB_TIME          :       SYS_DATE_TIME;
      SDB_OPPLAN        :       SYS_OPPLAN;
      SDB_RECORD        :       SYS_DB_SIZE;
   end record;
```

DATA BASE: BLUEFOR_UNIT_LOC_SOURCE

TYPE: FIXED ASCII

Description
Initial unit locations for the BLUEFOR units.

Column	Field Name	Type	Width	Dec
14	Unit Name	Character	12	
36	UTM Letters	Character	2	
38	UTM X Coordinate	Numeric	3	0
41	UTM Y Coordinate	Numeric	3	0

Note: A date/time record is used to assign a date/time to the unit location data. Format: *DDHHMM MON starting in column 1. (Example: *021800 SEP). The date/time record is followed by two comment records.

DATA BASE: BLUEFOR_UNIT_LOC

TYPE: Ada

Description
BLUEFOR unit location data base.

```
type SDB_UNIT_LOCATION is
   record
      SDB_UNIT_ID       :       SDB_UNIT;
      SDB_TIME          :       SYS_DATE_TIME;
      SDB_OPPLAN        :       SYS_OPPLAN;
      SDB_LOCATION      :       SDB_LOCATION_REC;
   end record;
```

DATA BASE: BLUEFOR_UNIT_EDIT_MENU

TYPE: FIXED ASCII

Description
Description of the walking menu to display when a BLUEFOR unit is selected on a tactical map in a window with edit capability.

Column	Field Name	Type	Width	Dec
3	Menu Option Title	Character	20	
35	Unit Option (SYS_UNIT_OPTION)	Character	15	

Note: Submenu Menu Option Titles must be indented 1 character from their parent and must appear immediately after the parent. Using a question mark "?" as the Menu Option Title will cause a blank item in the menu. The first record of this file is a comment.

DATA BASE: BLUEFOR_UNIT_NAME

TYPE: DELIMITED ASCII

Description
List of the BLUEFOR unit names. This file is used to assign names to the unit transactions in the situation recorded data.

Field #	Field Name	Type
1	Unit Number	Numeric
2	Unit Name	Character

DATA BASE: BLUEFOR_UNIT_STATUS

TYPE: Ada

Description
BLUEFOR unit status.

```
type SDB_BLUE_UNIT_STATUS is
   record
      SDB_UNIT_ID         :    SDB_BLUEFOR_UNIT_ID;
      SDB_TIME            :    SYS_DATE_TIME;
      SDB_OPPLAN          :    SYS_OPPLAN;
      SDB_NAME            :    string (SDB_UNIT_NAME_LEN);
      SDB_ECHELON         :    SDB_FORCE_ECHELON;
      SDB_TYPE            :    SDB_UNIT_TYPE;
      SDB_BATTLE_FUNC     :    SDB_BATTLE_FUNCTION;
      SDB_TO_RELATE       :    SDB_BLUEFOR_TO_RELATE;
      SDB_PARENT          :    SDB_BLUEFOR_UNIT_ID;
      SDB_HIGHER_ECH      :    SDB_BLUEFOR_UNIT_ID;
      SDB_NEXT_SIBLING    :    SDB_BLUEFOR_UNIT_ID;
      SDB_ASSET_SIBLING   :    SDB_BLUEFOR_UNIT_ID;
      SDB_FIRST_CHILD     :    SDB_BLUEFOR_UNIT_ID;
      SDB_ACTIVITY        :    SDB_FORCE_ACTIVITY;
      SDB_MISSION         :    SDB_FORCE_MISSION;
   end record;
```

DATA BASE: BLUEFOR_UNIT_STATUS_INDEX

TYPE: Ada

Description
Index file for the BLUEFOR unit status data base.

```
type SDB_BLUEFOR_STATUS_PTR is
   record
      SDB_UNIT_ID       :       SDB_BLUEFOR_UNIT_ID;
      SDB_TIME          :       SYS_DATE_TIME;
      SDB_OPPLAN        :       SYS_OPPLAN;
      SDB_RECORD        :       SYS_DB_SIZE;
   end record;
```

DATA BASE: BLUEFOR_UNIT_VIEW_MENU

TYPE: FIXED ASCII

Description
Description of the walking menu to display when a BLUEFOR unit is selected on the tactical map in a window with view only capability.

Column	Field Name	Type	Width	Dec
3	Menu Option Title	Character	20	
35	Unit Option (SYS_UNIT_OPTION)	Character	15	

Note: Submenu Menu Option Titles must be indented 1 character from their parent and must appear immediately after the parent. Using a question mark "?" as the Menu Option Title will cause a blank item in the menu. The first record of this file is a comment.

DATA BASE: C2_PRODUCT

TYPE: Ada

Description
Command and control product data base. Includes the products in the view situation, build and view message windows.

```
type CDB_PRODUCT_TYPE  is
   record
      CDB_REPT_NUMBER_CHAR    :  SYS_PRODUCT_LENGTH  range 0..
                                 CDB_PRODUCT_SIZE;
      CDB_PRODUCT_TEXT        :  string (1..CDB_PRODUCT_SIZE);
   end record;
```

DATA BASE: C2_PRODUCT_DESC

TYPE: Ada

Description
Command and control product description data base. This data base irdicates which record from the C2_PRODUCT data base to use for a product.

```
type CDB_PRODUCT_DESC_TYPE  is
   record
      CDB_PRODUCT_CAT         :  SYS_PRODUCT_CAT;
      CDB_PRODUCT_HDR_START   :  CDB_NUM_HEADER_REC;
      CDB_PRODUCT_HDR_END     :  CDB_NUM_HEADER_REC;
      CDB_PRODUCT_START       :  CDB_NUM_PRODUCT_REC;
      CDB_PRODUCT_END         :  CDB_NUM_PRODUCT_REC;
      CDB_PRODUCT_DATE        :  SYS_DATE_TIME;
      CDB_PRODUCT_OPPLAN      :  SYS_OPPLAN;
   end record;
```

DATA BASE: C2_PRODUCT_HEADER

TYPE: Ada

Description
Command and control report headers. The report headers only applies to those products in the view situation window.

```
type CDB_HEADER_TYPE  is
   record
      CDB_HEAD_NUMBER_CHAR    :  SYS_HEADER_LENGTH  range 0..
                                 CDB_HEADER_SIZE;
      CDB_HEADER_TEXT         :  string (1..CDB_HEADER_SIZE);
   end record;
```

DATA BASE: C2_PRODUCT_NAME

TYPE: DELIMITED ASCII

Description
List of the command and control product names. This file is used to assign names to the command and control transactions in the C2 product recorded data.

Field #	Field Name	Type
1	Report Number	Numeric
2	Functional Area	Character
3	Data Category	Character
4	Date Element	Character
5	Level of detail (D=Detail; A=Aggregate; S=Summary)	Character
6	Date/Time	Character

DATA BASE: C2_PRODUCT_RECORD

TYPE: BINARY

Description
Command and control data recording transactions.

This data base contains binary images of the messages in MSG_C2_RECORD_LIST. The message type is contained in MSG_RECORD_TYPE and the length is in MSG_BYTES_IN_MSG. The UUX_IO utilities should be used to interact with this data base.

DATA BASE: C2_PRODUCT_SOURCE

TYPE: VARIABLE ASCII

Description
Description of the command and control products to include in the view situation and build windows.

Column	Field Name	Type	Width	Dec
Record 1 (Functional Area)				
1	Slash "/"	Character	1	
2	"F"	Character	1	
3	Window Display Code 1 = View Situation 2 = Build 3 = View Situation and Build	Numeric	1	0
4	Routing Code 1 = G2 2 = G3 3 = G2 & G3 4 = G4 5 = G2 & G4 6 = G3 & G4 7 = G2, G3, & G4	Numeric	1	0
5	Functional Area Title	Character	20	
Record 2 (Data Category)				
1	Slash "/"	Character	1	
2	"C"	Character	1	
3	Window Display Code 1 = View Situation 2 = Build 3 = View Situation and Build	Numeric	1	0

4	Routing Code 1 = G2 2 = G3 3 = G2 & G3 4 = G4 5 = G2 & G4 6 = G3 & G4 7 = G2, G3, & G4	Numeric	1	0
5	Data Category Title	Character	20	

Record 3 (Data Element)

1	Slash "/"	Character	1	
2	"E"	Character	1	
3	Window Display Code 1 = View Situation 2 = Build 3 = View Situation and Build	Numeric	1	0
4	Routing Code 1 = G2 2 = G3 3 = G2 & G3 4 = G4 5 = G2 & G4 6 = G3 & G4 7 = G2, G3, & G4	Numeric	1	0
5	Level of Detail (D=Detail, A=Aggregate, S=Summary)	Character	1	
6	Colon ":"	Character	1	
7	Data Element Title	Character	20	

Record 4 (Date/Time)

1	Slash "/"	Character	1	
2	"D"	Character	1	
3	Window Display Code 1 = View Situation 2 = Build 3 = View Situation and Build	Numeric	1	0
4	Routing Code 1 = G2 2 = G3 3 = G2 & G3 4 = G4 5 = G2 & G4 6 = G3 & G4 7 = G2, G3, & G4	Numeric	1	0

5	Date/Time (Format ddhhmm mon, Example: 021800 SEP)	Character	10	

Record 5 (Report Header)

1	Percent Sign "%"	Character	1	
2	Report Header Line	Character	80	

Record 6 (Report)

1	Textual Report Line	Character	80	

Record 7 (Computer Generated Report)

1	Dollar Sign "$"	Character	1	
2	Report Type 1 = Task Organization 2 = Personnel Strengths 3 = OPFOR Committed 4 = OPFOR Reinforcements 5 = Equipment Status 6 = Class III Status 7 = Class V Status 8 = Tactical Map	Numeric	1	0
3	Comma ","	Character	1	
4	Force (RED , or BLUE)	Character	4	
5	Comma ","	Character	1	
6	Unit Name	Character	12	

Note: The Functional Area, Data Category, Data Element, and Date/Time records are used to build the product selection walking menu description files for the view situation and build windows.

DATA BASE: CNTRL_MSR_POINT

TYPE: Ada

Description

Point control measures.

```
type SDB_CNTRL_MSR_POINT_REC is
   record
      SDB_ID              :    SDB_CONTROL_MEASURE_ID;
      SDB_OPPLAN          :    SYS_OPPLAN;
      SDB_NAME            :    string (SDB_CNTL_MSR_NAME_LEN);
      SDB_SIDE            :    SDB_SIDE_TYPE;
      SDB_OWNER_BLUE      :    SDB_BLUEFOR_UNIT_ID;
      SDB_OWNER_OPFOR     :    SDB_OPFOR_UNIT_ID;
      SDB_TYPE            :    SDB_CONTROL_MEASURE_TYPE;
      SDB_LOCATION_TYPE   :    SDB_CONTROL_MEASURE_LOC_TYPE;
      SDB_SCALE           :    SDB_CONTROL_MEASURE_SCALES;
      SDB_STATUS          :    SDB_CONTROL_MEASURE_STATUS;
```

```
      SDB_EFF_FROM_DATE :     SYS_DATE_TIME;
      SDB_EFF_TO_DATE   :     SYS_DATE_TIME;
      SDB_LABEL_ECHELON :     SDB_FORCE_ECHELON;
      SDB_LOCATION      :     SDB_LOCATION_REC;
   end record;
```

DATA BASE: CNTRL_MSR_POINT_INDEX

TYPE: Ada

Description
Index file for the point control measure data base.

```
type SDB_CNTRL_MSR_POINT_PTR is
   record
      SDB_CNTRL_MSR_ID  :       SDB_CONTROL_MEASURE_ID;
      SDB_OPPLAN        :       SYS_OPPLAN;
      SDB_EFF_FROM      :       SYS_DATE_TIME;
      SDB_EFF_TO        :       SYS_DATE_TIME;
      SDB_RECORD        :       SYS_DB_SIZE;
   end record;
```

DATA BASE: CNTRL_MSR_POINT_NAME

TYPE: DELIMITED ASCII

Description
List of the point control measure names. This file is used to assign names to the point control measure transactions in the situation recorded data.

Field #	Field Name	Type
1	Point Control Measure Number	Numeric
2	Point Control Measure Name	Character

DATA BASE: CONTOUR_1TO160

TYPE: BINARY

Description
Map contour image file for the 1:160,000 map scale.

This contour image file is a bitmap contour image. If the bit is ON the contour is displayed and if the bit is OFF the contour is not displayed. The data is organized in column/row order (columns within rows) from northwest to southeast with 40 columns and 25 rows. Each record is organized in column/row order from northwest to southeast with 64 columns and 80 rows (640 bytes).

DATA BASE: CONTOUR_1TO400

TYPE: BINARY

Description
Map contour image file for the 1:400,000 map scale.

This contour image file is a bitmap contour image. If the bit is ON the contour is displayed and if the bit is OFF the contour is not displayed. The data is organized in column/row order (columns within rows) from northwest to southeast with 16 columns and 10 rows. Each record is organized in column/row order from northwest to southeast with 64 columns and 80 rows (640 bytes).

DATA BASE: CONTOUR_1TO80

TYPE: BINARY

Description
Map contour image file for the 1:80,000 map scale.

This contour image file is a bitmap contour image. If the bit is ON the contour is displayed and if the bit is OFF the contour is not displayed. The data is organized in column/row order (columns within rows) from northwest to southeast with 79 columns and 49 rows. Each record is organized in column/row order from northwest to southeast with 64 columns and 80 rows (640 bytes).

DATA BASE: CONTOUR_1TO800

TYPE: BINARY

Description
Map contour image file for the 1:800,000 map scale.

This contour image file is a bitmap contour image. If the bit is ON the contour is displayed and if the bit is OFF the contour is not displayed. The data is organized in column/row order (columns within rows) from northwest to southeast with 8 columns and 6 rows. Each record is organized in column/row order from northwest to southeast with 64 columns and 80 rows (640 bytes).

DATA BASE: CONTOUR_DESC

TYPE: FIXED ASCII

Description
Description of the contour files to include in the tactical map system.

Column	Field Name	Type	Width	Dec
1	Map Scale (SYS_MAP_SCALES)	Character	9	
12	Contour description file for this map scale.	Character	50	

Note: The first record of this file is a comment.

DATA BASE: CONTOUR_DESC_1TO160

TYPE: VARIABLE ASCII

Description

Description of the 1:160,000 contour image file.

Column	Field Name	Type	Width	Dec
Record 1 (Image File)				
1	Contour image file name for this map scale	Character	60	
Record 2 (Data Base Size)				
1	Number of contour image records in the X direction for this map scale	Numeric	5	0
6	Number of contour image records in the Y direction for this map scale	Numeric	5	0
Record 3 (Record Size)				
1	Number of contour image points in a record in the X direction	Numeric	5	0
6	Number of contour image points in a record in the Y direction	Numeric	5	0
Record 4 (Data Base Point Size)				
1	Number of contour image points in the data base in the X direction	Numeric	6	0
7	Number of contour image points in the data base in the Y direction	Numeric	6	0
14	Number of meters per pixel for this map scale	Numeric	7	3
Record 5 (Map Origin)				
4	Number of meters in the X direction from MA000000 to the northwest corner of the contour image	Numeric	7	0
14	Number of meters in the Y direction from MA000000 to the northwest corner of the contour image	Numeric	7	0
Record 6 (Grid Interval)				
1	Grid interval for this map scale (in meters)	Numeric	5	0

DATA BASE: CONTOUR_DESC_1TO400

TYPE: VARIABLE ASCII

Description
Description of the 1:400,000 contour image file.

Column	Field Name	Type	Width	Dec
Record 1 (Image File)				
1	Contour image file name for this map scale	Character	60	
Record 2 (Data Base Size)				
1	Number of contour image records in the X direction for this map scale	Numeric	5	0
6	Number of contour image records in the Y direction for this map scale	Numeric	5	0
Record 3 (Record Size)				
1	Number of contour image points in a record in the X direction	Numeric	5	0
6	Number of contour image points in a record in the Y direction	Numeric	5	0
Record 4 (Data Base Point Size)				
1	Number of contour image points in the data base in the X direction	Numeric	6	0
7	Number of contour image points in the data base in the Y direction	Numeric	6	0
14	Number of meters per pixel for this map scale	Numeric	7	3
Record 5 (Map Origin)				
4	Number of meters in the X direction from MA000000 to the northwest corner of the contour image	Numeric	7	0
14	Number of meters in the Y direction from MA000000 to the northwest corner of the contour image	Numeric	7	0
Record 6 (Grid Interval)				
1	Grid interval for this map scale (in meters)	Numeric	5	0

DATA BASE: CONTOUR_DESC_1TO80

TYPE: VARIABLE ASCII

Description
Description of the 1:80,000 contour image file.

Column	Field Name	Type	Width	Dec
Record 1 (Image File)				
1	Contour image file name for this map scale	Character	60	
Record 2 (Data Base Size)				
1	Number of contour image records in the X direction for this map scale	Numeric	5	0
6	Number of contour image records in the Y direction for this map scale	Numeric	5	0
Record 3 (Record Size)				
1	Number of contour image points in a record in the X direction	Numeric	5	0
6	Number of contour image points in a record in the Y direction	Numeric	5	0
Record 4 (Data Base Point Size)				
1	Number of contour image points in the data base in the X direction	Numeric	6	0
7	Number of contour image points in the data base in the Y direction	Numeric	6	0
14	Number of meters per pixel for this map scale	Numeric	7	3
Record 5 (Map Origin)				
4	Number of meters in the X direction from MA000000 to the northwest corner of the contour image	Numeric	7	0
14	Number of meters in the Y direction from MA000000 to the northwest corner of the contour image	Numeric	7	0
Record 6 (Grid Interval)				
1	Grid interval for this map scale (in meters)	Numeric	5	0

DATA BASE: CONTOUR_DESC_1TO800

TYPE: VARIABLE ASCII

Description
Description of the 1:800,000 contour image file.

Column	Field Name	Type	Width	Dec
Record 1 (Image File)				
1	Contour image file name for this map scale	Character	60	
Record 2 (Data Base Size)				
1	Number of contour image records in the X direction for this map scale	Numeric	5	0
6	Number of contour image records in the Y direction for this map scale	Numeric	5	0
Record 3 (Record Size)				
1	Number of contour image points in a record in the X direction	Numeric	5	0
6	Number of contour image points in a record in the Y direction	Numeric	5	0
Record 4 (Data Base Point Size)				
1	Number of contour image points in the data base in the X direction	Numeric	6	0
7	Number of contour image points in the data base in the Y direction	Numeric	6	0
14	Number of meters per pixel for this map scale	Numeric	7	3
Record 5 (Map Origin)				
4	Number of meters in the X direction from MA000000 to the northwest corner of the contour image	Numeric	7	0
14	Number of meters in the Y direction from MA000000 to the northwest corner of the contour image	Numeric	7	0
Record 6 (Grid Interval)				
1	Grid interval for this map scale (in meters)	Numeric	5	0

DATA BASE: CONTROL_MEASURE

TYPE: Ada

Description
Control measures.

```
type SDB_CONTROL_MEASURE_POINTS is array
        (SDB_CONTROL_MEASURE_PT) of SDB_LOCATION_REC;

type SDB_CONTROL_MEASURE_SCALES is array (SYS_MAP_SCALES) of
                                BOOLEAN;

type SDB_CONTROL_MEASURE_REC is
   record
      SDB_ID              :    SDB_CONTROL_MEASURE_ID;
      SDB_OPPLAN          :    SYS_OPPLAN;
      SDB_NAME            :    string (SDB_CNTL_MSR_NAME_LEN);
      SDB_SIDE            :    SDB_SIDE_TYPE;
      SDB_OWNER_BLUE      :    SDB_BLUEFOR_UNIT_ID;
      SDB_OWNER_OPFOR     :    SDB_OPFOR_UNIT_ID;
      SDB_TYPE            :    SDB_CONTROL_MEASURE_TYPE;
      SDB_LOCATION_TYPE   :    SDB_CONTROL_MEASURE_LOC_TYPE;
      SDB_SCALE           :    SDB_CONTROL_MEASURE_SCALES;
      SDB_STATUS          :    SDB_CONTROL_MEASURE_STATUS;
      SDB_EFF_FROM_DATE   :    SYS_DATE_TIME;
      SDB_EFF_TO_DATE     :    SYS_DATE_TIME;
      SDB_LABEL_ECHELON   :    SDB_FORCE_ECHELON;
      SDB_NUMBER_POINTS   :    SDB_CONTROL_MEASURE_PT;
      SDB_LOCATION        :    SDB_CONTROL_MEASURE_POINTS;
   end record;
```

DATA BASE: CONTROL_MEASURE_NAME

TYPE: DELIMITED ASCII

Description
List of the control measure names. This file is used to assign names to the control measure transactions in the situation recorded data.

Column	Field Name	Type
1	Control Measure Number	Numeric
2	Control Measure Name	Character

DATA BASE: CONTROL_MEASURE_SOURCE

TYPE: VARIABLE ASCII

Description
Initial control measures.

Column	Field Name	Type	Width	Dec

Record 1 (CM Type)

1	Control Measure Name	Character	12	
14	Control Measure Type	Character	6	
21	Control Measure Echelon	Character	6	
28	Side (BLUE, RED)	Character	4	
33	Display on 1:40000 map flag (1 = Yes, 0 = No)	Numeric	1	0
34	Display on 1:80000 map flag (1 = Yes, 0 = No)	Numeric	1	0
35	Display on 1:160000 map flag (1 = Yes, 0 = No)	Numeric	1	0
36	Display on 1:400000 map flag (1 = Yes, 0 = No)	Numeric	1	0
37	Display on 1:800000 map flag (1 = Yes, 0 = No)	Numeric	1	0

Record 2 (Points 1 - 8)

1	UTM coordinate of point 1	Character	8
10	UTM coordinate of point 2	Character	8
19	UTM coordinate of point 3	Character	8
28	UTM coordinate of point 4	Character	8
37	UTM coordinate of point 5	Character	8
46	UTM coordinate of point 6	Character	8
55	UTM coordinate of point 7	Character	8
64	UTM coordinate of point 8	Character	8

Record 3 (Points 9 - 15)

1	UTM coordinate of point 9	Character	8
10	UTM coordinate of point 10	Character	8
19	UTM coordinate of point 11	Character	8
28	UTM coordinate of point 12	Character	8
37	UTM coordinate of point 13	Character	8
46	UTM coordinate of point 14	Character	8
55	UTM coordinate of point 15	Character	8

Note: A date/time record is used to assign a date/time to the control measure data. Format: *DDHHMM MON starting in column 1. (Example: *021800 SEP). The date/time record is followed by one comment record.

DATA BASE: CONTROL_MEASURE_INDEX

TYPE: Ada

Description
Index file for the control measure data base.

```
type SDB_CONTROL_MEASURE_PTR is
   record
      SDB_CNTRL_MSR_ID  :      SDB_CONTROL_MEASURE_ID;
      SDB_OPPLAN        :      SYS_OPPLAN;
      SDB_EFF_FROM      :      SYS_DATE_TIME;
      SDB_EFF_TO        :      SYS_DATE_TIME;
      SDB_RECORD        :      SYS_DB_SIZE;
   end record;
```

DATA BASE: ELEVATION_1TO400

TYPE: BINARY

Description
Elevation file for the 1:400,000 map scale.

This elevation consists of 16-bit values representing the elevation in meters. The data is organized in column/row order (columns within rows) from northwest to southeast with 16 columns and 10 rows. Each record is organized in column/row order from northwest to southeast with 64 columns and 80 rows (10240 bytes).

DATA BASE: ELEVATION_DESC_1TO400

TYPE: VARIABLE ASCII

Description
Description of the 1:400,000 elevation file.

Column	Field Name	Type	Width	Dec
Record 1 (Image File)				
1	Elevation file name for this map scale	Character	60	
Record 2 (Data Base Size)				
1	Number of elevation records in the X direction for this map scale	Numeric	5	0
6	Number of elevation records in the Y direction for this map scale	Numeric	5	0

Record 3 (Record Size)				
1	Number of elevation points in a record in the X direction	Numeric	5	0
6	Number of elevation points in a record in the Y direction	Numeric	5	0
Record 4 (Data Base Point Size)				
1	Number of elevation points in the data base in the X direction	Numeric	6	0
7	Number of elevation points in the data base in the Y direction	Numeric	6	0
14	Number of meters per pixel for this map scale	Numeric	7	3
Record 5 (Map Origin)				
4	Number of meters in the X direction from MA000000 to the northwest corner of the elevation	Numeric	7	0
14	Number of meters in the Y direction from MA000000 to the northwest corner of the elevation	Numeric	7	0

DATA BASE: ELEV_BAND_1TO160

TYPE: BINARY

Description
Elevation band image file for the 1:160,000 map scale.

This elevation banded data base consists of byte values representing the color lookup table value to use to represent the elevation bands. The data is organized in column/row order (columns within rows) from northwest to southeast with 40 columns and 25 rows. Each record is organized in column/row order from northwest to southeast with 64 columns and 80 rows (5120 bytes).

DATA BASE: ELEV_BAND_1TO400

TYPE: BINARY

Description
Elevation band image file for the 1:400,000 map scale.

This elevation banded data base consists of byte values representing the color lookup table value to use to represent the elevation bands. The data is organized in column/row order (columns within rows) from northwest to southeast with 16 columns and 10 rows. Each record is organized in column/row order from northwest to southeast with 64 columns and 80 rows (5120 bytes).

DATA BASE: ELEV_BAND_1TO80

TYPE: BINARY

Description
Elevation band image file for the 1:80,000 map scale.

This elevation banded data base consists of byte values representing the color lookup table value to use to represent the elevation bands. The data is organized in column/row order (columns within rows) from northwest to southeast with 79 columns and 49 rows. Each record is organized in column/row order from northwest to southeast with 64 columns and 80 rows (5120 bytes).

DATA BASE: ELEV_BAND_1TO800

TYPE: BINARY

Description
Elevation band image file for the 1:800,000 map scale.

This elevation banded data base consists of byte values representing the color lookup table value to use to represent the elevation bands. The data is organized in column/row order (columns within rows) from northwest to southeast with 8 columns and 6 rows. Each record is organized in column/row order from northwest to southeast with 64 columns and 80 rows (5120 bytes).

DATA BASE: ELEV_BAND_DESC_1TO160

TYPE: VARIABLE ASCII

Description
Description of the 1:160,000 elevation band image file.

Column	Field Name	Type	Width	Dec
Record 1 (Image File)				
1	Elevation banding image file name for this map scale	Character	60	
Record 2 (Data Base Size)				
1	Number of elevation banding image records in the X direction for this map scale	Numeric	5	0
6	Number of elevation banding image records in the Y direction for this map scale	Numeric	5	0

Record 3 (Record Size)				
1	Number of elevation banding image points in a record in the X direction	Numeric	5	0
6	Number of elevation banding image points in a record in the Y direction	Numeric	5	0
Record 4 (Data Base Point Size)				
1	Number of elevation banding image points in the data base in the X direction	Numeric	6	0
7	Number of elevation banding image points in the data base in the Y direction	Numeric	6	0
14	Number of meters per pixel for this map scale	Numeric	7	3
Record 5 (Map Origin)				
4	Number of meters in the X direction from MA000000 to the northwest corner of the elevation banding image	Numeric	7	0
14	Number of meters in the Y direction from MA000000 to the northwest corner of the elevation banding image	Numeric	7	0
Record 6 (Grid Interval)				
1	Grid interval for this map scale (in meters)	Numeric	5	0

DATA BASE: ELEV_BAND_DESC_1TO400

TYPE: VARIABLE ASCII

Description

Description of the 1:400,000 elevation band image file.

Column	Field Name	Type	Width	Dec
Record 1 (Image File)				
1	Elevation banding image file name for this map scale	Character	60	

Record 2 (Data Base Size)				
1	Number of elevation banding image records in the X direction for this map scale	Numeric	5	0
6	Number of elevation banding image records in the Y direction for this map scale	Numeric	5	0
Record 3 (Record Size)				
1	Number of elevation banding image points in a record in the X direction	Numeric	5	0
6	Number of elevation banding image points in a record in the Y direction	Numeric	5	0
Record 4 (Data Base Point Size)				
1	Number of elevation banding image points in the data base in the X direction	Numeric	6	0
7	Number of elevation banding image points in the data base in the Y direction	Numeric	6	0
14	Number of meters per pixel for this map scale	Numeric	7	3
Record 5 (Map Origin)				
4	Number of meters in the X direction from MA000000 to the northwest corner of the elevation banding image	Numeric	7	0
14	Number of meters in the Y direction from MA000000 to the northwest corner of the elevation banding image	Numeric	7	0
Record 6 (Grid Interval)				
1	Grid interval for this map scale (in meters)	Numeric	5	0

DATA BASE: ELEV_BAND_DESC_1TO80

TYPE: VARIABLE ASCII

Description

Description of the 1:80,000 elevation band image file.

Column	Field Name	Type	Width	Dec
Record 1 (Image File)				
1	Elevation banding image file name for this map scale	Character	60	
Record 2 (Data Base Size)				
1	Number of elevation banding image records in the X direction for this map scale	Numeric	5	0
6	Number of elevation banding image records in the Y direction for this map scale	Numeric	5	0
Record 3 (Record Size)				
1	Number of elevation banding image points in a record in the X direction	Numeric	5	0
6	Number of elevation banding image points in a record in the Y direction	Numeric	5	0
Record 4 (Data Base Point Size)				
1	Number of elevation banding image points in the data base in the X direction	Numeric	6	0
7	Number of elevation banding image points in the data base in the Y direction	Numeric	6	0
14	Number of meters per pixel for this map scale	Numeric	7	3
Record 5 (Map Origin)				
4	Number of meters in the X direction from MA000000 to the northwest corner of the elevation banding image	Numeric	7	0
14	Number of meters in the Y direction from MA000000 to the northwest corner of the elevation banding image	Numeric	7	0
Record 6 (Grid Interval)				
1	Grid interval for this map scale (in meters)	Numeric	5	0

DATA BASE: ELEV_BAND_DESC_1TO800

TYPE: VARIABLE ASCII

Description
Description of the 1:800,000 elevation band image file.

Column	Field Name	Type	Width	Dec
Record 1 (Image File)				
1	Elevation banding image file name for this map scale	Character	60	
Record 2 (Data Base Size)				
1	Number of elevation banding image records in the X direction for this map scale	Numeric	5	0
6	Number of elevation banding image records in the Y direction for this map scale	Numeric	5	0
Record 3 (Record Size)				
1	Number of elevation banding image points in a record in the X direction	Numeric	5	0
6	Number of elevation banding image points in a record in the Y direction	Numeric	5	0
Record 4 (Data Base Point Size)				
1	Number of elevation banding image points in the data base in the X direction	Numeric	6	0
7	Number of elevation banding image points in the data base in the Y direction	Numeric	6	0
14	Number of meters per pixel for this map scale	Numeric	7	3
Record 5 (Map Origin)				
4	Number of meters in the X direction from MA000000 to the northwest corner of the elevation banding image	Numeric	7	0
14	Number of meters in the Y direction from MA000000 to the northwest corner of the elevation banding image	Numeric	7	0

Record 6 (Grid Interval)

1	Grid interval for this map scale (in meters)	Numeric	5	0

DATA BASE: EXP_CONTROL_MENU

TYPE: FIXED ASCII

Description

Description of the experiment control product walking menu. This file is created from the product names in the experiment control source file.

Column	Field Name	Type	Width	Dec
3	Menu Option Title	Character	20	
35	Experiment Control Product Number	Numeric	6	0

Note: Submenu Menu Option Titles must be indented 1 character from their parent and must appear immediately after the parent. Using a question mark "?" as the Menu Option Title will cause a blank item in the menu. The first record of this file is a comment.

DATA BASE: EXP_CONTROL_NAME

TYPE: DELIMITED ASCII

Description

List of the experiment control product names. This file is used to assign names to the experiment control transactions in the experiment control recorded data.

Field #	Field Name	Type
1	Product Number	Numeric
2	Functional Area	Character
3	Data Category	Character
4	Data Element	Character
5	Data Component	Character

DATA BASE: EXP_CONTROL_PARTICIPANT

TYPE: Ada

Description
List of participants that the experimenter can send experiment control messages to.

```
subtype CTL_PART_NAME_LEN      is INTEGER     range 1..10;
subtype CTL_PART_NAME_TEXT is string (CTL_PART_NAME_LEN);
type CTL_PART_REC is
   record
      CTL_TEXT          :   CTL_PART_NAME_TEXT;
      CTL_PART          :   SYS_PARTICIPANTS;
end record;
```

DATA BASE: EXP_CONTROL_PRODUCT

TYPE: Ada

Description
Experiment control products.

```
type CTL_PRODUCT_TYPE  is
   record
      CTL_REPT_NUMBER_CHAR   :  SYS_PRODUCT_LENGTH  range 0..
                                CTL_PRODUCT_SIZE;
      CTL_PRODUCT_TEXT       :  string (1..CTL_PRODUCT_SIZE);
   end record;
```

DATA BASE: EXP_CONTROL_PROD_DESC

TYPE: Ada

Description
Experiment control product description data base. This data base indicates which record from the experiment control data base to use for a product.

```
type CTL_PRODUCT_DESC_TYPE  is
   record
      CTL_PRODUCT_TYPE       :  SYS_PRODUCT;
      CTL_PRODUCT_START      :  CTL_NUM_PRODUCT_REC;
      CTL_PRODUCT_END        :  CTL_NUM_PRODUCT_REC;
      CTL_PRODUCT_DATE       :  SYS_DATE_TIME;
   end record;
```

DATA BASE: EXP_CONTROL_RECORD

TYPE: BINARY

Description
Experiment control data recording transactions.

This data base contains binary images of the messages in MSG_EC_RECORD_LIST. The message type is contained in MSG_RECORD_TYPE and the length is in MSG_BYTES_IN_MSG. The UUX_IO utilities should be used to interact with this data base.

DATA BASE: EXP_CONTROL_SOURCE

TYPE: VARIABLE ASCII

Description
Description of the products to include in the experiment control window.

Column	Field Name	Type	Width	Dec
Record 1 (Functional Area)				
1	Slash "/"	Character	1	
2	"F"	Character	1	
3	Message Type Code 1 = Informative 2 = Requires Answer	Numeric	1	0
4	Functional Area Title	Character	20	
Record 2 (Data Category)				
1	Slash "/"	Character	1	
2	"C"	Character	1	
3	Message Type Code 1 = Informative 2 = Requires Answer	Numeric	1	0
4	Data Category Title	Character	20	
Record 3 (Data Element)				
1	Slash "/"	Character	1	
2	"E"	Character	1	
3	Message Type Code 1 = Informative 2 = Requires Answer	Numeric	1	0
4	Data Element Title	Character	20	

Record 4 (Data Component)

1	Slash "/"	Character	1	
2	"D"	Character	1	
3	Message Type Code 1 = Informative 2 = Requires Answer	Numeric	1.	0
4	Data Component Title	Character	20	

Record 5 (Report)

1	Textual Report Line	Character	80

Note: The Functional Area, Data Category, Data Element, and Data Component records are used to build the product selection walking menu description files for the experimenter's experiment control window.

DATA BASE: FORM_DESCRIPTION

TYPE: VARIABLE ASCII

Description

Description and layout of EDDIC form.

The EDDIC Forms Manager shall accept a ASCII buffer that contains the static text, editor descriptors, and line, box, and circle descriptors. The first part of the buffer describes the static text and the location of the editors. The second section of the buffer contains the description of the geometric symbols to include in the form. The last part of the buffer describes the editors being used.

The static text is identified by a vertical bar '|' and should be typed in just as it is to appear in the form. The editors are identified by a backslash '\' followed by a unique identifier and terminated by a space. The identifier must be the same as in the editor description section. The editor will be located where the identifier is located in the static text. The static text section is terminated by line containing only a period '.'.

The geometric symbol section describes the type, size, and location of the geometric symbols to include in the form. The geometric symbol locations are in pixels from the upper left corner of the form. Only one symbol can be defined per line in the buffer. The geometric symbol section is terminated by line containing only a period '.'. The following describes the available geometric symbols and the parameters for each.

LINE - Line described by two end points

Parameter	Description
1	Starting X location in pixels
2	Starting Y location in pixels
3	Ending X location in pixels
4	Ending Y location in pixels
5	Width of the line in pixels

BOX - Box described by two corners

Parameter	Description
1	Starting corner X location in pixels
2	Starting corner Y location in pixels
3	Ending corner X location in pixels
4	Ending corner Y location in pixels
5	Width of the line in pixels

CIRCLE - Cirlce described by a center point and radius

Parameter	Description
1	Center X location in pixels
2	Center Y location in pixels
3	Radius in pixels
4	Width of the line in pixels

The editor description section shall describe each editor identifier that is used in the static text section. Each Editor Descriptor is started and terminated by a backslash '\'. The descriptor contains the editor type followed by parameters required to define the editor. Editor parameters are separated by a comma and default values will be provided for parameters not provided. The following describes the available editors and the parameters for each (default values are enclosed in brackets []):

MEMO_TEXT - Full page text editor

Parameter	Description
1	Width of Memo Editor in character positions [80]
2	Height of Memo Editor in character positions [24]
3	Read-Only attribute (ON=Read Only, [OFF]=Read/Write)
4	Memo Editor Initial text

NUMERIC_FIELD - Numeric field editor

Parameter	Description
1	Width of Numeric Editor
2	Numeric Field Title
3	Numeric Field Initial Value
4	Minimum Value
5	Maximum Value

STRING_FIELD - String field editor

Parameter	Description
1	Width of String Editor
2	String Field Title
3	String Field Initial Value

RADIO_BUTTON - One of a series of buttons that are logically connected in a way that only one is on at a time.

Parameter	Description
1	Initial State of button (ON, OFF)
2	Identification of next related radio button

(Null = End of list)

PUSH_BUTTON - Push Button Editor.

Parameter	Description
1	Number of Columns to use for the Buttons
2	Index into list of pushbuttons for the default (-1 = No Default)
3,4..etc	Title to display for each button

CHECKLIST - One of a series of buttons that are logically connected in a way that none to all on at a time.

Parameter	Description
1	Initial State of button (ON, OFF)
2	Identification of next related checklist button (Null = End of list)

BUTTON_WALK - Walking Menu initiated by a button

Parameter	Description
1	Button Title
2,4..etc	Title to display in the walking menu (1 per line with single character indentation for submenus)
3,5..etc	Value to return for this selection

FORM_WALK - Walking Menu initiated by right button on the form

Parameter	Description
1	Menu Title
2,4..etc	Title to display in the walking menu (1 per line with single character indentation for submenus)
3,5..etc	Value to return for this selection

MULTIPLE_SELECT_MENU - Menu where multiple selections can be made

Parameter	Description
1	Menu Title
2	Number of menu options to display at a time (If there are more menu options than this amount, a scroll bar will be displayed)
3,5..etc	Menu Option label
4,6..etc	Menu Option initial status (ON, OFF)

SINGLE_SELECT_MENU - Menu where only single selection can be made

Parameter	Description
1	Menu Title
2	Number of menu options to display at a time (If there are more menu options than this amount, a scroll bar will be displayed)
3,4..etc	Menu Option label
n+1	Index into to options for the default selection

DIGITAL_MAP

Parameter	Description
1	Width of Digital Map in character positions
2	Height of Digital Map in character positions
3	Date Time of situation data
4	OPPLAN Number [0]
5	Background Type [vegetation]
6	Map Scale [1:160000]
7	Grid Status ([0]=Off; 1=On)
8	Contour Status ([0]=Off; 1=On)
9	X Map Center
10	Y Map Center
11	BLUEFOR Unit Division Status ([0]=Off; 1=On)
12	BLUEFOR Unit Brigade Status ([0]=Off; 1=On)
13	BLUEFOR Unit Battalion Status ([0]=Off; 1=On)
14	BLUEFOR Unit Company Status ([0]=Off; 1=On)
15	BLUEFOR Unit Combat Status ([0]=Off; 1=On)
16	BLUEFOR Unit Combat Support Status ([0]=Off; 1=On)
17	BLUEFOR Unit CSS Status ([0]=Off; 1=On)
18	BLUEFOR Unit Name Status ([0]=Off; 1=On)
19	BLUEFOR Unit Symbol Status ([0]=Off; 1=On)
20	BLUEFOR Cntrl Msr Points Status ([0]=Off; 1=On)
21	BLUEFOR Cntrl Msr Lines Status ([0]=Off; 1=On)
22	BLUEFOR Cntrl Msr Areas Status ([0]=Off; 1=On)
23	BLUEFOR Cntrl Msr Routes Status ([0]=Off; 1=On)
24	BLUEFOR Obstacle Status ([0]=Off; 1=On)
25	BLUEFOR Cntrl Msr Crossings Status ([0]=Off; 1=On)
26	BLUEFOR Cntrl Msr Fire Plan Status ([0]=Off; 1=On)
27	BLUEFOR Cntrl Msr Map Feature Status ([0]=Off; 1=On)
28	BLUEFOR Cntrl Msr EAC Status ([0]=Off; 1=On)
29	BLUEFOR Cntrl Msr Corps Status ([0]=Off; 1=On)
30	BLUEFOR Cntrl Msr Division Status ([0]=Off; 1=On)
31	BLUEFOR Cntrl Msr Brigade Status ([0]=Off; 1=On)
32	BLUEFOR Cntrl Msr Battalion Status ([0]=Off; 1=On)
33	BLUEFOR Cntrl Msr Company Status ([0]=Off; 1=On)
34	OPFOR Unit Division Status ([0]=Off; 1=On)
35	OPFOR Unit Regiment Status ([0]=Off; 1=On)
36	OPFOR Unit Battalion Status ([0]=Off; 1=On)
37	OPFOR Unit Company Status ([0]=Off; 1=On)
38	OPFOR Unit Committed Status ([0]=Off; 1=On)
39	OPFOR Unit Reinforcing Status ([0]=Off; 1=On)
40	OPFOR Unit Artillery Status ([0]=Off; 1=On)
41	OPFOR Unit Name Status ([0]=Off; 1=On)
42	OPFOR Unit Symbol Status ([0]=Off; 1=On)
43	OPFOR Cntrl Msr Points Status ([0]=Off; 1=On)
44	OPFOR Cntrl Msr Lines Status ([0]=Off; 1=On)
45	OPFOR Cntrl Msr Areas Status ([0]=Off; 1=On)
46	OPFOR Cntrl Msr Routes Status ([0]=Off; 1=On)
47	OPFOR Obstacle Status ([0]=Off; 1=On)
48	OPFOR Cntrl Msr Crossings Status ([0]=Off; 1=On)
49	OPFOR Cntrl Msr Fire Plan Status ([0]=Off; 1=On)
50	OPFOR Cntrl Msr Map Feature Status ([0]=Off; 1=On)
51	OPFOR Cntrl Msr Army Status ([0]=Off; 1=On)
52	OPFOR Cntrl Msr Division Status ([0]=Off; 1=On)
53	OPFOR Cntrl Msr Regiment Status ([0]=Off; 1=On)
54	OPFOR Cntrl Msr Battalion Status ([0]=Off; 1=On)
55	OPFOR Cntrl Msr Company Status ([0]=Off; 1=On)

56 Map Option File Name [/edata/maps/menu/map.view]
57 BLUEFOR Unit Option File Name
[/edata/maps/menu/blue_unit.view]
58 BLUEFOR Control Measure Option File Name
[/edata/maps/menu/blue_cm.view]
59 BLUEFOR Obstacle Option File Name
[/edata/maps/menu/blue_obs.view]
60 OPFOR Unit Option File Name
[/edata/maps/menu/opfor_unit.view]
61 OPFOR Control Measure Option File Name
[/edata/maps/menu/opfor_cm.view]
62 OPFOR Obstacle Option File Name
[/edata/maps/menu/opfor_obs.view]

DATA BASE: G2_BUILD_MENU

TYPE:

Description
Description of the build product walking menu for the G2 workstation. This file is created from the command and control product source file.

DATA BASE: G2_REFERENCE_MENU

TYPE: FIXED ASCII

Description
Description of the reference product walking menu for the G2 workstation. This file is created from the reference product source file.

Column	Field Name	Type	Width	Dec
3	Menu Option Title	Character	20	
35	Reference Product Number	Numeric	6	

Note: Submenu Menu Option Titles must be indented 1 character from their parent and must appear immediately after the parent. Using a question mark "?" as the Menu Option Title will cause a blank item in the menu. The first record of this file is a comment.

DATA BASE: G2_VIEW_C2_MENU

TYPE: FIXED ASCII

Description
Description of the view situation product walking menu for the G2 workstation. This file is created from the command and control product source file.

Column	Field Name	Type	Width	Dec
3	Menu Option Title	Character	20	

35	C2 Product Number	Numeric	6

Note: Submenu Menu Option Titles must be indented 1 character from their parent and must appear immediately after the parent. Using a question mark "?" as the Menu Option Title will cause a blank item in the menu. The first record of this file is a comment.

DATA BASE: G3_BUILD_MENU

TYPE: FIXED ASCII

Description
Description of the build product walking menu for the G3 workstation. This file is created from the command and control product source file.

Column	Field Name	Type	Width	Dec
3	Menu Option Title	Character	20	
35	C2 Product Number	Numeric	6	

Note: Submenu Menu Option Titles must be indented 1 character from their parent and must appear immediately after the parent. Using a question mark "?" as the Menu Option Title will cause a blank item in the menu. The first record of this file is a comment.

DATA BASE: G3_REFERENCE_MENU

TYPE: FIXED ASCII

Description
Description of the reference product walking menu for the G3 workstation. This file is created from the reference product source file.

Column	Field Name	Type	Width	Dec
3	Menu Option Title	Character	20	
35	Reference Product Number	Numeric	6	

Note: Submenu Menu Option Titles must be indented 1 character from their parent and must appear immediately after the parent. Using a question mark "?" as the Menu Option Title will cause a blank item in the menu. The first record of this file is a comment.

DATA BASE: G3_VIEW_C2_MENU

TYPE: FIXED ASCII

Description
Description of the view situation product walking menu for the G3 workstation. This file is created from the command and control product source file.

Column	Field Name	Type	Width	Dec
3	Menu Option Title	Character	20	
35	C2 Product Number	Numeric	6	

Note: Submenu Menu Option Titles must be indented 1 character from their parent and must appear immediately after the parent. Using a question mark "?" as the Menu Option Title will cause a blank item in the menu. The first record of this file is a comment.

DATA BASE: G4_BUILD_MENU

TYPE: FIXED ASCII

Description
Description of the build product walking menu for the G4 workstation. This file is created from the command and control product source file.

Column	Field Name	Type	Width	Dec
3	Menu Option Title	Character	20	
35	C2 Product Number	Numeric	6	

Note: Submenu Menu Option Titles must be indented 1 character from their parent and must appear immediately after the parent. Using a question mark "?" as the Menu Option Title will cause a blank item in the menu. The first record of this file is a comment.

DATA BASE: G4_REFERENCE_MENU

TYPE: FIXED ASCII

Description
Description of the reference product walking menu for the G4 workstation. This file is created from the reference product source file.

Column	Field Name	Type	Width	Dec
3	Menu Option Title	Character	20	
35	Reference Product Number	Numeric	6	

Note: Submenu Menu Option Titles must be indented 1 character from their parent and must appear immediately after the parent. Using a question mark "?" as the Menu Option Title will cause a blank item in the menu. The first record of this file is a comment.

DATA BASE: G4_VIEW_C2_MENU

TYPE: FIXED ASCII

Description
Description of the view situation product walking menu for the G4 workstation. This file is created from the command and control product source file.

Column	Field Name	Type	Width	Dec
3	Menu Option Title	Character	20	
35	C2 Product Number	Numeric	6	

Note: Submenu Menu Option Titles must be indented 1 character from their parent and must appear immediately after the parent. Using a question mark "?" as the Menu Option Title will cause a blank item in the menu. The first record of this file is a comment.

DATA BASE: HELP_MENU

TYPE: FIXED ASCII

Description
Description of the help product walking menu. This file is created from the help product source file.

Column	Field Name	Type	Width	Dec
3	Menu Option Title	Character	20	
35	Help Product Number	Numeric	6	

Note: Submenu Menu Option Titles must be indented 1 character from their parent and must appear immediately after the parent. Using a question mark "?" as the Menu Option Title will cause a blank item in the menu. The first record of this file is a comment.

DATA BASE: HELP_NAME

TYPE: DELIMITED ASCII

Description
List of the help product names. This file is used to assign names to the help transactions in the reference recorded data.

Field #	Field Name	Type
1	Product Number	Numeric
2	Functional Area	Character
3	Data Category	Character
4	Data Element	Character

DATA BASE: HELP_PROD_DESC

TYPE: Ada

Description
Help product description data base. This data base indicates which record from the help product data base to use for a product.

```
type HDB_PRODUCT_DESC_TYPE  is
   record
      HDB_PRODUCT_CAT        :  SYS_PRODUCT_CAT;
      HDB_PRODUCT_START      :  HDB_NUM_PRODUCT_REC;
      HDB_PRODUCT_END        :  HDB_NUM_PRODUCT_REC;
   end record;
```

DATA BASE: HELP_PRODUCT

TYPE: Ada

Description
Help products.

```
type HDB_PRODUCT_TYPE  is
   record
      HDB_REPT_NUMBER_CHAR      :  SYS_PRODUCT_LENGTH  range 0..
                                   HDB_PRODUCT_SIZE;
      HDB_PRODUCT_TEXT           :  string (1..HDB_PRODUCT_SIZE);
   end record;
```

DATA BASE: HELP_SOURCE

TYPE: VARIABLE ASCII

Description
Description of the products to include in the help window.

Column	Field Name	Type	Width	Dec
Record 1 (Functional Area)				
1	Slash "/"	Character	1	
2	"F"	Character	1	
3	Functional Area Title	Character	20	
Record 2 (Data Category)				
1	Slash "/"	Character	1	
2	"C"	Character	1	
3	Data Category Title	Character	20	
Record 3 (Data Element)				
1	Slash "/"	Character	1	

2	"E"	Character	1
3	Data Element Title	Character	20

Record 4 (Report)

1	Textual Report Line	Character	80

Note: The Functional Area, Data Category, and Data Element records are used to build the product selection walking menu description files for the help button.

DATA BASE: ICON_STACK_DB

TYPE: BINARY

Description

Icon stack status data base. Indicates which stack positions are used and which ones are free (C format).

The Icon Stack Data Base is needed to keep track of the dynamic allocation of the icon stacks associated with each base window creation icon. It is a file pointed to by the environment variable Icon_Path. The screen manager is responsible for creating the data file, if it does not exist. The format consists of the X coordinate of the origin for each base icon, and then the process id of the background process associated with each of the maximum number of processes allowed per base window creation icon. A null value for the process id indicates that the stack position is not currently in use. The X origin coordinate for each base icon is used by the process window creation procedure to offset a new icon onto a stack.

DATA BASE: LUT_HILITE_DESC

TYPE: FIXED ASCII

Description

Description of the color lookup table files to use when features are hilighted.

Column	Field Name	Type	Width	Dec
1	Background Type (LUT_SHADE_VEG or LUT_NONE)	Character	13	
21	Lookup description file name	Character	60	

Note: The first record in this file is a comment.

DATA BASE: LUT_HILITE_MAP_ON

TYPE: FIXED ASCII

Description
Color lookup table to use when a map background (elevation band, shaded relief, or vegetation) is displayed and map features are hilighted.

Column	Field Name	Type	Width	Dec
3	Lookup Table Start Position	Numeric	3	0
8	Lookup Table End Position	Numeric	3	0
13	Red Intensity	Numeric	3	0
18	Green Intensity	Numeric	3	0
23	Blue Intensity	Numeric	3	0
29	Description	Character	52	

Note: The first record on this file is a comment.

DATA BASE: LUT_HILITE_MAP_OFF

TYPE: FIXED ASCII

Description
Color lookup table to use when a map with a null background is displayed and map features are hilighted.

Column	Field Name	Type	Width	Dec
3	Lookup Table Start Position	Numeric	3	0
8	Lookup Table End Position	Numeric	3	0
13	Red Intensity	Numeric	3	0
18	Green Intensity	Numeric	3	0
23	Blue Intensity	Numeric	3	0
29	Description	Character	52	

Note: The first record on this file is a comment.

DATA BASE: LUT_OVERLAY

TYPE: FIXED ASCII

Description
Color lookup table for the overlay planes.

Column	Field Name	Type	Width	Dec
3	Lookup Table Start Position	Numeric	3	0
8	Lookup Table End Position	Numeric	3	0
13	Red Intensity	Numeric	3	0
18	Green Intensity	Numeric	3	0
23	Blue Intensity	Numeric	3	0
29	Description	Character	52	

Note: The first record on this file is a comment.

DATA BASE: LUT_UNHILITE_DESC

TYPE: FIXED ASCII

Description
Description of the color lookup table files to use when features are not hilighted.

Column	Field Name	Type	Width	Dec
1	Background Type (LUT_SHADE_VEG or LUT_NONE)	Character	13	
21	Lookup description file name	Character	60	

Note: The first record in this file is a comment.

DATA BASE: LUT_UNHILITE_MAP_ON

TYPE: FIXED ASCII

Description
Color lookup table to use when a map background (elevation band, shaded relief, or vegetation) is displayed and map features are not hilighted.

Column	Field Name	Type	Width	Dec
3	Lookup Table Start Position	Numeric	3	0
8	Lookup Table End Position	Numeric	3	0
13	Red Intensity	Numeric	3	0
18	Green Intensity	Numeric	3	0
23	Blue Intensity	Numeric	3	0

29	Description	Character	52

Note: The first record on this file is a comment.

DATA BASE: LUT_UNHILITE_MAP_OFF

TYPE: FIXED ASCII

Description
Color lookup table to use when a map with a null background is displayed and map features are not hilighted.

Column	Field Name	Type	Width	Dec
3	Lookup Table Start Position	Numeric	3	0
8	Lookup Table End Position	Numeric	3	0
13	Red Intensity	Numeric	3	0
18	Green Intensity	Numeric	3	0
23	Blue Intensity	Numeric	3	0
29	Description	Character	52	

Note: The first record on this file is a comment.

DATA BASE: MAP_BUILD_MENU

TYPE: FIXED ASCII

Description
Description of the map options walking menu for the build window.

Column	Field Name	Type	Width	Dec
3	Menu Option Title	Character	20	
35	Map Option (SYS_MAP_CONTROL)	Character	25	

Note: Submenu Menu Option Titles must be indented 1 character from their parent and must appear immediately after the parent. Using a question mark "?" as the Menu Option Title will cause a blank item in the menu. The first record of this file is a comment. The map control menu also allows the integration of multiple selection menus into the walking menu options. The multiple selection menu definitions has the following format:

Column	Field Name	Type	Width	Dec
1	Multiple Selection Menu Code "M"	Character	1	

2	Multiple Selection Menu Record Code (M=Main; S=Subordinate). The Main option appears in the walking menu and as a title on the multiple selection menu. The subordinate appear as options on the multiple selection menu.	Character	1
3	Menu Option Title	Character	20
31	Map Option to use when an on-state is returned from the button menu manager (SYS_MAP_CONTROL)	Character	25
56	Map Option to use when an off-state is returned from the button menu manager (SYS_MAP_CONTROL)	Character	25

DATA BASE: MAP_DESC

TYPE: FIXED ASCII

Description
Description of the map image files to include in the tactical map system.

Column	Field Name	Type	Width	Dec
1	Background Type (SYS_MAP_BACKGROUND)	Character	16	
21	Map Scale (SYS_MAP_SCALES)	Character	9	
32	Name of map description file for this background type and map scale	Character	50	

Note: The first record of this file is a comment.

DATA BASE: MAP_LEGEND

TYPE: FIXED ASCII

Description
Description of what to display in the map legend.

Column	Field Name	Type	Width	Dec
3	Lookup Table Index	Numeric	3	0

8	Description	Character	20

Note: The first record contains the number of entries in the map legend in columns 3 through 5.

DATA BASE: MAP_MESSAGE_MENU

TYPE: FIXED ASCII

Description
Description of the map options walking menu for the view message window.

Column	Field Name	Type	Width	Dec
3	Menu Option Title	Character	20	
35	Map Options (SYS_MAP_CONTROL)	Character	25	

Note: Submenu Menu Option Titles must be indented 1 character from their parent and must appear immediately after the parent. Using a question mark "?" as the Menu Option Title will cause a blank item in the menu. The first record of this file is a comment. The map control menu also allows the integration of multiple selection menus into the walking menu options. The multiple selection menu definitions has the following format:

Column	Field Name	Type	Width	Dec
1	Multiple Selection Menu Code "M"	Character	1	
2	Multiple Selection Menu Record Code (M=Main; S=Subordinate). The Main option appears in the walking menu and as a title on the multiple selection menu. The subordinate appear as options on the multiple selection menu.	Character	1	
3	Menu Option Title	Character	20	
31	Map Option to use when an on-state is returned from the button menu manager (SYS_MAP_CONTROL)	Character	25	

56	Map Option to use when an off-state is returned from the button menu manager (SYS_MAP_CONTROL)	Character	25

DATA BASE: MAP_VIEW_C2_MENU

TYPE: FIXED ASCII

Description
description of the map options walking menu for the view situation window.

Column	Field Name	Type	Width	Dec
3	Menu Option Title	Character	20	
35	Map Option (SYS_MAP_CONTROL)	Character	25	

Note: Submenu Menu Option Titles must be indented 1 character from their parent and must appear immediately after the parent. Using a question mark "?" as the Menu Option Title will cause a blank item in the menu. The first record of this file is a comment. The map control menu also allows the integration of multiple selection menus into the walking menu options. The multiple selection menu definitions has the following format:

Column	Field Name	Type	Width	Dec
1	Multiple Selection Menu Code "M"	Character	1	
2	Multiple Selection Menu Record Code (M=Main; S=Subordinate). The Main option appears in the walking menu and as a title on the multiple selection menu. The subordinate appear as options on the multiple selection menu.	Character	1	
3	Menu Option Title	Character	20	
31	Map Option to use when an on-state is returned from the button menu manager (SYS_MAP_CONTROL)	Character	25	

56	Map Option to use when an off-state is returned from the button menu manager (SYS_MAP_CONTROL)	Character	25

DATA BASE: MESSAGE_LOG
TYPE: Ada

Description
Log of all the messages sent.

```
type CDB_LOG_LIMIT is  range 0..100;
type CDB_MESSAGE_LOG is  array (CDB_LOG_LIMIT) of
     CDB_SUM_MESSAGE_REC;

type CDB_MESSAGE_LOG_REC is
   record
      CDB_COUNT          :   CDB_LOG_LIMIT;
      CDB_LIST           :   CDB_MESSAGE_LOG;
   end record;
```

DATA BASE: OBSTACLE

TYPE: Ada

Description
Obstacles.

```
type SDB_OBSTACLE_REC is
   record
      SDB_ID             :       SDB_OBSTACLE_ID;
      SDB_OPPLAN         :       SYS_OPPLAN;
      SDB_SIDE           :       SDB_SIDE_TYPE;
      SDB_TYPE           :       SDB_OBSTACLE_TYPE;
      SDB_STATUS         :       SDB_OBSTACLE_STATUS;
      SDB_EFF_FROM_DATE  :       SYS_DATE_TIME;
      SDB_EFF_TO_DATE    :       SYS_DATE_TIME;
      SDB_LOCATION       :       SDB_LOCATION_REC;
      SDB_FRONTAGE       :       SYS_WIDTH_DEPTH;
      SDB_DEPTH          :       SYS_WIDTH_DEPTH;
      SDB_ORIENTATION    :       SYS_DEGREE;
      SDB_LANES_OR_GAPS  :       boolean;
      SDB_ECHELON        :       SDB_FORCE_ECHELON;
   end record;
```

DATA BASE: OBSTACLE_INDEX

TYPE: Ada

Description
Index for the obstacle data base.

```
type SDB_OBSTACLE_PTR is
   record
      SDB_ID            :      SDB_OBSTACLE_ID;
      SDB_OPPLAN        :      SYS_OPPLAN;
      SDB_EFF_FROM      :      SYS_DATE_TIME;
      SDB_EFF_TO        :      SYS_DATE_TIME;
      SDB_RECORD        :      SYS_DB_SIZE;
   end record;
```

DATA BASE: OBSTACLE_SOURCE

TYPE: VARIABLE ASCII

Description
Initial obstacles.

Column	Field Name	Type	Width	Dec
Record 1 (Obstacle Type)				
1	Obstacle Name	Character	12	
14	Obstacle Type	Character	6	
21	Echelon	Character	6	
28	Force (BLUE or RED)	Character	4	
Record 2 (Location)				
1	UTM Location	Character	8	
11	Frontage	Numeric	4	0
17	Depth	Numeric	4	0
22	Orientation	Numeric	3	0
26	Gap Flag (T=Yes, F=No)	Character	1	

Note: A date/time record is used to assign a date/time to the obstacle data. Format: *DDHHMM MON starting in column 1. (Example: *021800 SEP). The date/time record is followed by one comment record.

DATA BASE: OPFOR_AUTH_EQUIP

TYPE: Ada

Description
OPFOR authorized equipment levels.

```
type SDB_EQUIP_REC is
   record
      SDB_ID              :       SDB_EQUIPMENT;
      SDB_NAME            :       string (SDB_EQUIP_NAME_LEN);
      SDB_AUTHORIZED      :       SYS_QUANTITY;
      SDB_CATEGORY        :       SDB_EQUIP_CATEGORY;
   end record;

type SDB_EQUIP_ARRAY is array (SDB_EQUIPMENT) of
                                   SDB_EQUIP_REC;

type SDB_EQUIP_AUTH_LIST is
   record
      SDB_UNIT_ID         :       SDB_UNIT;
      SDB_TIME            :       SYS_DATE_TIME;
      SDB_OPPLAN          :       SYS_OPPLAN;
      SDB_COUNT           :       SDB_EQUIPMENT;
      SDB_LIST            :       SDB_EQUIP_ARRAY;
   end record;
```

DATA BASE: OPFOR_AUTH_EQUIP_INDEX

TYPE: Ada

Description
Index file for the OPFOR authorized equipment levels data base.

```
type SDB_OPFOR_EQUIP_PTR is
   record
      SDB_UNIT_ID         :       SDB_OPFOR_UNIT_ID;
      SDB_TIME            :       SYS_DATE_TIME;
      SDB_OPPLAN          :       SYS_OPPLAN;
      SDB_RECORD          :       SYS_DB_SIZE;
   end record;
```

DATA BASE: OPFOR_CM_EDIT_MENU

TYPE: FIXED ASCII

Description
Description of the walking menu to display when a OPFOR control measure is selected on the tactical map in a window with edit capability.

Column	Field Name	Type	Width	Dec
3	Menu Option Title	Character	20	

35	Control Measure Option (SYS_CM_OPTION)	Character	16

Note: Submenu Menu Option Titles must be indented 1 character from their parent and must appear immediately after the parent. Using a question mark "?" as the Menu Option Title will cause a blank item in the menu. The first record of this file is a comment.

DATA BASE: OPFOR_CM_VIEW_MENU

TYPE: FIXED ASCII

Description
Description of the walking menu to display when a OPFOR control measure is selected on the tactical map in a window with view only capability.

Column	Field Name	Type	Width	Dec
3	Menu Option Title	Character	20	
35	Control Measure Option (SYS_CM_OPTION)	Character	16	

Note: Submenu Menu Option Titles must be indented 1 character from their parent and must appear immediately after the parent. Using a question mark "?" as the Menu Option Title will cause a blank item in the menu. The first record of this file is a comment.

DATA BASE: OPFOR_CURR_EQUIP_INDEX

TYPE: Ada

Description
Index file for the OPFOR current equipment levels data base.

```
type SDB_OPFOR_EQUIP_QTY_PTR is
   record
      SDB_UNIT_ID       :       SDB_OPFOR_UNIT_ID;
      SDB_EQUIP_ID      :       SDB_OPFOR_EQUIP_ID;
      SDB_TIME          :       SYS_DATE_TIME;
      SDB_OPPLAN        :       SYS_OPPLAN;
      SDB_RECORD        :       SYS_DB_SIZE;
   end record;
```

DATA BASE: OPFOR_CURR_EQUIP

TYPE: Ada

Description
OPFOR current equipment levels.

```
type SDB_OPFOR_EQUIP_QTY is
   record
```

```
    SDB_UNIT_ID        :        SDB_OPFOR_UNIT_ID;
    SDB_EQUIP_ID       :        SDB_OPFOR_EQUIP_ID;
    SDB_TIME           :        SYS_DATE_TIME;
    SDB_OPPLAN         :        SYS_OPPLAN;
    SDB_OPERATIONAL    :        SYS_QUANTITY;
end record;
```

DATA BASE: OPFOR_EQUIP_NAME

TYPE: FIXED ASCII

Description
List of the OPFOR equipment names. This file is used to assign names to the equipment types in the situation data base.

Column	Field Name	Type	Width	Dec
1	Equipment Name	Character	12	

Note: The first record in this file is a comment.

DATA BASE: OPFOR_EQUIP_SOURCE

TYPE: VARIABLE ASCII

Description
Initial equipment levels for OPFOR units.

Column	Field Name	Type	Width	Dec
Record 1 (Unit)				
14	Unit Name	Character	12	
35	Number of Equipment Types	Numeric	2	0
Record 2 (Equip)				
15	Equipment Name	Character	12	
30	Authorized Amount	Numeric	5	0
40	Operational Amount	Numeric	5	0

Note: The Equip records must immediately follow the Unit record. The number of equip records must equal the number of equipment types in the Unit record. A date/time record is used to assign a date/time to the ammunition data. Format: *DDHHMM MON starting in column 1. (Example: *021800 SEP). The date/time record is followed by two comment records.

DATA BASE: OPFOR_OBS_EDIT_MENU

TYPE: FIXED ASCII

Description
Description of the walking menu to display when a OPFOR obstacle is selected on the tactical map in a window with view only capability.

Column	Field Name	Type	Width	Dec
3	Menu Option Title	Character	20	
35	Obstacle Option (SYS_OBS_OPTION)	Character	15	

Note: Submenu Menu Option Titles must be indented 1 character from their parent and must appear immediately after the parent. Using a question mark "?" as the Menu Option Title will cause a blank item in the menu. The first record of this file is a comment.

DATA BASE: OPFOR_OBS_VIEW_MENU

TYPE: FIXED ASCII

Description
Description of the walking menu to display when a OPFOR obstacle is selected on the tactical map in a window with edit capability.

Column	Field Name	Type	Width	Dec
3	Menu Option Title	Character	20	
35	Obstacle Option (SYS_OBS_OPTION)	Character	15	

Note: Submenu Menu Option Titles must be indented 1 character from their parent and must appear immediately after the parent. Using a question mark "?" as the Menu Option Title will cause a blank item in the menu. The first record of this file is a comment.

DATA BASE: OPFOR_ORGANIC_TASK_ORG

TYPE: VARIABLE ASCII

Description
Organic task organization for the OPFOR units.

Column	Field Name	Type	Width	Dec
1	Unit Name	Character	12	

Note: Subordinate unit names must be indented 2 spaces from their parent units name.

DATA BASE: OPFOR_REINFORCE_TIME

TYPE: FIXED ASCII

Description
Initial reinforcing times for OPFOR units.

Column	Field Name	Type	Width	Dec
5	Unit Number	Numeric	3	0
14	Unit Name	Character	12	
35	Reinforcing Hours	Numeric	4	0
53	Percent Strength	Numeric	3	0

Note: The first record in this file is a comment.

DATA BASE: OPFOR_TASK_ORG_SOURCE

TYPE: VARIABLE ASCII

Description
Initial task organization for the OPFOR units.

Column	Field Name	Type	Width	Dec
Record 1 (Echelon Count)				
1	Echelon Count	Numeric	3	0
Record 2 (Echelon Name)				
1	Echelon Name	Character	20	
Record 3 (Unit)				
1	Unit Name	Character	12	

Note: The Echelon Name records must appear directly after the Echelon Count Record. Each subsequent echelon record must be indented 2 spaces from the previous one. Subordinate unit names in the Unit record must be indented 2 spaces from their parent units name. A date/time record is used to assign a date/time to the task organization data. Format: *DDHHMM MON starting in column 1. (Example: *021800 SEP). The date/time record is followed by one comment record and the Echelon Count and Echelon Name records.

DATA BASE: OPFOR_UNIT_CONVERT

TYPE: Ada

Description
Data base to convert OPFOR unit names to unit numbers.

```
type OPFOR_ORGANIC_UNIT is
   record
      OLD_ID          :       SDB_OPFOR_UNIT_ID;
      NEW_ID          :       SDB_OPFOR_UNIT_ID;
      NAME            :       string (SDB_UNIT_NAME_LEN);
end record;
```

DATA BASE: OPFOR_UNIT_EDIT_MENU

TYPE: FIXED ASCII

Description
Description of the walking menu to display when a OPFOR unit is selected on a tactical map in a window with edit capability.

Column	Field Name	Type	Width	Dec
3	Menu Option Title	Character	20	
35	Unit Option (SYS_UNIT_OPTION)	Character	15	

Note: Submenu Menu Option Titles must be indented 1 character from their parent and must appear immediately after the parent. Using a question mark "?" as the Menu Option Title will cause a blank item in the menu. The first record of this file is a comment.

DATA BASE: OPFOR_UNIT_LOC

TYPE: Ada

Description
OPFOR unit location data base.

```
type SDB_UNIT_LOCATION is
   record
      SDB_UNIT_ID       :       SDB_UNIT;
      SDB_TIME          :       SYS_DATE_TIME;
      SDB_OPPLAN        :       SYS_OPPLAN;
      SDB_LOCATION      :       SDB_LOCATION_REC;
   end record;
```

DATA BASE: OPFOR_UNIT_LOC_INDEX

TYPE: Ada

Description
Index file for the OPFOR unit location data base.

```
type SDB_OPFOR_LOCATION_PTR is
   record
      SDB_UNIT_ID       :       SDB_OPFOR_UNIT_ID;
      SDB_TIME          :       SYS_DATE_TIME;
      SDB_OPPLAN        :       SYS_OPPLAN;
      SDB_RECORD        :       SYS_DB_SIZE;
   end record;
```

DATA BASE: OPFOR_UNIT_LOC_SOURCE

TYPE: FIXED ASCII

Description
Initial unit locations for the OPFOR units.

Column	Field Name	Type	Width	Dec
14	Unit Name	Character	12	
36	UTM Letters	Character	2	
38	UTM X Coordinate	Numeric	3	0
41	UTM Y Coordinate	Numeric	3	0

Note: A date/time record is used to assign a date/time to the unit location data. Format: *DDHHMM MON starting in column 1. (Example: *021800 SEP). The date/time record is followed by two comment records.

DATA BASE: OPFOR_UNIT_NAME

TYPE: DELIMITED ASCII

Description
List of the OPFOR unit names. This file is used to assign names to the unit transactions in the situation recorded data.

Field #	Field Name	Type
1	Unit Number	Numeric
2	Unit Name	Character

DATA BASE: OPFOR_UNIT_STATUS_INDEX

TYPE: Ada

Description
Index file for the OPFOR unit status data base.

```
type SDB_OPFOR_STATUS_PTR is
   record
      SDB_UNIT_ID       :       SDB_OPFOR_UNIT_ID;
      SDB_TIME          :       SYS_DATE_TIME;
      SDB_OPPLAN        :       SYS_OPPLAN;
      SDB_RECORD        :       SYS_DB_SIZE;
   end record;
```

DATA BASE: OPFOR_UNIT_STATUS

TYPE: Ada

Description
OPFOR unit status.

```
type SDB_OPFOR_UNIT_STATUS is
   record
      SDB_UNIT_ID       :       SDB_BLUEFOR_UNIT_ID;
      SDB_TIME          :       SYS_DATE_TIME;
      SDB_OPPLAN        :       SYS_OPPLAN;
      SDB_NAME          :       string (SDB_UNIT_NAME_LEN);
      SDB_ECHELON       :       SDB_FORCE_ECHELON;
      SDB_TYPE          :       SDB_UNIT_TYPE;
      SDB_PARENT        :       SDB_OPFOR_UNIT_ID;
      SDB_HIGHER_ECH    :       SDB_OPFOR_UNIT_ID;
      SDB_NEXT_SIBLING  :       SDB_OPFOR_UNIT_ID;
      SDB_FIRST_CHILD   :       SDB_OPFOR_UNIT_ID;
      SDB_MISSION       :       SDB_FORCE_MISSION;
      SDB_ACTIVITY      :       SDB_FORCE_ACTIVITY;
      SDB_REINFORCE_HR  :       SYS_HOUR;
      SDB_PERCENT_STR   :       SYS_PERCENT;
   end record;
```

DATA BASE: OPFOR_UNIT_STATUS_SOURCE

TYPE: FIXED ASCII

Description
Initial status of the OPFOR units.

Column	Field Name	Type	Width	Dec
5	Unit Number	Numeric	3	0
14	Unit Name	Character	12	
34	Unit Size	Numeric	1	0

47	Unit Branch/Duty	Numeric	1	0
60	Unit Mission	Numeric	1	0

Note: A date/time record is used to assign a date/time to the unit location data. Format: *DDHHMM MON starting in column 1. (Example: *021800 SEP). The date/time record is followed by four comment records.

DATA BASE: OPFOR_UNIT_VIEW_MENU

TYPE: FIXED ASCII

Description
Description of the walking menu to display when a OPFOR unit is selected on the tactical map in a window with view only capability.

Column	Field Name	Type	Width	Dec
3	Menu Option Title	Character	20	
35	Unit Option (SYS_UNIT_OPTION)	Character	15	

Note: Submenu Menu Option Titles must be indented 1 character from their parent and must appear immediately after the parent. Using a question mark "?" as the Menu Option Title will cause a blank item in the menu. The first record of this file is a comment.

DATA BASE: OPLAN_LIST

TYPE: Ada

Description
List of existing Operational plans in the system.

```
type SDB_OPPLAN_REC is
   record
      SDB_OPPLAN_ID          :  SYS_OPPLAN;
      SDB_TYPE               :  SDB_OPPLAN_TYPE;
      SDB_OPPLAN_NAME        :  STRING (SYS_POP_UP_TEXT);
      SDB_BASE               :  SYS_OPPLAN;
      SDB_DATE_TIME          :  SYS_DATE_TIME;
end record;
```

DATA BASE: OPLAN_LIST_SOURCE

TYPE: FIXED ASCII

Description
Operational plans to initially have in the system.

Column	Field Name	Type	Width	Dec
1	OPLAN Name	Character	20	
2	OPLAN Type (BASE_SCENARIO, G2_PERSONAL, G3_PERSONAL, G4_PERSONAL, SHARED)	Character	13	

DATA BASE: PRODUCT_HARDCOPY

TYPE: FIXED ASCII

Description
ASCII output file of the products printed by CDB_HARDCOPY.

Column	Field Name	Type	Width	Dec
1	ASCII Output	Character	80	

DATA BASE: REFERENCE_HEADER

TYPE: Ada

Description
Reference report headers.

```
type FDB_HEADER_TYPE  is
   record
      FDB_HEAD_NUMBER_CHAR    :  SYS_HEADER_LENGTH  range 0..
                                 FDB_HEADER_SIZE;
      FDB_HEADER_TEXT         :  string (1..FDB_HEADER_SIZE);
   end record;
```

DATA BASE: REFERENCE_NAME

TYPE: DELIMITED ASCII

Description
List of the reference product names. This file is used to assign names to the reference transactions in the reference recorded data.

Field #	Field Name	Type
1	Product Number	Numeric
2	Functional Area	Character
3	Data Category	Character
4	Data Element	Character
5	Level of Detail (D= Detail; A=Aggregate, S=Summary)	Character

DATA BASE: REFERENCE_PROD_DESC

TYPE: Ada

Description
Reference product description data base. This data base indicates which records from the reference product data base to use for a product.

```
type FDB_PRODUCT_DESC_TYPE  is
   record
      FDB_PRODUCT_CAT          :  SYS_PRODUCT_CAT;
      FDB_PRODUCT_HDR_START    :  FDB_NUM_HEADER_REC;
      FDB_PRODUCT_HDR_END      :  FDB_NUM_HEADER_REC;
      FDB_PRODUCT_START        :  FDB_NUM_PRODUCT_REC;
      FDB_PRODUCT_END          :  FDB_NUM_PRODUCT_REC;
   end record;
```

DATA BASE: REFERENCE_PRODUCT

TYPE: Ada

Description
Reference product data base.

```
type FDB_PRODUCT_TYPE  is
   record
      FDB_REPT_NUMBER_CHAR     :  SYS_PRODUCT_LENGTH  range 0..
                                  FDB_PRODUCT_SIZE;
      FDB_PRODUCT_TEXT         :  string (1..FDB_PRODUCT_SIZE);
   end record;
```

DATA BASE: REFERENCE_RECORD

TYPE: BINARY

Description
Reference data recording transactions.

This data base contains binary images of the messages in MSG_RF_RECORD_LIST. The message type is contained in MSG_RECORD_TYPE and the length is in MSG_BYTES_IN_MSG. The UUX_IO utilities should be used to interact with this data base.

DATA BASE: REFERENCE_SOURCE

TYPE: VARIABLE ASCII

Description
Description of the reference products to include in the view reference window.

Column	Field Name	Type	Width	Dec
Record 1 (Functional Area)				
1	Slash "/"	Character	1	
2	"F"	Character	1	
3	Routing Code 1 = G2 2 = G3 3 = G2 & G3 4 = G4 5 = G2 & G4 6 = G3 & G4 7 = G2, G3, & G4	Numeric	1	0
4	Functional Area Title	Character	20	
Record 2 (Data Category)				
1	Slash "/"	Character	1	
2	"C"	Character	1	
3	Routing Code 1 = G2 2 = G3 3 = G2 & G3 4 = G4 5 = G2 & G4 6 = G3 & G4 7 = G2, G3, & G4	Numeric	1	0
4	Data Category Title	Character	20	
Record 3 (Data Element)				
1	Slash "/"	Character	1	
2	"E"	Character	1	
3	Routing Code 1 = G2 2 = G3 3 = G2 & G3 4 = G4 5 = G2 & G4 6 = G3 & G4 7 = G2, G3, & G4	Numeric	1	0
4	Level of Detail (D=Detail, A=Aggregate, S=Summary)	Character	1	
5	Colon ":"	Character	1	
6	Data Element Title	Character	20	

Record 4 (Report Header)

1	Percent Sign "%"	Character	1
2	Report Header Line	Character	80

Record 5 (Report)

1	Textual Report Line	Character	80

Note: The Functional Area, Data Category, and Data Element records are used to build the product selection walking menu description files for the view reference window.

DATA BASE: ROOT_WINDOW_MENU

TYPE: FIXED ASCII

Description
Description of the walking menu to display in the root window. The root window is any part of the screen where a window or button is not displayed.

Column	Field Name	Type	Width	Dec
3	Menu Option Title	Character	20	
35	Root Window Option (SCL_ROOT_OPTION)	Character	16	

Note: Submenu Menu Option Titles must be indented 1 character from their parent and must appear immediately after the parent. Using a question mark "?" as the Menu Option Title will cause a blank item in the menu. The first record of this file is a comment.

DATA BASE: SCREEN_DUMP_IMAGE

TYPE: BINARY

Description
Bitmap image of a screen of a Sun workstation (Bitmap format).

DATA BASE: SEND_PARTICIPANT_SOURCE

TYPE: FIXED ASCII

Description
List of the participants that messages can be sent to.

Column	Field Name	Type	Width	Dec
1	Participant Name	Character	10	
12	Participant Type (G2, G3, G4, EXPERIMENTER)	Character	20	

DATA BASE: SHAD_RELF_1TO160

TYPE: BINARY

Description
Shaded relief image file for the 1:160,000 map scale.

This shaded relief data base consists of byte values representing the color lookup table value to use to represent the relief shading. The data is organized in column/row order (columns within rows) from northwest to southeast with 40 columns and 25 rows. Each record is organized in column/row order from northwest to southeast with 64 columns and 80 rows (5120 bytes).

DATA BASE: SHAD_RELF_1TO400

TYPE: BINARY

Description
Shaded relief image file for the 1:400,000 map scale.

This shaded relief data base consists of byte values representing the color lookup table value to use to represent the relief shading. The data is organized in column/row order (columns within rows) from northwest to southeast with 16 columns and 10 rows. Each record is organized in column/row order from northwest to southeast with 64 columns and 80 rows (5120 bytes).

DATA BASE: SHAD_RELF_1TO80

TYPE: BINARY

Description
Shaded relief image file for the 1:80,000 map scale.

This shaded relief data base consists of byte values representing the color lookup table value to use to represent the relief shading. The data is organized in column/row order (columns within rows) from northwest to southeast with 79 columns and 49 rows. Each record is organized in column/row order from northwest to southeast with 64 columns and 80 rows (5120 bytes).

DATA BASE: SHAD_RELF_1TO800

TYPE: BINARY

Description
Shaded relief image file for the 1:800,000 map scale.

This shaded relief data base consists of byte values representing the color lookup table value to use to represent the relief shading. The data is organized in column/row order (columns within rows) from northwest to southeast with 8 columns and 6 rows. Each record is organized in column/row order from northwest to southeast with 64 columns and 80 rows (5120 bytes).

DATA BASE: SHAD_RELF_DESC_1TO160

TYPE: VARIABLE ASCII

Description
Description of the 1:160,000 shaded relief image file.

Column	Field Name	Type	Width	Dec
Record 1 (Image File)				
1	Shaded relief image file name for this map scale	Character	60	
Record 2 (Data Base Size)				
1	Number of shaded relief image records in the X direction for this map scale	Numeric	5	0
6	Number of shaded relief image records in the Y direction for this map scale	Numeric	5	0
Record 3 (Record Size)				
1	Number of shaded relief image points in a record in the X direction	Numeric	5	0
6	Number of shaded relief image points in a record in the Y direction	Numeric	5	0
Record 4 (Data Base Point Size)				
1	Number of shaded relief image points in the data base in the X direction	Numeric	6	0
7	Number of shaded relief image points in the data base in the Y direction	Numeric	6	0
14	Number of meters per pixel for this map scale	Numeric	7	3
Record 5 (Map Origin)				
4	Number of meters in the X direction from MA000000 to the northwest corner of the shaded relief image	Numeric	7	0
14	Number of meters in the Y direction from MA000000 to the northwest corner of the shaded relief image	Numeric	7	0
Record 6 (Grid Interval)				

1	Grid interval for this map scale (in meters)	Numeric	5	0

DATA BASE: SHAD_RELF_DESC_1TO400

TYPE: VARIABLE ASCII

Description
Description of the 1:400,000 shaded relief image file.

Column	Field Name	Type	Width	Dec
Record 1 (Image File)				
1	Shaded relief image file name for this map scale	Character	60	
Record 2 (Data Base Size)				
1	Number of shaded relief image records in the X direction for this map scale	Numeric	5	0
6	Number of shaded relief image records in the Y direction for this map scale	Numeric	5	0
Record 3 (Record Size)				
1	Number of shaded relief image points in a record in the X direction	Numeric	5	0
6	Number of shaded relief image points in a record in the Y direction	Numeric	5	0
Record 4 (Data Base Point Size)				
1	Number of shaded relief image points in the data base in the X direction	Numeric	6	0
7	Number of shaded relief image points in the data base in the Y direction	Numeric	6	0
14	Number of meters per pixel for this map scale	Numeric	7	3
Record 5 (Map Origin)				
4	Number of meters in the X direction from MA000000 to the northwest corner of the shaded relief image	Numeric	7	0
14	Number of meters in the Y direction from MA000000 to the northwest corner of the shaded	Numeric	7	0

relief image

Column	Field Name	Type	Width	Dec
Record 6 (Grid Interval)				
1	Grid interval for this map scale (in meters)	Numeric	5	0

DATA BASE: SHAD_RELF_DESC_1TO80

TYPE: VARIABLE ASCII

Description

Description of the 1:80,000 shaded relief image file.

Column	Field Name	Type	Width	Dec
Record 1 (Image File)				
1	Shaded relief image file name for this map scale	Character	60	
Record 2 (Data Base Size)				
1	Number of shaded relief image records in the X direction for this map scale	Numeric	5	0
6	Number of shaded relief image records in the Y direction for this map scale	Numeric	5	0
Record 3 (Record Size)				
1	Number of shaded relief image points in a record in the X direction	Numeric	5	0
6	Number of shaded relief image points in a record in the Y direction	Numeric	5	0
Record 4 (Data Base Point Size)				
1	Number of shaded relief image points in the data base in the X direction	Numeric	6	0
7	Number of shaded relief image points in the data base in the Y direction	Numeric	6	0
14	Number of meters per pixel for this map scale	Numeric	7	3
Record 5 (Map Origin)				

4	Number of meters in the X direction from MA000000 to the northwest corner of the shaded relief image	Numeric	7	0
14	Number of meters in the Y direction from MA000000 to the northwest corner of the shaded relief image	Numeric	7	0
Record 6 (Grid Interval)				
1	Grid interval for this map scale (in meters)	Numeric	5	0

DATA BASE: SHAD_RELF_DESC_1TO800

TYPE: VARIABLE ASCII

Description
Description of the 1:800,000 shaded relief image file.

Column	Field Name	Type	Width	Dec
Record 1 (Image File)				
1	Shaded relief image file name for this map scale	Character	60	
Record 2 (Data Base Size)				
1	Number of shaded relief image records in the X direction for this map scale	Numeric	5	0
6	Number of shaded relief image records in the Y direction for this map scale	Numeric	5	0
Record 3 (Record Size)				
1	Number of shaded relief image points in a record in the X direction	Numeric	5	0
6	Number of shaded relief image points in a record in the Y direction	Numeric	5	0
Record 4 (Data Base Point Size)				
1	Number of shaded relief image points in the data base in the X direction	Numeric	6	0
7	Number of shaded relief image points in the data base in the Y direction	Numeric	6	0
14	Number of meters per pixel for	Numeric	7	3

this map scale

Record 5 (Map Origin)				
4	Number of meters in the X direction from MA000000 to the northwest corner of the shaded relief image	Numeric	7	0
14	Number of meters in the Y direction from MA000000 to the northwest corner of the shaded relief image	Numeric	7	0
Record 6 (Grid Interval)				
1	Grid interval for this map scale (in meters)	Numeric	5	0

DATA BASE: SITUATION_RECORD

TYPE: BINARY

Description
Situation data recording transactions.

This data base contains binary images of the messages in MSG_SD_RECORD_LIST. The message type is contained in MSG_RECORD_TYPE and the length is in MSG_BYTES_IN_MSG. The UUX_IO utilities should be used to interact with this data base.

DATA BASE: TASK_ORG_TOOL_MENU

TYPE: FIXED ASCII

Description
Description of the walking menu to display as a popup menu for the task organization tool.

Column	Field Name	Type	Width	Dec
3	Menu Option Title	Character	20	

35	Task Organization Tool Option (VPUM_OPTS)	Character	16

Note: Submenu Menu Option Titles must be indented 1 character from their parent and must appear immediately after the parent. Using a question mark "?" as the Menu Option Title will cause a blank item in the menu. The first record of this file is a comment.

DATA BASE: TASK_ORG_TOP_UNIT_MENU

TYPE: FIXED ASCII

Description
Description of the walking menu to display when the top unit button is selected in the task organization tool.

Column	Field Name	Type	Width	Dec
3	Menu Option Title	Character	20	
35	Unit Number	Numeric	6	

Note: Submenu Menu Option Titles must be indented 1 character from their parent and must appear immediately after the parent. Using a question mark "?" as the Menu Option Title will cause a blank item in the menu. The first record of this file is a comment.

DATA BASE: TASK_ORG_UNIT_MENU

TYPE: FIXED ASCII

Description
Description of the walking menu to display as a popup menu when a unit is selected in the task organization tool.

Column	Field Name	Type	Width	Dec
3	Menu Option Title	Character	20	

35	Task Organization Tool Unit Option (UPUM_OPTS)	Character	16

Note: Submenu Menu Option Titles must be indented 1 character from their parent and must appear immediately after the parent. Using a question mark "?" as the Menu Option Title will cause a blank item in the menu. The first record of this file is a comment.

DATA BASE: TASK_ORG_UNIT_TYPE_MENU

TYPE: FIXED ASCII

Description
Description of the multiple selection menu to display when the unit type button is selected in the task organization tool.

Column	Field Name	Type	Width	Dec
1	Menu Option Title	Character	20	
22	Unit Type Option (UTB_OPTS)	Character	22	
56	Initial Status (On or Off)	Character	3	

Note: The first record of this file contains the number of menu items in columns 1 and 2.

DATA BASE: TOOL_MENU

TYPE: FIXED ASCII

Description
Description of the walking menu defining the tools available in the tool window.

Column	Field Name	Type	Width	Dec
3	Menu Option Title	Character	20	
35	Tool Option (SYS_TOOLS)	Character	16	

Note: Submenu Menu Option Titles must be indented 1 character from their parent and must appear immediately after the parent. Using a question mark "?" as the Menu Option Title will cause a blank item in the menu. The first record of this file is a comment.

DATA BASE: TRAN_ACTIVITY

TYPE: DELIMITED ASCII

Description
Unit activity update recorded transactions.

Field #	Field Name	Type
1	Participant (G2, G3, G4, EX)	Character
2	Message Date (CCyymmdd)	Character
3	Message Time (hhmmss)	Numeric
4	Window Type	Character
5	Window Stack Index	Numeric
6	Scenario Date (CCyymmdd)	Character
7	Scenario Time (hhmm)	Numeric
8	OPLAN Number	Numeric
9	Force (SDB_SIDE_TYPE)	Character
10	Unit Number	Numeric
11	Unit Activity (SDB_FORCE_ACTIVITY)	Character

DATA BASE: TRAN_AMMUNITION

TYPE: DELIMITED ASCII

Description
Unit ammunition update recorded transactions.

Field #	Field Name	Type
1	Participant (G2, G3, G4, EX)	Character
2	Message Date (CCyymmdd)	Character
3	Message Time (hhmmss)	Numeric
4	Window Type	Character
5	Window Stack Index	Numeric
6	Scenario Date (CCyymmdd)	Character
7	Scenario Time (hhmm)	Numeric
8	OPLAN Number	Numeric
9	Force (SDB_SIDE_TYPE)	Character
10	Unit Number	Numeric
11	Ammunition Number	Numeric

12	Ammunition Quantity	Numeric

DATA BASE: TRAN_BLUEFOR_TASK_ORG

TYPE: DELIMITED ASCII

Description
BLUEFOR task organization update recorded transactions.

Field #	Field Name	Type
1	Participant (G2, G3, G4, EX)	Character
2	Message Date (CCyymmdd)	Character
3	Message Time (hhmmss)	Numeric
4	Window Type	Character
5	Window Stack Index	Numeric
6	Scenario Date (CCyymmdd)	Character
7	Scenario Time (hhmm)	Numeric
8	OPLAN Number	Numeric
9	Force (SDB_SIDE_TYPE)	Character
10	Unit Number	Numeric
11	Higher Echelon Unit Number	Numeric
12	Relationship (ORG, ATCH, DS, GS, GSR, OPCN)	Character

DATA BASE: TRAN_C2_REQUEST

TYPE: DELIMITED ASCII

Description
Request for command and control product recorded transactions.

Field #	Field Name	Type
1	Participant (G2, G3, G4, EX)	Character
2	Message Date (CCyymmdd)	Character
3	Message Time (hhmmss)	Numeric
4	Window Type	Character

5	Window Stack Index	Numeric
6	C2 Product Number	Numeric

DATA BASE: TRAN_C2_WINDOW

TYPE: DELIMITED ASCII

Description
View situation, build, and view message window manipulation recorded transactions.

Field #	Field Name	Type
1	Participant (G2, G3, G4, EX)	Character
2	Message Date (CCyymmdd)	Character
3	Message Time (hhmmss)	Numeric
4	Window Type	Character
5	Window Stack Index	Numeric
6	Window Action (S=Stop, T=Close Socket, N=Connect, O=Open, C=Close)	Character

DATA BASE: TRAN_CNTRL_MSR_DEL

TYPE: DELIMITED ASCII

Description
Control measure delete recorded transactions.

Field #	Field Name	Type
1	Participant (G2, G3, G4, EX)	Character
2	Message Date (CCyymmdd)	Character
3	Message Time (hhmmss)	Numeric
4	Window Type	Character
5	Window Stack Index	Numeric
6	Scenario Date (CCyymmdd)	Character
7	Scenario Time (hhmm)	Numeric
8	OPLAN Number	Numeric
9	Control Measure Number	Numeric

DATA BASE: TRAN_CNTRL_MSR_EFF_TIME

TYPE: DELIMITED ASCII

Description
Control measure effective time update recorded transactions.

Field #	Field Name	Type
1	Participant (G2, G3, G4, EX)	Character
2	Message Date (CCyymmdd)	Character
3	Message Time (hhmmss)	Numeric
4	Window Type	Character
5	Window Stack Index	Numeric
6	Scenario Date (CCyymmdd)	Character
7	Scenario Time (hhmm)	Numeric
8	OPLAN Number	Numeric
9	Control Measure Number	Numeric
10	Control Measure Effective Date (CCyymmdd)	Character
11	Control Measure Effective Time (hhmm)	Numeric

DATA BASE: TRAN_CNTRL_MSR_LOC

TYPE: DELIMITED ASCII

Description
Control measure location update recorded transactions.

Field #	Field Name	Type
1	Participant (G2, G3, G4, EX)	Character
2	Message Date (CCyymmdd)	Character
3	Message Time (hhmmss)	Numeric
4	Window Type	Character
5	Window Stack Index	Numeric
6	Scenario Date (CCyymmdd)	Character

7	Scenario Time (hhmm)	Numeric
8	OPLAN Number	Numeric
9	Control Measure Number	Numeric
10	Point 1 X Coordinate	Numeric
11	Point 1 Y Coordinate	Numeric
12	Point 2 X Coordinate	Numeric
13	Point 2 Y Coordinate	Numeric
14	Point 3 X Coordinate	Numeric
15	Point 3 Y Coordinate	Numeric
16	Point 4 X Coordinate	Numeric
17	Point 4 Y Coordinate	Numeric
18	Point 5 X Coordinate	Numeric
19	Point 5 Y Coordinate	Numeric
20	Point 6 X Coordinate	Numeric
21	Point 6 Y Coordinate	Numeric
22	Point 7 X Coordinate	Numeric
23	Point 7 Y Coordinate	Numeric
24	Point 8 X Coordinate	Numeric
25	Point 8 Y Coordinate	Numeric
26	Point 9 X Coordinate	Numeric
27	Point 9 Y Coordinate	Numeric
28	Point 10 X Coordinate	Numeric
29	Point 10 Y Coordinate	Numeric
30	Point 11 X Coordinate	Numeric
31	Point 11 Y Coordinate	Numeric
32	Point 12 X Coordinate	Numeric
33	Point 12 Y Coordinate	Numeric
34	Point 13 X Coordinate	Numeric

35	Point 13 Y Coordinate	Numeric
36	Point 14 X Coordinate	Numeric
37	Point 14 Y Coordinate	Numeric
38	Point 15 X Coordinate	Numeric
39	Point 15 Y Coordinate	Numeric

DATA BASE: TRAN_CNTRL_MSR_STAT

TYPE: DELIMITED ASCII

Description
Control measure status update recorded transactions.

Field #	Field Name	Type
1	Participant (G2, G3, G4, EX)	Character
2	Message Date (CCyymmdd)	Character
3	Message Time (hhmmss)	Numeric
4	Window Type	Character
5	Window Stack Index	Numeric
6	Scenario Date (CCyymmdd)	Character
7	Scenario Time (hhmm)	Numeric
8	OPLAN Number	Numeric
9	Control Measure Number	Numeric
10	Control Measure Status (SDB_CONTROL_MEASURE_STATUS)	Character

DATA BASE: TRAN_CONTROL_REQUEST

TYPE: DELIMITED ASCII

Description
Request for experiment control product recorded transactions.

Field #	Field Name	Type
1	Participant (G2, G3, G4, EX)	Character
2	Message Date (CCyymmdd)	Character

3	Message Time (hhmmss)	Numeric
4	Window Type	Character
5	Window Stack Index	Numeric
6	Experiment Control Product Number	Numeric

DATA BASE: TRAN_CONTROL_WINDOW

TYPE: DELIMITED ASCII

Description
Tool and experiment control window manipulation recorded transactions.

Field #	Field Name	Type
1	Participant (G2, G3, G4, EX)	Character
2	Message Date (CCyymmdd)	Character
3	Message Time (hhmmss)	Numeric
4	Window Type	Character
5	Window Stack Index	Numeric
6	Window Action (S=Stop, T=Close Socket, N=Connect, O=Open, C=Close)	Character

DATA BASE: TRAN_EQUIPMENT

TYPE: DELIMITED ASCII

Description
Unit equipment update recorded transactions.

Field #	Field Name	Type
1	Participant (G2, G3, G4, EX)	Character
2	Message Date (CCyymmdd)	Character
3	Message Time (hhmmss)	Numeric
4	Window Type	Character
5	Window Stack Index	Numeric
6	Scenario Date (CCyymmdd)	Character

7	Scenario Time (hhmm)	Numeric
8	OPLAN Number	Numeric
9	Force (SDB_SIDE_TYPE)	Character
10	Unit Number	Numeric
11	Equipment Number	Numeric
12	Equipment Quantity	Numeric

DATA BASE: TRAN_FUEL

TYPE: DELIMITED ASCII

Description
Unit fuel update recorded transactions.

Field #	Field Name	Type
1	Participant (G2, G3, G4, EX)	Character
2	Message Date (CCyymmdd)	Character
3	Message Time (hhmmss)	Numeric
4	Window Type	Character
5	Window Stack Index	Numeric
6	Scenario Date (CCyymmdd)	Character
7	Scenario Time (hhmm)	Numeric
8	OPLAN Number	Numeric
9	Force (SDB_SIDE_TYPE)	Character
10	Unit Number	Numeric
11	MOGAS Quantity	Numeric
12	AVGAS Quantity	Numeric

13	Diesel Quantity	Numeric

DATA BASE: TRAN_LOOKUP_TABLE

TYPE: DELIMITED ASCII

Description
Color lookup table update recorded transactions.

Field #	Field Name	Type
1	Participant (G2, G3, G4, EX)	Character
2	Message Date (CCyymmdd)	Character
3	Message Time (hhmmss)	Numeric
4	Window Type	Character
5	Window Stack Index	Numeric
6	Background Type (N=None, F=Full Background Color)	Character
7	Background Status (N=No Change, H=Hilite, U=Unhilite)	Character
8	Road Status (N=No Change, H=Hilite, U=Unhilite)	Character
9	Hydrography Status (N=No Change, H=Hilite, U=Unhilite)	Character
10	Urban Status (N=No Change, H=Hilite, U=Unhilite)	Character
11	Miscellaneous Status (N=No Change, H=Hilite, U=Unhilite)	Character

DATA BASE: TRAN_MAP

TYPE: DELIMITED ASCII

Description
Tactical map control recorded transactions.

Field #	Field Name	Type
1	Participant (G2, G3, G4, EX)	Character
2	Message Date (CCyymmdd)	Character
3	Message Time (hhmmss)	Numeric

4	Window Type	Character
5	Window Stack Index	Numeric
6	Background Type (CCM, ELEV, SHAD, 3D, VEG, NONE)	Character
7	Map Scale (1:40, 1:80, 1:160, 1:400, 1:800)	Character
8	Map Center X Coordinate	Numeric
9	Map Center Y Coordinate	Numeric
10	Map Grid Status (T=On, False=Off)	Character
11	Contour Status (T=On, False=Off)	Character
12	BLUEFOR Division Unit Status (T=On, False=Off)	Character
13	BLUEFOR Brigade Unit Status (T=On, False=Off)	Character
14	BLUEFOR Regiment Unit Status (T=On, False=Off)	Character
15	BLUEFOR Battalion Unit Status (T=On, False=Off)	Character
16	BLUEFOR Company Unit Status (T=On, False=Off)	Character
17	BLUEFOR Combat Unit Status (T=On, False=Off)	Character
18	BLUEFOR CS Unit Status (T=On, False=Off)	Character
19	BLUEFOR CSS Unit Status (T=On, False=Off)	Character
20	BLUEFOR Unit Name Status (T=On, False=Off)	Character
21	BLUEFOR Unit Symbol Status (T=On, False=Off)	Character
22	OPFOR Division Unit Status (T=On, False=Off)	Character
23	OPFOR Brigade Unit Status (T=On, False=Off)	Character
24	OPFOR Regiment Unit Status (T=On, False=Off)	Character

25	OPFOR Battalion Unit Status (T=On, False=Off)	Character
26	OPFOR Company Unit Status (T=On, False=Off)	Character
27	OPFOR Committed Unit Status (T=On, False=Off)	Character
28	OPFOR Reinforcing Unit Status (T=On, False=Off)	Character
29	OPFOR Artillery Unit Status (T=On, False=Off)	Character
30	OPFOR Unit Name Status (T=On, False=Off)	Character
31	OPFOR Unit Symbol Status (T=On, False=Off)	Character
32	BLUEFOR Echelon Above Corps Control Measure Status (T=On, False=Off)	Character
33	BLUEFOR Division Control Measure Status (T=On, False=Off)	Character
34	BLUEFOR Brigade Control Measure Status (T=On, False=Off)	Character
35	BLUEFOR Battalion Control Measure Status (T=On, False=Off)	Character
36	BLUEFOR Company Control Measure Status (T=On, False=Off)	Character
37	BLUEFOR Point Control Measure Status (T=On, False=Off)	Character
38	BLUEFOR Line Control Measure Status (T=On, False=Off)	Character
39	BLUEFOR Area Control Measure Status (T=On, False=Off)	Character
40	BLUEFOR Route Control Measure Status (T=On, False=Off)	Character
41	BLUEFOR Crossing Control Measure Status (T=On, False=Off)	Character
42	BLUEFOR Fire Plan Control Measure Status (T=On, False=Off)	Character

43	BLUEFOR Map Feature Control Measure Status (T=On, False=Off)	Character
44	OPFOR Army Control Measure Status (T=On, False=Off)	Character
45	OPFOR Division Control Measure Status (T=On, False=Off)	Character
46	OPFOR Regiment Control Measure Status (T=On, False=Off)	Character
47	OPFOR Battalion Control Measure Status (T=On, False=Off)	Character
48	OPFOR Company Control Measure Status (T=On, False=Off)	Character
49	OPFOR Point Control Measure Status (T=On, False=Off)	Character
50	OPFOR Line Control Measure Status (T=On, False=Off)	Character
51	OPFOR Area Control Measure Status (T=On, False=Off)	Character
52	OPFOR Route Control Measure Status (T=On, False=Off)	Character
53	OPFOR Crossing Control Measure Status (T=On, False=Off)	Character
54	OPFOR Fire Plan Control Measure Status (T=On, False=Off)	Character
55	OPFOR Map Feature Control Measure Status (T=On, False=Off)	Character

DATA BASE: TRAN_NEW_C2

TYPE: DELIMITED ASCII

Description

New command and control product recorded transactions.

Field #	Field Name	Type
1	Participant (G2, G3, G4, EX)	Character
2	Message Date (CCyymmdd)	Character
3	Message Time (hhmmss)	Numeric
4	Window Type	Character

5	Window Stack Index	Numeric
6	Message Product Number	Numeric
7	Send to G2 Flag (T=True, F=False)	Character
8	Send to G3 Flag (T=True, F=False)	Character
9	Send to G4 Flag (T=True, F=False)	Character
10	Send to Experimenter Flag (T=True, F=False)	Character

DATA BASE: TRAN_NEW_CNTRL_MSR

TYPE: DELIMITED ASCII

Description
New control measure recorded transactions.

Field #	Field Name	Type
1	Participant (G2, G3, G4, EX)	Character
2	Message Date (CCyymmdd)	Character
3	Message Time (hhmmss)	Numeric
4	Window Type	Character
5	Window Stack Index	Numeric
6	Scenario Date (CCyymmdd)	Character
7	Scenario Time (hhmm)	Numeric
8	OPLAN Number	Numeric
9	Control Measure Number	Numeric
10	Control Measure Name	Character
11	Color (SDB_SIDE_TYPE)	Character
12	Control Measure Type	Character
13	Display on Map Scale 1:40000 (T=True, F=False)	Character
14	Display on Map Scale 1:80000 (T=True, F=False)	Character

15	Display on Map Scale 1:160000 (T=True, F=False)	Character
16	Display on Map Scale 1:400000 (T=True, F=False)	Character
17	Display on Map Scale 1:800000 (T=True, F=False)	Character
18	Control Measure Status (SDB_CONTROL_MEASURE_STATUS)	Character
19	Effective Date (CCyymmdd)	Character
20	Effective Time (hhmm)	Numeric
21	Point 1 X Coordinate	Numeric
22	Point 1 Y Coordinate	Numeric
23	Point 2 X Coordinate	Numeric
24	Point 2 Y Coordinate	Numeric
25	Point 3 X Coordinate	Numeric
26	Point 3 Y Coordinate	Numeric
27	Point 4 X Coordinate	Numeric
28	Point 4 Y Coordinate	Numeric
29	Point 5 X Coordinate	Numeric
30	Point 5 Y Coordinate	Numeric
31	Point 6 X Coordinate	Numeric
32	Point 6 Y Coordinate	Numeric
33	Point 7 X Coordinate	Numeric
34	Point 7 Y Coordinate	Numeric
35	Point 8 X Coordinate	Numeric
36	Point 8 Y Coordinate	Numeric
37	Point 9 X Coordinate	Numeric
38	Point 9 Y Coordinate	Numeric
39	Point 10 X Coordinate	Numeric
40	Point 10 Y Coordinate	Numeric

41	Point 11 X Coordinate	Numeric
42	Point 11 Y Coordinate	Numeric
43	Point 12 X Coordinate	Numeric
44	Point 12 Y Coordinate	Numeric
45	Point 13 X Coordinate	Numeric
46	Point 13 Y Coordinate	Numeric
47	Point 14 X Coordinate	Numeric
48	Point 14 Y Coordinate	Numeric
49	Point 15 X Coordinate	Numeric
50	Point 15 Y Coordinate	Numeric

DATA BASE: TRAN_NEW_OBSTACLE

TYPE: DELIMITED ASCII

Description
New obstacle recorded transactions.

Field #	Field Name	Type
1	Participant (G2, G3, G4, EX)	Character
2	Message Date (CCyymmdd)	Character
3	Message Time (hhmmss)	Numeric
4	Window Type	Character
5	Window Stack Index	Numeric
6	Scenario Date (CCyymmdd)	Character
7	Scenario Time (hhmm)	Numeric
8	OPLAN Number	Numeric
9	Obstacle Number	Numeric
10	Color (SDB_SIDE_TYPE)	Character
11	Obstacle Type	Character
12	Obstacle Status (SDB_OBSTACLE_STATUS)	Character
13	Effective Date (CCyymmdd)	Character

14	Effective Time (hhmm)	Numeric
15	Center X Coordinate	Numeric
16	Center Y Coordinate	Numeric
17	Frontage	Numeric
18	Depth	Numeric
19	Orientation	Numeric
20	Lane or Gap flag (T=True, F=False)	Character
21	Echelon	Character

DATA BASE: TRAN_OBSTACLE_DEL

TYPE: DELIMITED ASCII

Description
Obstacle delete recorded transactions.

Field #	Field Name	Type
1	Participant (G2, G3, G4, EX)	Character
2	Message Date (CCyymmdd)	Character
3	Message Time (hhmmss)	Numeric
4	Window Type	Character
5	Window Stack Index	Numeric
6	Scenario Date (CCyymmdd)	Character
7	Scenario Time (hhmm)	Numeric
8	OPLAN Number	Numeric
9	Obstacle Number	Numeric

DATA BASE: TRAN_OBSTACLE_EFF_TIME

TYPE: DELIMITED ASCII

Description
Obstacle effective time update recorded transactions.

Field #	Field Name	Type
1	Participant (G2, G3, G4, EX)	Character
2	Message Date (CCyymmdd)	Character
3	Message Time (hhmmss)	Numeric
4	Window Type	Character
5	Window Stack Index	Numeric
6	Scenario Date (CCyymmdd)	Character
7	Scenario Time (hhmm)	Numeric
8	OPLAN Number	Numeric
9	Obstacle Number	Numeric
10	Effective Date (CCyymmdd)	Character
11	Effective Time (hhmm)	Character

DATA BASE: TRAN_OBSTACLE_LOC

TYPE: DELIMITED ASCII

Description
Obstacle location update recorded transactions.

Field #	Field Name	Type
1	Participant (G2, G3, G4, EX)	Character
2	Message Date (CCyymmdd)	Character
3	Message Time (hhmmss)	Numeric
4	Window Type	Character
5	Window Stack Index	Numeric
6	Scenario Date (CCyymmdd)	Character
7	Scenario Time (hhmm)	Numeric
8	OPLAN Number	Numeric

9	Obstacle Number	Numeric
10	Center X Coordinate	Numeric
11	Center Y Coordinate	Numeric

DATA BASE: TRAN_OBSTACLE_STATUS

TYPE: DELIMITED ASCII

Description
Obstacle status update recorded transactions.

Field #	Field Name	Type
1	Participant (G2, G3, G4, EX)	Character
2	Message Date (CCyymmdd)	Character
3	Message Time (hhmmss)	Numeric
4	Window Type	Character
5	Window Stack Index	Numeric
6	Scenario Date (CCyymmdd)	Character
7	Scenario Time (hhmm)	Numeric
8	OPLAN Number	Numeric
9	Obstacle Number	Numeric
10	Obstacle Status (SDB_OBSTACLE_STATUS)	Character

DATA BASE: TRAN_OPFOR_REINFORCE

TYPE: DELIMITED ASCII

Description
OPFOR unit reinforcing time update recorded transactions.

Field #	Field Name	Type
1	Participant (G2, G3, G4, EX)	Character
2	Message Date (CCyymmdd)	Character
3	Message Time (hhmmss)	Numeric
4	Window Type	Character

5	Window Stack Index	Numeric
6	Scenario Date (CCyymmdd)	Character
7	Scenario Time (hhmm)	Numeric
8	OPLAN Number	Numeric
9	Force (SDB_SIDE_TYPE)	Character
10	Unit Number	Numeric
11	Reinforcing Hours	Numeric

DATA BASE: TRAN_OPFOR_STRENGTH

TYPE: DELIMITED ASCII

Description
OPFOR unit strength update recorded transactions.

Field #	Field Name	Type
1	Participant (G2, G3, G4, EX)	Character
2	Message Date (CCyymmdd)	Character
3	Message Time (hhmmss)	Numeric
4	Window Type	Character
5	Window Stack Index	Numeric
6	Scenario Date (CCyymmdd)	Character
7	Scenario Time (hhmm)	Numeric
8	OPLAN Number	Numeric
9	Force (SDB_SIDE_TYPE)	Character
10	Unit Number	Numeric
11	Strength Percent	Numeric

DATA BASE: TRAN_OPFOR_TASK_ORG

TYPE: DELIMITED ASCII

Description
OPFOR task organization update recorded transactions.

Field #	Field Name	Type
1	Participant (G2, G3, G4, EX)	Character
2	Message Date (CCyymmdd)	Character
3	Message Time (hhmmss)	Numeric
4	Window Type	Character
5	Window Stack Index	Numeric
6	Scenario Date (CCyymmdd)	Character
7	Scenario Time (hhmm)	Numeric
8	OPLAN Number	Numeric
9	Force (SDB_SIDE_TYPE)	Character
10	Unit Number	Numeric
11	Higher Echelon Unit Number	Numeric

DATA BASE: TRAN_PERSONNEL

TYPE: DELIMITED ASCII

Description
Unit personnel update recorded transactions.

Field #	Field Name	Type
1	Participant (G2, G3, G4, EX)	Character
2	Message Date (CCyymmdd)	Character
3	Message Time (hhmmss)	Numeric
4	Window Type	Character
5	Window Stack Index	Numeric
6	Scenario Date (CCyymmdd)	Character
7	Scenario Time (hhmm)	Numeric
8	OPLAN Number	Numeric
9	Force (SDB_SIDE_TYPE)	Character
10	Unit Number	Numeric
11	officer strength Change	Numeric

12	Enlisted Strength Change	Numeric

DATA BASE: TRAN_REF_REQUEST

TYPE: DELIMITED ASCII

Description
Request for reference product recorded transaction.

Field #	Field Name	Type
1	Participant (G2, G3, G4, EX)	Character
2	Message Date (CCyymmdd)	Character
3	Message Time (hhmmss)	Numeric
4	Window Type	Character
5	Window Stack Index	Numeric
6	Reference Product Number	Numeric

DATA BASE: TRAN_REF_WINDOW

TYPE: DELIMITED ASCII

Description
View reference window manipulation recorded transactions.

Field #	Field Name	Type
1	Participant (G2, G3, G4, EX)	Character
2	Message Date (CCyymmdd)	Character
3	Message Time (hhmmss)	Numeric
4	Window Type	Character
5	Window Stack Index	Numeric
6	Window Action (S=Stop, T=Close Socket, N=Connect, O=Open, C=Close)	Character

DATA BASE: TRAN_SITUATION_REQUEST

TYPE: DELIMITED ASCII

Description
Request for situation data recorded transactions.

Field #	Field Name	Type
1	Participant (G2, G3, G4, EX)	Character
2	Message Date (CCyymmdd)	Character
3	Message Time (hhmmss)	Numeric
4	Window Type	Character
5	Window Stack Index	Numeric
6	Message Type Requested (MSG_MESSAGES)	Character

DATA BASE: TRAN_SITUATION_WINDOW

TYPE: DELIMITED ASCII

Description
Window manipulation recorded transactions.

Field #	Field Name	Type
1	Participant (G2, G3, G4, EX)	Character
2	Message Date (CCyymmdd)	Character
3	Message Time (hhmmss)	Numeric
4	Window Type	Character
5	Window Stack Index	Numeric
6	Window Action (S=Stop, T=Close Socket, N=Connect, O=Open, C=Close)	Character

DATA BASE: TRAN_UNIT_MISSION

TYPE:

Description
Unit mission update recorded transactions.

Field #	Field Name	Type
1	Participant (G2, G3, G4, EX)	character
2	Message Date (CCyymmdd)	Charact
3	Message Time (hhmmss)	Numeric
4	Window Type	character

5	Window Stack Index	Numeric
6	Scenario Date (CCyymmdd)	Character
7	Scenario Time (hhmm)	Numeric
8	OPLAN Number	Numeric
9	Force (SDB_SIDE_TYPE)	Character
10	Unit Number	Numeric
11	Unit Mission (SDB_FORCE_MISSION)	Character

DATA BASE: TRAN_UNIT_LOCATION

TYPE: DELIMITED ASCII

Description
Unit location update recorded transactions.

Field #	Field Name	Type
1	Participant (G2, G3, G4, EX)	Character
2	Message Date (CCyymmdd)	Character
3	Message Time (hhmmss)	Numeric
4	Window Type	Character
5	Window Stack Index	Numeric
6	Scenario Date (CCyymmdd)	Character
7	Scenario Time (hhmm)	Numeric
8	OPLAN Number	Numeric
9	Force (SDB_SIDE_TYPE)	Character
10	Unit Number	Numeric
11	X Coordinate	Numeric
12	Y Coordinate	Numeric

DATA BASE: VEGETATION_1TO160

TYPE: BINARY

Description
Vegetation image file for the 1:160,000 map scale.

This vegetation data base consists of byte values representing the color lookup table value to use to represent the vegetation type. The data is organized in column/row order (columns within rows) from northwest to southeast with 40 columns and 25 rows. Each record is organized in column/row order from northwest to southeast with 64 columns and 80 rows (5120 bytes).

DATA BASE: VEGETATION_1TO400

TYPE: BINARY

Description
Vegetation image file for the 1:400,000 map scale.

This vegetation data base consists of byte values representing the color lookup table value to use to represent the vegetation type. The data is organized in column/row order (columns within rows) from northwest to southeast with 16 columns and 10 rows. Each record is organized in column/row order from northwest to southeast with 64 columns and 80 rows (5120 bytes).

DATA BASE: VEGETATION_1TO80

TYPE: BINARY

Description
Vegetation image file for the 1:80,000 map scale.

This vegetation data base consists of byte values representing the color lookup table value to use to represent the vegetation type. The data is organized in column/row order (columns within rows) from northwest to southeast with 79 columns and 49 rows. Each record is organized in column/row order from northwest to southeast with 64 columns and 80 rows (5120 bytes).

DATA BASE: VEGETATION_1TO800

TYPE: BINARY

Description
Vegetation image file for the 1:800,000 map scale.

This vegetation data base consists of byte values representing the color lookup table value to use to represent the vegetation type. The data is organized in column/row order (columns within rows) from northwest to southeast with 8 columns and 6 rows. Each record is organized in column/row order from northwest to southeast with 64 columns and 80 rows (5120 bytes).

DATA BASE: VEGETATION_DESC_1TO160

TYPE: VARIABLE ASCII

Description
Description of the 1:160,000 vegetation image file.

Column	Field Name	Type	Width	Dec
Record 1 (Image File)				
1	Vegetation image file name for this map scale	Character	60	
Record 2 (Data Base Size)				
1	Number of vegetation image records in the X direction for this map scale	Numeric	5	0
6	Number of vegetation image records in the Y direction for this map scale	Numeric	5	0
Record 3 (Record Size)				
1	Number of vegetation image points in a record in the X direction	Numeric	5	0
6	Number of vegetation image points in a record in the Y direction	Numeric	5	0
Record 4 (Data Base Point Size)				
1	Number of vegetation image points in the data base in the X direction	Numeric	6	0
7	Number of vegetation image points in the data base in the Y direction	Numeric	6	0
14	Number of meters per pixel for this map scale	Numeric	7	3
Record 5 (Map Origin)				
4	Number of meters in the X direction from MA000000 to the northwest corner of the vegetation image	Numeric	7	0
14	Number of meters in the Y direction from MA000000 to the northwest corner of the vegetation image	Numeric	7	0
Record 6 (Grid Interval)				
1	Grid interval for this map scale (in meters)	Numeric	5	0

DATA BASE: VEGETATION_DESC_1TO400

TYPE: VARIABLE ASCII

Description

Description of the 1:400,000 vegetation image file.

Column	Field Name	Type	Width	Dec
Record 1 (Image File)				
1	Vegetation image file name for this map scale	Character	60	
Record 2 (Data Base Size)				
1	Number of vegetation image records in the X direction for this map scale	Numeric	5	0
6	Number of vegetation image records in the Y direction for this map scale	Numeric	5	0
Record 3 (Record Size)				
1	Number of vegetation image points in a record in the X direction	Numeric	5	0
6	Number of vegetation image points in a record in the Y direction	Numeric	5	0
Record 4 (Data Base Point Size)				
1	Number of vegetation image points in the data base in the X direction	Numeric	6	0
7	Number of vegetation image points in the data base in the Y direction	Numeric	6	0
14	Number of meters per pixel for this map scale	Numeric	7	3
Record 5 (Map Origin)				
4	Number of meters in the X direction from MA000000 to the northwest corner of the vegetation image	Numeric	7	0
14	Number of meters in the Y direction from MA000000 to the northwest corner of the vegetation image	Numeric	7	0
Record 6 (Grid Interval)				
1	Grid interval for this map scale (in meters)	Numeric	5	0

DATA BASE: VEGETATION_DESC_1TO80

TYPE: VARIABLE ASCII

Description
Description of the 1:80,000 vegetation image file.

Column	Field Name	Type	Width	Dec
Record 1 (Image File)				
1	Vegetation image file name for this map scale	Character	60	
Record 2 (Data Base Size)				
1	Number of vegetation image records in the X direction for this map scale	Numeric	5	0
6	Number of vegetation image records in the Y direction for this map scale	Numeric	5	0
Record 3 (Record Size)				
1	Number of vegetation image points in a record in the X direction	Numeric	5	0
6	Number of vegetation image points in a record in the Y direction	Numeric	5	0
Record 4 (Data Base Point Size)				
1	Number of vegetation image points in the data base in the X direction	Numeric	6	0
7	Number of vegetation image points in the data base in the Y direction	Numeric	6	0
14	Number of meters per pixel for this map scale	Numeric	7	3
Record 5 (Map Origin)				
4	Number of meters in the X direction from MA000000 to the northwest corner of the vegetation image	Numeric	7	0
14	Number of meters in the Y direction from MA000000 to the northwest corner of the vegetation image	Numeric	7	0
Record 6 (Grid Interval)				
1	Grid interval for this map scale (in meters)	Numeric	5	0

DATA BASE: VEGETATION_DESC_1TO800

TYPE: VARIABLE ASCII

Description

Description of the 1:800,000 vegetation image file.

Column	Field Name	Type	Width	Dec
Record 1 (Image File)				
1	Vegetation image file name for this map scale	Character	60	
Record 2 (Data Base Size)				
1	Number of vegetation image records in the X direction for this map scale	Numeric	5	0
6	Number of vegetation image records in the Y direction for this map scale	Numeric	5	0
Record 3 (Record Size)				
1	Number of vegetation image points in a record in the X direction	Numeric	5	0
6	Number of vegetation image points in a record in the Y direction	Numeric	5	0
Record 4 (Data Base Point Size)				
1	Number of vegetation image points in the data base in the X direction	Numeric	6	0
7	Number of vegetation image points in the data base in the Y direction	Numeric	6	0
14	Number of meters per pixel for this map scale	Numeric	7	3
Record 5 (Map Origin)				
4	Number of meters in the X direction from MA000000 to the northwest corner of the vegetation image	Numeric	7	0
14	Number of meters in the Y direction from MA000000 to the northwest corner of the vegetation image	Numeric	7	0
Record 6 (Grid Interval)				
1	Grid interval for this map scale (in meters)	Numeric	5	0

APPENDIX E - EDDIC dBASE DATA BASES

This appendix describes the EDDIC PC-based data bases. Table E-1 lists the PC-based data base. This appendix also includes the record layouts for the data bases.

Table E-1. EDDIC Sun-Based Data Bases

Data Base Name	Description
BLUEFOR_AMMO_SOURCE	Initial Ammunition levels for BLUEFOR units. (ASCII format).
BLUEFOR_AMMO_TRACK	List of ammunition types to include in the graphical unit status report (ASCII format).
BLUEFOR_ASSET_UNIT	List of BLUEFOR units that have initial levels of assets assigned to them (ASCII format).
BLUEFOR_AUTH_AMMO_INDEX	Index file for the BLUEFOR authorized ammunition levels data base (Ada format).
BLUEFOR_AUTH_AMMO	BLUEFOR authorized ammunition levels (Ada format).
BLUEFOR_AUTH_EQUIP_INDEX	Index file for the BLUEFOR authorized equipment levels data base (Ada format).
BLUEFOR_AUTH_EQUIP	BLUEFOR authorized equipment levels (Ada format).
BLUEFOR_CM_EDIT_MENU	Description of the walking menu to display when a BLUEFOR control measure is selected on the tactical map in a window with edit capability (ASCII format).
BLUEFOR_CM_VIEW_MENU	Description of the walking menu to display when a BLUEFOR control measure is selected on the tactical map in a window with view only capability (ASCII format).
BLUEFOR_CURR_AMMO	BLUEFOR current ammunition levels (Ada format).
BLUEFOR_CURR_AMMO_INDEX	Index file for the BLUEFOR current ammunition levels data base (Ada format).
BLUEFOR_CURR_EQUIP_INDEX	Index file for the BLUEFOR current equipment levels data base (Ada format).
BLUEFOR_CURR_EQUIP	BLUEFOR current equipment levels (Ada format).

BLUEFOR_EQUIP_SOURCE	Initial equipment levels for BLUEFOR units (ASCII format).
BLUEFOR_EQUIP_TRACK	List of BLUEFOR equipment types to include in the graphical unit status report (ASCII format).
BLUEFOR_FUEL	BLUEFOR authorized and current fuel levels (Ada format).
BLUEFOR_FUEL_INDEX	Index file for the BLUEFOR fuel level data base (Ada format).
BLUEFOR_FUEL_SOURCE	Initial fuel levels for BLUEFOR units (ASCII format).
BLUEFOR_OBS_EDIT_MENU	Description of the walking menu to display when a BLUEFOR obstacle is selected on the tactical map in a window with view only capability (ASCII format).
BLUEFOR_OBS_VIEW_MENU	Description of the walking menu to display when a BLUEFOR obstacle is selected on the tactical map in a window with edit capability (ASCII format).
BLUEFOR_ORGANIC_TASK_ORG	Organic task organization for the BLUEFOR units (ASCII format).
BLUEFOR_PERSONNEL	BLUEFOR authorized and current personnel levels (Ada format).
BLUEFOR_PERSONNEL_INDEX	Index file for the BLUEFOR personnel level data base (Ada format).
BLUEFOR_PERSONNEL_SOURCE	Initial personnel levels for BLUEFOR units (ASCII format).
BLUEFOR_TASK_ORG_SOURCE	Initial task organization and status for the BLUEFOR units (ASCII format).
BLUEFOR_UNIT_CONVERT	Data base to convert BLUEFOR unit names to unit numbers (Ada format).
BLUEFOR_UNIT_LOC_INDEX	Index file for the BLUEFOR unit location data base (Ada format).
BLUEFOR_UNIT_LOC_SOURCE	Initial unit locations for the BLUEFOR units (ASCII format).
BLUEFOR_UNIT_LOC	BLUEFOR unit location data base (Ada format).

BLUEFOR_UNIT_EDIT_MENU	Description of the walking menu to display when a BLUEFOR unit is selected on a tactical map in a window with edit capability (ASCII format).
BLUEFOR_UNIT_NAME	List of the BLUEFOR unit names. This file is used to assign names to the unit transactions in the situation recorded data (ASCII format).
BLUEFOR_UNIT_STATUS	BLUEFOR unit status (Ada format).
BLUEFOR_UNIT_STATUS_INDEX	Index file for the BLUEFOR unit status data base (Ada format).
BLUEFOR_UNIT_VIEW_MENU	Description of the walking menu to display when a BLUEFOR unit is selected on the tactical map in a window with view only capability (ASCII format).
C2_PRODUCT	Command and control product data base. Includes the products in the view situation, build and view message windows (Ada format).
C2_PRODUCT_DESC	Command and control product description data base. This data base indicates which record from the C2_PRODUCT data base to use for a product (Ada format).
C2_PRODUCT_HEADER	Command and control report headers. The report headers only applies to those products in the view situation window (Ada format).
C2_PRODUCT_NAME	List of the command and control product names. This file is used to assign names to the command and control transactions in the C2 product recorded data (ASCII format).
C2_PRODUCT_RECORD	Command and control data recording transactions (Ada format).
C2_PRODUCT_SOURCE	Description of the command and control products to include in the view situation and build windows (ASCII format).
CNTRL_MSR_POINT	Point control measures (Ada format).
CNTRL_MSR_POINT_INDEX	Index file for the point control measure data base (Ada format).
CNTRL_MSR_POINT_NAME	List of the point control measure names. This file is used to assign names to the point control measure transactions in the situation recorded data (ASCII format).

CONTOUR_1TO160	Map contour image file for the 1:160,000 map scale (Binary format).
CONTOUR_1TO400	Map contour image file for the 1:400,000 map scale (Binary format).
CONTOUR_1TO80	Map contour image file for the 1:80,000 map scale (Binary format).
CONTOUR_1TO800	Map contour image file for the 1:800,000 map scale (Binary format).
CONTOUR_DESC	Description of the contour files to include in the tactical map system (ASCII format).
CONTOUR_DESC_1TO160	Description of the 1:160,000 contour image file (ASCII format).
CONTOUR_DESC_1TO400	Description of the 1:400,000 contour image file (ASCII format).
CONTOUR_DESC_1TO80	Description of the 1:80,000 contour image file (ASCII format).
CONTOUR_DESC_1TO800	Description of the 1:800,000 contour image file (ASCII format).
CONTROL_MEASURE	Control measures (Ada format).
CONTROL_MEASURE_NAME	List of the control measure names. This file is used to assign names to the control measure transactions in the situation recorded data (ASCII format).
CONTROL_MEASURE_SOURCE	Initial control measures (ASCII format).
CONTROL_MEASURE_INDEX	Index file for the control measure data base (Ada format).
ELEVATION_1TO400	Elevation file for the 1:400,000 map scale (Binary format).
ELEVATION_DESC_1TO400	Description of the 1:400,000 elevation file (ASCII format).
ELEV_BAND_1TO160	Elevation band image file for the 1:160,000 map scale (Binary format).
ELEV_BAND_1TO400	Elevation band image file for the 1:400,000 map scale (Binary format).
ELEV_BAND_1TO80	Elevation band image file for the 1:80,000 map scale (Binary format).
ELEV_BAND_1TO800	Elevation band image file for the 1:800,000 map scale (Binary format).

ELEV_BAND_DESC_1TO160	Description of the 1:160,000 elevation band image file (ASCII format).
ELEV_BAND_DESC_1TO400	Description of the 1:400,000 elevation band image file (ASCII format).
ELEV_BAND_DESC_1TO80	Description of the 1:80,000 elevation band image file (ASCII format).
ELEV_BAND_DESC_1TO800	Description of the 1:800,000 elevation band image file (ASCII format).
EXP_CONTROL_MENU	Description of the experiment control product walking menu. This file is created from the product names in the experiment control source file (ASCII format).
EXP_CONTROL_NAME	List of the experiment control product names. This file is used to assign names to the experiment control transactions in the experiment control recorded data (ASCII format).
EXP_CONTROL_PARTICIPANT	List of participants that the experimenter can send experiment control messages to (Ada format).
EXP_CONTROL_PRODUCT	Experiment control products (Ada format).
EXP_CONTROL_PROD_DESC	Experiment control product description data base. This data base indicates which record from the experiment control data base to use for a product (Ada format).
EXP_CONTROL_RECORD	Experiment control data recording transactions (Ada format).
EXP_CONTROL_SOURCE	Description of the products to include in the experiment control window (ASCII format).
FORM_DESCRIPTION	Description and layout of EDDIC form.
G2_BUILD_MENU	Description of the build product walking menu for the G2 workstation. This file is created from the command and control product source file (ASCII format).
G2_REFERENCE_MENU	Description of the reference product walking menu for the G2 workstation. This file is created from the reference product source file (ASCII format).

G2_VIEW_C2_MENU	Description of the view situation product walking menu for the G2 workstation. This file is created from the command and control product source file (ASCII format).
G3_BUILD_MENU	Description of the build product walking menu for the G3 workstation. This file is created from the command and control product source file (ASCII format).
G3_REFERENCE_MENU	Description of the reference product walking menu for the G3 workstation. This file is created from the reference product source file (ASCII format).
G3_VIEW_C2_MENU	Description of the view situation product walking menu for the G3 workstation. This file is created from the command and control product source file (ASCII format).
G4_BUILD_MENU	Description of the build product walking menu for the G4 workstation. This file is created from the command and control product source file (ASCII format).
G4_REFERENCE_MENU	Description of the reference product walking menu for the G4 workstation. This file is created from the reference product source file (ASCII format).
G4_VIEW_C2_MENU	Description of the view situation product walking menu for the G3 workstation. This file is created from the command and control product source file (ASCII format).
HELP_MENU	Description of the help product walking menu for the G3 workstation. This file is created from the help product source file (ASCII format).
HELP_NAME	List of the help product names. This file is used to assign names to the help transactions in the reference recorded data (ASCII format).
HELP_PROD_DESC	Help product description data base. This data base indicates which record from the help product data base to use for a product (Ada format).
HELP_PRODUCT	Help products (Ada format).
HELP_SOURCE	Description of the products to include in the help window (ASCII format).

ICON_STACK_DB	Icon stack status data base. Indicates which stack positions are used and which ones are free (C format).
LUT_HILITE_DESC	Description of the color lookup table files to use when features are hilighted (ASCII format).
LUT_HILITE_MAP_ON	Color lookup table to use when a map background (elevation band, shaded relief, or vegetation) is displayed and map features are hilighted (ASCII format).
LUT_HILITE_MAP_OFF	Color lookup table to use when a map with a null background is displayed and map features are hilighted (ASCII format).
LUT_OVERLAY	Color lookup table for the overlay planes (ASCII format).
LUT_UNHILITE_DESC	Description of the color lookup table files to use when features are not hilighted (ASCII format).
LUT_UNHILITE_MAP_ON	Color lookup table to use when a map background (elevation band, shaded relief, or vegetation) is displayed and map features are not hilighted (ASCII format).
LUT_UNHILITE_MAP_OFF	Color lookup table to use when a map with a null background is displayed and map features are not hilighted (ASCII format).
MAP_BUILD_MENU	Description of the map options walking menu for the build window (ASCII format).
MAP_DESC	Description of the map image files to include in the tactical map system (ASCII format).
MAP_LEGEND	Description of what to display in the map legend (ASCII format).
MAP_MESSAGE_MENU	Description of the map options walking menu for the view message window (ASCII format).
MAP_VIEW_C2_MENU	description of the map options walking menu for the view situation window (ASCII format).
MESSAGE_LOG	Log of all the messages sent (Ada format).
OBSTACLE	Obstacles (Ada format).
OBSTACLE_INDEX	Index for the obstacle data base (Ada format).

OBSTACLE_SOURCE	Initial obstacles (ASCII format).
OPFOR_AUTH_EQUIP	OPFOR authorized equipment levels (Ada format).
OPFOR_AUTH_EQUIP_INDEX	Index file for the OPFOR authorized equipment levels data base (Ada format).
OPFOR_CM_EDIT_MENU	Description of the walking menu to display when a OPFOR control measure is selected on the tactical map in a window with edit capability (ASCII format).
OPFOR_CM_VIEW_MENU	Description of the walking menu to display when a OPFOR control measure is selected on the tactical map in a window with view only capability (ASCII format).
OPFOR_CURR_EQUIP_INDEX	Index file for the OPFOR current equipment levels data base (Ada format).
OPFOR_CURR_EQUIP	OPFOR current equipment levels (Ada format).
OPFOR_EQUIP_NAME	List of the OPFOR equipment names. This file is used to assign names to the equipment types in the situation data base (ASCII format).
OPFOR_EQUIP_SOURCE	Initial equipment levels for OPFOR units (ASCII format).
OPFOR_OBS_EDIT_MENU	Description of the walking menu to display when a OPFOR obstacle is selected on the tactical map in a window with view only capability (ASCII format).
OPFOR_OBS_VIEW_MENU	Description of the walking menu to display when a OPFOR obstacle is selected on the tactical map in a window with edit capability (ASCII format).
OPFOR_ORGANIC_TASK_ORG	Organic task organization for the OPFOR units (ASCII format).
OPFOR_REINFORCE_TIME	Initial reinforcing times for OPFOR units (ASCII format).
OPFOR_TASK_ORG_SOURCE	Initial task organization for the OPFOR units (ASCII format).
OPFOR_UNIT_CONVERT	Data base to convert OPFOR unit names to unit numbers (Ada format).

OPFOR_UNIT_EDIT_MENU	Description of the walking menu to display when a OPFOR unit is selected on a tactical map in a window with edit capability (ASCII format).
OPFOR_UNIT_LOC	OPFOR unit location data base (Ada format).
OPFOR_UNIT_LOC_INDEX	Index file for the OPFOR unit location data base (Ada format).
OPFOR_UNIT_LOC_SOURCE	Initial unit locations for the OPFOR units (ASCII format).
OPFOR_UNIT_NAME	List of the OPFOR unit names. This file is used to assign names to the unit transactions in the situation recorded data (ASCII format).
OPFOR_UNIT_STATUS_INDEX	Index file for the OPFOR unit status data base (Ada format).
OPFOR_UNIT_STATUS	OPFOR unit status (Ada format).
OPFOR_UNIT_STATUS_SOURCE	Initial status of the OPFOR units (ASCII format).
OPFOR_UNIT_VIEW_MENU	Description of the walking menu to display when a OPFOR unit is selected on the tactical map in a window with view only capability (ASCII format).
OPLAN_LIST	List of existing Operational plans in the system (Ada format).
OPLAN_LIST_SOURCE	Operational plans to initially have in the system (ASCII format).
PRODUCT_HARDCOPY	ASCII output file of the products printed by CDB_HARDCOPY.
REFERENCE_HEADER	Reference report headers (Ada format).
REFERENCE_NAME	List of the reference product names. This file is used to assign names to the reference transactions in the reference recorded data (ASCII format).
REFERENCE_PROD_DESC	Reference product description data base. This data base indicates which records from the reference product data base to use for a product (Ada format).
REFERENCE_PRODUCT	Reference product data base (Ada format).

REFERENCE_RECORD	Reference data recording transactions (Ada format).
REFERENCE_SOURCE	Description of the reference products to include in the view reference window (ASCII format).
ROOT_WINDOW_MENU	Description of the walking menu to display in the root window. The root window is any part of the screen where a window or button is not displayed (ASCII format).
SCREEN_DUMP_IMAGE	Bitmap image of a screen of a Sun workstation (Bitmap format).
SEND_PARTICIPANT_SOURCE	List of the participants that messages can be sent to (ASCII format).
SHAD_RELF_1TO160	Shaded relief image file for the 1:160,000 map scale (Binary format).
SHAD_RELF_1TO400	Shaded relief image file for the 1:400,000 map scale (Binary format).
SHAD_RELF_1TO80	Shaded relief image file for the 1:80,000 map scale (Binary format).
SHAD_RELF_1TO800	Shaded relief image file for the 1:800,000 map scale (Binary format).
SHAD_RELF_DESC_1TO160	Description of the 1:160,000 shaded relief image file (ASCII format).
SHAD_RELF_DESC_1TO400	Description of the 1:400,000 shaded relief image file (ASCII format).
SHAD_RELF_DESC_1TO80	Description of the 1:80,000 shaded relief image file (ASCII format).
SHAD_RELF_DESC_1TO800	Description of the 1:800,000 shaded relief image file (ASCII format).
SITUATION_RECORD	Situation data recording transactions (Ada format).
TASK_ORG_TOOL_MENU	Description of the walking menu to display as a popup menu for the task organization tool (ASCII format).
TASK_ORG_TOP_UNIT_MENU	Description of the walking menu to display when the top unit button is selected in the task organization tool (ASCII format).

TASK_ORG_UNIT_MENU	Description of the walking menu to display as a popup menu when a unit is selected in the task organization tool (ASCII format).
TASK_ORG_UNIT_TYPE_MENU	Description of the multiple selection menu to display when the unit type button is selected in the task organization tool (ASCII format).
TOOL_MENU	Description of the walking menu defining the tools available in the tool window (ASCII format).
TRAN_ACTIVITY	Unit activity update recorded transactions (ASCII format).
TRAN_AMMUNITION	Unit ammunition update recorded transactions (ASCII format).
TRAN_BLUEFOR_TASK_ORG	BLUEFOR task organization update recorded transactions (ASCII format).
TRAN_C2_REQUEST	Request for command and control product recorded transactions (ASCII format).
TRAN_C2_WINDOW	View situation, build, and view message window manipulation recorded transactions (ASCII format).
TRAN_CNTRL_MSR_DEL	Control measure delete recorded transactions (ASCII format).
TRAN_CNTRL_MSR_EFF_TIME	Control measure effective time update recorded transactions (ASCII format).
TRAN_CNTRL_MSR_LOC	Control measure location update recorded transactions (ASCII format).
TRAN_CNTRL_MSR_STAT	Control measure status update recorded transactions (ASCII format).
TRAN_CONTROL_REQUEST	Request for experiment control product recorded transactions (ASCII format).
TRAN_CONTROL_WINDOW	Tool and experiment control window manipulation recorded transactions (ASCII format).
TRAN_EQUIPMENT	Unit equipment update recorded transactions (ASCII format).
TRAN_FUEL	Unit fuel update recorded transactions (ASCII format).
TRAN_LOOKUP_TABLE	Color lookup table update recorded transactions (ASCII format).

TRAN_MAP	Tactical map control recorded transactions (ASCII format).
TRAN_NEW_C2	New command and control product recorded transactions (ASCII format).
TRAN_NEW_CNTRL_MSR	New control measure recorded transactions (ASCII format).
TRAN_NEW_OBSTACLE	New obstacle recorded transactions (ASCII format).
TRAN_OBSTACLE_DEL	Obstacle delete recorded transactions (ASCII format).
TRAN_OBSTACLE_EFF_TIME	Obstacle effective time update recorded transactions (ASCII format).
TRAN_OBSTACLE_LOC	Obstacle location update recorded transactions (ASCII format).
TRAN_OBSTACLE_STATUS	Obstacle status update recorded transactions (ASCII format).
TRAN_OPFOR_REINFORCE	OPFOR unit reinforcing time update recorded transactions (ASCII format).
TRAN_OPFOR_STRENGTH	OPFOR unit strength update recorded transactions (ASCII format).
TRAN_OPFOR_TASK_ORG	OPFOR task organization update recorded transactions (ASCII format).
TRAN_PERSONNEL	Unit personnel update recorded transactions (ASCII format).
TRAN_REF_REQUEST	Request for reference product recorded transaction (ASCII format).
TRAN_REF_WINDOW	View reference window manipulation recorded transactions (ASCII format).
TRAN_SITUATION_REQUEST	Request for situation data recorded transactions (ASCII format).
TRAN_SITUATION_WINDOW	Window manipulation recorded transactions (ASCII format).
TRAN_UNIT_MISSION	Unit mission update recorded transactions (ASCII format).
TRAN_UNIT_LOCATION	Unit location update recorded transactions (ASCII format).
VEGETATION_1TO160	Vegetation image file for the 1:160,000 map scale (Binary format).

VEGETATION_1TO400	Vegetation image file for the 1:400,000 map scale (Binary format).
VEGETATION_1TO80	Vegetation image file for the 1:80,000 map scale (Binary format).
VEGETATION_1TO800	Vegetation image file for the 1:800,000 map scale (Binary format).
VEGETATION_DESC_1TO160	Description of the 1:160,000 vegetation image file (ASCII format).
VEGETATION_DESC_1TO400	Description of the 1:400,000 vegetation image file (ASCII format).
VEGETATION_DESC_1TO80	Description of the 1:80,000 vegetation image file (ASCII format).
VEGETATION_DESC_1TO800	Description of the 1:800,000 vegetation image file (ASCII format).

Table E-2. EDDIC PC-Based Data Bases

Data Base Name	Description
BASEUNIT	Authorized assets for the basic BLUEFOR company level unit types. All company level units that are added to the scenario start with the assets that are contained in its base unit.
BATTAL1	Battalion level units in day 1 of the scenario. There is a different version of this data base for each day of the scenario (identified with a 2..n suffix).
BDEDISP	Temporary data base to print the brigade personnel strength report.
BRIGADE1	Brigade level units in day 1 of the scenario. There is a different version of this data base for each day of the scenario (identified with a 2..n suffix).
BUNXREF	Conversion data base for converting BLUEFOR unit IDs to names. This data base is used to assign unit names to recorded data.
C2_RQST	Data recording of the Command and Control (C2) product requests.
CCAB	Summary results of a CCAB exercise.

CMXREF	Conversion data base for converting control measure IDs to names. This data base is used to assign control measure names to recorded data.
CNTLMSR1	Control measures in day 1 of the scenario. A different version of this data base exists for each day of the scenario.
COAATCAN	COAAT canned critical events. These are used to initialize the COAAT data bases for the manual structured condition.
COAATM1	Critical events identified by the participant in module 1 of COAAT.
COAATM2	War-gaming results from module 2 of COAAT.
COAATSC	Scaling factors assigned to the war-gaming categories.
COAATWT	Weights assigned to the war-gaming factors.
COMPANY1	Company level units in day 1 of the scenario. Contains the current asset levels. There is a different version of this data base for each day of the scenario.
CTL_RQST	Data recording of the experiment control product requests.
CTL_XREF	Conversion data base for converting experiment control product *IDs to names*. This data base is used to assign experiment control names to recorded data.
DAY	List of days in the scenario and associated data bases.
DIVISN1	Division level units in day 1 of the scenario. There is a different version of this data base for each day of the scenario.
DUMMY	Dummy data base to assign to the top-level menu in the EDDIC data analysis program.
ED_LUT	Color lookup table updates received from the Sun system for an experiment.
ED_MAP	Digital map interactions received from the Sun system for an experiment.
ED_WIND	EDDIC window interactions received from the Sun system for an experiment.
EDC2RQ	EDDIC C2 product requests received from the Sun system for an experiment.
EDCOTM1	COAAT module 1 data received from the Sun system for an experiment.

EDCOTM2	COAAT module 2 data received from the Sun system for an experiment.
EDCOTSC	COAAT scales received from the Sun system for an experiment.
EDCOTWT	COAAT weights received from the Sun system for an experiment.
EDCTLRQ	Experiment control product requests received from the Sun system for an experiment.
EDNEWC2	New C2 products received from the Sun system for an experiment.
EDREFRQ	Reference product requests received from the Sun system for an experiment.
EDSTACTV	Unit activity updates received from the Sun system for an experiment.
EDSTAMMO	Ammunition level updates received from the Sun system for an experiment.
EDSTBLTO	BLUEFOR task organization updates received from the Sun system for an experiment.
EDSTCMDEL	Control measure delete transactions received from the Sun system for an experiment.
EDSTCMEF	Control measure effective time updates received from the Sun system for an experiment.
EDSTCMLC	Control measure location updates received from the Sun system for an experiment.
EDSTCMST	Control measure status updates received from the Sun system for an experiment.
EDSTEQP	Unit equipment strength updates received from the Sun system for an experiment.
EDSTFUEL	Unit fuel level updates received from the Sun system for an experiment.
EDSTMISS	Unit mission updates received from the Sun system for an experiment.
EDSTNWCM	New control measures received from the Sun system for an experiment.
EDSTNWOB	New obstalces received from the Sun system for an experiment.
EDSTOBDL	Obstacle delete transactions received from the Sun system for an experiment.

EDSTOBEF	Obstacle effective time update received from the Sun system for an experiment.
EDSTOBLC	Obstacle location update received from the Sun system for an experiment.
EDSTOBST	Obstacle status updates received from the Sun system for an experiment.
EDSTOPTO	OPFOR task organization updates received from the Sun system for an experiment.
EDSTPERS	Unit personnel strength updates received from the Sun system for an experiment.
EDSTRENF	OPFOR unit reinforcing time updates received from the Sun system for an experiment.
EDSTRQST	Situation data requests received from the Sun system for an experiment.
EDSTSTNG	OPFOR unit strength update received from the Sun system for an experiment.
EDSTULOC	Unit location update received from the Sun system for an experiment.
EQDISPLA	Temporary data base for displaying unit equipment strength report from the scenario data.
HLP_XREF	Conversion data base for converting help product IDs to names. This data base is used to assign help product names to recorded data.
HMIED	EDDIC Human Machine Interface (HMI) questionnaire results.
HMIEDCT	EDDIC/COAAT HMI questionnaire results.
HST_XREF	Conversion data base for converting reference product IDs to names. This data base is used to assign reference names to recorded data.
LOSSRAT2	Loss rates to use to create day 2 of the scenario from day 1. There are different versions of this data base for creating other days of the scenario.
LUT_CTRL	Color lookup table updates recorded data.
MAP_CTRL	Digital map interaction recorded data.
MISSION	Names to use for main attack, supporting attack, and reserve in the Army forces scoring report.
NEW_C2	New C2 product recorded data.

OPLAN	List of initial OPLAN's in the scenario.
PERCENT	Temporary data base for adjusting unit strength by a percent.
PERDISP	Temporary data base for displaying the personnel strength report from the scenario data base.
PERSTYLE	Personal Style questionnaire data.
RBASEUNI	Authorized assets for the basic OPFOR company level unit types. All company level units that are added to the scenario start with the assets that are contained in its base unit.
RBATTAL1	Battalion level units in the scenario. There is a different version of this data base for each day of the scenario.
RBRIGAD1	Brigade level units in the scenario. There is a different version of this data base for each day of the scenario.
RCOMPNY1	Company level units in the scenario. The assets are assigned to units at this level. There is a different version of this data base for each day of the scenario.
REF_RQST	Reference product request recorded data.
REF_XREF	Conversion data base for converting reference product IDs to names. This data base is used to assign reference names to recorded data.
RUNXREF	Conversion data base for converting OPFOR unit IDs to names. This data base is used to assign unit names to recorded data.
SCCNOP	Experiment scores for the Concept of operations questionnaire.
SCCRTEVT	Experiment scores for identifying critical events.
SCFACTS	Experiment scores for gathering pertinent facts.
SCFORCE	Experiment scores for arraying the forces.
SCJUST	Experiment scores for COA justification.
SCPOWER	Combat power assigned to each unit. This data base is not part of the menu system and must be updated using dBASE (if required).
SITAWARE	Situation awareness questionnaire data.
SITCMDEL	Control measure delete recorded data.

SITNEWCM	New control measure recorded data.
SITRQST	Situation data request recorded data.
SITTASKO	Task organization update recorded data.
SITULOC	Unit location update recorded data.
TASKEVAL	Experiment task evaluation data.
TEAMPRF	Team profile observation data.
TIMELINE	Experiment time line data.
VERTASK	Temporary data base to print the task organization validation report.
WINDOW	EDDIC window interaction recorded data.
WORKASMT	Workload assessment questionnaire data.

The following section describes the record format of the dBASE data bases. These data bases are maintained on a PC and consist mostly of scenario and experiment analysis data.

DATA BASE: BASEUNIT.DBF

Field	Field Name	Type	Width	Dec	Index
1	UNIT_NAME	Character	12		N
2	OFFICER	Numeric	3		N
3	ENLISTED	Numeric	3		N
4	EQ_NAME_1	Character	8		N
5	EQ_QTY_1	Numeric	3		N
6	EQ_NAME_2	Character	8		N
7	EQ_QTY_2	Numeric	3		N
8	EQ_NAME_3	Character	8		N
9	EQ_QTY_3	Numeric	3		N
10	EQ_NAME_4	Character	8		N
11	EQ_QTY_4	Numeric	3		N
12	EQ_NAME_5	Character	8		N
13	EQ_QTY_5	Numeric	3		N
14	EQ_NAME_6	Character	8		N
15	EQ_QTY_6	Numeric	3		N
16	EQ_NAME_7	Character	8		N
17	EQ_QTY_7	Numeric	3		N
18	EQ_NAME_8	Character	8		N
19	EQ_QTY_8	Numeric	3		N
20	EQ_NAME_9	Character	8		N
21	EQ_QTY_9	Numeric	3		N
22	EQ_NAME_10	Character	8		N
23	EQ_QTY_10	Numeric	3		N
24	EQ_NAME_11	Character	8		N
25	EQ_QTY_11	Numeric	3		N
26	EQ_NAME_12	Character	8		N

27	EQ_QTY_12	Numeric	3	N
28	AM_NAME_1	Character	8	N
29	AM_QTY_1	Numeric	6	N
30	AM_NAME_2	Character	8	N
31	AM_QTY_2	Numeric	6	N
32	AM_NAME_3	Character	8	N
33	AM_QTY_3	Numeric	6	N
34	AM_NAME_4	Character	8	N
35	AM_QTY_4	Numeric	6	N
36	AM_NAME_5	Character	8	N
37	AM_QTY_5	Numeric	6	N
38	AM_NAME_6	Character	8	N
39	AM_QTY_6	Numeric	6	N
40	MOGAS	Numeric	5	N
41	DIESEL	Numeric	5	N
42	AVGAS	Numeric	5	N
** Total **			250	

DATA BASE: BATTAL1.DBF

Field	Field Name	Type	Width	Dec	Index
1	BN_NAME	Character	12		N
2	ECHELON	Character	6		N
3	TYPE	Character	6		N
4	BATL_FUNC	Character	6		N
5	ACTIVITY	Character	6		N
6	MISSION	Character	6		N
7	LOCATION	Character	8		N
8	CO_NAME_1	Character	12		N
9	CO_REL_1	Character	6		N
10	CO_NAME_2	Character	12		N
11	CO_REL_2	Character	6		N
12	CO_NAME_3	Character	12		N
13	CO_REL_3	Character	6		N
14	CO_NAME_4	Character	12		N
15	CO_REL_4	Character	6		N
16	CO_NAME_5	Character	12		N
17	CO_REL_5	Character	6		N
18	CO_NAME_6	Character	12		N
19	CO_REL_6	Character	6		N
20	CO_NAME_7	Character	12		N
21	CO_REL_7	Character	6		N
22	CO_NAME_8	Character	12		N
23	CO_REL_8	Character	6		N
** Total **			195		

DATA BASE: BDEDISP.DBF

Field	Field Name	Type	Width	Dec	Index
1	UNIT_NAME	Character	12		N
2	OFF_LOSS	Numeric	4		N
3	ENL_LOSS	Numeric	4		N
4	OFF_GAIN	Numeric	4		N
5	ENL_GAIN	Numeric	4		N

6	OFF_AUTH	Numeric	7	N
7	ENL_AUTH	Numeric	7	N
8	OFF_CURR	Numeric	7	N
9	ENL_CURR	Numeric	7	N
** Total **			57	

DATA BASE: BRIGADE1.DBF

Field	Field Name	Type	Width	Dec	Index
1	BDE_NAME	Character	12		N
2	ECHELON	Character	6		N
3	TYPE	Character	6		N
4	BATL_FUNC	Character	6		N
5	ACTIVITY	Character	6		N
6	MISSION	Character	6		N
7	LOCATION	Character	8		N
8	BN_NAME_1	Character	12		N
9	BN_REL_1	Character	6		N
10	BN_NAME_2	Character	12		N
11	BN_REL_2	Character	6		N
12	BN_NAME_3	Character	12		N
13	BN_REL_3	Character	6		N
14	BN_NAME_4	Character	12		N
15	BN_REL_4	Character	6		N
16	BN_NAME_5	Character	12		N
17	BN_REL_5	Character	6		N
18	BN_NAME_6	Character	12		N
19	BN_REL_6	Character	6		N
20	BN_NAME_7	Character	12		N
21	BN_REL_7	Character	6		N
22	BN_NAME_8	Character	12		N
23	BN_REL_8	Character	6		N
24	BN_NAME_9	Character	12		N
25	BN_REL_9	Character	6		N
26	BN_NAME_10	Character	12		N
27	BN_REL_10	Character	6		N
28	BN_NAME_11	Character	12		N
29	BN_REL_11	Character	6		N
30	BN_NAME_12	Character	12		N
31	BN_REL_12	Character	6		N
** Total **			267		

DATA BASE: BUNXREF.DBF

Field	Field Name	Type	Width	Dec	Index
1	UNIT_ID	Numeric	3		Y
2	NAME	Character	15		N
** Total **			19		

DATA BASE: C2_RQST.DBF

Field	Field Name	Type	Width	Dec	Index
1	SEQ_NO	Character	5		N

2	DATE	Date	8	N
3	TIME	Character	6	N
4	WINDOW	Character	7	N
5	STACK	Numeric	1	N
6	FUNC_AREA	Character	20	N
7	DATA_CAT	Character	20	N
8	DATA_ELE	Character	20	N
9	DATA_SUB	Character	20	N
10	DATA_LVL	Character	1	N
** Total **			109	

DATA BASE: CCAB.DBF

Field	Field Name	Type	Width	Dec	Index
1	SEQ_NO	Character	5		Y
2	CABTP	Numeric	3		N
3	CABFD	Numeric	3		N
4	CABWA	Numeric	3		N
5	CABLR	Numeric	3		N
6	CABMN	Numeric	3		N
7	CABNW	Numeric	3		N
8	CABIP	Numeric	3		N
9	CABRP	Numeric	3		N
10	CABMI	Numeric	3		N
11	MEAN	Numeric	3		N
12	STD	Numeric	3		N
** Total **			39		

DATA BASE: CMXREF.DBF

Field	Field Name	Type	Width	Dec	Index
1	CM_ID	Numeric	3		Y
2	NAME	Character	12		N
** Total **			16		

DATA BASE: CNTLMSR1.DBF

Field	Field Name	Type	Width	Dec	Index
1	NAME	Character	12		N
2	TYPE	Character	6		N
3	ECHELON	Character	6		N
4	SIDE	Character	4		N
5	SCALE1	Logical	1		N
6	SCALE2	Logical	1		N
7	SCALE3	Logical	1		N
8	SCALE4	Logical	1		N
9	SCALE5	Logical	1		N
10	LOC1	Character	8		N
11	LOC2	Character	8		N
12	LOC3	Character	8		N
13	LOC4	Character	8		N
14	LOC5	Character	8		N
15	LOC6	Character	8		N

16	LOC7	Character	8		N
17	LOC8	Character	8		N
18	LOC9	Character	8		N
19	LOC10	Character	8		N
20	LOC11	Character	8		N
21	LOC12	Character	8		N
22	LOC13	Character	8		N
23	LOC14	Character	8		N
24	LOC15	Character	8		N
** Total **			154		

DATA BASE: COAATCAN.DBF

Field	Field Name	Type	Width	Dec	Index
1	BALANCE	Character	1		N
2	CE	Character	7		N
3	COA	Numeric	1		N
4	AVENUE	Character	8		N
5	TYPE	Character	20		N
6	OBJECTIVE	Character	20		N
7	COMMENT	Character	20		N
** Total **			78		

DATA BASE: COAATM1.DBF

Field	Field Name	Type	Width	Dec	Index
1	SEQ_NO	Character	5		N
2	CE	Character	7		N
3	COA	Numeric	1		N
4	AVENUE	Character	8		N
5	TYPE	Character	20		N
6	OBJECTIVE	Character	20		N
7	COMMENT	Character	20		N
** Total **			82		

DATA BASE: COAATM2.DBF

Field	Field Name	Type	Width	Dec	Index
1	SEQ_NO	Character	5		N
2	CE	Character	7		N
3	COA	Numeric	1		N
4	AVENUE	Character	8		N
5	TYPE	Character	20		N
6	OBJECTIVE	Character	20		N
7	COMMENT	Character	20		N
8	FR_PERS	Numeric	4		N
9	FR_EQUIP	Numeric	4		N
10	EN_PERS	Numeric	4		N
11	EN_EQUIP	Numeric	4		N
12	POL	Numeric	3		N
13	AMMO	Numeric	3		N
14	FEBA	Numeric	2		N
15	TIME	Numeric	4	1	N

** Total ** 110

DATA BASE: COAATSC.DBF

Field	Field Name	Type	Width	Dec	Index
1	SEQ_NO	Character	5		Y
2	COA	Numeric	1		N
3	SFR_PERS	Numeric	1		N
4	SFR_EQUIP	Numeric	1		N
5	SEN_PERS	Numeric	1		N
6	SEN_EQUIP	Numeric	1		N
7	SPOL	Numeric	1		N
8	SAMMO	Numeric	1		N
9	SFEBA	Numeric	1		N
10	STIME	Numeric	1		N
11	SSUB_A	Numeric	1		N
12	SSUB_B	Numeric	1		N
13	SSUB_C	Numeric	1		N
14	SSUB_D	Numeric	1		N
15	SSUB_E	Numeric	1		N
16	SSUB_F	Numeric	1		N
17	SSUB_G	Numeric	1		N
18	SSUB_H	Numeric	1		N
** Total **			23		

DATA BASE: COAATWT.DBF

Field	Field Name	Type	Width	Dec	Index
1	SEQ_NO	Character	5		Y
2	FR_PERS	Numeric	3		N
3	FR_EQUIP	Numeric	3		N
4	EN_PERS	Numeric	3		N
5	EN_EQUIP	Numeric	3		N
6	POL	Numeric	3		N
7	AMMO	Numeric	3		N
8	FEBA	Numeric	3		N
9	TIME	Numeric	3		N
10	SUB_A	Numeric	3		N
11	SUB_B	Numeric	3		N
12	SUB_C	Numeric	3		N
13	SUB_D	Numeric	3		N
14	SUB_E	Numeric	3		N
15	SUB_F	Numeric	3		N
16	SUB_G	Numeric	3		N
17	SUB_H	Numeric	3		N
18	SUB_NAM1	Character	25		N
19	SUB_NAM2	Character	25		N
20	SUB_NAM3	Character	25		N
21	SUB_NAM4	Character	25		N
22	SUB_NAM5	Character	25		N
23	SUB_NAM6	Character	25		N
24	SUB_NAM7	Character	25		N
25	SUB_NAM8	Character	25		N
** Total **			254		

DATA BASE: COMPANY1.DBF

Field	Field Name	Type	Width	Dec	Index
1	CO_NAME	Character	12		N
2	BASE_NAME	Character	12		N
3	ECHELON	Character	6		N
4	TYPE	Character	6		N
5	BATL_FUNC	Character	6		N
6	ACTIVITY	Character	6		N
7	MISSION	Character	6		N
8	LOCATION	Character	8		N
9	OFFICER	Numeric	3		N
10	ENLISTED	Numeric	3		N
11	EQ_NAME_1	Character	8		N
12	EQ_QTY_1	Numeric	3		N
13	EQ_NAME_2	Character	8		N
14	EQ_QTY_2	Numeric	3		N
15	EQ_NAME_3	Character	8		N
16	EQ_QTY_3	Numeric	3		N
17	EQ_NAME_4	Character	8		N
18	EQ_QTY_4	Numeric	3		N
19	EQ_NAME_5	Character	8		N
20	EQ_QTY_5	Numeric	3		N
21	EQ_NAME_6	Character	8		N
22	EQ_QTY_6	Numeric	3		N
23	EQ_NAME_7	Character	8		N
24	EQ_QTY_7	Numeric	3		N
25	EQ_NAME_8	Character	8		N
26	EQ_QTY_8	Numeric	3		N
27	EQ_NAME_9	Character	8		N
28	EQ_QTY_9	Numeric	3		N
29	EQ_NAME_10	Character	8		N
30	EQ_QTY_10	Numeric	3		N
31	EQ_NAME_11	Character	8		N
32	EQ_QTY_11	Numeric	3		N
33	EQ_NAME_12	Character	8		N
34	EQ_QTY_12	Numeric	3		N
35	AM_NAME_1	Character	8		N
36	AM_QTY_1	Numeric	6		N
37	AM_NAME_2	Character	8		N
38	AM_QTY_2	Numeric	6		N
39	AM_NAME_3	Character	8		N
40	AM_QTY_3	Numeric	6		N
41	AM_NAME_4	Character	8		N
42	AM_QTY_4	Numeric	6		N
43	AM_NAME_5	Character	8		N
44	AM_QTY_5	Numeric	6		N
45	AM_NAME_6	Character	8		N
46	AM_QTY_6	Numeric	6		N
47	MOGAS	Numeric	5		N
48	DIESEL	Numeric	5		N
49	AVGAS	Numeric	5		N
** Total **			300		

DATA BASE: CTL_RQST.DBF

Field	Field Name	Type	Width	Dec	Index
1	SEQ_NO	Character	5		N
2	DATE	Date	8		N
3	TIME	Character	6		N
4	WINDOW	Character	7		N
5	STACK	Numeric	1		N
6	FUNC_AREA	Character	20		N
7	DATA_CAT	Character	20		N
8	DATA_ELE	Character	20		N
9	DATA_SUB	Character	20		N
** Total **			108		

DATA BASE: CTL_XREF.DBF

Field	Field Name	Type	Width	Dec	Index
1	PROD	Character	4		Y
2	FUNC_AREA	Character	20		N
3	DATA_CAT	Character	20		N
4	DATA_ELE	Character	20		N
5	DATA_SUB	Character	20		N
** Total **			85		

DATA BASE: DAY.DBF

Field	Field Name	Type	Width	Dec	Index
1	DATE	Character	10		N
2	OPLAN	Numeric	2		N
3	BCO_FILE	Character	8		N
4	BCO_WRITE	Logical	1		N
5	BBN_FILE	Character	8		N
6	BBN_WRITE	Logical	1		N
7	BBDE_FILE	Character	8		N
8	BBDE_WRITE	Logical	1		N
9	BDIV_FILE	Character	8		N
10	BDIV_WRITE	Logical	1		N
11	RCO_FILE	Character	8		N
12	RCO_WRITE	Logical	1		N
13	RBN_FILE	Character	8		N
14	RBN_WRITE	Logical	1		N
15	RBDE_FILE	Character	8		N
16	RBDE_WRITE	Logical	1		N
17	RDIV_FILE	Character	8		N
18	RDIV_WRITE	Logical	1		N
19	CM_FILE	Character	8		N
20	CM_WRITE	Logical	1		N
21	LOS_FILE	Character	8		N
22	LOS_WRITE	Logical	1		N
** Total **			103		

DATA BASE: DIVISN1.DBF

Field	Field Name	Type	Width	Dec	Index
1	DIV_NAME	Character	12		N
2	ECHELON	Character	6		N
3	TYPE	Character	6		N
4	BATL_FUNC	Character	6		N
5	ACTIVITY	Character	6		N
6	MISSION	Character	6		N
7	LOCATION	Character	8		N
8	BDE_NAME1	Character	12		N
9	BDE_REL_1	Character	6		N
10	BDE_NAME2	Character	12		N
11	BDE_REL_2	Character	6		N
12	BDE_NAME3	Character	12		N
13	BDE_REL_3	Character	6		N
14	BDE_NAME4	Character	12		N
15	BDE_REL_4	Character	6		N
16	BDE_NAME5	Character	12		N
17	BDE_REL_5	Character	6		N
18	BDE_NAME6	Character	12		N
19	BDE_REL_6	Character	6		N
20	BDE_NAME7	Character	12		N
21	BDE_REL_7	Character	6		N
22	BDE_NAME8	Character	12		N
23	BDE_REL_8	Character	6		N
24	BDE_NAME9	Character	12		N
25	BDE_REL_9	Character	6		N
26	BDE_NAME10	Character	12		N
27	BDE_REL_10	Character	6		N
28	BDE_NAME11	Character	12		N
29	BDE_REL_11	Character	6		N
30	BDE_NAME12	Character	12		N
31	BDE_REL_12	Character	6		N
** Total **			267		

DATA BASE: DUMMY.DBF

Field	Field Name	Type	Width	Dec	Index
1	EDDIC	Character	2		N
** Total **			3		

DATA BASE: ED_LUT.DBF

Field	Field Name	Type	Width	Dec	Index
1	PART	Character	4		N
2	DATE	Date	8		N
3	TIME	Character	6		N
4	WINDOW	Character	7		N
5	STACK	Numeric	1		N
6	BACK_TYPE	Character	1		N
7	BACK_ACT	Character	1		N
8	ROAD_ACT	Character	1		N
9	WATER_ACT	Character	1		N
10	URBAN_ACT	Character	1		N

11	MISC_ACT	Character	1		N
12	SEQ_NO	Character	5		N
** Total **			38		

DATA BASE: ED_MAP.DBF

Field	Field Name	Type	Width	Dec	Index
1	PART	Character	4		N
2	DATE	Date	8		N
3	TIME	Character	6		N
4	WINDOW	Character	7		N
5	STACK	Numeric	1		N
6	BACK_TYPE	Character	4		N
7	SCALE	Character	5		N
8	CENTER_X	Numeric	6		N
9	CENTER_Y	Numeric	6		N
10	GRID	Logical	1		N
11	CONTOUR	Logical	1		N
12	BL_UN_DIV	Logical	1		N
13	BL_UN_BDE	Logical	1		N
14	BL_UN_RGMT	Logical	1		N
15	BL_UN_BN	Logical	1		N
16	BL_UN_CO	Logical	1		N
17	BL_UN_CBT	Logical	1		N
18	BL_UN_CS	Logical	1		N
19	BL_UN_CSS	Logical	1		N
20	BL_UN_NAME	Logical	1		N
21	BL_UN_SYM	Logical	1		N
22	OP_UN_DIV	Logical	1		N
23	OP_UN_BDE	Logical	1		N
24	OP_UN_RGMT	Logical	1		N
25	OP_UN_BN	Logical	1		N
26	OP_UN_CO	Logical	1		N
27	OP_UN_COMM	Logical	1		N
28	OP_UN_RENF	Logical	1		N
29	OP_UN_ARTL	Logical	1		N
30	OP_UN_NAME	Logical	1		N
31	OP_UN_SYM	Logical	1		N
32	BL_CM_EAC	Logical	1		N
33	BL_CM_CORP	Logical	1		N
34	BL_CM_DIV	Logical	1		N
35	BL_CM_BDE	Logical	1		N
36	BL_CM_BN	Logical	1		N
37	BL_CM_CO	Logical	1		N
38	BL_CM_PNT	Logical	1		N
39	BL_CM_LINE	Logical	1		N
40	BL_CM_AREA	Logical	1		N
41	BL_CM_RTE	Logical	1		N
42	BL_CM_XNG	Logical	1		N
43	BL_CM_FPLN	Logical	1		N
44	BL_CM_MAPF	Logical	1		N
45	OP_CM_ARMY	Logical	1		N
46	OP_CM_DIV	Logical	1		N
47	OP_CM_RGMT	Logical	1		N
48	OP_CM_BN	Logical	1		N

49	OP_CM_CO	Logical	1	N
50	OP_CM_PNT	Logical	1	N
51	OP_CM_LINE	Logical	1	N
52	OP_CM_AREA	Logical	1	N
53	OP_CM_RTE	Logical	1	N
54	OP_CM_XNG	Logical	1	N
55	OP_CM_FPLN	Logical	1	N
56	OP_CM_MAPF	Logical	1	N
57	SEQ_NO	Character	5	N
** Total **			100	

DATA BASE: ED_WIND.DBF

Field	Field Name	Type	Width	Dec	Index
1	PART	Character	4		N
2	DATE	Date	8		N
3	TIME	Character	6		N
4	WINDOW	Character	7		N
5	STACK	Numeric	1		N
6	ACTION	Character	1		N
7	SEQ_NO	Character	5		N
** Total **			33		

DATA BASE: EDC2RQ.DBF

Field	Field Name	Type	Width	Dec	Index
1	PART	Character	4		N
2	DATE	Date	8		N
3	TIME	Character	6		N
4	WINDOW	Character	7		N
5	STACK	Numeric	1		N
6	PROD	Character	4		N
7	SEQ_NO	Character	5		N
8	FUNC_AREA	Character	20		N
9	DATA_CAT	Character	20		N
10	DATA_ELE	Character	20		N
11	DATA_SUB	Character	20		N
12	DATA_LVL	Character	1		N
** Total **			117		

DATA BASE: EDCOTM1.DBF

Field	Field Name	Type	Width	Dec	Index
1	CE	Character	7		N
2	COA	Numeric	1		N
3	AVENUE	Character	8		N
4	TYPE	Character	20		N
5	OBJECTIVE	Character	20		N
6	COMMENT	Character	20		N
7	SEQ_NO	Character	5		N
** Total **			82		

DATA BASE: EDCOTM2.DBF

Field	Field Name	Type	Width	Dec	Index
1	CE	Character	7		N
2	COA	Numeric	1		N
3	AVENUE	Character	8		N
4	TYPE	Character	20		N
5	OBJECTIVE	Character	20		N
6	COMMENT	Character	20		N
7	FR_PERS	Numeric	4		N
8	FR_EQUIP	Numeric	4		N
9	EN_PERS	Numeric	4		N
10	EN_EQUIP	Numeric	4		N
11	POL	Numeric	3		N
12	AMMO	Numeric	3		N
13	FEBA	Numeric	2		N
14	TIME	Numeric	4	1	N
15	SEQ_NO	Character	5		N
** Total **			110		

DATA BASE: EDCOTSC.DBF

Field	Field Name	Type	Width	Dec	Index
1	COA	Numeric	1		N
2	SFR_PERS	Numeric	1		N
3	SFR_EQUIP	Numeric	1		N
4	SEN_PERS	Numeric	1		N
5	SEN_EQUIP	Numeric	1		N
6	SPOL	Numeric	1		N
7	SAMMO	Numeric	1		N
8	SFEBA	Numeric	1		N
9	STIME	Numeric	1		N
10	SSUB_A	Numeric	1		N
11	SSUB_B	Numeric	1		N
12	SSUB_C	Numeric	1		N
13	SSUB_D	Numeric	1		N
14	SSUB_E	Numeric	1		N
15	SSUB_F	Numeric	1		N
16	SSUB_G	Numeric	1		N
17	SSUB_H	Numeric	1		N
18	SEQ_NO	Character	5		N
** Total **			23		

DATA BASE: EDCOTWT.DBF

Field	Field Name	Type	Width	Dec	Index
1	FR_PERS	Numeric	3		N
2	FR_EQUIP	Numeric	3		N
3	EN_PERS	Numeric	3		N
4	EN_EQUIP	Numeric	3		N
5	POL	Numeric	3		N
6	AMMO	Numeric	3		N
7	FEBA	Numeric	3		N
8	TIME	Numeric	3		N

9	SUB_A	Numeric	3	N
10	SUB_B	Numeric	3	N
11	SUB_C	Numeric	3	N
12	SUB_D	Numeric	3	N
13	SUB_E	Numeric	3	N
14	SUB_F	Numeric	3	N
15	SUB_G	Numeric	3	N
16	SUB_H	Numeric	3	N
17	SUB_NAM1	Character	25	N
18	SUB_NAM2	Character	25	N
19	SUB_NAM3	Character	25	N
20	SUB_NAM4	Character	25	N
21	SUB_NAM5	Character	25	N
22	SUB_NAM6	Character	25	N
23	SUB_NAM7	Character	25	N
24	SUB_NAM8	Character	25	N
25	SEQ_NO	Character	5	N
** Total **			254	

DATA BASE: EDCTLRQ.DBF

Field	Field Name	Type	Width	Dec	Index
1	PART	Character	4		N
2	DATE	Date	8		N
3	TIME	Character	6		N
4	WINDOW	Character	7		N
5	STACK	Numeric	1		N
6	PROD	Character	4		N
7	SEQ_NO	Character	5		N
8	FUNC_AREA	Character	20		N
9	DATA_CAT	Character	20		N
10	DATA_ELE	Character	20		N
11	DATA_SUB	Character	20		N
** Total **			116		

DATA BASE: EDNEWC2.DBF

Field	Field Name	Type	Width	Dec	Index
1	PART	Character	4		N
2	DATE	Date	8		N
3	TIME	Character	6		N
4	WINDOW	Character	7		N
5	STACK	Numeric	1		N
6	PROD	Character	4		N
7	TO_G2	Logical	1		N
8	TO_G3	Logical	1		N
9	TO_G4	Logical	1		N
10	TO_EX	Logical	1		N
11	SEQ_NO	Character	5		N
** Total **			40		

DATA BASE: EDREFRQ.DBF

Field	Field Name	Type	Width	Dec	Index
1	PART	Character	4		N
2	DATE	Date	8		N
3	TIME	Character	6		N
4	WINDOW	Character	7		N
5	STACK	Numeric	1		N
6	PROD	Character	4		N
7	SEQ_NO	Character	5		N
8	FUNC_AREA	Character	20		N
9	DATA_CAT	Character	20		N
10	DATA_ELE	Character	20		N
11	DATA_LVL	Character	1		N
** Total **			97		

DATA BASE: EDSTACTV.DBF

Field	Field Name	Type	Width	Dec	Index
1	PART	Character	4		N
2	DATE	Date	8		N
3	TIME	Character	6		N
4	WINDOW	Character	7		N
5	STACK	Numeric	1		N
6	SIT_DATE	Date	8		N
7	SIT_TIME	Character	6		N
8	OPLAN_NO	Numeric	2		N
9	FORCE	Character	7		N
10	UNIT_NO	Numeric	3		N
11	ACTIVITY	Character	12		N
12	SEQ_NO	Character	5		N
13	OPLAN	Character	20		N
14	UNIT	Character	15		N
** Total **			105		

DATA BASE: EDSTAMMO.DBF

Field	Field Name	Type	Width	Dec	Index
1	PART	Character	4		N
2	DATE	Date	8		N
3	TIME	Character	6		N
4	WINDOW	Character	7		N
5	STACK	Numeric	1		N
6	SIT_DATE	Date	8		N
7	SIT_TIME	Character	6		N
8	OPLAN_NO	Numeric	2		N
9	FORCE	Character	7		N
10	UNIT_NO	Numeric	3		N
11	AMMO_NO	Numeric	3		N
12	AMOUNT	Numeric	5		N
13	SEQ_NO	Character	5		N
14	OPLAN	Character	20		N
15	UNIT	Character	15		N
16	AMMO	Character	12		N
** Total **			113		

DATA BASE: EDSTBLTO.DBF

Field	Field Name	Type	Width	Dec	Index
1	PART	Character	4		N
2	DATE	Date	8		N
3	TIME	Character	6		N
4	WINDOW	Character	7		N
5	STACK	Numeric	1		N
6	SIT_DATE	Date	8		N
7	SIT_TIME	Character	6		N
8	OPLAN_NO	Numeric	2		N
9	FORCE	Character	7		N
10	UNIT_NO	Numeric	3		N
11	HI_ECH_NO	Numeric	3		N
12	RELATE	Character	4		N
13	SEQ_NO	Character	5		N
14	OPLAN	Character	20		N
15	UNIT	Character	15		N
16	HIGH_ECH	Character	15		N
** Total **			115		

DATA BASE: EDSTCMDL.DBF

Field	Field Name	Type	Width	Dec	Index
1	PART	Character	4		N
2	DATE	Date	8		N
3	TIME	Character	6		N
4	WINDOW	Character	7		N
5	STACK	Numeric	1		N
6	SIT_DATE	Date	8		N
7	SIT_TIME	Character	6		N
8	OPLAN_NO	Numeric	2		N
9	CM_ID	Numeric	3		N
10	SEQ_NO	Character	5		N
11	OPLAN	Character	20		N
12	NAME	Character	12		N
** Total **			83		

DATA BASE: EDSTCMEF.DBF

Field	Field Name	Type	Width	Dec	Index
1	PART	Character	4		N
2	DATE	Date	8		N
3	TIME	Character	6		N
4	WINDOW	Character	7		N
5	STACK	Numeric	1		N
6	SIT_DATE	Date	8		N
7	SIT_TIME	Character	6		N
8	OPLAN_NO	Numeric	2		N
9	CM_ID	Numeric	3		N
10	EFF2DATE	Date	8		N
11	EFF2TIME	Character	6		N
12	SEQ_NO	Character	5		N

13	OPLAN	Character	20		N
14	NAME	Character	12		N
** Total **			97		

DATA BASE: EDSTCMLC.DBF

Field	Field Name	Type	Width	Dec	Index
1	PART	Character	4		N
2	DATE	Date	8		N
3	TIME	Character	6		N
4	WINDOW	Character	7		N
5	STACK	Numeric	1		N
6	SIT_DATE	Date	8		N
7	SIT_TIME	Character	6		N
8	OPLAN_NO	Numeric	2		N
9	CM_ID	Numeric	3		N
10	LOCX1	Numeric	6		N
11	LOCY1	Numeric	6		N
12	LOCX2	Numeric	6		N
13	LOCY2	Numeric	6		N
14	LOCX3	Numeric	6		N
15	LOCY3	Numeric	6		N
16	LOCX4	Numeric	6		N
17	LOCY4	Numeric	6		N
18	LOCX5	Numeric	6		N
19	LOCY5	Numeric	6		N
20	LOCX6	Numeric	6		N
21	LOCY6	Numeric	6		N
22	LOCX7	Numeric	6		N
23	LOXY7	Numeric	6		N
24	LOCX8	Numeric	6		N
25	LOCY8	Numeric	6		N
26	LOCX9	Numeric	6		N
27	LOCY9	Numeric	6		N
28	LOCX10	Numeric	6		N
29	LOCY10	Numeric	6		N
30	LOCX11	Numeric	6		N
31	LOCY11	Numeric	6		N
32	LOCX12	Numeric	6		N
33	LOCY12	Numeric	6		N
34	LOCX13	Numeric	6		N
35	LOCY13	Numeric	6		N
36	LOCX14	Numeric	6		N
37	LOCY14	Numeric	6		N
38	LOCX15	Numeric	6		N
39	LOCY15	Numeric	6		N
40	SEQ_NO	Character	5		N
41	OPLAN	Character	20		N
42	NAME	Character	12		N
** Total **			263		

DATA BASE: EDSTCMST.DBF

Field	Field Name	Type	Width	Dec	Index

1	PART	Character	4	N
2	DATE	Date	8	N
3	TIME	Character	6	N
4	WINDOW	Character	7	N
5	STACK	Numeric	1	N
6	SIT_DATE	Date	8	N
7	SIT_TIME	Character	6	N
8	OPLAN_NO	Numeric	2	N
9	CM_ID	Numeric	3	N
10	STATUS	Character	12	N
11	SEQ_NO	Character	5	N
12	OPLAN	Character	20	N
13	NAME	Character	12	N
** Total **			95	

DATA BASE: EDSTEQP.DBF

Field	Field Name	Type	Width	Dec	Index
1	PART	Character	4		N
2	DATE	Date	8		N
3	TIME	Character	6		N
4	WINDOW	Character	7		N
5	STACK	Numeric	1		N
6	SIT_DATE	Date	8		N
7	SIT_TIME	Character	6		N
8	OPLAN_NO	Numeric	2		N
9	FORCE	Character	7		N
10	UNIT_NO	Numeric	3		N
11	EQUIP_NO	Numeric	3		N
12	AMOUNT	Numeric	5		N
13	SEQ_NO	Character	5		N
14	OPLAN	Character	20		N
15	UNIT	Character	15		N
16	EQUIP	Character	12		N
** Total **			113		

DATA BASE: EDSTFUEL.DBF

Field	Field Name	Type	Width	Dec	Index
1	PART	Character	4		N
2	DATE	Date	8		N
3	TIME	Character	6		N
4	WINDOW	Character	7		N
5	STACK	Numeric	1		N
6	SIT_DATE	Date	8		N
7	SIT_TIME	Character	6		N
8	OPLAN_NO	Numeric	2		N
9	FORCE	Character	7		N
10	UNIT_NO	Numeric	3		N
11	MOGAS	Numeric	5		N
12	AVGAS	Numeric	5		N
13	DIESEL	Numeric	5		N
14	SEQ_NO	Character	5		N
15	OPLAN	Character	20		N

16	UNIT	Character	15		N
** Total **			108		

DATA BASE: EDSTMISS.DBF

Field	Field Name	Type	Width	Dec	Index
1	PART	Character	4		N
2	DATE	Date	8		N
3	TIME	Character	6		N
4	WINDOW	Character	7		N
5	STACK	Numeric	1		N
6	SIT_DATE	Date	8		N
7	SIT_TIME	Character	6		N
8	OPLAN_NO	Numeric	2		N
9	FORCE	Character	7		N
10	UNIT_NO	Numeric	3		N
11	MISSION	Character	15		N
12	SEQ_NO	Character	5		N
13	OPLAN	Character	20		N
14	UNIT	Character	15		N
** Total **			108		

DATA BASE: EDSTNWCM.DBF

Field	Field Name	Type	Width	Dec	Index
1	PART	Character	4		N
2	DATE	Date	8		N
3	TIME	Character	6		N
4	WINDOW	Character	7		N
5	STACK	Numeric	1		N
6	SIT_DATE	Date	8		N
7	SIT_TIME	Character	6		N
8	OPLAN_NO	Numeric	2		N
9	CM_ID	Numeric	3		N
10	NAME	Character	12		N
11	FORCE	Character	7		N
12	TYPE	Character	6		N
13	SCAL1_40	Logical	1		N
14	SCAL1_80	Logical	1		N
15	SCAL1_160	Logical	1		N
16	SCAL1_400	Logical	1		N
17	SCAL1_800	Logical	1		N
18	STATUS	Character	12		N
19	EFF2DATE	Date	8		N
20	EFF2TIME	Character	6		N
21	ECHELON	Character	6		N
22	LOCX1	Numeric	6		N
23	LOCY1	Numeric	6		N
24	LOCX2	Numeric	6		N
25	LOCY2	Numeric	6		N
26	LOCX3	Numeric	6		N
27	LOCY3	Numeric	6		N
28	LOCX4	Numeric	6		N
29	LOCY4	Numeric	6		N

30	LOCX5	Numeric	6	N
31	LOCY5	Numeric	6	N
32	LOCX6	Numeric	6	N
33	LOCY6	Numeric	6	N
34	LOCX7	Numeric	6	N
35	LOCY7	Numeric	6	N
36	LOCX8	Numeric	6	N
37	LOCY8	Numeric	6	N
38	LOCX9	Numeric	6	N
39	LOCY9	Numeric	6	N
40	LOCX10	Numeric	6	N
41	LOCY10	Numeric	6	N
42	LOCX11	Numeric	6	N
43	LOCY11	Numeric	6	N
44	LOCX12	Numeric	6	N
45	LOCY12	Numeric	6	N
46	LOCX13	Numeric	6	N
47	LOCY13	Numeric	6	N
48	LOCX14	Numeric	6	N
49	LOCY14	Numeric	6	N
50	LOCX15	Numeric	6	N
51	LOCY15	Numeric	6	N
52	SEQ_NO	Character	5	N
53	OPLAN	Character	20	N
** Total **			313	

DATA BASE: EDSTNWOB.DBF

Field	Field Name	Type	Width	Dec	Index
1	PART	Character	4		N
2	DATE	Date	8		N
3	TIME	Character	6		N
4	WINDOW	Character	7		N
5	STACK	Numeric	1		N
6	SIT_DATE	Date	8		N
7	SIT_TIME	Character	6		N
8	OPLAN_NO	Numeric	2		N
9	OB_ID	Numeric	3		N
10	FORCE	Character	7		N
11	TYPE	Character	6		N
12	STATUS	Character	12		N
13	EFF2DATE	Date	8		N
14	EFF2TIME	Character	6		N
15	LOCX	Numeric	6		N
16	LOCY	Numeric	6		N
17	FRONTAGE	Numeric	5		N
18	DEPTH	Numeric	5		N
19	ORIENT	Numeric	3		N
20	GAPS	Logical	1		N
21	ECHELON	Character	6		N
22	SEQ_NO	Character	5		N
23	OPLAN	Character	20		N
** Total **			142		

DATA BASE: EDSTOBDL.DBF

Field	Field Name	Type	Width	Dec	Index
1	PART	Character	4		N
2	DATE	Date	8		N
3	TIME	Character	6		N
4	WINDOW	Character	7		N
5	STACK	Numeric	1		N
6	SIT_DATE	Date	8		N
7	SIT_TIME	Character	6		N
8	OPLAN_NO	Numeric	2		N
9	OB_ID	Numeric	3		N
10	SEQ_NO	Character	5		N
11	OPLAN	Character	20		N
** Total **			71		

DATA BASE: EDSTOBEF.DBF

Field	Field Name	Type	Width	Dec	Index
1	PART	Character	4		N
2	DATE	Date	8		N
3	TIME	Character	6		N
4	WINDOW	Character	7		N
5	STACK	Numeric	1		N
6	SIT_DATE	Date	8		N
7	SIT_TIME	Character	6		N
8	OPLAN_NO	Numeric	2		N
9	OB_ID	Numeric	3		N
10	EFF2DATE	Date	8		N
11	EFF2TIME	Character	6		N
12	SEQ_NO	Character	5		N
13	OPLAN	Character	20		N
** Total **			85		

DATA BASE: EDSTOBLC.DBF

Field	Field Name	Type	Width	Dec	Index
1	PART	Character	4		N
2	DATE	Date	8		N
3	TIME	Character	6		N
4	WINDOW	Character	7		N
5	STACK	Numeric	1		N
6	SIT_DATE	Date	8		N
7	SIT_TIME	Character	6		N
8	OPLAN_NO	Numeric	2		N
9	OB_ID	Numeric	3		N
10	LOCX	Numeric	6		N
11	LOCY	Numeric	6		N
12	SEQ_NO	Character	5		N
13	OPLAN	Character	20		N
** Total **			83		

DATA BASE: EDSTOBST.DBF

Field	Field Name	Type	Width	Dec	Index
1	PART	Character	4		N
2	DATE	Date	8		N
3	TIME	Character	6		N
4	WINDOW	Character	7		N
5	STACK	Numeric	1		N
6	SIT_DATE	Date	8		N
7	SIT_TIME	Character	6		N
8	OPLAN_NO	Numeric	2		N
9	OB_ID	Numeric	3		N
10	STATUS	Character	12		N
11	SEQ_NO	Character	5		N
12	OPLAN	Character	20		N
** Total **			83		

DATA BASE: EDSTOPTO.DBF

Field	Field Name	Type	Width	Dec	Index
1	PART	Character	4		N
2	DATE	Date	8		N
3	TIME	Character	6		N
4	WINDOW	Character	7		N
5	STACK	Numeric	1		N
6	SIT_DATE	Date	8		N
7	SIT_TIME	Character	6		N
8	OPLAN_NO	Numeric	2		N
9	FORCE	Character	7		N
10	UNIT_NO	Numeric	3		N
11	HI_ECH_NO	Numeric	3		N
12	SEQ_NO	Character	5		N
13	OPLAN	Character	20		N
14	UNIT	Character	15		N
15	HIGH_ECH	Character	15		N
** Total **			111		

DATA BASE: EDSTPERS.DBF

Field	Field Name	Type	Width	Dec	Index
1	PART	Character	4		N
2	DATE	Date	8		N
3	TIME	Character	6		N
4	WINDOW	Character	7		N
5	STACK	Numeric	1		N
6	SIT_DATE	Date	8		N
7	SIT_TIME	Character	6		N
8	OPLAN_NO	Numeric	2		N
9	FORCE	Character	7		N
10	UNIT_NO	Numeric	3		N
11	OFF_AMT	Numeric	4		N
12	ENL_AMT	Numeric	4		N
13	SEQ_NO	Character	5		N
14	OPLAN	Character	20		N
15	UNIT	Character	15		N

** Total ** 101

DATA BASE: EDSTRENF.DBF

Field	Field Name	Type	Width	Dec	Index
1	PART	Character	4		N
2	DATE	Date	8		N
3	TIME	Character	6		N
4	WINDOW	Character	7		N
5	STACK	Numeric	1		N
6	SIT_DATE	Date	8		N
7	SIT_TIME	Character	6		N
8	OPLAN_NO	Numeric	2		N
9	FORCE	Character	7		N
10	UNIT_NO	Numeric	3		N
11	REINF	Numeric	3		N
12	SEQ_NO	Character	5		N
13	OPLAN	Character	20		N
14	UNIT	Character	15		N
** Total **			96		

DATA BASE: EDSTRQST.DBF

Field	Field Name	Type	Width	Dec	Index
1	PART	Character	4		N
2	DATE	Date	8		N
3	TIME	Character	6		N
4	WINDOW	Character	7		N
5	STACK	Numeric	1		N
6	SIT_DATE	Date	8		N
7	SIT_TIME	Character	6		N
8	OPLAN_NO	Numeric	2		N
9	FORCE	Character	7		N
10	UNIT_NO	Numeric	3		N
11	MESSAGE	Character	20		N
12	SEQ_NO	Character	5		N
13	OPLAN	Character	20		N
14	UNIT	Character	15		N
** Total **			113		

DATA BASE: EDSTSTNG.DBF

Field	Field Name	Type	Width	Dec	Index
1	PART	Character	4		N
2	DATE	Date	8		N
3	TIME	Character	6		N
4	WINDOW	Character	7		N
5	STACK	Numeric	1		N
6	SIT_DATE	Date	8		N
7	SIT_TIME	Character	6		N
8	OPLAN_NO	Numeric	2		N
9	FORCE	Character	7		N
10	UNIT_NO	Numeric	3		N

11	STRENGTH	Numeric	3	N
12	SEQ_NO	Character	5	N
13	OPLAN	Character	20	N
14	UNIT	Character	15	N
** Total **			96	

DATA BASE: EDSTULOC.DBF

Field	Field Name	Type	Width	Dec	Index
1	PART	Character	4		N
2	DATE	Date	8		N
3	TIME	Character	6		N
4	WINDOW	Character	7		N
5	STACK	Numeric	1		N
6	SIT_DATE	Date	8		N
7	SIT_TIME	Character	6		N
8	OPLAN_NO	Numeric	2		N
9	FORCE	Character	7		N
10	UNIT_NO	Numeric	3		N
11	X_LOC	Numeric	6		N
12	Y_LOC	Numeric	6		N
13	SEQ_NO	Character	5		N
14	OPLAN	Character	20		N
15	UNIT	Character	15		N
** Total **			105		

DATA BASE: EQDISPLA.DBF

Field	Field Name	Type	Width	Dec	Index
1	EQ_NAME	Character	8		N
2	EQ_AUTH	Numeric	6		N
3	EQ_CURR	Numeric	6		N
** Total **			21		

DATA BASE: HLP_XREF.DBF

Field	Field Name	Type	Width	Dec	Index
1	PROD	Character	4		Y
2	FUNC_AREA	Character	20		N
3	DATA_CAT	Character	20		N
4	DATA_ELE	Character	20		N
** Total **			65		

DATA BASE: HMIED.DBF

Field	Field Name	Type	Width	Dec	Index
1	SEQ_NO	Character	5		Y
2	LABORG	Numeric	1		N
3	MAPORG	Numeric	1		N
4	GRIDS	Numeric	1		N
5	CONTOUR	Numeric	1		N
6	ROADS	Numeric	1		N

7	HYDRO	Numeric	1	N
8	BUILTUP	Numeric	1	N
9	MISC	Numeric	1	N
10	MPSCL80	Numeric	1	N
11	MPSCL160	Numeric	1	N
12	MPSCL400	Numeric	1	N
13	MPSCL800	Numeric	1	N
14	SHADRELF	Numeric	1	N
15	VEG	Numeric	1	N
16	ELEVBAND	Numeric	1	N
17	NONE	Numeric	1	N
18	BLUEUNIT	Numeric	1	N
19	BLUECM	Numeric	1	N
20	OPFORUNIT	Numeric	1	N
21	OPFORCM	Numeric	1	N
22	CONTERM	Logical	1	N
23	SUNCE	Logical	1	N
24	CONCOOR	Numeric	1	N
25	EASMENU	Numeric	1	N
26	EASTORG	Numeric	1	N
** Total **			31	

DATA BASE: HMIEDCT.DBF

Field	Field Name	Type	Width	Dec	Index
1	SEQ_NO	Character	5		Y
2	LABORG	Numeric	1		N
3	MAPORG	Numeric	1		N
4	GRIDS	Numeric	1		N
5	CONTOUR	Numeric	1		N
6	ROADS	Numeric	1		N
7	HYDRO	Numeric	1		N
8	BUILTUP	Numeric	1		N
9	MISC	Numeric	1		N
10	MPSCL80	Numeric	1		N
11	MPSCL160	Numeric	1		N
12	MPSCL400	Numeric	1		N
13	MPSCL800	Numeric	1		N
14	SHADRELF	Numeric	1		N
15	VEG	Numeric	1		N
16	ELEVBAND	Numeric	1		N
17	NONE	Numeric	1		N
18	BLUEUNIT	Numeric	1		N
19	BLUECM	Numeric	1		N
20	OPFORUNIT	Numeric	1		N
21	OPFORCM	Numeric	1		N
22	CONTERM	Logical	1		N
23	COMCOOR	Numeric	1		N
24	SYMCE	Numeric	1		N
25	SYMWG	Numeric	1		N
26	SYMEASE	Numeric	1		N
** Total **			31		

DATA BASE: HST_XREF.DBF

Field	Field Name	Type	Width	Dec	Index
1	PROD	Character	4		Y
2	FUNC_AREA	Character	20		N
3	DATA_CAT	Character	20		N
4	DATA_ELE	Character	20		N
5	DATA_LVL	Character	1		N
6	DATA_SUB	Character	20		N
** Total **			86		

DATA BASE: LOSSRAT2.DBF

Field	Field Name	Type	Width	Dec	Index
1	UNIT_NAME	Character	12		N
2	OFF_LRATE	Numeric	6	4	N
3	ENL_LRATE	Numeric	6	4	N
4	GAIN_RATE	Numeric	6	4	N
** Total **			31		

DATA BASE: LUT_CTRL.DBF

Field	Field Name	Type	Width	Dec	Index
1	SEQ_NO	Character	5		N
2	DATE	Date	8		N
3	TIME	Character	6		N
4	WINDOW	Character	7		N
5	STACK	Numeric	1		N
6	BACK_TYPE	Character	1		N
7	BACK_ACT	Character	1		N
8	ROAD_ACT	Character	1		N
9	WATER_ACT	Character	1		N
10	URBAN_ACT	Character	1		N
11	MISC_ACT	Character	1		N
** Total **			34		

DATA BASE: MAP_CTRL.DBF

Field	Field Name	Type	Width	Dec	Index
1	SEQ_NO	Character	5		N
2	DATE	Date	8		N
3	TIME	Character	6		N
4	WINDOW	Character	7		N
5	STACK	Numeric	1		N
6	BACK_TYPE	Character	4		N
7	SCALE	Character	5		N
8	CENTER_X	Numeric	6		N
9	CENTER_Y	Numeric	6		N
10	GRID	Logical	1		N
11	CONTOUR	Logical	1		N
12	BL_UN_DIV	Logical	1		N
13	BL_UN_BDE	Logical	1		N
14	BL_UN_RGMT	Logical	1		N
15	BL_UN_BN	Logical	1		N

16	BL_UN_CO	Logical	1	N
17	BL_UN_CBT	Logical	1	N
18	BL_UN_CS	Logical	1	N
19	BL_UN_CSS	Logical	1	N
20	BL_UN_NAME	Logical	1	N
21	BL_UN_SYM	Logical	1	N
22	OP_UN_DIV	Logical	1	N
23	OP_UN_BDE	Logical	1	N
24	OP_UN_RGMT	Logical	1	N
25	OP_UN_BN	Logical	1	N
26	OP_UN_CO	Logical	1	N
27	OP_UN_COMM	Logical	1	N
28	OP_UN_RENF	Logical	1	N
29	OP_UN_ARTL	Logical	1	N
30	OP_UN_NAME	Logical	1	N
31	OP_UN_SYM	Logical	1	N
32	BL_CM_EAC	Logical	1	N
33	BL_CM_CORP	Logical	1	N
34	BL_CM_DIV	Logical	1	N
35	BL_CM_BDE	Logical	1	N
36	BL_CM_BN	Logical	1	N
37	BL_CM_CO	Logical	1	N
38	BL_CM_PNT	Logical	1	N
39	BL_CM_LINE	Logical	1	N
40	BL_CM_AREA	Logical	1	N
41	BL_CM_RTE	Logical	1	N
42	BL_CM_XNG	Logical	1	N
43	BL_CM_FPLN	Logical	1	N
44	BL_CM_MAPF	Logical	1	N
45	OP_CM_ARMY	Logical	1	N
46	OP_CM_DIV	Logical	1	N
47	OP_CM_RGMT	Logical	1	N
48	OP_CM_BN	Logical	1	N
49	OP_CM_CO	Logical	1	N
50	OP_CM_PNT	Logical	1	N
51	OP_CM_LINE	Logical	1	N
52	OP_CM_AREA	Logical	1	N
53	OP_CM_RTE	Logical	1	N
54	OP_CM_XNG	Logical	1	N
55	OP_CM_FPLN	Logical	1	N
56	OP_CM_MAPF	Logical	1	N
** Total **			96	

DATA BASE: MISSION.DBF

Field	Field Name	Type	Width	Dec	Index
1	MISSID	Character	1		Y
2	MISSNAME	Character	17		N
** Total **			19		

DATA BASE: NEW_C2.DBF

Field	Field Name	Type	Width	Dec	Index
1	SEQ_NO	Character	5		N

2	DATE	Date	8	N
3	TIME	Character	6	N
4	WINDOW	Character	7	N
5	STACK	Numeric	1	N
6	PROD	Character	4	N
7	TO_G2	Logical	1	N
8	TO_G3	Logical	1	N
9	TO_G4	Logical	1	N
10	TO_EX	Logical	1	N
** Total **			36	

DATA BASE: OPLAN.DBF

Field	Field Name	Type	Width	Dec	Index
1	NUMBER	Numeric	2		N
2	NAME	Character	18		N
3	TYPE	Character	13		N
** Total **			34		

DATA BASE: PERCENT.DBF

Field	Field Name	Type	Width	Dec	Index
1	OFF_PER_1	Numeric	7	3	N
2	ENL_PER_1	Numeric	7	3	N
3	EQ_PER_1	Numeric	7	3	N
4	AM_PER_1	Numeric	7	3	N
5	FL_PER_1	Numeric	7	3	N
6	OFF_PER_2	Numeric	7	3	N
7	ENL_PER_2	Numeric	7	3	N
8	EQ_PER_2	Numeric	7	3	N
9	AM_PER_2	Numeric	7	3	N
10	FL_PER_2	Numeric	7	3	N
11	OFF_PER_3	Numeric	7	3	N
12	ENL_PER_3	Numeric	7	3	N
13	EQ_PER_3	Numeric	7	3	N
14	AM_PER_3	Numeric	7	3	N
15	FL_PER_3	Numeric	7	3	N
16	OFF_PER_4	Numeric	7	3	N
17	ENL_PER_4	Numeric	7	3	N
18	EQ_PER_4	Numeric	7	3	N
19	AM_PER_4	Numeric	7	3	N
20	FL_PER_4	Numeric	7	3	N
21	OFF_PER_5	Numeric	7	3	N
22	ENL_PER_5	Numeric	7	3	N
23	EQ_PER_5	Numeric	7	3	N
24	AM_PER_5	Numeric	7	3	N
25	FL_PER_5	Numeric	7	3	N
26	OFF_PER_6	Numeric	7	3	N
27	ENL_PER_6	Numeric	7	3	N
28	EQ_PER_6	Numeric	7	3	N
29	AM_PER_6	Numeric	7	3	N
30	FL_PER_6	Numeric	7	3	N
31	OFF_PER_7	Numeric	7	3	N
32	ENL_PER_7	Numeric	7	3	N

33	EQ_PER_7	Numeric	7	3	N
34	AM_PER_7	Numeric	7	3	N
35	FL_PER_7	Numeric	7	3	N
36	OFF_PER_8	Numeric	7	3	N
37	ENL_PER_8	Numeric	7	3	N
38	EQ_PER_8	Numeric	7	3	N
39	AM_PER_8	Numeric	7	3	N
40	FL_PER_8	Numeric	7	3	N
** Total **			281		

DATA BASE: PERDISP.DBF

Field	Field Name	Type	Width	Dec	Index
1	UNIT_NAME	Character	12		N
2	OFF_LOSS	Numeric	4		N
3	ENL_LOSS	Numeric	4		N
4	OFF_GAIN	Numeric	4		N
5	ENL_GAIN	Numeric	4		N
6	OFF_AUTH	Numeric	7		N
7	ENL_AUTH	Numeric	7		N
8	OFF_CURR	Numeric	7		N
9	ENL_CURR	Numeric	7		N
** Total **			57		

DATA BASE: PERSON.DBF

Field	Field Name	Type	Width	Dec	Index
1	SEQ_NO	Character	5		Y
2	DATE	Date	8		N
3	NAME	Character	30		N
4	RANK	Character	4		N
5	CURR_POS	Character	20		N
6	BRANCH	Character	15		N
7	TIG_YR	Numeric	4	1	N
8	TIS_YR	Numeric	4	1	N
9	EDUCATION	Character	2		N
10	AREA_STUDY	Character	15		N
11	OFF_BASIC	Numeric	4		N
12	OFF_ADV	Numeric	4		N
13	CAS3	Numeric	4		N
14	CGSOC	Numeric	4		N
15	WAR_COLLGE	Numeric	4		N
16	ASG_FULDA	Logical	1		N
17	EXR_FULDA	Logical	1		N
18	CMP_COURSE	Logical	1		N
19	MINI_FREQ	Character	1		N
20	WORK_FREQ	Character	1		N
21	PC_FREQ	Character	1		N
22	CURS_CNTRL	Logical	1		N
23	PRGM_SELF	Logical	1		N
24	PRGM_OTHER	Logical	1		N
25	OWN_CMPUTR	Logical	1		N
** Total **			138		

DATA BASE: PERSTYLE.DBF

Field	Field Name	Type	Width	Dec	Index
1	SEQ_NO	Character	5		Y
2	A_1	Numeric	1		N
3	B_1	Numeric	1		N
4	A_2	Numeric	1		N
5	B_2	Numeric	1		N
6	A_3	Numeric	1		N
7	B_3	Numeric	1		N
8	A_4	Numeric	1		N
9	B_4	Numeric	1		N
10	A_5	Numeric	1		N
11	B_5	Numeric	1		N
12	A_6	Numeric	1		N
13	B_6	Numeric	1		N
14	A_7	Numeric	1		N
15	B_7	Numeric	1		N
16	A_8	Numeric	1		N
17	B_8	Numeric	1		N
18	A_9	Numeric	1		N
19	B_9	Numeric	1		N
20	A_10	Numeric	1		N
21	B_10	Numeric	1		N
22	A_11	Numeric	1		N
23	B_11	Numeric	1		N
24	A_12	Numeric	1		N
25	B_12	Numeric	1		N
26	A_13	Numeric	1		N
27	B_13	Numeric	1		N
28	A_14	Numeric	1		N
29	B_14	Numeric	1		N
30	A_15	Numeric	1		N
31	B_15	Numeric	1		N
32	A_16	Numeric	1		N
33	B_16	Numeric	1		N
34	A_17	Numeric	1		N
35	B_17	Numeric	1		N
36	A_18	Numeric	1		N
37	B_18	Numeric	1		N
38	A_19	Numeric	1		N
39	B_19	Numeric	1		N
40	A_20	Numeric	1		N
41	B_20	Numeric	1		N
42	A_21	Numeric	1		N
43	B_21	Numeric	1		N
44	A_22	Numeric	1		N
45	B_22	Numeric	1		N
46	A_23	Numeric	1		N
47	B_23	Numeric	1		N
48	A_24	Numeric	1		N
49	B_24	Numeric	1		N
50	A_25	Numeric	1		N
51	B_25	Numeric	1		N
52	A_26	Numeric	1		N

53	B_26	Numeric	1		N
54	A_27	Numeric	1		N
55	B_27	Numeric	1		N
56	A_28	Numeric	1		N
57	B_28	Numeric	1		N
58	A_29	Numeric	1		N
59	B_29	Numeric	1		N
60	A_30	Numeric	1		N
61	B_30	Numeric	1		N
62	A_31	Numeric	1		N
63	B_31	Numeric	1		N
64	A_32	Numeric	1		N
65	B_32	Numeric	1		N
** Total **			70		

DATA BASE: RBASEUNI.DBF

Field	Field Name	Type	Width	Dec	Index
1	UNIT_NAME	Character	12		N
2	OFFICER	Numeric	3		N
3	ENLISTED	Numeric	3		N
4	EQ_NAME_1	Character	8		N
5	EQ_QTY_1	Numeric	3		N
6	EQ_NAME_2	Character	8		N
7	EQ_QTY_2	Numeric	3		N
8	EQ_NAME_3	Character	8		N
9	EQ_QTY_3	Numeric	3		N
10	EQ_NAME_4	Character	8		N
11	EQ_QTY_4	Numeric	3		N
12	EQ_NAME_5	Character	8		N
13	EQ_QTY_5	Numeric	3		N
14	EQ_NAME_6	Character	8		N
15	EQ_QTY_6	Numeric	3		N
16	EQ_NAME_7	Character	8		N
17	EQ_QTY_7	Numeric	3		N
18	EQ_NAME_8	Character	8		N
19	EQ_QTY_8	Numeric	3		N
20	EQ_NAME_9	Character	8		N
21	EQ_QTY_9	Numeric	3		N
22	EQ_NAME_10	Character	8		N
23	EQ_QTY_10	Numeric	3		N
24	EQ_NAME_11	Character	8		N
25	EQ_QTY_11	Numeric	3		N
26	EQ_NAME_12	Character	8		N
27	EQ_QTY_12	Numeric	3		N
** Total **			151		

DATA BASE: RBATTAL1.DBF

Field	Field Name	Type	Width	Dec	Index
1	BN_NAME	Character	12		N
2	ECHELON	Character	6		N
3	TYPE	Character	6		N
4	BATL_FUNC	Character	6		N

5	ACTIVITY	Character	6		N
6	MISSION	Character	6		N
7	LOCATION	Character	8		N
8	CO_NAME_1	Character	12		N
9	CO_REL_1	Character	6		N
10	CO_NAME_2	Character	12		N
11	CO_REL_2	Character	6		N
12	CO_NAME_3	Character	12		N
13	CO_REL_3	Character	6		N
14	CO_NAME_4	Character	12		N
15	CO_REL_4	Character	6		N
16	CO_NAME_5	Character	12		N
17	CO_REL_5	Character	6		N
18	CO_NAME_6	Character	12		N
19	CO_REL_6	Character	6		N
20	CO_NAME_7	Character	12		N
21	CO_REL_7	Character	6		N
22	CO_NAME_8	Character	12		N
23	CO_REL_8	Character	6		N
** Total **			195		

DATA BASE: RBRIGAD1.DBF

Field	Field Name	Type	Width	Dec	Index
1	BDE_NAME	Character	12		N
2	ECHELON	Character	6		N
3	TYPE	Character	6		N
4	BATL_FUNC	Character	6		N
5	ACTIVITY	Character	6		N
6	MISSION	Character	6		N
7	LOCATION	Character	6		N
8	BN_NAME_1	Character	12		N
9	BN_REL_1	Character	6		N
10	BN_NAME_2	Character	12		N
11	BN_REL_2	Character	6		N
12	BN_NAME_3	Character	12		N
13	BN_REL_3	Character	6		N
14	BN_NAME_4	Character	12		N
15	BN_REL_4	Character	6		N
16	BN_NAME_5	Character	12		N
17	BN_REL_5	Character	6		N
18	BN_NAME_6	Character	12		N
19	BN_REL_6	Character	6		N
20	BN_NAME_7	Character	12		N
21	BN_REL_7	Character	6		N
22	BN_NAME_8	Character	12		N
23	BN_REL_8	Character	6		N
24	BN_NAME_9	Character	12		N
25	BN_REL_9	Character	6		N
26	BN_NAME_10	Character	12		N
27	BN_REL_10	Character	6		N
28	BN_NAME_11	Character	12		N
29	BN_REL_11	Character	6		N
30	BN_NAME_12	Character	12		N
31	BN_REL_12	Character	6		N

** Total ** 265

DATA BASE: RCOMPNY1.DBF

Field	Field Name	Type	Width	Dec	Index
1	CO_NAME	Character	12		N
2	BASE_NAME	Character	12		N
3	ECHELON	Character	6		N
4	TYPE	Character	6		N
5	BATL_FUNC	Character	6		N
6	ACTIVITY	Character	6		N
7	MISSION	Character	6		N
8	LOCATION	Character	8		N
9	OFFICER	Numeric	3		N
10	ENLISTED	Numeric	3		N
11	EQ_NAME_1	Character	8		N
12	EQ_QTY_1	Numeric	3		N
13	EQ_NAME_2	Character	8		N
14	EQ_QTY_2	Numeric	3		N
15	EQ_NAME_3	Character	8		N
16	EQ_QTY_3	Numeric	3		N
17	EQ_NAME_4	Character	8		N
18	EQ_QTY_4	Numeric	3		N
19	EQ_NAME_5	Character	8		N
20	EQ_QTY_5	Numeric	3		N
21	EQ_NAME_6	Character	8		N
22	EQ_QTY_6	Numeric	3		N
23	EQ_NAME_7	Character	8		N
24	EQ_QTY_7	Numeric	3		N
25	EQ_NAME_8	Character	8		N
26	EQ_QTY_8	Numeric	3		N
27	EQ_NAME_9	Character	8		N
28	EQ_QTY_9	Numeric	3		N
29	EQ_NAME_10	Character	8		N
30	EQ_QTY_10	Numeric	3		N
31	EQ_NAME_11	Character	8		N
32	EQ_QTY_11	Numeric	3		N
33	EQ_NAME_12	Character	8		N
34	EQ_QTY_12	Numeric	3		N
** Total **			201		

DATA BASE: REF_RQST.DBF

Field	Field Name	Type	Width	Dec	Index
1	SEQ_NO	Character	5		N
2	DATE	Date	8		N
3	TIME	Character	6		N
4	WINDOW	Character	7		N
5	STACK	Numeric	1		N
6	FUNC_AREA	Character	20		N
7	DATA_CAT	Character	20		N
8	DATA_ELE	Character	20		N
9	DATA_LVL	Character	1		N
** Total **			89		

DATA BASE: REF_XREF.DBF

Field	Field Name	Type	Width	Dec	Index
1	PROD	Character	4		Y
2	FUNC_AREA	Character	20		N
3	DATA_CAT	Character	20		N
4	DATA_ELE	Character	20		N
5	DATA_LVL	Character	1		N
** Total **			66		

DATA BASE: RUNXREF.DBF

Field	Field Name	Type	Width	Dec	Index
1	UNIT_ID	Numeric	3		Y
2	NAME	Character	15		N
** Total **			19		

DATA BASE: SCCNOP.DBF

Field	Field Name	Type	Width	Dec	Index
1	SEQ_NO	Character	5		Y
2	C1A	Character	1		N
3	C1B	Character	1		N
4	C1C	Character	1		N
5	C1D	Character	1		N
6	C1E	Character	1		N
7	C1F	Character	1		N
8	C1G	Character	1		N
9	C1H	Character	1		N
10	C1I	Character	1		N
11	C1J	Character	1		N
12	C1K	Character	1		N
13	C2A	Character	1		N
14	C2B	Character	1		N
15	C2C	Character	1		N
16	C2D	Character	1		N
17	C2E	Character	1		N
** Total **			22		

DATA BASE: SCCRTEVT.DBF

Field	Field Name	Type	Width	Dec	Index
1	SEQ_NO	Character	5		N
2	COA	Numeric	1		N
3	CE	Character	7		N
4	MATCH	Character	1		N
5	BALANCE	Character	1		N
** Total **			16		

DATA BASE: SCFACTS.DBF

Field	Field Name	Type	Width	Dec	Index
1	SEQ_NO	Character	5		Y
2	A1	Character	1		N
3	B1	Character	1		N
4	C1	Character	1		N
5	A2	Character	1		N
6	B2	Character	1		N
7	C2	Character	1		N
8	D2	Character	1		N
9	E2	Character	1		N
10	F2	Character	1		N
11	A3	Character	1		N
12	B3	Character	1		N
13	C3	Character	1		N
14	D3	Character	1		N
15	E3	Character	1		N
16	F3	Character	1		N
17	G3	Character	1		N
18	A4	Character	1		N
19	B4	Character	1		N
20	C4	Character	1		N
21	D4	Character	1		N
22	E4	Character	1		N
23	F4	Character	1		N
24	A5	Character	1		N
25	B5	Character	1		N
** Total **			30		

DATA BASE: SCFORCE.DBF

Field	Field Name	Type	Width	Dec	Index
1	SEQ_NO	Character	5		N
2	COA	Numeric	1		N
3	UNIT	Character	12		N
4	POWER	Numeric	6	2	N
5	MISSION	Character	1		N
** Total **			26		

DATA BASE: SCJUST.DBF

Field	Field Name	Type	Width	Dec	Index
1	SEQ_NO	Character	5		Y
2	MISSION	Character	1		N
3	EN_ECH1	Character	1		N
4	EN_ECH2	Character	1		N
5	EN_EQUIP	Character	1		N
6	AVENUE	Character	1		N
7	RIVER_OBS	Character	1		N
8	CITY	Character	1		N
9	LOC	Character	1		N
10	DIST_OBJ	Character	1		N
11	BRIDGE	Character	1		N
12	RIVER_CRS	Character	1		N

13	FLANKS	Character	1		N
14	REDISPOSE	Character	1		N
15	FR_EQUIP	Character	1		N
16	TIME	Character	1		N
17	COA	Character	1		N
** Total **			22		

DATA BASE: SCPOWER.DBF

Field	Field Name	Type	Width	Dec	Index
1	UNIT	Character	12		Y
2	POWER	Numeric	6	2	N
** Total **			19		

DATA BASE: SITAWARE.DBF

Field	Field Name	Type	Width	Dec	Index
1	SEQ_NO	Character	5		Y
2	Q1	Character	1		N
3	Q2	Character	1		N
4	Q3	Character	1		N
5	Q4	Character	1		N
6	Q5	Character	1		N
7	Q6	Character	1		N
8	Q7	Character	1		N
9	Q8	Character	1		N
10	Q9	Character	1		N
11	Q10	Character	1		N
12	Q11	Character	1		N
13	Q12	Character	1		N
14	Q13	Character	1		N
15	Q14	Character	1		N
16	Q15	Character	1		N
17	Q16	Character	1		N
18	Q17	Character	1		N
19	Q18	Character	1		N
20	Q19	Character	1		N
21	Q20	Character	1		N
22	Q21	Character	1		N
23	Q22	Character	1		N
24	Q23	Character	1		N
25	Q24	Character	1		N
26	Q25	Character	1		N
27	Q26	Character	1		N
28	Q27	Character	1		N
29	Q28	Character	1		N
30	Q29	Character	1		N
31	Q30	Character	1		N
32	Q31	Character	1		N
33	Q32	Character	1		N
** Total **			38		

DATA BASE: SITCMDEL.DBF

Field	Field Name	Type	Width	Dec	Index
1	SEQ_NO	Character	5		N
2	DATE	Date	8		N
3	TIME	Character	6		N
4	WINDOW	Character	7		N
5	STACK	Numeric	1		N
6	SIT_DATE	Date	8		N
7	SIT_TIME	Character	6		N
8	OPLAN_NO	Numeric	2		N
9	CM_ID	Numeric	3		N
10	OPLAN	Character	20		N
11	NAME	Character	12		N
** Total **			79		

DATA BASE: SITCMLOC.DBF

Field	Field Name	Type	Width	Dec	Index
1	SEQ_NO	Character	5		N
2	DATE	Date	8		N
3	TIME	Character	6		N
4	WINDOW	Character	7		N
5	STACK	Numeric	1		N
6	SIT_DATE	Date	8		N
7	SIT_TIME	Character	6		N
8	OPLAN_NO	Numeric	2		N
9	CM_ID	Numeric	3		N
10	LOCX1	Numeric	6		N
11	LOCY1	Numeric	6		N
12	LOCX2	Numeric	6		N
13	LOCY2	Numeric	6		N
14	LOCX3	Numeric	6		N
15	LOCY3	Numeric	6		N
16	LOCX4	Numeric	6		N
17	LOCY4	Numeric	6		N
18	LOCX5	Numeric	6		N
19	LOCY5	Numeric	6		N
20	LOCX6	Numeric	6		N
21	LOCY6	Numeric	6		N
22	LOCX7	Numeric	6		N
23	LOXY7	Numeric	6		N
24	LOCX8	Numeric	6		N
25	LOCY8	Numeric	6		N
26	LOCX9	Numeric	6		N
27	LOCY9	Numeric	6		N
28	LOCX10	Numeric	6		N
29	LOCY10	Numeric	6		N
30	LOCX11	Numeric	6		N
31	LOCY11	Numeric	6		N
32	LOCX12	Numeric	6		N
33	LOCY12	Numeric	6		N
34	LOCX13	Numeric	6		N
35	LOCY13	Numeric	6		N
36	LOCX14	Numeric	6		N
37	LOCY14	Numeric	6		N

38	LOCX15	Numeric	6	N
39	LOCY15	Numeric	6	N
40	OPLAN	Character	20	N
41	NAME	Character	12	N
** Total **			259	

DATA BASE: SITNEWCM.DBF

Field	Field Name	Type	Width	Dec	Index
1	SEQ_NO	Character	5		N
2	DATE	Date	8		N
3	TIME	Character	6		N
4	WINDOW	Character	7		N
5	STACK	Numeric	1		N
6	SIT_DATE	Date	8		N
7	SIT_TIME	Character	6		N
8	OPLAN_NO	Numeric	2		N
9	CM_ID	Numeric	3		N
10	NAME	Character	12		N
11	FORCE	Character	7		N
12	TYPE	Character	6		N
13	SCAL1_40	Logical	1		N
14	SCAL1_80	Logical	1		N
15	SCAL1_160	Logical	1		N
16	SCAL1_400	Logical	1		N
17	SCAL1_800	Logical	1		N
18	STATUS	Character	12		N
19	EFF2DATE	Date	8		N
20	EFF2TIME	Character	6		N
21	ECHELON	Character	6		N
22	LOCX1	Numeric	6		N
23	LOCY1	Numeric	6		N
24	LOCX2	Numeric	6		N
25	LOCY2	Numeric	6		N
26	LOCX3	Numeric	6		N
27	LOCY3	Numeric	6		N
28	LOCX4	Numeric	6		N
29	LOCY4	Numeric	6		N
30	LOCX5	Numeric	6		N
31	LOCY5	Numeric	6		N
32	LOCX6	Numeric	6		N
33	LOCY6	Numeric	6		N
34	LOCX7	Numeric	6		N
35	LOXY7	Numeric	6		N
36	LOCX8	Numeric	6		N
37	LOCY8	Numeric	6		N
38	LOCX9	Numeric	6		N
39	LOCY9	Numeric	6		N
40	LOCX10	Numeric	6		N
41	LOCY10	Numeric	6		N
42	LOCX11	Numeric	6		N
43	LOCY11	Numeric	6		N
44	LOCX12	Numeric	6		N
45	LOCY12	Numeric	6		N
46	LOCX13	Numeric	6		N

47	LOCY13	Numeric	6	N
48	LOCX14	Numeric	6	N
49	LOCY14	Numeric	6	N
50	LOCX15	Numeric	6	N
51	LOCY15	Numeric	6	N
52	OPLAN	Character	20	N
** Total **			309	

DATA BASE: SITRQST.DBF

Field	Field Name	Type	Width	Dec	Index
1	SEQ_NO	Character	5		N
2	DATE	Date	8		N
3	TIME	Character	6		N
4	WINDOW	Character	7		N
5	STACK	Numeric	1		N
6	SIT_DATE	Date	8		N
7	SIT_TIME	Character	6		N
8	OPLAN_NO	Numeric	2		N
9	FORCE	Character	7		N
10	UNIT_NO	Numeric	3		N
11	MESSAGE	Character	20		N
12	OPLAN	Character	20		N
13	UNIT	Character	15		N
** Total **			109		

DATA BASE: SITTASKO.DBF

Field	Field Name	Type	Width	Dec	Index
1	SEQ_NO	Character	5		N
2	DATE	Date	8		N
3	TIME	Character	6		N
4	WINDOW	Character	7		N
5	STACK	Numeric	1		N
6	SIT_DATE	Date	8		N
7	SIT_TIME	Character	6		N
8	OPLAN_NO	Numeric	2		N
9	FORCE	Character	7		N
10	UNIT_NO	Numeric	3		N
11	HI_ECH_NO	Numeric	3		N
12	RELATE	Character	4		N
13	OPLAN	Character	20		N
14	UNIT	Character	15		N
15	HIGH_ECH	Character	15		N
** Total **			111		

DATA BASE: SITULOC.DBF

Field	Field Name	Type	Width	Dec	Index
1	SEQ_NO	Character	5		N
2	DATE	Date	8		N
3	TIME	Character	6		N
4	WINDOW	Character	7		N

5	STACK	Numeric	1	N
6	SIT_DATE	Date	8	N
7	SIT_TIME	Character	6	N
8	OPLAN_NO	Numeric	2	N
9	FORCE	Character	7	N
10	UNIT_NO	Numeric	3	N
11	X_LOC	Numeric	6	N
12	Y_LOC	Numeric	6	N
13	OPLAN	Character	20	N
14	UNIT	Character	15	N
** Total **			101	

DATA BASE: TASKEVAL.DBF

Field	Field Name	Type	Width	Dec	Index
1	SEQ_NO	Character	5		Y
2	STPRSTR	Numeric	1		N
3	STPRLSGN	Numeric	1		N
4	STPROTHER	Numeric	1		N
5	STPREST	Numeric	1		N
6	STOPCURR	Numeric	1		N
7	STOPOVLY	Numeric	1		N
8	STOPTSKO	Numeric	1		N
9	STOPFRAG	Numeric	1		N
10	STOPGUID	Numeric	1		N
11	STINCOMP	Numeric	1		N
12	STINCOMM	Numeric	1		N
13	STINREIN	Numeric	1		N
14	STINARTY	Numeric	1		N
15	STINEST	Numeric	1		N
16	STINRPT	Numeric	1		N
17	STINWHST	Numeric	1		N
18	STINWFOR	Numeric	1		N
19	STLGCIII	Numeric	1		N
20	STLGCV	Numeric	1		N
21	STLGEQP	Numeric	1		N
22	STLGEST	Numeric	1		N
23	RFPRSTR	Numeric	1		N
24	RFPRLOSS	Numeric	1		N
25	RFPRPO[illegible]	Numeric	1		N
26	RFOPORG	Numeric	1		N
27	RFOPEQP	Numeric	1		N
28	RFOPECHR	Numeric	1		N
29	RFOPMIN[illegible]	Numeric	1		N
30	RFINCOMP	Numeric	1		N
31	RFINSTR	Numeric	1		N
32	RFINEQP	Numeric	1		N
33	RFINECHR	Numeric	1		N
34	RFLGSPLY	Numeric	1		N
35	RFLGTRNS	Numeric	1		N
36	DIFF2	Numeric	1		N
37	DIFF4	Numeric	1		N
38	DIFF5	Numeric	1		N
39	DIFF6	Numeric	1		N
40	DIFF7	Numeric	1		N

41	DIFF8	Numeric	1	N
42	DIFF9	Numeric	1	N
43	CNCLP8	Logical	1	N
44	CNCLAGR	Logical	1	N
45	CNCL2	Logical	1	N
46	CNCL4	Logical	1	N
47	CNCL5	Logical	1	N
48	CNCL6	Logical	1	N
49	CNCL7	Logical	1	N
50	SCALE	Character	1	N
51	CFFPRS	Numeric	1	N
52	CFFEQP	Numeric	1	N
53	CFEPRS	Numeric	1	N
54	CFEEQP	Numeric	1	N
55	CFPOL	Numeric	1	N
56	CFAMMO	Numeric	1	N
57	CFFEBA	Numeric	1	N
58	CFBDUR	Numeric	1	N
59	VALRANG	Logical	1	N
** Total **			64	

DATA BASE: TEAMPRF.DBF

Field	Field Name	Type	Width	Dec	Index
1	SEQ_NO	Character	5		Y
2	ORG	Logical	1		N
3	MINS	Numeric	3		N
4	FOLORG	Logical	1		N
5	REFORG	Logical	1		N
6	IMORG	Logical	1		N
7	MANTIME	Logical	1		N
8	DISTRACT	Logical	1		N
9	GETBACK	Logical	1		N
10	EQSTS	Logical	1		N
11	LSUB	Logical	1		N
12	SUBA	Logical	1		N
13	LADQ	Logical	1		N
14	ROLCON	Logical	1		N
15	ADQKNOW	Logical	1		N
16	DOMPER	Logical	1		N
17	TEAMCON	Logical	1		N
18	GT	Logical	1		N
19	CAPRES	Logical	1		N
** Total **			26		

DATA BASE: TIMELINE.DBF

Field	Field Name	Type	Width	Dec	Index
1	SEQ_NO	Character	5		Y
2	STEP1	Numeric	3		N
3	STEP2	Numeric	3		N
4	STEP3	Numeric	3		N
5	STEP4	Numeric	3		N
6	STEP5	Numeric	3		N

7	STEP6	Numeric	3		N
8	STEP7	Numeric	3		N
9	STEP8	Numeric	3		N
10	STEP9	Numeric	3		N
11	STEP10	Numeric	3		N
** Total **			36		

DATA BASE: VERTASK.DBF

Field	Field Name	Type	Width	Dec	Index
1	VFY_BDE	Character	12		N
2	VFY_BN	Character	12		N
3	VFY_CO	Character	12		N
4	VFY_BASE	Character	12		N
5	VFY_MESS	Character	10		N
** Total **			59		

DATA BASE: WINDOW.DBF

Field	Field Name	Type	Width	Dec	Index
1	SEQ_NO	Character	5		N
2	DATE	Date	8		N
3	TIME	Character	6		Y
4	WINDOW	Character	7		N
5	STACK	Numeric	1		N
6	ACTION	Character	1		N
** Total **			29		

DATA BASE: WORKASMT.DBF

Field	Field Name	Type	Width	Dec	Index
1	SEQ_NO	Character	5		Y
** Total **			6		

APPENDIX F - EDDIC ENVIRONMENT VARIABLES

This appendix describes the Unix environment variables used in the EDDIC system.

Environment Variable	Description
BLUE_ASSET_UNIT	Data base BLUEFOR_ASSET_UNIT.
BLUEFOR_AMMO_AUTH	Data base BLUEFOR_AUTH_AMMO.
BLUEFOR_AMMO_AUTH_NDX	Data base BLUEFOR_AUTH_AMMO_INDEX.
BLUEFOR_AMMO_CURR	Data base BLUEFOR_CURR_AMMO.
BLUEFOR_AMMO_CURR_NDX	Data base BLUEFOR_CURR_AMMO_INDEX.
BLUEFOR_AMMO_TRACK	Data base BLUEFOR_AMMO_TRACK.
BLUEFOR_EQUIP_AUTH	Data base BLUEFOR_AUTH_EQUIP.
BLUEFOR_EQUIP_AUTH_NDX	Data base BLUEFOR_AUTH_EQUIP_INDEX.
BLUEFOR_EQUIP_CURR	Data base BLUEFOR_CURR_EQUIP.
BLUEFOR_EQUIP_CURR_NDX	Data base BLUEFOR_CURR_EQUIP_INDEX.
BLUEFOR_EQUIP_TRACK	Data base BLUEFOR_EQUIP_TRACK.
BLUEFOR_FUEL	Data base BLUEFOR_FUEL.
BLUEFOR_FUEL_NDX	Data base BLUEFOR_FUEL_INDEX.
BLUEFOR_LOCATION	Data base BLUEFOR_UNIT_LOC.
BLUEFOR_LOCATION_NDX	Data base BLUEFOR_UNIT_LOC_INDEX.
BLUEFOR_ORGANIC_UNIT	Data base BLUEFOR_ORGANIC_TASK_ORG.
BLUEFOR_PERS	Data base BLUEFOR_PERSONNEL.
BLUEFOR_PERS_NDX	Data base BLUEFOR_PERSONNEL_INDEX.
BLUEFOR_TOP_UNIT	Data base TASK_ORG_TOP_UNIT_MENU.
BLUEFOR_UNIT_CONVERSION	Data base BLUEFOR_UNIT_CONVERT.
BLUEFOR_UNIT_STATUS	Data base BLUEFOR_UNIT_STATUS.
BLUEFOR_UNIT_STATUS_NDX	Data base BLUEFOR_UNIT_STATUS_INDEX.
BLUEFOR_UNIT_XREF	Data base BLUEFOR_UNIT_NAME.

BUILD_BLUE_AMMO	Logical flag to indicate if the BLUEFOR ammunition data base should be built as part of the situation data base build.
BUILD_BLUE_EQUIP	Logical flag to indicate if the BLUEFOR equipment data base should be built as part of the situation data base build.
BUILD_BLUE_FUEL	Logical flag to indicate if the BLUEFOR fuel data base should be built as part of the situation data base build.
BUILD_BLUE_PERS	Logical flag to indicate if the BLUEFOR personnel data base should be built as part of the situation data base build.
BUILD_BLUE_STATUS	Logical flag to indicate if the BLUEFOR unit status data base should be built as part of the situation data base build.
BUILD_BLUE_ULOC	Logical flag to indicate if the BLUEFOR unit location data base should be built as part of the situation data base build.
BUILD_C2_MAP_MENU	Data base MAP_BUILD_MENU.
BUILD_CNTRL_MSR	Logical flag to indicate if the control measure data base should be built as part of the situation data base build.
BUILD_EX	Data base G3_BUILD_MENU.
BUILD_G2	Data base G2_BUILD_MENU.
BUILD_G3	Data base G3_BUILD_MENU.
BUILD_G4	Data base G4_BUILD_MENU.
BUILD_OBSTACLE	Logical flag to indicate if the obstacle data base should be built as part of the situation data base build.
BUILD_ONE	Data base G2_BUILD_MENU.
BUILD_OPFOR_EQUIP	Logical flag to indicate if the OPFOR equipment data base should be built as part of the situation data base build.
BUILD_OPFOR_REINF	Logical flag to indicate if the OPFOR reinforcing time data base should be built as part of the situation data base build.
BUILD_OPFOR_STATUS	Logical flag to indicate if the OPFOR unit status data base should be built as part of the situation data base build.

BUILD_OPFOR_ULOC	Logical flag to indicate if the OPFOR unit location data base should be built as part of the situation data base build.
BUILD_OPPLAN	Logical flag to indicate if the OPLAN data base should be built as part of the situation data base build.
BUILD_THREE	Data base G4_BUILD_MENU.
BUILD_TWO	Data base G3_BUILD_MENU.
C2_NEW_PROD	data base TRAN_NEW_C2.
C2_PRODUCT_RECORD_DB	Data base C2_PRODUCT_RECORD.
C2_PRODUCT_ROUTER_HOST	Name of the computer where the C2 product router is running.
C2_PRODUCT_ROUTER_SERV	Entry in the services file that is reserved for the C2 product router.
C2_RECORD	Data base C2_PRODUCT_RECORD.
C2_REQUEST	Data base TRAN_C2_REQUEST.
C2_WINDOW	Data base TRAN_C2_WINDOW.
C2LAB_BLUE_TASK_ORG	Data base BLUEFOR_TASK_ORG_SOURCE.
C2LAB_BLUEFOR_AMMO	Data base BLUEFOR_AMMO_SOURCE.
C2LAB_BLUEFOR_EQUIP	Data base BLUEFOR_EQUIP_SOURCE.
C2LAB_BLUEFOR_FUEL	Data base BLUEFOR_FUEL_SOURCE.
C2LAB_BLUEFOR_LOCATION	Data base BLUEFOR_UNIT_LOC_SOURCE.
C2LAB_BLUEFOR_PERS	Data base BLUEFOR_PERSONNEL_SOURCE.
C2LAB_CONTROL_MEASURE	Data base CONTROL_MEASURE_SOURCE.
C2LAB_DB	Data base C2_PRODUCT_SOURCE.
C2LAB_DB	Data base REFERENCE_SOURCE.
C2LAB_OBSTACLE	Data base OBSTACLE_SOURCE.
C2LAB_OPFOR_EQUIP	Data base OPFOR_EQUIP_SOURCE.
C2LAB_OPFOR_LOCATION	Data base OPFOR_UNIT_LOC_SOURCE.
C2LAB_OPFOR_REINFORCE	Data base OPFOR_REINFORCE_TIME.
C2LAB_OPFOR_STATUS	Data base OPFOR_UNIT_STATUS_SOURCE.

C2LAB_OPFOR_TASK_ORG	Data base OPFOR_TASK_ORG_SOURCE.
CDB_HEADER_DB	Data base C2_PRODUCT_HEADER.
CDB_PARTICIPANT_DB	Data base SEND_PARTICIPANT_SOURCE.
CDB_PROD_DESC_DB	Data base C2_PRODUCT_DESC.
CDB_PRODUCT_DB	Data base C2_PRODUCT.
CHARACTER_FONT_FILE	Name of the font file to use for text on the tactical map, task organization tool, and graphical status report.
CNTRL_MSR_POINT_NDX	Data base CNTRL_MSR_POINT_INDEX.
CNTRL_MSR_XREF	Data base CONTROL_MEASURE_NAME.
CNTRL_POINT_XREF	Data base CNTRL_MSR_POINT_NAME.
CONTOUR_DESCRIPTION_FILE	Data base CONTOUR_DESC.
CONTROL_DB	Data base EXP_CONTROL_SOURCE.
CONTROL_DISPLAY_MANAGER	Path and file name of the experiment control display manager executable.
CONTROL_MEASURE	Data base CONTROL_MEASURE.
CONTROL_MEASURE_POINT	Data base CNTRL_MSR_POINT.
CONTROL_MEASURE_NDX	Data base CONTROL_MEASURE_INDEX.
CONTROL_MENU	Data base EXP_CONTROL_MENU.
CONTROL_RECORD	Data base EXP_CONTROL_RECORD.
CONTROL_RECORD_DB	Data base EXP_CONTROL_RECORD.
CONTROL_REQUEST	Data base TRAN_CONTROL_REQUEST.
CONTROL_ROUTER_HOST	Name of the computer where the experiment control router is running.
CONTROL_ROUTER_SERV	Entry in the services file that is reserved for the experiment control router.
CONTROL_WINDOW	Data base TRAN_CONTROL_WINDOW.
CTL_PARTICIPANT_DB	Data base EXP_CONTROL_PARTICIPANT.
CTL_PROD_DESC_DB	Data base EXP_CONTROL_PROD_DESC.
CTL_PRODUCT_DB	Data base EXP_CONTROL_PRODUCT.

DB_MANAGER	Name of the C2 product data base manager. Used for printing the C2 products (C2_DB_MANAGER).
DB_MANAGER	Name of the reference data base manager. Used for printing the reference products (REFERENCE_DB_MANAGER).
EDDIC_STATION_USER	Identification of the user of the workstation. Legal values are g2_plans, g3_plans, g4_plans, and experimenter.
ELEV_DESCRIPTION_FILE	Data base ELEVATION_DESC_1TO400.
FDB_HEADER_DB	Data base REFERENCE_HEADER.
FDB_PROD_DESC_DB	Data base REFERENCE_PROD_DESC.
FDB_PRODUCT_DB	Data base REFERENCE_PRODUCT.
FORM_TOOL_FILE	Data base FORM_DESCRIPTION.
HDB_HELP_DESC_DB	Data base HELP_PROD_DESC.
HDB_HELP_TEXT_DB	Data base HELP_PRODUCT.
HEADER_DB	Name of the C2 product report header data base. Used for building the C2 products (C2_PRODUCT_HEADER).
HEADER_DB	Name of the reference report header data base. Used for building the reference products (REFERENCE_HEADER).
HELP_MENU	Data base HELP_MENU.
HELP_MENU_FILE	Data base HELP_MENU.
HELP_SOURCE	Data base HELP_SOURCE.
HILITE_DESCRIPTION_FILE	Data base LUT_HILITE_DESC.
ICON_PATH	Data base ICON_STACK_DB.
LASER_SERVER	Name of the laserwriter print server computer. Used for workstation screen dumps.
LUT_UPDATE	Data base TRAN_LOOKUP_TABLE.
MAP_DESCRIPTION_FILE	Data base MAP_DESC.
MAP_LEGEND	Data base MAP_LEGEND.
MAP_STATUS	Data base TRAN_MAP.

MESSAGE_CREATED_BY_USER	Logical flag to indicate if a message window is created by the user or by the system. If the window is created by the user (= true), the window starts in the open position. If the window is created by the system (= false), the window starts as an icon.
MESSAGE_DISPLAY_MANAGER	Path and file name of the message window display manager executable.
MESSAGE_LOG_DB	Data base MESSAGE_LOG.
MESSAGE_MAP_MENU	Data base MAP_MESSAGE_MENU.
OBSTACLE	Data base OBSTACLE.
OBSTACLE_NDX	Data base OBSTACLE_INDEX.
OBSTACLE_XREF	Data base OBSTACLE_NAME.
OPFOR_EQUIP_AUTH	Data base OPFOR_AUTH_EQUIP.
OPFOR_EQUIP_AUTH_NDX	Data base OPFOR_AUTH_EQUIP_INDEX.
OPFOR_EQUIP_CURR	Data base OPFOR_CURR_EQUIP.
OPFOR_EQUIP_CURR_NDX	Data base OPFOR_CURR_EQUIP_INDEX.
OPFOR_EQUIP_LIST	Data base OPFOR_EQUIP_NAME.
OPFOR_LOCATION	Data base OPFOR_UNIT_LOC.
OPFOR_LOCATION_NDX	Data base OPFOR_UNIT_LOC_INDEX.
OPFOR_ORGANIC_UNIT	Data base OPFOR_ORGANIC_TASK_ORG.
OPFOR_UNIT_CONVERSION	Data base OPFOR_UNIT_CONVERT.
OPFOR_UNIT_STATUS	Data base OPFOR_UNIT_STATUS.
OPFOR_UNIT_STATUS_NDX	Data base OPFOR_UNIT_STATUS_INDEX.
OPFOR_UNIT_XREF	Data base OPFOR_UNIT_NAME.
OPPLAN_DB	Data base OPLAN_LIST.
OPPLAN_SOURCE	Data base OPLAN_LIST_SOURCE.
OVERLAY_LOOKUP_TABLE	Data base LUT_OVERLAY.
PRODUCT_DB	Name of the experiment control data base. Used for building the experiment control data base (EXP_CONTROL_PRODUCT).

PRODUCT_DB	Name of the help data base. Used for building the help data base (HELP_PRODUCT).
PRODUCT_DB	Name of the C2 product data base. Used for building the C2 product data base (C2_PRODUCT).
PRODUCT_DB	Name of the reference data base. Used for building the reference data base (REFERENCE_PRODUCT).
PRODUCT_DESC_DB	Name of the experiment control description data base. Used for building the experiment control data base (EXP_CONTROL_PROD_DESC).
PRODUCT_DESC_DB	Name of the help description data base. Used for building the help data base (HELP_PROD_DESC).
PRODUCT_DESC_DB	Name of the C2 product description data base. Used for building the C2 product data base (C2_PROD_DESC).
PRODUCT_DESC_DB	Name of the reference description data base. Used for building the reference data base (REFERENCE_PROD_DESC).
PRODUCT_XREF	Name of experiment control product name data base. Built as part of the experiment control data base build process (EXP_CONTROL_NAME).
PRODUCT_XREF	Name of the help product name database. Built as part of the help data base build process (HELP_NAME).
PRODUCT_XREF	Name of the C2 product name data base. Built as part of the C2 product data base build process (C2_PRODUCT_NAME).
PRODUCT_XREF	Name of the reference name data base. Built as part of the reference data base build process (REFERENCE_NAME).
RECORD_MAP_INTERACTION	Logical flag to indicate if the interactions with the tactical map should be recorded.
RECORD_SESSION	Logical flag to indicate if an EDDIC session should be recorded.
REF_RECORD	Data base Reference record.
REF_REQUEST	Data base TRAN_REF_REQUEST.
REF_VIEW_EX	Data base G3_REFERENCE_MENU.

REF_VIEW_ONE	Data base G2_REFERENCE_MENU.
REF_VIEW_THREE	Data base G4_REFERENCE_MENU.
REF_VIEW_TWO	Data base G3_REFERENCE_MENU.
REF_WINDOW	Data base TRAN_REF_WINDOW.
REFERENCE_RECORD_DB	Data base REFERENCE_RECORD.
REFERENCE_ROUTER_HOST	Name of the computer where the reference router is running.
REFERENCE_ROUTER_SERV	Entry in the services file that is reserved for the reference router.
REPORT_OUTPUT	Name of the file to print the C2 product reports to (PRODUCT_HARDCOPY).
REPORT_OUTPUT	Name of the file to print the reference product reports to (PRODUCT_HARDCOPY).
ROOT_MENU	Data base ROOT_WINDOW_MENU.
ROUTER_HOST	Name of the computer where the C2 product router is running. Used for printing the C2 products.
ROUTER_HOST	Name of the computer where the reference router is running. Used for printing the reference products.
ROUTER_SERV	Entry in the services file that is reserved for the C2 product router. Used for printing the C2 products.
ROUTER_SERV	Entry in the services file that is reserved for the reference router. Used for printing the reference products.
SIT_ACTIVITY	Data base TRAN_ACTIVITY.
SIT_AMMO	Data base TRAN_AMMUNITION.
SIT_BLUE_TASK_ORG	Data base TRAN_BLUEFOR_TASK_ORG.
SIT_CNTRL_MSR_DELETE	Data base TRAN_CNTRL_MSR_DEL.
SIT_CNTRL_MSR_EFFECT	Data base TRAN_CNTRL_MSR_EFF_TIME.
SIT_CNTRL_MSR_LOCATE	Data base TRAN_CNTRL_MSR_LOC.
SIT_CNTRL_MSR_STATUS	Data base TRAN_CNTRL_MSR_STAT.
SIT_EQUIP	Data base TRAN_EQUIPMENT.

SIT_FUEL	Data base TRAN_FUEL.
SIT_MISSION	Data base TRAN_UNIT_MISSION.
SIT_NEW_CNTRL_MSR	Data base TRAN_NEW_CNTRL_MSR.
SIT_NEW_OBSTACLE	Data base TRAN_NEW_OBSTACLE.
SIT_OBSTACLE_DELETE	Data base TRAN_OBSTACLE_DEL.
SIT_OBSTACLE_EFFECT	Data base TRAN_OBSTACLE_EFF_TIME.
SIT_OBSTACLE_LOCATE	Data base TRAN_OBSTACLE_LOC.
SIT_OBSTACLE_STATUS	Data base TRAN_OBSTACLE_STAT.
SIT_OPFOR_TASK_ORG	Data base TRAN_OPFOR_TASK_ORG.
SIT_PERS SIT_RECORD	Data base TRAN_PERSONNEL.
SIT_RECORD	Data base SITUATION_RECORD.
SIT_REINF	Data base TRAN_OPFOR_REINFORCE.
SIT_REQUEST	Data base TRAN_SITUATION_REQUEST.
SIT_STRENGTH	Data base TRAN_OPFOR_STRENGTH.
SIT_UNIT_LOC	Data base TRAN_UNIT_LOCATION.
SIT_WINDOW	Data base TRAN_SITUATION_WINDOW.
SITUATION_RECORD_DB	Data base SITUATION_RECORD.
SITUATION_ROUTER_HOST	Name of the computer where the situation data router is running.
SITUATION_ROUTER_SERV	Entry in the services file that is reserved for the situation data router.
SPOOL_PATH	Path name to use as a repository for screendump bitmap image files.
START_DATE	Experiment start time. (format: HHMM/DD/MM/YY).
SYMBOL_FONT_FILE	Name of the font file to use for displaying unit symbols on the tactical map and task organization tool.
TOOLS	Data base TOOL_MENU.
TOP_UNIT_MENU	Data base TASK_ORG_TOP_UNIT_MENU.
UNHILITE_DESCRIPTION_FILE	Data base LUT_UNHILITE_DESC.

UNIT_MENU	Data base TASK_ORG_UNIT_MENU.
UNIT_TYPE_BTN_MENU	Data base TASK_ORG_UNIT_TYPE_MENU.
USE_DBASE_BLUE_STATUS	Flag to indicate if the BLUEFOR unit status information is located in the task organization source file.
USE_DBASE_CNTRL_MSR	Flag to indicate if the control measure source file was created from the dBASE scenario manager.
VIEW_C2_MAP_MENU	Data base MAP_VIEW_C2_MENU.
VIEW_EX	Data base G3_VIEW_C2_MENU.
VIEW_G2	Data base to use as input to the C2 product print program. This file is the same format as the view C2 menu data base.
VIEW_G2	Data base G2_VIEW_C2_MENU.
VIEW_G3	Data base G3_VIEW_C2_MENU.
VIEW_G4	Data base G4_VIEW_C2_MENU.
VIEW_MENU	Data base TASK_ORG_TOOL_MENU.
VIEW_ONE	Name of the view C2 menu file to be created by the C2 product build process for the G2 workstation (G2_VIEW_C2_MENU).
VIEW_ONE	Name of the reference menu file to be created by the reference build process for the G2 workstation (G2_REFERENCE_MENU).
VIEW_THREE	Name of the view C2 menu file to be created by the C2 product build process for the G4 workstation (G4_VIEW_C2_MENU).
VIEW_THREE	Name of the reference menu file to be created by the reference build process for the G4 workstation (G4_REFERENCE_MENU).
VIEW_TWO	Name of the view C3 menu file to be created by the C2 product build process for the G3 workstation (G3_VIEW_C2_MENU).
VIEW_TWO	Name of the reference menu file to be created by the reference build process for the G3 workstation (G3_REFERENCE_MENU).