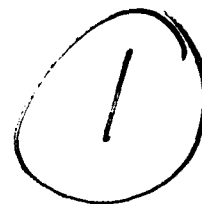
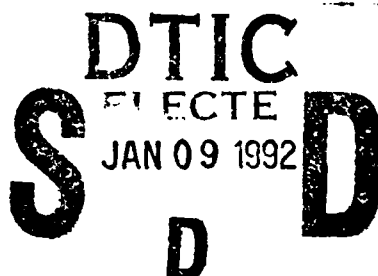


AD-A244 974



**SOFTWARE DESIGN DOCUMENT
SAF Parameter Editor CSCI (7)**

June, 1991



Prepared by:

BBN Systems and Technologies,
A Division of Bolt Beranek and Newman Inc.
10 Moulton Street
Cambridge, MA 02138
(617) 873-3000 FAX: (617) 873-4315

Prepared for:

Defense Advanced Research Projects Agency (DARPA)
Information and Science Technology Office
1400 Wilson Blvd., Arlington, VA 22209-2308
(202) 694-8232, AUTOVON 224-8232

Program Manager for Training Devices (PM TRADE)
12350 Research Parkway
Orlando, FL 32826-3276
(407) 380-4518

92 1 6 054

92-00246



**APPROVED FOR PUBLIC RELEASE
DISTRIBUTION UNLIMITED**

SOFTWARE DESIGN DOCUMENT

SAF Parameter Editor CSCI (7)

June, 1991

Prepared by:

BBN Systems and Technologies,
A Division of Bolt Beranek and Newman Inc.
10 Moulton Street
Cambridge, MA 02138
(617) 873-3000 FAX: (617) 873-4315



Prepared for:

Defense Advanced Research Projects Agency (DARPA)
Information and Science Technology Office
1400 Wilson Blvd., Arlington, VA 22209-2308
(202) 694-8232, AUTOVON 224-8232

Program Manager for Training Devices (PM TRADE)
12350 Research Parkway
Orlando, FL 32826-3276
(407) 380-4518

Accession For	
NTIS	CRA&I
DTIC	TAB
Unannounced	
Justification	
By	
Distribution/	
Availability	
Dist	Availability
A-1	Spec

APPROVED FOR PUBLIC RELEASE
DISTRIBUTION UNLIMITED

REPORT DOCUMENTATION PAGE

Form Approved
OPM No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Information and Regulatory Affairs, Office of Management and Budget, Washington, DC 20503.

1. AGENCY USE ONLY (Leave Blank)		2. REPORT DATE June 1991	3. REPORT TYPE AND DATES COVERED Software Design Document
4. TITLE AND SUBTITLE Software Design Document SAF Parameter Editor CSCI (7)			5. FUNDING NUMBERS Contract Numbers: MDA972-89-C-0060 MDA972-89-C-0061
6. AUTHOR(S) Author not specified.			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Bolt Beranek and Newman, Inc. (BBN) Systems and Technologies; Advanced Simulation 10 Moulton Street Cambridge, MA 02138			8. PERFORMING ORGANIZATION REPORT NUMBER Advanced Simulation #: 9110
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Defense Advanced Research Projects Agency (DARPA) 3701 North Fairfax Drive Arlington, VA 22203-1714			10. SPONSORING/MONITORING AGENCY REPORT NUMBER DARPA Report Number: None.
11. SUPPLEMENTARY NOTES None			
12a. DISTRIBUTION/AVAILABILITY STATEMENT Distribution Statement A: Approved for public release; distribution is unlimited.			12b. DISTRIBUTION CODE Distribution Code: A
13. ABSTRACT (Maximum 200 words) A Simulation Network (SIMNET) project Software Design Document that describes the Semi-Automated Forces (SAF) Parameter Editor Computer Software Configuration Item (CSCI number 7) of the SIMNET hardware and software training system for vehicle crew training and operational training.			
14. SUBJECT TERMS SIMNET Software Design Document for the SAF Parameter Editor CSCI (CSCI 7).			15. NUMBER OF PAGES
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT Same as report.

Table of Contents

1	INTRODUCTION: SAF PARAMETER EDITOR CSCI DESCRIPTION	1
1.1	BACKGROUND.....	1
1.2	EXTERNAL COMMUNICATIONS	1
1.3	INTERNAL STRUCTURE.....	2
1.4	CONFIGURATION AND CONFIGURATION MANAGEMENT	3
1.5	TERMINOLOGY AND DOCUMENTATION	4
2	CSC DESCRIPTIONS	4
2.1	MODEL EDITOR CSC.....	4
2.1.1	Parameters, Globals, and Macros.....	5
2.1.1.1	*OLD-SELECTION*.....	5
2.1.1.2	*HULL-GRAPH*	6
2.1.1.3	*STATIONARY-GRAPH*.....	6
2.1.1.4	*MOVING-GRAPH*	6
2.1.1.5	*CATASTROPHIC-GRAPH*	7
2.1.1.6	*MOBILITY-GRAPH*	7
2.1.1.7	*FIREPOWER-GRAPH*.....	7
2.1.1.8	*HULL-POINTS*	8
2.1.1.9	*STATIONARY-POINTS*.....	8
2.1.1.10	*MOVING-POINTS*	9
2.1.1.11	*CATASTROPHIC-POINTS*	9
2.1.1.12	*MOBILITY-POINTS*	10
2.1.1.14	*PART*	10
2.1.1.15	*ASPECT*	11
2.1.1.16	*TABLE-TYPE*	11
2.1.1.17	*MODE*	12
2.1.1.18	*WEAPON-NAME*.....	14
2.1.1.19	*PARAMETER-DISPLAY-PANE*	15
2.1.1.20	*TABULAR-DISPLAY-PANE*	16
2.1.1.21	*MODEL-FRAME*.....	17
2.1.1.22	*DISPLAY-LOGO-PANE*.....	17
2.1.1.23	*TABULAR-LOGO-PANE*	17
2.1.1.24	*HIT-FILE-NAME*	18

2.1.1.25	*DETECTION-FILE-NAME*	18
2.1.1.26	*INDIRECT-FIRE-DAMAGE-FILE-NAME*	19
2.1.1.27	*DIRECT-FIRE-DAMAGE-FILE-NAME*	19
2.1.1.28	*DF-VEHICLE*	20
2.1.1.29	*IF-VEHICLE*	20
2.1.1.30	*IND-FIR-TABLES*	20
2.1.1.31	*DIR-FIR-TABLES*	21
2.1.1.32	*HIT-TABLES*	22
2.1.1.33	*DIRECT-FIRE-DAMAGE-DATA*	22
2.1.1.34	*INDIRECT-FIRE-DAMAGE-DATA*	23
2.1.1.35	TRUNCATE-TO-N-POSITIONS	23
2.1.1.36	GIMME-VAR-NAME-OF-CURRENT-GRAPH	23
2.1.1.37	GET-CURRENT-GRAPH-POINTS	24
2.1.1.38	GIMME-VAR-NAME-OF-CURRENT-POINTS	26
2.1.1.39	GET-CURRENT-LINE-POINTS	26
2.1.2	Graphing code	27
2.1.2.1	*X-AXIS-LENGTH*	27
2.1.2.2	*Y-AXIS-LENGTH*	28
2.1.2.3	*X-ORIGIN-INCREMENT*	28
2.1.2.4	*TOTAL-X-RANGE*	28
2.1.2.5	*TOTAL-Y-RANGE*	29
2.1.2.6	*DELTA-X*	29
2.1.2.7	*DELTA-Y*	30
2.1.2.8	*ABSOLUTE-ORIGIN*	30
2.1.2.9	*ABSOLUTE-X-ORIGIN*	31
2.1.2.10	*ABSOLUTE-Y-ORIGIN*	31
2.1.2.11	DRAW-X-SCALE-BAR	32
2.1.2.12	DRAW-Y-SCALE-BAR	32
2.1.2.13	DRAW-SCALE-BAR	33
2.1.2.14	DRAW-X-TICKS	33
2.1.2.15	DRAW-Y-TICKS	34
2.1.2.16	DRAW-TICKS	35
2.1.2.17	CLEAN-UP-AXIS-AND-REDRAW	35
2.1.2.18	PRESENT-GRAPH	36
2.1.2.19	CALCULATE-POINT-DATA	36

2.1.2.20	CALCULATE-POINT-DATA-FROM-PIXEL-TO-REAL	37
2.1.2.21	POINT	37
2.1.2.22	(METHOD DRAW POINT)	37
2.1.2.23	(METHOD DRAW-WITH-CHECK POINT)	38
2.1.2.24	(METHOD ON-POINT? POINT)	38
2.1.2.25	(METHOD ERASE POINT)	38
2.1.2.26	(METHOD EXPUNGE POINT)	38
2.1.3	Graphics manipulation functions	39
2.1.3.1	DELETE-POINT-IF-THERE	39
2.1.3.2	LAST-CAR	40
2.1.3.3	MAKE-GAP-BETWEEN-POINTS	40
2.1.3.4	DRAW-LINE-BETWEEN-POINTS	40
2.1.3.5	GET-POINT-RIGHT	41
2.1.3.6	GET-POINT-LEFT	42
2.1.3.7	FIND-SURROUNDING-POINTS	42
2.1.3.8	SPLICE-IN-LIST-AFTER	43
2.1.3.9	SPLICE-IN-LIST-AT-POSITION	43
2.1.3.10	UPDATE-POINT-LIST	43
2.1.3.11	RESCALE-POINT-LIST	44
2.1.3.12	ADD-CORRESPONDING-NEW-POINT-IN-POINT-LIST	45
2.1.3.13	RECORD-NEW-POINT	45
2.1.3.14	ADD-NEW-POINT	46
2.1.3.15	(METHOD DRAG POINT)	47
2.1.3.16	(METHOD ERASE-POINT-AND-LINES POINT)	48
2.1.3.17	GET-POINT	49
2.1.3.18	SELECT-AND-DRAG-POINT	49
2.1.3.19	LIST-INSTANCE-VARIABLES	50
2.1.3.20	LIST-INSTANCE-VARIABLE-VALUES	50
2.1.3.21	CREATE-MAKE-INSTANCE-FORM-ARGS	50
2.1.3.22	MAKE-COPY-OF-INSTANCE-POINT	51
2.1.3.23	*GRAPH-BEFORE-LAST-STEP*	51
2.1.3.24	UPDATE-BACKTRACKING-CAPABILITY	52
2.1.3.25	COPY-POINT-LIST	53
2.1.3.26	GO-BACK-TO-PREVIOUS-STEP	53

2.1.3.27	REVERT-TO-FACTORY-VERSION	54
2.1.3.28	EXTRACT-FILE-NAME	54
2.1.3.29	GRAPH-UNDO.....	55
2.1.4	Frame and screen configurations.....	56
2.1.4.1	MODEL-MENU.....	56
2.1.4.2	(METHOD MAKE-INSTANCE MODEL-MENU AFTER)	58
2.1.4.3	*MODEL-IO*	58
2.1.4.4	*HITMODELS-DICTIONARY*.....	58
2.1.4.5	(METHOD TOP-LEVEL MODEL-MENU).....	59
2.1.4.6	MAKE-DOCUMENTATION.....	59
2.1.5	Command processor commands	60
2.1.5.1	(COM-SELECT-MODEL MENU- ACCELERATOR Select Model MENU-LEVEL COM)	60
2.1.5.2	(COM-READ-TABLE MENU-ACCELERATOR Read Table MENU-LEVEL COM).....	60
2.1.5.3	(COM-WRITE-TABLE MENU-ACCELERATOR Write Table MENU-LEVEL COM)	60
2.1.5.4	(COM-UNDO MENU-ACCELERATOR Undo MENU-LEVEL COM).....	61
2.1.5.5	(COM-SELECT-MODEL-DIR MENU- ACCELERATOR Select Model MENU-LEVEL COM-DIR)	61
2.1.5.6	(COM-SELECT-ASPECT-DIR MENU- ACCELERATOR Select Aspect MENU-LEVEL COM-DIR)	61
2.1.5.7	(COM-READ-TABLE-DIR MENU- ACCELERATOR Read Table MENU-LEVEL COM-DIR)	61
2.1.5.8	(COM-WRITE-TABLE-DIR MENU- ACCELERATOR Write Table MENU-LEVEL COM-DIR)	62
2.1.5.9	(COM-UNDO-DIR MENU-ACCELERATOR Undo MENU-LEVEL COM-DIR)	62
2.1.5.10	(COM-SELECT-MODEL-IND MENU- ACCELERATOR Select Model MENU-LEVEL COM-IND).....	62
2.1.5.11	(COM-READ-TABLE-IND MENU- ACCELERATOR Read Table MENU-LEVEL COM-IND).....	62

2.1.5.12	(COM-WRITE-TABLE-IND MENU-ACCELERATOR Write Table MENU-LEVEL COM-IND).....	63
2.1.5.13	(COM-UNDO-IND MENU-ACCELERATOR Undo MENU-LEVEL COM-IND).....	63
2.1.5.14	(COM-HULL MENU-ACCELERATOR Hull MENU-LEVEL PAR)	63
2.1.5.15	(COM-STATIONARY MENU-ACCELERATOR Stationary MENU-LEVEL PAR).....	63
2.1.5.16	(COM-MOVING MENU-ACCELERATOR Moving MENU-LEVEL PAR)	64
2.1.5.17	(COM-CATASTROPHIC MENU-ACCELERATOR Catastrophic MENU-LEVEL PAR-1)	64
2.1.5.18	(COM-MOBILITY MENU-ACCELERATOR Mobility MENU-LEVEL PAR-1)	64
2.1.5.19	(COM-FIREPOWER MENU-ACCELERATOR Firepower MENU-LEVEL PAR-1).....	65
2.1.5.20	MAKE-GRAPH-GIVEN-POINTS	65
2.1.5.21	SWITCH-HIGHLIGHT	66
2.1.5.22	CHANGE-MODE.....	66
2.1.5.23	SELECT-HOST	67
2.1.5.24	SELECT-TABLE.....	68
2.1.5.25	DISPATCH-TO-TABLE	68
2.1.5.26	USER-SELECT-MODEL	69
2.1.5.27	USER-SELECT-ASPECT	70
2.1.6	Panes and their presentations	70
2.1.6.1	DYNAMIC-WINDOW-WITHOUT-SCROLL-BARS.....	71
2.1.6.2	MS-DYNAMIC-WINDOW-PANE	71
2.1.6.3	PARAMETER-DISPLAY-PANE	72
2.1.6.4	POINT.....	72
2.1.6.5	MOVE-POINT.....	75
2.1.6.6	DELETE-POINT	75
2.1.6.7	INSERT-POINT	76
2.1.6.8	INSERT-POINT-1.....	76
2.1.6.9	UP-DOWN-POINT.....	76
2.1.6.10	MOVE-UP-DOWN-POINT.....	77
2.1.6.11	SELECT-AND-DRAG-UP-DOWN-POINT.....	77
2.1.6.12	(METHOD DRAG-UP-DOWN POINT).....	77

2.1.6.13	MS-PANE.....	78
2.1.6.14	RANGE	79
2.1.6.15	PROBABILITY	79
2.1.6.16	REPLACE-RANGE	79
2.1.6.17	REPLACE-RANGE	80
2.1.6.18	(METHOD ALTER-RANGE POINT).....	80
2.1.6.19	REPLACE-PROBABILITY	81
2.1.6.20	REPLACE-PROBABILITY	81
2.1.6.21	(METHOD ALTER-PROBABILITY POINT).....	81
2.1.6.22	(METHOD XREPLACE-OBJECT-VALUE DYNAMIC-WINDOW)	82
2.1.6.23	COM-MODELS	82
2.1.6.24	SELECT-MODEL-MENU	82
2.1.7	File input/output and database preparation	83
2.1.7.1	HIT-TABLE	83
2.1.7.2	IND-FIR-TABLE	84
2.1.7.3	DIR-FIR-TABLE.....	84
2.1.7.4	MAKE-SIMHOST-READTABLE	84
2.1.7.5	*SAVE-READTABLE*	85
2.1.7.6	MAKE-HITMODELS-DICTIONARY	85
2.1.7.7	ELIMINATE-DUPPLICATES	86
2.1.7.8	MAKE-DICTIONARY-PAIRS.....	86
2.1.7.9	COPY-LIST-ALL-LEVELS	87
2.1.7.10	RESTORE-UPLOWCASE.....	87
2.1.7.11	LOOKUP-DICTIONARY.....	87
2.1.7.12	READ-HIT-DATA	88
2.1.7.13	READ-HIT-DATA-FILE.....	88
2.1.7.14	READ-HIT-DATA-FILE-AUX	89
2.1.7.15	READ-GROUND-DETECTION-DATA	89
2.1.7.16	READ-GROUND-DATA-FROM-FILE.....	90
2.1.7.17	READ-AIR-DETECTION-DATA.....	91
2.1.7.18	READ-AIR-DATA-FROM-FILE.....	91
2.1.7.19	READ-INDIRECT-FIRE-DAMAGE-DATA	92
2.1.7.20	FETCH-IF-DATA.....	93
2.1.7.21	READ-INDIRECT-FIRE-DAMAGE-DATA- FROM-FILE	93
2.1.7.22	READ-DIRECT-FIRE-DAMAGE-DATA.....	94

2.1.7.23	FETCH-DF-DATA.....	95
2.1.7.24	READ-DIRECT-FIRE-DAMAGE-DATA-FROM- FILE	95
2.1.7.25	BUILD-A-HIT-TABLE	95
2.1.7.26	BUILD-A-DIR-FIR-TABLE.....	96
2.1.7.27	BUILD-A-IND-FIR-TABLE	96
2.1.7.28	WRITE-CORRESPONDING-TABLE	96
2.1.7.29	*DEVELOPERS-FLG*	97
2.1.7.30	WRITE-HIT-DATA.....	97
2.1.7.31	WRITE-HIT-DATA-FILE	98
2.1.7.32	WRITE-DETECTION-DATA	98
2.1.7.33	WRITE-DETECTION-DATA-FILE	99
2.1.7.34	WRITE-INDIRECT-FIRE-DAMAGE-DATA.....	99
2.1.7.35	WRITE-INDIRECT-FIRE-DAMAGE-DATA- FILE	100
2.1.7.36	WRITE-DIRECT-FIRE-DAMAGE-DATA	100
2.1.7.37	WRITE-DIRECT-FIRE-DAMAGE-DATA-FILE.....	101
2.1.7.38	GATHER-DATABASE	101
2.1.7.39	GATHER-INDIRECT-FIRE-DATABASE	102
2.1.7.40	EXTRACT-RANGE-AND-DICE-NUMBERS	102
2.1.7.41	GATHER-DIRECT-FIRE-DATABASE	102
2.1.7.42	EXTRACT-DICE-NUMBERS.....	103
2.1.7.43	GET-TABLE.....	103
2.1.7.44	GET-HIT-TABLE.....	104
2.1.7.45	REMEMBER-TABLE	104
2.1.7.46	REMEMBER-CURRENT-HIT-TABLE	105
2.1.7.47	MAKE-DOTTED-SUBLISTS.....	105
2.1.7.48	MAKE-UNDOTTED-SUBLISTS	106
2.1.7.49	QUERY-WINDOW.....	106
2.1.7.50	*DEFAULT-FILE-NAME*.....	106
2.1.7.51	*QUERY-WINDOW*.....	107
2.1.7.52	QUERY-USER.....	107
2.1.7.53	QUERY-FOR-FILENAME.....	107
2.1.7.54	GET-NEW-WEAPON-NAME	108
2.1.8	High level functions	108
2.1.8.1	SET-CURRENT-MODEL	108
2.1.8.2	HIGHLIGHT-SELECTION.....	110

2.1.8.3	PRESENT-DATA.....	110
2.1.8.4	GET-PLATFORM	111
2.1.8.5	PRESENT-TABLE	112
2.2	WEAPONS SYSTEMS EDITOR CSC.....	113
2.2.1	File I/O Operations CSU	114
2.2.2	Editor Frame CSU	115
2.2.3	Presentations CSU.....	117
2.2.4	Auxiliaries CSU.....	124
2.2.5	Program Frame CSU	127
2.3	FORMATIONS EDITOR CSC.....	129
2.3.1	Parameters and Globals CSU.....	130
2.3.2	File Input/Output CSU	130
2.3.3	Formations object CSU	130
2.3.4	Panes and Frame CSU	131
2.3.5	Presentations and commands CSU	131
2.3.6	Drawing Routines CSU	131
APPENDIX A: SAF COMMAND PROTOCOL.....		A-1
APPENDIX B: SAF PARAMETERS FILES		B-1
APPENDIX C: SAF TERRAIN FILES.....		C-1
INDEX BY SECTION NUMBER		INDEX-1

1 INTRODUCTION: SAF PARAMETER EDITOR CSCI DESCRIPTION

1.1 BACKGROUND

The function of the Semi-Automated Forces (SAF) Segment is to allow a single operator at a SAF workstation to control a large number of vehicles out on the SIMNET battlefield. The SAF Segment runs on two hardware platforms, the workstation computer and the simulation host computer. It consists of three CSCIs, The SAF Parameter Editor CSCI and the SAF Workstation CSCI which run on the workstation computer and the SAF Simulation Host CSCI which runs on the simulation host (Simhost) computer. The SAF Parameter Editor is the component which is used to reconfigure the SAF and its operating characteristics without the need to reprogram it.

1.2 EXTERNAL COMMUNICATIONS

The way the SAF Parameter Editor communicates with itself and other SAF CSCIs is shown in Figure 1.2-1.

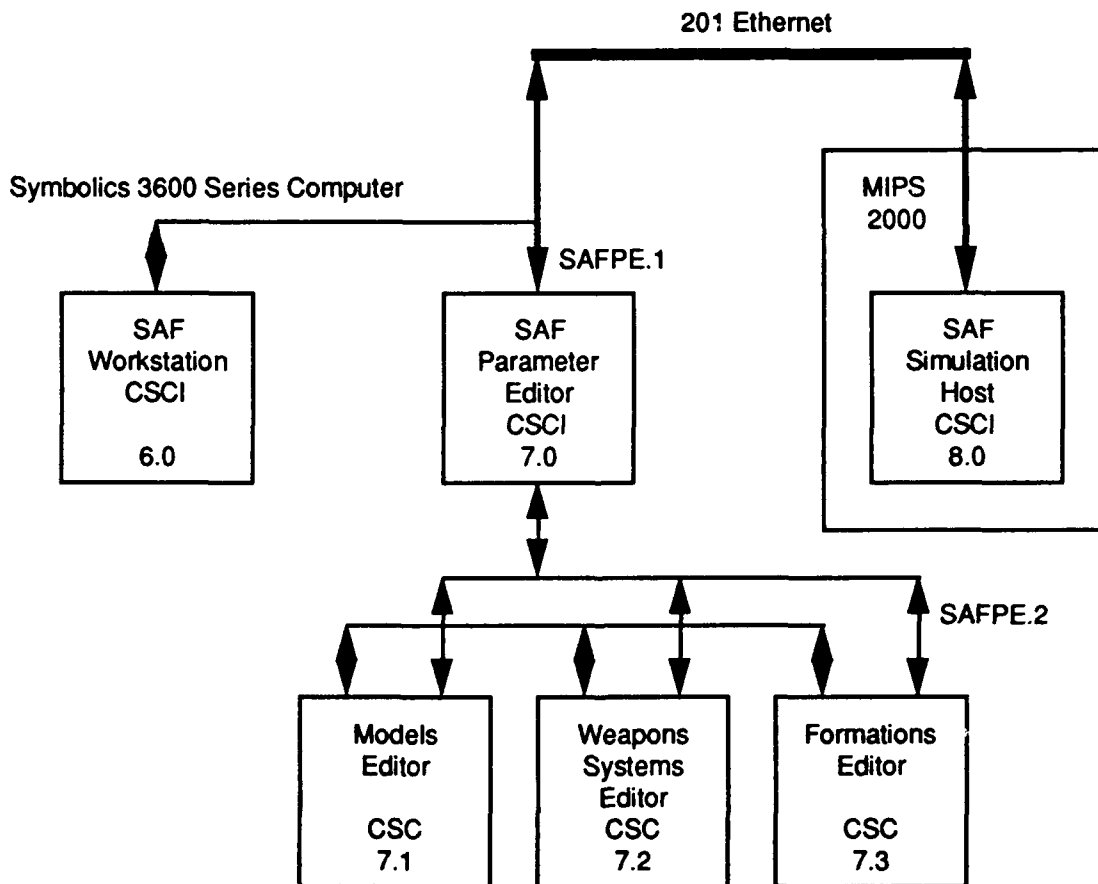


Figure 1.2-1 Software Interface Organization

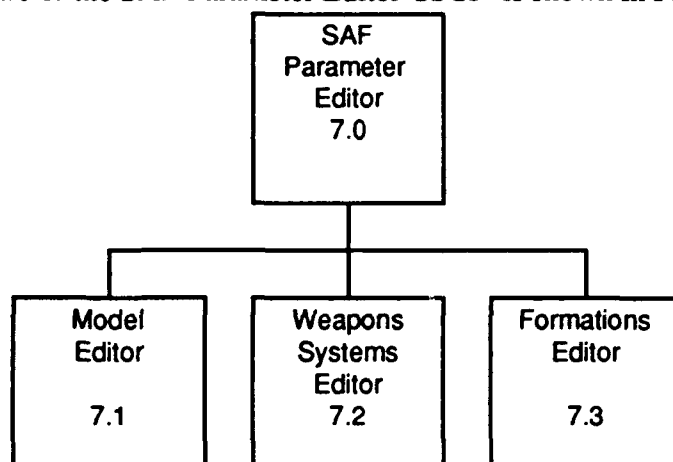
The SAF Parameter Editor CSCI interfaces with the SAF Workstation CSCI [6.0] and the SAF Simhost CSCI [8.0]. The SAF Parameter Editor receives mouse and keyboard input from the Workstation CSCI and sends screen information to the workstation. The Parameter Editor CSCI retrieves SAF Parameter Data Files, updates them, and stores these files via the Ethernet. These SAF Parameter Data Files are stored on the SAF Simhost computer. Reference the Appendices A.LA SAF SimHost.

1.3 INTERNAL STRUCTURE

The function of the Parameter Editor is to allow a user to modify the default characteristics of the network entities under his control in a consistent and straight forward manner. The parameters under the users control can affect vehicle's capability (Model Editor), a weapons capability (Weapons Editor), or define how multiple vehicles interact (Formation Editor).

The parameters available for alteration on the Model Editor include the probability the vehicle will hit a target at a given range and the probability that the vehicle will be damaged by Indirect or Direct fire. The Weapons Systems Editor allows the user to change parameters such as effective range for the weapon. The Formations editor allows the user to update the fundamental elements of various multiple vehicle formations, such as the inter vehicle distance or relative angle between vehicles.

The structure of the SAF Parameter Editor CSCI is shown in Figure 1.3-1.



SAF Parameter Editor CSCI Structure

The SAF Parameter Editor is composed of three top-level CSCs:

- Model Editor
- Weapons Systems Editor
- Formations Editor

1.4 CONFIGURATION AND CONFIGURATION MANAGEMENT

The SAF Parameter Editor CSCI and the SAF Workstation CSCI are written in LISP. The SAF Simulation Host CSCI is written in C. The SAF Parameter Editor runs on Symbolics 3650 Series Computers.

1.5 TERMINOLOGY AND DOCUMENTATION

The following documents provide additional information about the SAF Segment:

BBN Report No. 7025, SIMNET SEMI-AUTOMATED FORCES COMBINED ARMS WORKSTATION USERS GUIDE

BBN Report No. 7310, SIMNET SEMI-AUTOMATED FORCES TECHNICAL OPERATIONS MANUAL

BBN Report No. 7311, SIMNET SEMI-AUTOMATED FORCES MODEL EDITOR USERS GUIDE

BBN Report No. 7312, SIMNET SEMI-AUTOMATED FORCES WEAPONS SYSTEMS EDITOR USERS GUIDE

Perceptronics PTR-4043-17-0200, SIMNET SEMI-AUTOMATED FORCES (Version 3.x) FUNCTIONAL SPECIFICATION

Other related documents are listed in section 1.6 of the SAF Workstation CSCI document. The SAF Workstation CSCI document also contains a number of sections that apply to the entire SAF Lisp code. These sections provide important information relevant to the SAF Parameter Editor CSCI. Section 1.4 of the SAF Workstation CSCI document is an index to all the SAF files, section 1.5 contains notes on the design of SAF software, including information on the automatically generated definition cross-reference (section 1.5.1), section 1.7 is a glossary of SAF terms, and section 1.8 is a master index of SAF Lisp definitions.

The SAF Parameter Editor CSCI enables the user to edit, off-line, the parameters of vehicles and other SAF objects. The user does not need to know programming, nor have any knowledge of a standard text editor, in order to make changes to these parameters. The SAF Parameter Editor CSCI passes the edited information to the SAF Simulator Host and Workstation CSCIs in the form of SAF parameter files.

2 CSC DESCRIPTIONS

The SAF Parameter Editor CSCI contains three top level CSCs. These CSCs are editors that enable modification of various SAF data files: the Model Editor CSC, Weapons Systems Editor CSC, and Formations Editor CSC.

2.1 MODEL EDITOR CSC

This CSC contains the code which allows the operator to edit detection probabilities, direct fire hit probability, direct fire damage probabilities, and indirect fire damage probabilities. The data is presented in graphical and tabular displays, and can be changed in either form. Changes can be undone or saved. The software for this editor is found in the file `saf>interface>model-menu.lisp`.

This CSC edits the following files in the Simhost directory `/usr/saf/config/`:

detection.lisp
hitmodels.lisp
tank_df.lisp
tank_if.lisp

The Models Editor is actually a multi-use editor for 5 different kinds of data, sharing a common thread:

Hit Model: probability that a weapon's shot will hit a target versus the target's distance, as a function of weapon kind and target attitude (moving, stationary, or hull defilade).

Ground Detection Model: probability of seeing a vehicle from the ground versus distance, as a function of arc of vision and target attitude (moving, stationary, or hull defilade).

Air Detection Model: probability of seeing a vehicle from the air versus distance, as a function of arc of vision and target attitude (moving, stationary, or hull defilade).

Direct Fire Damage Model: probability of damage given that a directly fired shot has hit a vehicle versus angle of incidence, as a function of kind of munition, side, structure being hit, and degree of damage (catastrophic, mobility, and firepower).

Indirect Fire Damage Model: probability of damage due to indirect fire (e.g.: near miss by a bomb) versus distance, as a function of kind of munition and degree of damage (catastrophic, mobility, and firepower).

The common thread is the probability function, which the editor represents both as an XY graph or plot, and as a numerical table. Users can change probability functions by manipulating points in the graph representation (deleting inserting and moving them), or by editing numbers in the table representation. The resulting changes can then be evaluated for their tactical effects.

This editor was originally conceived for Hit data only. When the need for editing detection data became apparent, the editor was augmented to handle it. Finally, damage data editing capabilities were added. The software is in the file saf:interface;model-menu.lisp.

This CSC consists of the following CSUs:

- Parameters, Globals, and Macros
- Graphing code
- Graphics manipulation functions
- Frame and screen configurations
- Command processor commands
- Panes and their presentations
- File input/output and database preparation
- High level functions

A description of each of these units follows. Each unit is located in the file following a comment header; an abbreviated form of this header is included below, in brackets, at the beginning of each section.

2.1.1 Parameters, Globals, and Macros

[; ; PARAMETERS]

Contains definitions of variables used globally (mostly within the Models editor code). Their names are enclosed in asterisks (*), their initial values are specified, and most have their purpose explained by a brief documentation string. A few macros of general utility are also defined in this CSU.

2.1.1.1 *OLD-SELECTION*

Type:	>saf>interface>model-menu.lisp
Arguments:	Parameter
Outputs:	()
Calls:	None
Called by:	HIGHLIGHT-SELECTION
	>saf>interface>model-menu.lisp
	SET-CURRENT-MODEL
	>saf>interface>model-menu.lisp
	SWITCH-HIGHLIGHT
	>saf>interface>model-menu.lisp
Description:	None

2.1.1.2 *HULL-GRAPH*

Type: >saf>interface>model-menu.lisp
Parameter
Arguments: ()
Outputs:
Calls: None
Called by: SET-CURRENT-MODEL
>saf>interface>model-menu.lisp
CHANGE-MODE
>saf>interface>model-menu.lisp
Description: None

2.1.1.3 *STATIONARY-GRAPH*

Type: >saf>interface>model-menu.lisp
Parameter
Arguments: ()
Outputs:
Calls: None
Called by: SET-CURRENT-MODEL
>saf>interface>model-menu.lisp
CHANGE-MODE
>saf>interface>model-menu.lisp
Description: None

2.1.1.4 *MOVING-GRAPH*

Type: >saf>interface>model-menu.lisp
Parameter
Arguments: ()
Outputs:
Calls: None
Called by: SET-CURRENT-MODEL
>saf>interface>model-menu.lisp
CHANGE-MODE
>saf>interface>model-menu.lisp
Description: None

2.1.1.5 *CATASTROPHIC-GRAPH*

Type: >saf>interface>model-menu.lisp
Parameter
Arguments: ()
Outputs:
Calls: None
Called by: SET-CURRENT-MODEL
>saf>interface>model-menu.lisp
CHANGE-MODE
>saf>interface>model-menu.lisp
Description: None

2.1.1.6 *MOBILITY-GRAPH*

Type: >saf>interface>model-menu.lisp
Parameter
Arguments: ()
Outputs:
Calls: None
Called by: SET-CURRENT-MODEL
>saf>interface>model-menu.lisp
CHANGE-MODE
>saf>interface>model-menu.lisp
Description: None

2.1.1.7 *FIREPOWER-GRAPH*

Type: >saf>interface>model-menu.lisp
Parameter
Arguments: ()
Outputs:
Calls: None
Called by: SET-CURRENT-MODEL
>saf>interface>model-menu.lisp
CHANGE-MODE
>saf>interface>model-menu.lisp
Description: None

2.1.1.8 *HULL-POINTS*

Type: >saf>interface>model-menu.lisp
Parameter
Arguments: ()
Outputs:
Calls: None
Called by: SET-CURRENT-MODEL
>saf>interface>model-menu.lisp
REMEMBER-CURRENT-HIT-TABLE
>saf>interface>model-menu.lisp
REMEMBER-TABLE
>saf>interface>model-menu.lisp
CHANGE-MODE
>saf>interface>model-menu.lisp
Description: None

2.1.1.9 *STATIONARY-POINTS*

Type: >saf>interface>model-menu.lisp
Parameter
Arguments: ()
Outputs:
Calls: None
Called by: SET-CURRENT-MODEL
>saf>interface>model-menu.lisp
REMEMBER-CURRENT-HIT-TABLE
>saf>interface>model-menu.lisp
REMEMBER-TABLE
>saf>interface>model-menu.lisp
CHANGE-MODE
>saf>interface>model-menu.lisp
Description: None

2.1.1.10 *MOVING-POINTS*

Type: >saf>interface>model-menu.lisp
Parameter
Arguments: ()
Outputs:
Calls: None
Called by: SET-CURRENT-MODEL
>saf>interface>model-menu.lisp
REMEMBER-CURRENT-HIT-TABLE
>saf>interface>model-menu.lisp
REMEMBER-TABLE
>saf>interface>model-menu.lisp
CHANGE-MODE
>saf>interface>model-menu.lisp
Description: None

2.1.1.11 *CATASTROPHIC-POINTS*

Type: >saf>interface>model-menu.lisp
Parameter
Arguments: ()
Outputs:
Calls: None
Called by: SET-CURRENT-MODEL
>saf>interface>model-menu.lisp
REMEMBER-TABLE
>saf>interface>model-menu.lisp
CHANGE-MODE
>saf>interface>model-menu.lisp
Description: None

2.1.1.12 *MOBILITY-POINTS*

Type: >saf>interface>model-menu.lisp
Parameter
Arguments: ()
Outputs:
Calls: None
Called by: SET-CURRENT-MODEL
>saf>interface>model-menu.lisp
REMEMBER-TABLE
>saf>interface>model-menu.lisp
CHANGE-MODE
>saf>interface>model-menu.lisp
Description: None

Type: >saf>interface>model-menu.lisp
Parameter
Arguments: ()
Outputs:
Calls: None
Called by: SET-CURRENT-MODEL
>saf>interface>model-menu.lisp
REMEMBER-TABLE
>saf>interface>model-menu.lisp
CHANGE-MODE
>saf>interface>model-menu.lisp
Description: None

2.1.1.14 *PART*

Type: >saf>interface>model-menu.lisp
Parameter
Arguments: ()
Outputs:
Calls: None
Called by: PRESENT-DATA
>saf>interface>model-menu.lisp
GET-TABLE
>saf>interface>model-menu.lisp
READ-DIRECT-FIRE-DAMAGE-DATA
>saf>interface>model-menu.lisp
USER-SELECT-ASPECT
>saf>interface>model-menu.lisp
Description: None

2.1.1.15 *ASPECT*

Type: >saf>interface>model-menu.lisp
 Arguments: Parameter
 Outputs: ()
 Calls: None
 Called by: PRESENT-DATA
 >saf>interface>model-menu.lisp
 GET-TABLE
 >saf>interface>model-menu.lisp
 READ-DIRECT-FIRE-DAMAGE-DATA
 >saf>interface>model-menu.lisp
 USER-SELECT-ASPECT
 >saf>interface>model-menu.lisp
 Description: None

2.1.1.16 *TABLE-TYPE*

Type: >saf>interface>model-menu.lisp
 Arguments: Parameter
 Outputs: ()
 Calls: None
 Called by: (PRESENTATION-MOUSE-HANDLER INSERT-POINT-1
 TESTER)
 No Source File Record
 (PRESENTATION-MOUSE-HANDLER INSERT-POINT
 TESTER)
 No Source File Record
 PRESENT-TABLE
 >saf>interface>model-menu.lisp
 PRESENT-DATA
 >saf>interface>model-menu.lisp
 SET-CURRENT-MODEL
 >saf>interface>model-menu.lisp
 REMEMBER-TABLE
 >saf>interface>model-menu.lisp
 GET-TABLE
 >saf>interface>model-menu.lisp
 WRITE-DETECTION-DATA-FILE
 >saf>interface>model-menu.lisp
 WRITE-CORRESPONDING-TABLE
 >saf>interface>model-menu.lisp
 READ-DIRECT-FIRE-DAMAGE-DATA
 >saf>interface>model-menu.lisp
 READ-INDIRECT-FIRE-DAMAGE-DATA
 >saf>interface>model-menu.lisp
 READ-AIR-DETECTION-DATA
 >saf>interface>model-menu.lisp
 READ-GROUND-DETECTION-DATA

>saf>interface>model-menu.lisp
 READ-HIT-DATA
 >saf>interface>model-menu.lisp
 USER-SELECT-MODEL
 >saf>interface>model-menu.lisp
 MAKE-GRAPH-GIVEN-POINTS
 >saf>interface>model-menu.lisp
 GRAPH-UNDO
 >saf>interface>model-menu.lisp
 REVERT-TO-FACTORY-VERSION
 >saf>interface>model-menu.lisp
 RESCALE-POINT-LIST
 >saf>interface>model-menu.lisp
 PRESENT-GRAPH
 >saf>interface>model-menu.lisp
 DRAW-X-TICKS
 >saf>interface>model-menu.lisp
 Description: None

2.1.1.17 *MODE*

Type: >saf>interface>model-menu.lisp
 Arguments: Parameter
 Outputs: ()
 Calls: None
 Called by: (PRESENTATION-MOUSE-HANDLER INSERT-POINT-1)
 No Source File Record
 (PRESENTATION-MOUSE-HANDLER INSERT-POINT)
 No Source File Record
 SET-CURRENT-MODEL
 >saf>interface>model-menu.lisp
 (METHOD ALTER-PROBABILITY POINT)
 >saf>interface>model-menu.lisp
 (METHOD ALTER-RANGE POINT)
 >saf>interface>model-menu.lisp
 (METHOD DRAG-UP-DOWN POINT)
 >saf>interface>model-menu.lisp
 SELECT-AND-DRAG-UP-DOWN-POINT
 >saf>interface>model-menu.lisp
 CHANGE-MODE
 >saf>interface>model-menu.lisp
 SWITCH-HIGHLIGHT
 >saf>interface>model-menu.lisp
 GO-BACK-TO-PREVIOUS-STEP
 >saf>interface>model-menu.lisp
 UPDATE-BACKTRACKING-CAPABILITY
 >saf>interface>model-menu.lisp
 SELECT-AND-DRAG-POINT
 >saf>interface>model-menu.lisp
 GET-POINT
 >saf>interface>model-menu.lisp

```
(METHOD ERASE-POINT-AND-LINES POINT)
>saf>interface>model-menu.lisp
(METHOD DRAG POINT)
>saf>interface>model-menu.lisp
ADD-NEW-POINT
>saf>interface>model-menu.lisp
RECORD-NEW-POINT
>saf>interface>model-menu.lisp
ADD-CORRESPONDING-NEW-POINT-IN-POINT-LIST
>saf>interface>model-menu.lisp
UPDATE-POINT-LIST
>saf>interface>model-menu.lisp
FIND-SURROUNDING-POINTS
>saf>interface>model-menu.lisp
DELETE-POINT-IF-THERE
>saf>interface>model-menu.lisp
(METHOD EXPUNGE POINT)
>saf>interface>model-menu.lisp
DRAW-X-TICKS
>saf>interface>model-menu.lisp
GET-CURRENT-LINE-POINTS
>saf>interface>model-menu.lisp
GET-CURRENT-GRAPH-POINTS
>saf>interface>model-menu.lisp
GIMME-VAR-NAME-OF-CURRENT-POINTS
>saf>interface>model-menu.lisp
GIMME-VAR-NAME-OF-CURRENT-GRAPH
>saf>interface>model-menu.lisp
None
```

Description:

2.1.1.18 *WEAPON-NAME*

Type:	>saf>interface>model-menu.lisp
Arguments:	Parameter
Outputs:	()
Calls:	None
Called by:	PRESENT-DATA >saf>interface>model-menu.lisp SET-CURRENT-MODEL >saf>interface>model-menu.lisp REMEMBER-CURRENT-HIT-TABLE >saf>interface>model-menu.lisp REMEMBER-TABLE >saf>interface>model-menu.lisp READ-DIRECT-FIRE-DAMAGE-DATA >saf>interface>model-menu.lisp READ-INDIRECT-FIRE-DAMAGE-DATA-FROM-FILE >saf>interface>model-menu.lisp READ-INDIRECT-FIRE-DAMAGE-DATA >saf>interface>model-menu.lisp READ-AIR-DATA-FROM-FILE >saf>interface>model-menu.lisp READ-AIR-DETECTION-DATA >saf>interface>model-menu.lisp READ-GROUND-DATA-FROM-FILE >saf>interface>model-menu.lisp READ-GROUND-DETECTION-DATA >saf>interface>model-menu.lisp READ-HIT-DATA-FILE >saf>interface>model-menu.lisp READ-HIT-DATA >saf>interface>model-menu.lisp USER-SELECT-ASPECT >saf>interface>model-menu.lisp GRAPH-UNDO >saf>interface>model-menu.lisp
Description:	None

2.1.1.19 *PARAMETER-DISPLAY-PANE*

Type:	>saf>interface>model-menu.lisp
Arguments:	Variable
Outputs:	()
Calls:	None
Called by:	(PRESENTATION-MOUSE-HANDLER INSERT-POINT-1) No Source File Record (PRESENTATION-MOUSE-HANDLER INSERT-POINT) No Source File Record READ-DIRECT-FIRE-DAMAGE-DATA >saf>interface>model-menu.lisp READ-INDIRECT-FIRE-DAMAGE-DATA >saf>interface>model-menu.lisp READ-AIR-DETECTION-DATA >saf>interface>model-menu.lisp READ-GROUND-DETECTION-DATA >saf>interface>model-menu.lisp READ-HIT-DATA >saf>interface>model-menu.lisp SELECT-MODEL-MENU >saf>interface>model-menu.lisp (METHOD ALTER-PROBABILITY POINT) >saf>interface>model-menu.lisp (METHOD ALTER-RANGE POINT) >saf>interface>model-menu.lisp (METHOD DRAG-UP-DOWN POINT) >saf>interface>model-menu.lisp (METHOD COM-FIREPOWER-INTERNAL MODEL-MENU) No Source File Record (METHOD COM-MOBILITY-INTERNAL MODEL-MENU) No Source File Record (METHOD COM-CATASTROPHIC-INTERNAL MODEL-MENU) No Source File Record (METHOD COM-MOVING-INTERNAL MODEL-MENU) No Source File Record (METHOD COM-STATIONARY-INTERNAL MODEL-MENU) No Source File Record (METHOD COM-HULL-INTERNAL MODEL-MENU) No Source File Record (METHOD COM-UNDO-IND-INTERNAL MODEL-MENU) No Source File Record

(METHOD COM-SELECT-MODEL-IND-INTERNAL MODEL-MENU)
 No Source File Record
 (METHOD COM-UNDO-DIR-INTERNAL MODEL-MENU)
 No Source File Record
 (METHOD COM-SELECT-ASPECT-DIR-INTERNAL MODEL-MENU)
 No Source File Record
 (METHOD COM-SELECT-MODEL-DIR-INTERNAL MODEL-MENU)
 No Source File Record
 (METHOD COM-UNDO-INTERNAL MODEL-MENU)
 No Source File Record
 (METHOD COM-SELECT-MODEL-INTERNAL MODEL-MENU)
 No Source File Record
 (METHOD TOP-LEVEL MODEL-MENU)
 >saf>interface>model-menu.lisp
 GRAPH-UNDO
 >saf>interface>model-menu.lisp
 (METHOD DRAG POINT)
 >saf>interface>model-menu.lisp
 SELECT-MODEL-MENU
 >saf>interface>model-menu.lisp
 Description: None

2.1.1.20 *TABULAR-DISPLAY-PANE*

>saf>interface>model-menu.lisp
 Type: Variable
 Arguments: ()
 Outputs:
 Calls: None
 Called by: PRESENT-DATA
 >saf>interface>model-menu.lisp
 (METHOD TOP-LEVEL MODEL-MENU)
 >saf>interface>model-menu.lisp
 ADD-CORRESPONDING-NEW-POINT-IN-POINT-LIST
 >saf>interface>model-menu.lisp
 DELETE-POINT-IF-THERE
 >saf>interface>model-menu.lisp
 Description: None

2.1.1.21 *MODEL-FRAME*

Type: >saf>interface>model-menu.lisp
Parameter
Arguments: ()
Outputs:
Calls: None
Called by: MAKE-DOCUMENTATION
>saf>interface>model-menu.lisp
(METHOD TOP-LEVEL MODEL-MENU)
>saf>interface>model-menu.lisp
(METHOD MAKE-INSTANCE MODEL-MENU AFTER)
>saf>interface>model-menu.lisp
Description: None

2.1.1.22 *DISPLAY-LOGO-PANE*

Type: >saf>interface>model-menu.lisp
Variable
Arguments: ()
Outputs:
Calls: None
Called by: PRESENT-DATA
>saf>interface>model-menu.lisp
(METHOD TOP-LEVEL MODEL-MENU)
>saf>interface>model-menu.lisp
Description: None

2.1.1.23 *TABULAR-LOGO-PANE*

Type: >saf>interface>model-menu.lisp
Variable
Arguments: ()
Outputs:
Calls: None
Called by: PRESENT-DATA
>saf>interface>model-menu.lisp
(METHOD TOP-LEVEL MODEL-MENU)
>saf>interface>model-menu.lisp
Description: None

2.1.1.24 *HIT-FILE-NAME*

Type: >saf>interface>model-menu.lisp
Parameter
Arguments: ()
Outputs:
Calls: None
Called by: WRITE-HIT-DATA
>saf>interface>model-menu.lisp
READ-HIT-DATA
>saf>interface>model-menu.lisp
MAKE-HITMODELS-DICTIONARY
>saf>interface>model-menu.lisp
Description: None

2.1.1.25 *DETECTION-FILE-NAME*

Type: >saf>interface>model-menu.lisp
Parameter
Arguments: ()
Outputs:
Calls: None
Called by: WRITE-DETECTION-DATA
>saf>interface>model-menu.lisp
READ-AIR-DETECTION-DATA
>saf>interface>model-menu.lisp
READ-GROUND-DETECTION-DATA
>saf>interface>model-menu.lisp
Description: None

2.1.1.26 *INDIRECT-FIRE-DAMAGE-FILE-NAME*

Type: >saf>interface>model-menu.lisp
Parameter
Arguments: ()
Outputs:
Calls: None
Called by: PRESENT-DATA
>saf>interface>model-menu.lisp
WRITE-INDIRECT-FIRE-DAMAGE-DATA
>saf>interface>model-menu.lisp
READ-INDIRECT-FIRE-DAMAGE-DATA
>saf>interface>model-menu.lisp
REVERT-TO-FACTORY-VERSION
>saf>interface>model-menu.lisp
Description: None

2.1.1.27 *DIRECT-FIRE-DAMAGE-FILE-NAME*

Type: >saf>interface>model-menu.lisp
Parameter
Arguments: ()
Outputs:
Calls: None
Called by: PRESENT-DATA
>saf>interface>model-menu.lisp
WRITE-DIRECT-FIRE-DAMAGE-DATA
>saf>interface>model-menu.lisp
READ-DIRECT-FIRE-DAMAGE-DATA
>saf>interface>model-menu.lisp
REVERT-TO-FACTORY-VERSION
>saf>interface>model-menu.lisp
Description: None

2.1.1.28 *DF-VEHICLE*

Type: >saf>interface>model-menu.lisp
Parameter
Arguments: ()
Outputs:
Calls: None
Called by: GET-PLATFORM
>saf>interface>model-menu.lisp
WRITE-DIRECT-FIRE-DAMAGE-DATA-FILE
>saf>interface>model-menu.lisp
READ-DIRECT-FIRE-DAMAGE-DATA
>saf>interface>model-menu.lisp
Description: None

2.1.1.29 *IF-VEHICLE*

Type: >saf>interface>model-menu.lisp
Parameter
Arguments: ()
Outputs:
Calls: None
Called by: GET-PLATFORM
>saf>interface>model-menu.lisp
WRITE-INDIRECT-FIRE-DAMAGE-DATA-FILE
>saf>interface>model-menu.lisp
READ-INDIRECT-FIRE-DAMAGE-DATA
>saf>interface>model-menu.lisp
Description: None

2.1.1.30 *IND-FIR-TABLES*

Type: >saf>interface>model-menu.lisp
Parameter
Arguments: ()
Outputs:
Calls: None
Called by: GATHER-INDIRECT-FIRE-DATABASE
>saf>interface>model-menu.lisp
GATHER-DATABASE
>saf>interface>model-menu.lisp
WRITE-INDIRECT-FIRE-DAMAGE-DATA-FILE
>saf>interface>model-menu.lisp

READ-INDIRECT-FIRE-DAMAGE-DATA
 >saf>interface>model-menu.lisp
 USER-SELECT-MODEL
 >saf>interface>model-menu.lisp
 (METHOD COM-SELECT-MODEL-IND-INTERNAL MODEL-
 MENU)
 No Source File Record
 GRAPH-UNDO
 >saf>interface>model-menu.lisp
 DRAW-X-TICKS
 >saf>interface>model-menu.lisp
 Description: None

2.1.1.31 *DIR-FIR-TABLES*

>saf>interface>model-menu.lisp
 Type: Parameter
 Arguments: ()
 Outputs:
 Calls: None
 Called by: GATHER-DIRECT-FIRE-DATABASE
 >saf>interface>model-menu.lisp
 GATHER-DATABASE
 >saf>interface>model-menu.lisp
 WRITE-DIRECT-FIRE-DAMAGE-DATA-FILE
 >saf>interface>model-menu.lisp
 READ-DIRECT-FIRE-DAMAGE-DATA
 >saf>interface>model-menu.lisp
 USER-SELECT-ASPECT
 >saf>interface>model-menu.lisp
 USER-SELECT-MODEL
 >saf>interface>model-menu.lisp
 (METHOD COM-SELECT-MODEL-DIR-INTERNAL MODEL-
 MENU)
 No Source File Record
 GRAPH-UNDO
 >saf>interface>model-menu.lisp
 DRAW-X-TICKS
 >saf>interface>model-menu.lisp
 Description: None

2.1.1.32 *HIT-TABLES*

Type: >saf>interface>model-menu.lisp
 Variable
 Arguments: ()
 Outputs:
 Calls: None
 Called by: REMEMBER-TABLE
 >saf>interface>model-menu.lisp
 GET-HIT-TABLE
 >saf>interface>model-menu.lisp
 GET-TABLE
 >saf>interface>model-menu.lisp
 WRITE-DETECTION-DATA-FILE
 >saf>interface>model-menu.lisp
 WRITE-HIT-DATA-FILE
 >saf>interface>model-menu.lisp
 READ-AIR-DETECTION-DATA
 >saf>interface>model-menu.lisp
 READ-GROUND-DETECTION-DATA
 >saf>interface>model-menu.lisp
 READ-HIT-DATA
 >saf>interface>model-menu.lisp
 USER-SELECT-MODEL
 >saf>interface>model-menu.lisp
 GRAPH-UNDO
 >saf>interface>model-menu.lisp
 DRAW-X-TICKS
 >saf>interface>model-menu.lisp
 Description: list of all known hit table defstructs

2.1.1.33 *DIRECT-FIRE-DAMAGE-DATA*

Type: >saf>interface>model-menu.lisp
 Parameter
 Arguments: ()
 Outputs:
 Calls: None
 Called by: WRITE-DIRECT-FIRE-DAMAGE-DATA-FILE
 >saf>interface>model-menu.lisp
 FETCH-DF-DATA
 >saf>interface>model-menu.lisp
 READ-DIRECT-FIRE-DAMAGE-DATA
 >saf>interface>model-menu.lisp
 Description: None

2.1.1.34 *INDIRECT-FIRE-DAMAGE-DATA*

Type: >saf>interface>model-menu.lisp
 Parameter
 Arguments: ()
 Outputs:
 Calls: None
 Called by: WRITE-INDIRECT-FIRE-DAMAGE-DATA-FILE
 >saf>interface>model-menu.lisp
 FETCH-IF-DATA
 >saf>interface>model-menu.lisp
 READ-INDIRECT-FIRE-DAMAGE-DATA
 >saf>interface>model-menu.lisp
 Description: None

2.1.1.35 TRUNCATE-TO-N-POSITIONS

Type: >saf>interface>model-menu.lisp
 Macro
 Arguments: (NUMBER N)
 Outputs:
 Calls: TRUNCATE-TO-N-POSITIONS
 >saf>interface>model-menu.lisp
 Called by: ADD-NEW-POINT
 >saf>interface>model-menu.lisp
 ADD-CORRESPONDING-NEW-POINT-IN-POINT-LIST
 >saf>interface>model-menu.lisp
 TRUNCATE-TO-N-POSITIONS
 >saf>interface>model-menu.lisp
 Description: None

2.1.1.36 GIMME-VAR-NAME-OF-CURRENT-GRAPH

Type: >saf>interface>model-menu.lisp
 Macro
 Arguments: ()
 Outputs:
 Calls: MKATOM
 >saf>sys>new-storage.lisp
 CONCAT
 >saf>sys>new-storage.lisp
 MODE
 >saf>interface>model-menu.lisp
 GIMME-VAR-NAME-OF-CURRENT-GRAPH
 >saf>interface>model-menu.lisp
 Called by: (PRESENTATION-MOUSE-HANDLER INSERT-POINT-1)
 No Source File Record
 (PRESENTATION-MOUSE-HANDLER INSERT-POINT)
 No Source File Record

```

(METHOD ALTER-PROBABILITY POINT)
>saf>interface>model-menu.lisp
(METHOD ALTER-RANGE POINT)
>saf>interface>model-menu.lisp
(METHOD DRAG-UP-DOWN POINT)
>saf>interface>model-menu.lisp
SELECT-AND-DRAG-UP-DOWN-POINT
>saf>interface>model-menu.lisp
GO-BACK-TO-PREVIOUS-STEP
>saf>interface>model-menu.lisp
UPDATE-BACKTRACKING-CAPABILITY
>saf>interface>model-menu.lisp
SELECT-AND-DRAG-POINT
>saf>interface>model-menu.lisp
GET-POINT
>saf>interface>model-menu.lisp
(METHOD ERASE-POINT-AND-LINES POINT)
>saf>interface>model-menu.lisp
(METHOD DRAG POINT)
>saf>interface>model-menu.lisp
ADD-NEW-POINT
>saf>interface>model-menu.lisp
RECORD-NEW-POINT
>saf>interface>model-menu.lisp
ADD-CORRESPONDING-NEW-POINT-IN-POINT-LIST
>saf>interface>model-menu.lisp
UPDATE-POINT-LIST
>saf>interface>model-menu.lisp
FIND-SURROUNDING-POINTS
>saf>interface>model-menu.lisp
DELETE-POINT-IF-THERE
>saf>interface>model-menu.lisp
(METHOD EXPUNGE POINT)
>saf>interface>model-menu.lisp
DRAW-X-TICKS
>saf>interface>model-menu.lisp
GET-CURRENT-GRAPH-POINTS
>saf>interface>model-menu.lisp
GIMME-VAR-NAME-OF-CURRENT-GRAPH
>saf>interface>model-menu.lisp

```

Description: None

2.1.1.37 GET-CURRENT-GRAPH-POINTS

```

Type: >saf>interface>model-menu.lisp
Subst
Arguments: ()
Outputs:
Calls: MKATOM
>saf>sys>new-storage.lisp
CONCAT
>saf>sys>new-storage.lisp

```

Called by: *MODE*
 >saf>interface>model-menu.lisp
 GIMME-VAR-NAME-OF-CURRENT-GRAPH
 >saf>interface>model-menu.lisp
 (PRESENTATION-MOUSE-HANDLER INSERT-POINT-1)
 No Source File Record
 (PRESENTATION-MOUSE-HANDLER INSERT-POINT)
 No Source File Record
 (METHOD ALTER-PROBABILITY POINT)
 >saf>interface>model-menu.lisp
 (METHOD ALTER-RANGE POINT)
 >saf>interface>model-menu.lisp
 (METHOD DRAG-UP-DOWN POINT)
 >saf>interface>model-menu.lisp
 SELECT-AND-DRAG-UP-DOWN-POINT
 >saf>interface>model-menu.lisp
 UPDATE-BACKTRACKING-CAPABILITY
 >saf>interface>model-menu.lisp
 SELECT-AND-DRAG-POINT
 >saf>interface>model-menu.lisp
 GET-POINT
 >saf>interface>model-menu.lisp
 (METHOD ERASE-POINT-AND-LINES POINT)
 >saf>interface>model-menu.lisp
 (METHOD DRAG POINT)
 >saf>interface>model-menu.lisp
 ADD-NEW-POINT
 >saf>interface>model-menu.lisp
 ADD-CORRESPONDING-NEW-POINT-IN-POINT-LIST
 >saf>interface>model-menu.lisp
 UPDATE-POINT-LIST
 >saf>interface>model-menu.lisp
 FIND-SURROUNDING-POINTS
 >saf>interface>model-menu.lisp
 DELETE-POINT-IF-THERE
 >saf>interface>model-menu.lisp
 (METHOD EXPUNGE POINT)
 >saf>interface>model-menu.lisp
 DRAW-X-TICKS
Description: >saf>interface>model-menu.lisp
 None

2.1.1.38 GIMME-VAR-NAME-OF-CURRENT-POINTS

Type: >saf>interface>model-menu.lisp
Macro
Arguments: ()
Outputs:
Calls: MKATOM
>saf>sys>new-storage.lisp
CONCAT
>saf>sys>new-storage.lisp
MODE
>saf>interface>model-menu.lisp
GIMME-VAR-NAME-OF-CURRENT-POINTS
Called by: >saf>interface>model-menu.lisp
ADD-CORRESPONDING-NEW-POINT-IN-POINT-LIST
>saf>interface>model-menu.lisp
UPDATE-POINT-LIST
>saf>interface>model-menu.lisp
DELETE-POINT-IF-THERE
>saf>interface>model-menu.lisp
GET-CURRENT-LINE-POINTS
>saf>interface>model-menu.lisp
GIMME-VAR-NAME-OF-CURRENT-POINTS
>saf>interface>model-menu.lisp
Description: None

2.1.1.39 GET-CURRENT-LINE-POINTS

Type: >saf>interface>model-menu.lisp
Subst
Arguments: ()
Outputs:
Calls: MKATOM
>saf>sys>new-storage.lisp
CONCAT
>saf>sys>new-storage.lisp

	MODE
	>saf>interface>model-menu.lisp
	GIMME-VAR-NAME-OF-CURRENT-POINTS
Called by:	>saf>interface>model-menu.lisp
	ADD-CORRESPONDING-NEW-POINT-IN-POINT-LIST
	>saf>interface>model-menu.lisp
	UPDATE-POINT-LIST
	>saf>interface>model-menu.lisp
	DELETE-POINT-IF-THERE
	>saf>interface>model-menu.lisp
Description:	None

2.1.2 Graphing code

[; ; ; GRAPHING code]

This is the code used to draw the probability graphs. It consists of functions to draw the X and Y axes (with their marks, calibrations, and labels), and functions to draw the graph proper. The top construct for the former is the macro CLEAN-UP-AXES-AND-REDRAW.

The top-level function for graph construction is PRESENT-GRAPH. It first draws the points of the graph and then the connecting straight-line segments.

Points are instances of the flavor POINT, and the principal methods are :DRAW and :DRAG

2.1.2.1 *X-AXIS-LENGTH*

Type:	>saf>interface>model-menu.lisp
Arguments:	Parameter
Outputs:	()
Calls:	None
Called by:	MAKE-GRAPH-GIVEN-POINTS
	>saf>interface>model-menu.lisp
	RESCALE-POINT-LIST
	>saf>interface>model-menu.lisp
	DRAW-X-TICKS
	>saf>interface>model-menu.lisp
	DRAW-X-SCALE-BAR
	>saf>interface>model-menu.lisp
Description:	None

2.1.2.2 *Y-AXIS-LENGTH*

Type: >saf>interface>model-menu.lisp
Parameter
Arguments: ()
Outputs:
Calls: None
Called by: DRAW-Y-TICKS
DRAW-Y-SCALE-BAR
>saf>interface>model-menu.lisp
Description: >saf>interface>model-menu.lisp
None

2.1.2.3 *X-ORIGIN-INCREMENT*

Type: >saf>interface>model-menu.lisp
Parameter
Arguments: ()
Outputs:
Calls: None
Called by: MAKE-GRAPH-GIVEN-POINTS
>saf>interface>model-menu.lisp
RESCALE-POINT-LIST
>saf>interface>model-menu.lisp
DRAW-X-TICKS
>saf>interface>model-menu.lisp
Description: None

2.1.2.4 *TOTAL-X-RANGE*

Type: >saf>interface>model-menu.lisp
Parameter
Arguments: ()
Outputs:
Calls: None
Called by: None
Description: None

2.1.2.5 *TOTAL-Y-RANGE*

Type: >saf>interface>model-menu.lisp
 Parameter
 Arguments: ()
 Outputs:
 Calls: None
 Called by: None
 Description: None

2.1.2.6 *DELTA-X*

Type: >saf>interface>model-menu.lisp
 Parameter
 Arguments: ()
 Outputs:
 Calls: None
 Called by: (METHOD ALTER-PROBABILITY POINT)
 >saf>interface>model-menu.lisp
 (METHOD ALTER-RANGE POINT)
 >saf>interface>model-menu.lisp
 (METHOD DRAG-UP-DOWN POINT)
 >saf>interface>model-menu.lisp
 MAKE-GRAPH-GIVEN-POINTS
 >saf>interface>model-menu.lisp
 (METHOD DRAG POINT)
 >saf>interface>model-menu.lisp
 ADD-NEW-POINT
 >saf>interface>model-menu.lisp
 RESCALE-POINT-LIST
 >saf>interface>model-menu.lisp
 CALCULATE-POINT-DATA-FROM-PIXEL-TO-REAL
 >saf>interface>model-menu.lisp
 CALCULATE-POINT-DATA
 >saf>interface>model-menu.lisp
 Description: None

2.1.2.7 *DELTA-Y*

Type: >saf>interface>model-menu.lisp
Parameter
Arguments: ()
Outputs:
Calls: None
Called by: (METHOD ALTER-PROBABILITY POINT)
>saf>interface>model-menu.lisp
(METHOD ALTER-RANGE POINT)
>saf>interface>model-menu.lisp
(METHOD DRAG-UP-DOWN POINT)
>saf>interface>model-menu.lisp
MAKE-GRAPH-GIVEN-POINTS
>saf>interface>model-menu.lisp
(METHOD DRAG POINT)
>saf>interface>model-menu.lisp
ADD-NEW-POINT
>saf>interface>model-menu.lisp
RESCALE-POINT-LIST
>saf>interface>model-menu.lisp
CALCULATE-POINT-DATA-FROM-PIXEL-TO-REAL
>saf>interface>model-menu.lisp
CALCULATE-POINT-DATA
>saf>interface>model-menu.lisp
Description: None

2.1.2.8 *ABSOLUTE-ORIGIN*

Type: >saf>interface>model-menu.lisp
Parameter
Arguments: ()
Outputs:
Calls: None
Called by: (METHOD ALTER-PROBABILITY POINT)
>saf>interface>model-menu.lisp
(METHOD ALTER-RANGE POINT)
>saf>interface>model-menu.lisp
(METHOD DRAG-UP-DOWN POINT)
>saf>interface>model-menu.lisp
MAKE-GRAPH-GIVEN-POINTS
>saf>interface>model-menu.lisp
(METHOD DRAG POINT)
>saf>interface>model-menu.lisp
ADD-NEW-POINT
>saf>interface>model-menu.lisp
RESCALE-POINT-LIST
>saf>interface>model-menu.lisp
CALCULATE-POINT-DATA-FROM-PIXEL-TO-REAL
>saf>interface>model-menu.lisp
CALCULATE-POINT-DATA

Description: >saf>interface>model-menu.lisp
None

2.1.2.9 *ABSOLUTE-X-ORIGIN*

Type: >saf>interface>model-menu.lisp
Parameter
Arguments: ()
Outputs:
Calls: None
Called by: DRAW-Y-TICKS
>saf>interface>model-menu.lisp
DRAW-X-TICKS
>saf>interface>model-menu.lisp
DRAW-Y-SCALE-BAR
>saf>interface>model-menu.lisp
DRAW-X-SCALE-BAR
>saf>interface>model-menu.lisp
Description: None

2.1.2.10 *ABSOLUTE-Y-ORIGIN*

Type: >saf>interface>model-menu.lisp
Parameter
Arguments: ()
Outputs:
Calls: None
Called by: DRAW-Y-TICKS
>saf>interface>model-menu.lisp
DRAW-X-TICKS
>saf>interface>model-menu.lisp
DRAW-Y-SCALE-BAR
>saf>interface>model-menu.lisp
DRAW-X-SCALE-BAR
>saf>interface>model-menu.lisp
Description: None

2.1.2.11 DRAW-X-SCALE-BAR

Type: >saf>interface>model-menu.lisp
Function
Arguments: (WINDOW &KEY (START-X *ABSOLUTE-X-ORIGIN*)
(START-Y *ABSOLUTE-Y-ORIGIN*)
(END-X (+ *ABSOLUTE-X-ORIGIN* *X-AXIS-LENGTH*)) (END-Y *ABSOLUTE-Y-ORIGIN*))
Outputs:
Calls: *X-AXIS-LENGTH*
>saf>interface>model-menu.lisp
ABSOLUTE-X-ORIGIN
>saf>interface>model-menu.lisp
ABSOLUTE-Y-ORIGIN
>saf>interface>model-menu.lisp
Called by: DRAW-SCALE-BAR
>saf>interface>model-menu.lisp
Description: None

2.1.2.12 DRAW-Y-SCALE-BAR

Type: >saf>interface>model-menu.lisp
Function
Arguments: (WINDOW &KEY (START-X *ABSOLUTE-X-ORIGIN*)
(START-Y *ABSOLUTE-Y-ORIGIN*)
(END-X *ABSOLUTE-X-ORIGIN*) (END-Y (- *ABSOLUTE-Y-ORIGIN* *Y-AXIS-LENGTH*)))
Outputs:
Calls: *Y-AXIS-LENGTH*
>saf>interface>model-menu.lisp
ABSOLUTE-X-ORIGIN
>saf>interface>model-menu.lisp
ABSOLUTE-Y-ORIGIN
>saf>interface>model-menu.lisp
Called by: DRAW-SCALE-BAR
>saf>interface>model-menu.lisp
Description: None

2.1.2.13 DRAW-SCALE-BAR

Type: >saf>interface>model-menu.lisp
 Function
 Arguments: (WINDOW)
 Outputs:
 Calls: DRAW-X-SCALE-BAR
 >saf>interface>model-menu.lisp
 DRAW-Y-SCALE-BAR
 >saf>interface>model-menu.lisp
 Called by: PRESENT-DATA
 >saf>interface>model-menu.lisp
 CLEAN-UP-AXIS-AND-REDRAW
 >saf>interface>model-menu.lisp
 Description: None

2.1.2.14 DRAW-X-TICKS

Type: >saf>interface>model-menu.lisp
 Function
 Arguments: (WINDOW &KEY (X-LENGTH *X-AXIS-LENGTH*)
 (INCREMENT *X-ORIGIN-INCREMENT*)
 (START-X *ABSOLUTE-X-ORIGIN*)
 (MAX-RANGE
 (IF (AND (OR *DIR-FIR-TABLES*
 IND-FIR-TABLES
 HIT-TABLES))
 (GET-CURRENT-GRAPH-POINTS)))
 (FLOAT (SEND (CAR (LAST (GET-CURRENT-GRAPH-POINTS))) REAL-X))
 (CASE *TABLE-TYPE*
 (DIRECT-FIRE-DAMAGE 90)
 (INDIRECT-FIRE-DAMAGE 50)
 (T 3500.0))))))
 Outputs:
 Calls: MKATOM
 >saf>sys>new-storage.lisp
 CONCAT
 >saf>sys>new-storage.lisp
 TABLE-TYPE
 >saf>interface>model-menu.lisp
 MODE
 >saf>interface>model-menu.lisp
 IND-FIR-TABLES
 >saf>interface>model-menu.lisp
 DIR-FIR-TABLES
 >saf>interface>model-menu.lisp
 HIT-TABLES
 >saf>interface>model-menu.lisp
 GIMME-VAR-NAME-OF-CURRENT-GRAPH
 >saf>interface>model-menu.lisp
 GET-CURRENT-GRAPH-POINTS

Called by: >saf>interface>model-menu.lisp
 X-AXIS-LENGTH
 >saf>interface>model-menu.lisp
 X-ORIGIN-INCREMENT
 >saf>interface>model-menu.lisp
 ABSOLUTE-X-ORIGIN
 >saf>interface>model-menu.lisp
 ABSOLUTE-Y-ORIGIN
 >saf>interface>model-menu.lisp
 PRESENT-DATA
 >saf>interface>model-menu.lisp
 DRAW-TICKS
Description: >saf>interface>model-menu.lisp
 None

2.1.2.15 DRAW-Y-TICKS

Type: >saf>interface>model-menu.lisp
Function
Arguments: (WINDOW &KEY (Y-LENGTH *Y-AXIS-LENGTH*)
 (START-Y *ABSOLUTE-Y-ORIGIN*) (GRANULARITY
 10))
Outputs:
Calls: *Y-AXIS-LENGTH*
 >saf>interface>model-menu.lisp
 ABSOLUTE-X-ORIGIN
 >saf>interface>model-menu.lisp
 ABSOLUTE-Y-ORIGIN
 >saf>interface>model-menu.lisp
Called by: PRESENT-DATA
 >saf>interface>model-menu.lisp
 DRAW-TICKS
 >saf>interface>model-menu.lisp
Description: None

2.1.2.16 DRAW-TICKS

Type: >saf>interface>model-menu.lisp
Macro
Arguments: (WINDOW)
Outputs:
Calls: DRAW-X-TICKS
>saf>interface>model-menu.lisp
DRAW-Y-TICKS
>saf>interface>model-menu.lisp
DRAW-TICKS
>saf>interface>model-menu.lisp
Called by: PRESENT-DATA
>saf>interface>model-menu.lisp
CLEAN-UP-AXIS-AND-REDRAW
>saf>interface>model-menu.lisp
DRAW-TICKS
>saf>interface>model-menu.lisp
Description: None

2.1.2.17 CLEAN-UP-AXIS-AND-REDRAW

Type: >saf>interface>model-menu.lisp
Macro
Arguments: (WINDOW)
Outputs:
Calls: DRAW-SCALE-BAR
>saf>interface>model-menu.lisp
DRAW-TICKS
>saf>interface>model-menu.lisp
CLEAN-UP-AXIS-AND-REDRAW
>saf>interface>model-menu.lisp
Called by: PRESENT-DATA
>saf>interface>model-menu.lisp
CLEAN-UP-AXIS-AND-REDRAW
>saf>interface>model-menu.lisp
Description: None

2.1.2.18 PRESENT-GRAPH

Type: >saf>interface>model-menu.lisp
Function
Arguments: (GRAPH WINDOW)
Outputs:
Calls: POINT
>saf>interface>model-menu.lisp
POINT
>saf>interface>model-menu.lisp
TABLE-TYPE
>saf>interface>model-menu.lisp
POINT
>saf>interface>model-menu.lisp
POINT
>saf>interface>model-menu.lisp
UP-DOWN-POINT
Called by: >saf>interface>model-menu.lisp
PRESENT-DATA
>saf>interface>model-menu.lisp
Description: None

2.1.2.19 CALCULATE-POINT-DATA

Type: >saf>interface>model-menu.lisp
Subst
Arguments: (X Y)
Outputs:
Calls: *DELTA-X*
>saf>interface>model-menu.lisp
DELTA-Y
>saf>interface>model-menu.lisp
ABSOLUTE-ORIGIN
>saf>interface>model-menu.lisp
Called by: (METHOD ALTER-PROBABILITY POINT)
>saf>interface>model-menu.lisp
(METHOD ALTER-RANGE POINT)
>saf>interface>model-menu.lisp
MAKE-GRAPH-GIVEN-POINTS
>saf>interface>model-menu.lisp
RESCALE-POINT-LIST
>saf>interface>model-menu.lisp
Description: None

2.1.2.20 CALCULATE-POINT-DATA-FROM-PIXEL-TO-REAL

Type: >saf>interface>model-menu.lisp
Subst
Arguments: (PIXEL-X PIXEL-Y)
Outputs:
Calls: *DELTA-X*
>saf>interface>model-menu.lisp
DELTA-Y
>saf>interface>model-menu.lisp
ABSOLUTE-ORIGIN
Called by: >saf>interface>model-menu.lisp
(METHOD DRAG-UP-DOWN POINT)
>saf>interface>model-menu.lisp
(METHOD DRAG POINT)
>saf>interface>model-menu.lisp
ADD-NEW-POINT
>saf>interface>model-menu.lisp
Description: None

2.1.2.21 POINT

Type: >saf>interface>model-menu.lisp
Flavor
Arguments: ()
Outputs:
Calls: None
Called by: None
Description: None

2.1.2.22 (METHOD DRAW POINT)

Type: >saf>interface>model-menu.lisp
Method
Arguments: (WINDOW &OPTIONAL NEW-X NEW-Y)
Outputs:
Calls: None
Called by: None
Description: None

2.1.2.23 (METHOD DRAW-WITH-CHECK POINT)

Type: >saf>interface>model-menu.lisp
Method
Arguments: (WINDOW)
Outputs:
Calls: None
Called by: None
Description: None

2.1.2.24 (METHOD ON-POINT? POINT)

Type: >saf>interface>model-menu.lisp
Method
Arguments: (X Y)
Outputs:
Calls: None
Called by: None
Description: None

2.1.2.25 (METHOD ERASE POINT)

Type: >saf>interface>model-menu.lisp
Method
Arguments: (WINDOW &OPTIONAL NEW-X NEW-Y)
Outputs:
Calls: None
Called by: None
Description: None

2.1.2.26 (METHOD EXPUNGE POINT)

Type: >saf>interface>model-menu.lisp
Method
Arguments: ()
Outputs:
Calls: MKATOM
>saf>sys>new-storage.lisp
CONCAT
>saf>sys>new-storage.lisp
MODE
>saf>interface>model-menu.lisp
GIMME-VAR-NAME-OF-CURRENT-GRAPH
>saf>interface>model-menu.lisp
GET-CURRENT-GRAPH-POINTS
>saf>interface>model-menu.lisp

Called by: None
 Description: None

2.1.3 Graphics manipulation functions

[; ; ; Code for DELETing, MOVing and INSERTing points in graphs]

Graph points can be deleted, inserted, and moved.

Inserting a new point entails finding first the pair in-between which the new point will be. The top-level point insertion functions are RECORD-NEW-POINT and ADD-NEW-POINT.

Moving a point relies on the :DRAG method. The top-level function is SELECT-AND-DRAG-POINT.

2.1.3.1 DELETE-POINT-IF-THERE

Type:	>saf>interface>model-menu.lisp
Arguments:	Function
Outputs:	(X Y WINDOW)
Calls:	POINT
	>saf>interface>model-menu.lisp
	MKATOM
	>saf>sys>new-storage.lisp
	CONCAT
	>saf>sys>new-storage.lisp
	MODE
	>saf>interface>model-menu.lisp
	TABULAR-DISPLAY-PANE
	>saf>interface>model-menu.lisp
	GIMME-VAR-NAME-OF-CURRENT-GRAPH
	>saf>interface>model-menu.lisp
	GET-CURRENT-GRAPH-POINTS
	>saf>interface>model-menu.lisp
	GIMME-VAR-NAME-OF-CURRENT-POINTS
	>saf>interface>model-menu.lisp
	GET-CURRENT-LINE-POINTS
	>saf>interface>model-menu.lisp
	POINT
	>saf>interface>model-menu.lisp
	PRESENT-TABLE
	>saf>interface>model-menu.lisp
Called by:	(PRESENTATION-MOUSE-HANDLER DELETE-POINT)
	No Source File Record
Description:	None

2.1.3.2 LAST-CAR

Type: >saf>interface>model-menu.lisp
Macro
Arguments: (LIST)
Outputs:
Calls: LAST-CAR
>saf>interface>model-menu.lisp
Called by: FIND-SURROUNDING-POINTS
>saf>interface>model-menu.lisp
LAST-CAR
>saf>interface>model-menu.lisp
Description: None

2.1.3.3 MAKE-GAP-BETWEEN-POINTS

Type: >saf>interface>model-menu.lisp
Subst
Arguments: (P1 P2 WINDOW)
Outputs:
Calls: DRAW-LINE-BETWEEN-POINTS
>saf>interface>model-menu.lisp
Called by: ADD-NEW-POINT
>saf>interface>model-menu.lisp
Description: None

2.1.3.4 DRAW-LINE-BETWEEN-POINTS

Type: >saf>interface>model-menu.lisp
Function
Arguments: (P1 P2 WINDOW)
Outputs:
Calls: None

Called by: (METHOD DRAG-UP-DOWN POINT)
>saf>interface>model-menu.lisp
(METHOD ERASE-POINT-AND-LINES POINT)
>saf>interface>model-menu.lisp
(METHOD DRAG POINT)
>saf>interface>model-menu.lisp
ADD-NEW-POINT
>saf>interface>model-menu.lisp
MAKE-GAP-BETWEEN-POINTS
>saf>interface>model-menu.lisp
Description: None

2.1.3.5 GET-POINT-RIGHT

Type: >saf>interface>model-menu.lisp
Subst
Arguments: (POINT GRAPH)
Outputs:
Calls: POINT
>saf>interface>model-menu.lisp
POINT
>saf>interface>model-menu.lisp
Called by: (METHOD DRAG-UP-DOWN POINT)
>saf>interface>model-menu.lisp
(METHOD ERASE-POINT-AND-LINES POINT)
>saf>interface>model-menu.lisp
(METHOD DRAG POINT)
>saf>interface>model-menu.lisp
FIND-SURROUNDING-POINTS
>saf>interface>model-menu.lisp
GET-POINT-LEFT
>saf>interface>model-menu.lisp
Description: None

2.1.3.6 GET-POINT-LEFT

Type:	>saf>interface>model-menu.lisp
Arguments:	Subst
Outputs:	(POINT GRAPH)
Calls:	POINT >saf>interface>model-menu.lisp GET-POINT-RIGHT >saf>interface>model-menu.lisp POINT >saf>interface>model-menu.lisp
Called by:	(METHOD DRAG-UP-DOWN POINT) >saf>interface>model-menu.lisp (METHOD ERASE-POINT-AND-LINES POINT) >saf>interface>model-menu.lisp (METHOD DRAG POINT) >saf>interface>model-menu.lisp FIND-SURROUNDING-POINTS >saf>interface>model-menu.lisp
Description:	None

2.1.3.7 FIND-SURROUNDING-POINTS

Type:	>saf>interface>model-menu.lisp
Arguments:	Function
Outputs:	(X Y)
Calls:	POINT >saf>interface>model-menu.lisp MKATOM >saf>sys>new-storage.lisp CONCAT >saf>sys>new-storage.lisp *MODE* >saf>interface>model-menu.lisp GIMME-VAR-NAME-OF-CURRENT-GRAPH >saf>interface>model-menu.lisp GET-CURRENT-GRAPH-POINTS >saf>interface>model-menu.lisp LAST-CAR >saf>interface>model-menu.lisp GET-POINT-RIGHT >saf>interface>model-menu.lisp GET-POINT-LEFT >saf>interface>model-menu.lisp POINT >saf>interface>model-menu.lisp
Called by:	ADD-NEW-POINT >saf>interface>model-menu.lisp
Description:	None

2.1.3.8 SPLICE-IN-LIST-AFTER

Type: >saf>interface>model-menu.lisp
 Function
 Arguments: (ELEMENT ELEMENT-AFTER LIST)
 Outputs:
 Calls: None
 Called by: RECORD-NEW-POINT
 >saf>interface>model-menu.lisp
 Description: None

2.1.3.9 SPLICE-IN-LIST-AT-POSITION

Type: >saf>interface>model-menu.lisp
 Function
 Arguments: (ELEMENT POSITION LIST)
 Outputs:
 Calls: None
 Called by: ADD-CORRESPONDING-NEW-POINT-IN-POINT-LIST
 >saf>interface>model-menu.lisp
 UPDATE-POINT-LIST
 >saf>interface>model-menu.lisp
 Description: None

2.1.3.10 UPDATE-POINT-LIST

Type: >saf>interface>model-menu.lisp
 Function
 Arguments: (OLD-R OLD-P ROW)
 Outputs:
 Calls: MKATOM
 >saf>sys>new-storage.lisp
 CONCAT
 >saf>sys>new-storage.lisp
 MODE
 >saf>interface>model-menu.lisp
 GIMME-VAR-NAME-OF-CURRENT-GRAPH
 >saf>interface>model-menu.lisp
 GET-CURRENT-GRAPH-POINTS
 >saf>interface>model-menu.lisp
 GIMME-VAR-NAME-OF-CURRENT-POINTS
 >saf>interface>model-menu.lisp
 GET-CURRENT-LINE-POINTS
 >saf>interface>model-menu.lisp
 SPLICE-IN-LIST-AT-POSITION
 >saf>interface>model-menu.lisp
 RESCALE-POINT-LIST
 >saf>interface>model-menu.lisp
 Called by: (METHOD ALTER-PROBABILITY POINT)

Description: >saf>interface>model-menu.lisp
(METHOD ALTER-RANGE POINT)
>saf>interface>model-menu.lisp
(METHOD DRAG-UP-DOWN POINT)
>saf>interface>model-menu.lisp
(METHOD DRAG POINT)
>saf>interface>model-menu.lisp
None

2.1.3.11 RESCALE-POINT-LIST

Type: >saf>interface>model-menu.lisp
Function
Arguments: (POINT-LIST)
Outputs:
Calls: POINT
>saf>interface>model-menu.lisp
TABLE-TYPE
>saf>interface>model-menu.lisp
X-AXIS-LENGTH
>saf>interface>model-menu.lisp
X-ORIGIN-INCREMENT
>saf>interface>model-menu.lisp
DELTA-X
>saf>interface>model-menu.lisp
DELTA-Y
>saf>interface>model-menu.lisp
ABSOLUTE-ORIGIN
>saf>interface>model-menu.lisp
CALCULATE-POINT-DATA
>saf>interface>model-menu.lisp
POINT
Called by: >saf>interface>model-menu.lisp
UPDATE-POINT-LIST
Description: >saf>interface>model-menu.lisp
None

2.1.3.12 ADD-CORRESPONDING-NEW-POINT-IN-POINT-LIST

Type: >saf>interface>model-menu.lisp
 Function
 Arguments: (X Y NEW-GRAPHICAL-INSTANCE)
 Outputs:
 Calls: MKATOM
 >saf>sys>new-storage.lisp
 CONCAT
 >saf>sys>new-storage.lisp
 MODE
 >saf>interface>model-menu.lisp
 TABULAR-DISPLAY-PANE
 >saf>interface>model-menu.lisp
 TRUNCATE-TO-N-POSITIONS
 >saf>interface>model-menu.lisp
 GIMME-VAR-NAME-OF-CURRENT-GRAPH
 >saf>interface>model-menu.lisp
 GET-CURRENT-GRAPH-POINTS
 >saf>interface>model-menu.lisp
 GIMME-VAR-NAME-OF-CURRENT-POINTS
 >saf>interface>model-menu.lisp
 GET-CURRENT-LINE-POINTS
 >saf>interface>model-menu.lisp
 SPLICE-IN-LIST-AT-POSITION
 >saf>interface>model-menu.lisp
 PRESENT-TABLE
 >saf>interface>model-menu.lisp
 Called by: ADD-NEW-POINT
 >saf>interface>model-menu.lisp
 Description: None

2.1.3.13 RECORD-NEW-POINT

Type: >saf>interface>model-menu.lisp
 Function
 Arguments: (POINT POINT-AFTER LIST)
 Outputs:
 Calls: POINT
 >saf>interface>model-menu.lisp
 MKATOM
 >saf>sys>new-storage.lisp
 CONCAT
 >saf>sys>new-storage.lisp
 MODE
 >saf>interface>model-menu.lisp
 GIMME-VAR-NAME-OF-CURRENT-GRAPH
 >saf>interface>model-menu.lisp
 SPLICE-IN-LIST-AFTER
 >saf>interface>model-menu.lisp
 POINT

Called by: >saf>interface>model-menu.lisp
ADD-NEW-POINT
Description: >saf>interface>model-menu.lisp
None

2.1.3.14 ADD-NEW-POINT

Type: >saf>interface>model-menu.lisp
Function
Arguments: (X Y WINDOW &KEY (UPDATE-TABLE T))
Outputs:
Calls: POINT
>saf>interface>model-menu.lisp
MKATOM
>saf>sys>new-storage.lisp
CONCAT
>saf>sys>new-storage.lisp
MODE
>saf>interface>model-menu.lisp
TRUNCATE-TO-N-POSITIONS
>saf>interface>model-menu.lisp
GIMME-VAR-NAME-OF-CURRENT-GRAPH
>saf>interface>model-menu.lisp
GET-CURRENT-GRAPH-POINTS
>saf>interface>model-menu.lisp
DELTA-X
>saf>interface>model-menu.lisp
DELTA-Y
>saf>interface>model-menu.lisp
ABSOLUTE-ORIGIN
>saf>interface>model-menu.lisp
CALCULATE-POINT-DATA-FROM-PIXEL-TO-REAL
>saf>interface>model-menu.lisp
MAKE-GAP-BETWEEN-POINTS
>saf>interface>model-menu.lisp
DRAW-LINE-BETWEEN-POINTS
>saf>interface>model-menu.lisp
FIND-SURROUNDING-POINTS
>saf>interface>model-menu.lisp
ADD-CORRESPONDING-NEW-POINT-IN-POINT-LIST
>saf>interface>model-menu.lisp

RECORD-NEW-POINT
 >saf>interface>model-menu.lisp
 POINT
 >saf>interface>model-menu.lisp
 Called by: (PRESENTATION-MOUSE-HANDLER INSERT-POINT-1)
 No Source File Record
 (PRESENTATION-MOUSE-HANDLER INSERT-POINT)
 No Source File Record
 Description: None

2.1.3.15 (METHOD DRAG POINT)

Type: >saf>interface>model-menu.lisp
 Arguments: Method
 Outputs: (WINDOW)
 Calls: POINT
 >saf>interface>model-menu.lisp
 MKATOM
 >saf>sys>new-storage.lisp
 CONCAT
 >saf>sys>new-storage.lisp
 MODE
 >saf>interface>model-menu.lisp
 PARAMETER-DISPLAY-PANE
 >saf>interface>model-menu.lisp
 GIMME-VAR-NAME-OF-CURRENT-GRAPH
 >saf>interface>model-menu.lisp
 GET-CURRENT-GRAPH-POINTS
 >saf>interface>model-menu.lisp
 DELTA-X
 >saf>interface>model-menu.lisp
 DELTA-Y
 >saf>interface>model-menu.lisp
 ABSOLUTE-ORIGIN
 >saf>interface>model-menu.lisp
 CALCULATE-POINT-DATA-FROM-PIXEL-TO-REAL
 >saf>interface>model-menu.lisp
 DRAW-LINE-BETWEEN-POINTS
 >saf>interface>model-menu.lisp
 GET-POINT-RIGHT
 >saf>interface>model-menu.lisp
 GET-POINT-LEFT
 >saf>interface>model-menu.lisp
 UPDATE-POINT-LIST
 >saf>interface>model-menu.lisp

POINT
>saf>interface>model-menu.lisp
PRESENT-DATA
>saf>interface>model-menu.lisp
Called by: None
Description: None

2.1.3.16 (METHOD ERASE-POINT-AND-LINES POINT)

>saf>interface>model-menu.lisp
Type: Method
Arguments: (WINDOW &OPTIONAL (DRAW-NEW-LINE? NIL))
Outputs:
Calls: POINT
>saf>interface>model-menu.lisp
MKATOM
>saf>sys>new-storage.lisp
CONCAT
>saf>sys>new-storage.lisp
MODE
>saf>interface>model-menu.lisp
GIMME-VAR-NAME-OF-CURRENT-GRAPH
>saf>interface>model-menu.lisp
GET-CURRENT-GRAPH-POINTS
>saf>interface>model-menu.lisp
DRAW-LINE-BETWEEN-POINTS
>saf>interface>model-menu.lisp
GET-POINT-RIGHT
>saf>interface>model-menu.lisp
GET-POINT-LEFT
>saf>interface>model-menu.lisp
POINT
>saf>interface>model-menu.lisp
Called by: None
Description: None

2.1.3.17 GET-POINT

Type: >saf>interface>model-menu.lisp
Subst
Arguments: (X Y)
Outputs:
Calls: MKATOM
>saf>sys>new-storage.lisp
CONCAT
>saf>sys>new-storage.lisp
MODE
>saf>interface>model-menu.lisp
GIMME-VAR-NAME-OF-CURRENT-GRAPH
>saf>interface>model-menu.lisp
GET-CURRENT-GRAPH-POINTS
>saf>interface>model-menu.lisp
Called by: SELECT-AND-DRAG-UP-DOWN-POINT
>saf>interface>model-menu.lisp
SELECT-AND-DRAG-POINT
>saf>interface>model-menu.lisp
Description: None

2.1.3.18 SELECT-AND-DRAG-POINT

Type: >saf>interface>model-menu.lisp
Function
Arguments: (X Y WINDOW)
Outputs:
Calls: MKATOM
>saf>sys>new-storage.lisp
CONCAT
>saf>sys>new-storage.lisp
MODE
>saf>interface>model-menu.lisp
GIMME-VAR-NAME-OF-CURRENT-GRAPH
>saf>interface>model-menu.lisp
GET-CURRENT-GRAPH-POINTS
>saf>interface>model-menu.lisp
GET-POINT
>saf>interface>model-menu.lisp
Called by: (PRESENTATION-MOUSE-HANDLER MOVE-POINT)
No Source File Record
Description: None

2.1.3.19 LIST-INSTANCE-VARIABLES

Type: >saf>interface>model-menu.lisp
Macro
Arguments: (INSTANCE)
Outputs:
Calls: LIST-INSTANCE-VARIABLES
>saf>interface>model-menu.lisp
Called by: MAKE-COPY-OF-INSTANCE-POINT
>saf>interface>model-menu.lisp
LIST-INSTANCE-VARIABLE-VALUES
>saf>interface>model-menu.lisp
CREATE-MAKE-INSTANCE-FORM-ARGS
>saf>interface>model-menu.lisp
LIST-INSTANCE-VARIABLES
>saf>interface>model-menu.lisp
Description: None

2.1.3.20 LIST-INSTANCE-VARIABLE-VALUES

Type: >saf>interface>model-menu.lisp
Function
Arguments: (INSTANCE)
Outputs:
Calls: LIST-INSTANCE-VARIABLES
>saf>interface>model-menu.lisp
Called by: MAKE-COPY-OF-INSTANCE-POINT
>saf>interface>model-menu.lisp
CREATE-MAKE-INSTANCE-FORM-ARGS
>saf>interface>model-menu.lisp
Description: see get-instance-variables

2.1.3.21 CREATE-MAKE-INSTANCE-FORM-ARGS

Type: >saf>interface>model-menu.lisp
Macro
Arguments: (INSTANCE)
Outputs:
Calls: LIST-INSTANCE-VARIABLES
>saf>interface>model-menu.lisp
LIST-INSTANCE-VARIABLE-VALUES
>saf>interface>model-menu.lisp
CREATE-MAKE-INSTANCE-FORM-ARGS
>saf>interface>model-menu.lisp
Called by: MAKE-COPY-OF-INSTANCE-POINT
>saf>interface>model-menu.lisp
CREATE-MAKE-INSTANCE-FORM-ARGS
>saf>interface>model-menu.lisp
Description: None

2.1.3.22 MAKE-COPY-OF-INSTANCE-POINT

Type: >saf>interface>model-menu.lisp
Function
Arguments: (PT &OPTIONAL (FLAVOR-NAME 'POINT))
Outputs:
Calls: POINT
>saf>interface>model-menu.lisp
LIST-INSTANCE-VARIABLES
>saf>interface>model-menu.lisp
LIST-INSTANCE-VARIABLE-VALUES
>saf>interface>model-menu.lisp
CREATE-MAKE-INSTANCE-FORM-ARGS
>saf>interface>model-menu.lisp
POINT
Called by: >saf>interface>model-menu.lisp
COPY-POINT-LIST
>saf>interface>model-menu.lisp
Description: None

2.1.3.23 *GRAPH-BEFORE-LAST-STEP*

Type: >saf>interface>model-menu.lisp
Parameter
Arguments: ()
Outputs:
Calls: None
Called by: SET-CURRENT-MODEL
>saf>interface>model-menu.lisp
GRAPH-UNDO
>saf>interface>model-menu.lisp
GO-BACK-TO-PREVIOUS-STEP
>saf>interface>model-menu.lisp
UPDATE-BACKTRACKING-CAPABILITY
>saf>interface>model-menu.lisp
Description: None

2.1.3.24 UPDATE-BACKTRACKING-CAPABILITY

Type:	>saf>interface>model-menu.lisp
Arguments:	Function
Outputs:	(&OPTIONAL (PTS (GET-CURRENT-GRAPH-POINTS)))
Calls:	MKATOM >saf>sys>new-storage.lisp CONCAT >saf>sys>new-storage.lisp *MODE* >saf>interface>model-menu.lisp GIMME-VAR-NAME-OF-CURRENT-GRAPH >saf>interface>model-menu.lisp GET-CURRENT-GRAPH-POINTS >saf>interface>model-menu.lisp *GRAPH-BEFORE-LAST-STEP* >saf>interface>model-menu.lisp COPY-POINT-LIST >saf>interface>model-menu.lisp
Called by:	(PRESENTATION-MOUSE-HANDLER INSERT-POINT-1) No Source File Record (PRESENTATION-MOUSE-HANDLER INSERT-POINT) No Source File Record (PRESENTATION-MOUSE-HANDLER REPLACE- PROBABILITY) No Source File Record (PRESENTATION-MOUSE-HANDLER REPLACE-RANGE) No Source File Record (PRESENTATION-MOUSE-HANDLER MOVE-UP-DOWN- POINT) No Source File Record (PRESENTATION-MOUSE-HANDLER DELETE-POINT) No Source File Record (PRESENTATION-MOUSE-HANDLER MOVE-POINT) No Source File Record
Description:	None

2.1.3.25 COPY-POINT-LIST

Type: >saf>interface>model-menu.lisp
Function
Arguments: (LIST)
Outputs:
Calls: MAKE-COPY-OF-INSTANCE-POINT
>saf>interface>model-menu.lisp
Called by: UPDATE-BACKTRACKING-CAPABILITY
>saf>interface>model-menu.lisp
Description: None

2.1.3.26 GO-BACK-TO-PREVIOUS-STEP

Type: >saf>interface>model-menu.lisp
Function
Arguments: (WINDOW)
Outputs:
Calls: MKATOM
>saf>sys>new-storage.lisp
CONCAT
>saf>sys>new-storage.lisp
MODE
>saf>interface>model-menu.lisp
GIMME-VAR-NAME-OF-CURRENT-GRAPH
>saf>interface>model-menu.lisp
GRAPH-BEFORE-LAST-STEP
>saf>interface>model-menu.lisp
PRESENT-DATA
>saf>interface>model-menu.lisp
Called by: GRAPH-UNDO
>saf>interface>model-menu.lisp
Description: None

2.1.3.27 REVERT-TO-FACTORY-VERSION

Type: >saf>interface>model-menu.lisp
Function
Arguments: ()
Outputs:
Calls: *HOST-FOR-CONFIG-DATA*
>saf>sys>interim-model.lisp
TABLE-TYPE
>saf>interface>model-menu.lisp
INDIRECT-FIRE-DAMAGE-FILE-NAME
>saf>interface>model-menu.lisp
DIRECT-FIRE-DAMAGE-FILE-NAME
>saf>interface>model-menu.lisp
EXTRACT-FILE-NAME
>saf>interface>model-menu.lisp
READ-HIT-DATA
>saf>interface>model-menu.lisp
READ-GROUND-DETECTION-DATA
>saf>interface>model-menu.lisp
READ-AIR-DETECTION-DATA
>saf>interface>model-menu.lisp
READ-INDIRECT-FIRE-DAMAGE-DATA
>saf>interface>model-menu.lisp
READ-DIRECT-FIRE-DAMAGE-DATA
>saf>interface>model-menu.lisp
Called by: GRAPH-UNDO
Description: >saf>interface>model-menu.lisp
None

2.1.3.28 EXTRACT-FILE-NAME

Type: >saf>interface>model-menu.lisp
Function
Arguments: (UNIX-FILE-SPEC)
Outputs:
Calls: None
Called by: REVERT-TO-FACTORY-VERSION
>saf>interface>model-menu.lisp
Description: None

2.1.3.29 GRAPH-UNDO

Type:	>saf>interface>model-menu.lisp
Arguments:	Function
Outputs:	(WINDOW)
Calls:	*TABLE-TYPE* >saf>interface>model-menu.lisp *WEAPON-NAME* >saf>interface>model-menu.lisp *PARAMETER-DISPLAY-PANE* >saf>interface>model-menu.lisp *IND-FIR-TABLES* >saf>interface>model-menu.lisp *DIR-FIR-TABLES* >saf>interface>model-menu.lisp *HIT-TABLES* >saf>interface>model-menu.lisp *GRAPH-BEFORE-LAST-STEP* >saf>interface>model-menu.lisp GO-BACK-TO-PREVIOUS-STEP >saf>interface>model-menu.lisp REVERT-TO-FACTORY-VERSION >saf>interface>model-menu.lisp GET-TABLE >saf>interface>model-menu.lisp SET-CURRENT-MODEL >saf>interface>model-menu.lisp
Called by:	(METHOD COM-UNDO-IND-INTERNAL MODEL-MENU) No Source File Record (METHOD COM-UNDO-DIR-INTERNAL MODEL-MENU) No Source File Record (METHOD COM-UNDO-INTERNAL MODEL-MENU) No Source File Record
Description:	None

2.1.4 Frame and screen configurations.

[;;; FRAME code]

This CSU contains the code that define the Models Editor Frame and its configurations, as well as the individual panes that appear within it, as seen on the screen.

The frame itself is defined as a program framework with three configurations: MAIN-WITH-LOGO, which is used for Hit and Detection data; MAIN-WITH-LOGO-DIRECT-DAMAGE, used for DIRECT DAMAGE data, and MAIN-WITH-LOGO-INDIRECT-DAMAGE. They differ mainly in the commands available in the top menu bar. The program framework is called MODEL-MENU, for historical reasons. See the Symbolics manuals for details on program frameworks and how they are defined by DEFINE-PROGRAM-FRAMEWORK.

The TOP-LEVEL method of MODEL-MENU sets up a bunch of global variables, provides mouse documentation data, and starts up the frame. It also creates *HITMODELS-DICTIONARY*, using the function MAKE-HITMODELS-DICTIONARY, from the File input/output CSU, section 2.1.7.

2.1.4.1 MODEL-MENU

Type:	>saf>interface>model-menu.lisp
Arguments:	DEFINE-PROGRAM-FRAMEWORK
Outputs:	()
Calls:	None
Called by:	DEFINE-MODEL-MENU-COMMAND >saf>patch>saf-6>saf-6-5.lisp SELECT-MODEL-MENU >saf>interface>model-menu.lisp LEFT-ON-FIREPOWER-AT-PAR-1-MODEL-MENU-MENU-COMMAND >saf>interface>model-menu.lisp LEFT-ON-MOBILITY-AT-PAR-1-MODEL-MENU-MENU-COMMAND >saf>interface>model-menu.lisp LEFT-ON-CATASTROPHIC-AT-PAR-1-MODEL-MENU-MENU-COMMAND >saf>interface>model-menu.lisp LEFT-ON-MOVING-AT-PAR-MODEL-MENU-MENU-COMMAND >saf>interface>model-menu.lisp LEFT-ON-STATIONARY-AT-PAR-MODEL-MENU-MENU-COMMAND >saf>interface>model-menu.lisp LEFT-ON-HULL-AT-PAR-MODEL-MENU-MENU-COMMAND >saf>interface>model-menu.lisp LEFT-ON-UNDO-AT-COM-IND-MODEL-MENU-MENU-COMMAND >saf>interface>model-menu.lisp LEFT-ON-WRITE TABLE-AT-COM-IND-MODEL-MENU-

MENU-COMMAND
>saf>interface>model-menu.lisp
LEFT-ON-READ TABLE-AT-COM-IND-MODEL-MENU-
MENU-COMMAND
>saf>interface>model-menu.lisp
LEFT-ON-SELECT MODEL-AT-COM-IND-MODEL-
MENU-MENU-COMMAND
>saf>interface>model-menu.lisp
LEFT-ON-UNDO-AT-COM-DIR-MODEL-MENU-MENU-
COMMAND
>saf>interface>model-menu.lisp
LEFT-ON-WRITE TABLE-AT-COM-DIR-MODEL-MENU-
MENU-COMMAND
>saf>interface>model-menu.lisp
LEFT-ON-READ TABLE-AT-COM-DIR-MODEL-MENU-
MENU-COMMAND
>saf>interface>model-menu.lisp
LEFT-ON-SELECT ASPECT-AT-COM-DIR-MODEL-
MENU-MENU-COMMAND
>saf>interface>model-menu.lisp
LEFT-ON-SELECT MODEL-AT-COM-DIR-MODEL-
MENU-MENU-COMMAND
>saf>interface>model-menu.lisp
LEFT-ON-UNDO-AT-COM-MODEL-MENU-MENU-
COMMAND
>saf>interface>model-menu.lisp
LEFT-ON-WRITE TABLE-AT-COM-MODEL-MENU-
MENU-COMMAND
>saf>interface>model-menu.lisp
LEFT-ON-READ TABLE-AT-COM-MODEL-MENU-
MENU-COMMAND
>saf>interface>model-menu.lisp
LEFT-ON-SELECT MODEL-AT-COM-MODEL-MENU-
MENU-COMMAND
>saf>interface>model-menu.lisp
None

Description:

2.1.4.2 (METHOD MAKE-INSTANCE MODEL-MENU AFTER)

Type: >saf>interface>model-menu.lisp
Method
Arguments: (&REST INIT-ARGS)
Outputs:
Calls: *MODEL-FRAME*
>saf>interface>model-menu.lisp
Called by: None
Description: None

2.1.4.3 *MODEL-IO*

Type: >saf>interface>model-menu.lisp
Parameter
Arguments: ()
Outputs:
Calls: None
Called by: (METHOD TOP-LEVEL MODEL-MENU)
>saf>interface>model-menu.lisp
Description: None

2.1.4.4 *HITMODELS-DICTIONARY*

Type: >saf>interface>model-menu.lisp
Parameter
Arguments: ()
Outputs:
Calls: None
Called by: WRITE-HIT-DATA-FILE
>saf>interface>model-menu.lisp
(METHOD TOP-LEVEL MODEL-MENU)
>saf>interface>model-menu.lisp
Description: None

2.1.4.5 (METHOD TOP-LEVEL MODEL-MENU)

Type: >saf>interface>model-menu.lisp
 Method
 Arguments: (&REST ARGS)
 Outputs:
 Calls: MOUSE-FLIP-SCREEN
 >saf>ui>mouse-interface.lisp
 FIND-MOUSE
 >saf>ui>mouse-interface.lisp
 SAF
 >saf>ui>frame.lisp
 PARAMETER-DISPLAY-PANE
 >saf>interface>model-menu.lisp
 TABULAR-DISPLAY-PANE
 >saf>interface>model-menu.lisp
 MODEL-FRAME
 >saf>interface>model-menu.lisp
 DISPLAY-LOGO-PANE
 >saf>interface>model-menu.lisp
 TABULAR-LOGO-PANE
 >saf>interface>model-menu.lisp
 MODEL-IO
 >saf>interface>model-menu.lisp
 HITMODELS-DICTIONARY
 >saf>interface>model-menu.lisp
 MAKE-DOCUMENTATION
 >saf>interface>model-menu.lisp
 MAKE-HITMODELS-DICTIONARY
 >saf>interface>model-menu.lisp
 Called by: None
 Description: None

2.1.4.6 MAKE-DOCUMENTATION

Type: >saf>interface>model-menu.lisp
 Function
 Arguments: (COMMAND-STRING DOCUMENTATION-STRING
 &OPTIONAL (FRAME *MODEL-FRAME*))
 Outputs:
 Calls: *MODEL-FRAME*
 >saf>interface>model-menu.lisp
 Called by: (METHOD TOP-LEVEL MODEL-MENU)
 >saf>interface>model-menu.lisp
 Description: None

2.1.5 Command processor commands

[; ; Command Processor User Commands]

The user commands in the top menu bar of all the Models Editors are provided via the system's Command Processor facility. They are created via the system's construct `DEFINE-[Program-Frame-Name]-COMMAND`, in our case `DEFINE-MODEL-MENU-COMMAND`. The core of the functionality is provided by the function call within, e.g.: `(USER-SELECT-MODEL ...)`, `(SELECT-TABLE)`, etc.

2.1.5.1 (COM-SELECT-MODEL MENU-ACCELERATOR Select Model MENU-LEVEL COM)

Type:	>saf>interface>model-menu.lisp
Arguments:	DEFINE-MODEL-MENU-COMMAND
Outputs:	()
Calls:	None
Called by:	None
Description:	None

2.1.5.2 (COM-READ-TABLE MENU-ACCELERATOR Read Table MENU-LEVEL COM)

Type:	>saf>interface>model-menu.lisp
Arguments:	DEFINE-MODEL-MENU-COMMAND
Outputs:	()
Calls:	None
Called by:	None
Description:	None

2.1.5.3 (COM-WRITE-TABLE MENU-ACCELERATOR Write Table MENU-LEVEL COM)

Type:	>saf>interface>model-menu.lisp
Arguments:	DEFINE-MODEL-MENU-COMMAND
Outputs:	()
Calls:	None
Called by:	None
Description:	None

2.1.5.4 (COM-UNDO MENU-ACCELERATOR Undo MENU-LEVEL COM)

Type: >saf>interface>model-menu.lisp
Arguments: DEFINE-MODEL-MENU-COMMAND
Outputs: ()
Calls: None
Called by: None
Description: None

2.1.5.5 (COM-SELECT-MODEL-DIR MENU-ACCELERATOR Select Model MENU-LEVEL COM-DIR)

Type: >saf>interface>model-menu.lisp
Arguments: DEFINE-MODEL-MENU-COMMAND
Outputs: ()
Calls: None
Called by: None
Description: None

2.1.5.6 (COM-SELECT-ASPECT-DIR MENU-ACCELERATOR Select Aspect MENU-LEVEL COM-DIR)

Type: >saf>interface>model-menu.lisp
Arguments: DEFINE-MODEL-MENU-COMMAND
Outputs: ()
Calls: None
Called by: None
Description: None

2.1.5.7 (COM-READ-TABLE-DIR MENU-ACCELERATOR Read Table MENU-LEVEL COM-DIR)

Type: >saf>interface>model-menu.lisp
Arguments: DEFINE-MODEL-MENU-COMMAND
Outputs: ()
Calls: None
Called by: None
Description: None

**2.1.5.8 (COM-WRITE-TABLE-DIR MENU-ACCELERATOR Write Table
MENU-LEVEL COM-DIR)**

Type: >saf>interface>model-menu.lisp
DEFINE-MODEL-MENU-COMMAND
Arguments: ()
Outputs:
Calls: None
Called by: None
Description: None

**2.1.5.9 (COM-UNDO-DIR MENU-ACCELERATOR Undo MENU-
LEVEL COM-DIR)**

Type: >saf>interface>model-menu.lisp
DEFINE-MODEL-MENU-COMMAND
Arguments: ()
Outputs:
Calls: None
Called by: None
Description: None

**2.1.5.10 (COM-SELECT-MODEL-IND MENU-ACCELERATOR Select
Model MENU-LEVEL COM-IND)**

Type: >saf>interface>model-menu.lisp
DEFINE-MODEL-MENU-COMMAND
Arguments: ()
Outputs:
Calls: None
Called by: None
Description: None

**2.1.5.11 (COM-READ-TABLE-IND MENU-ACCELERATOR Read Table
MENU-LEVEL COM-IND)**

Type: >saf>interface>model-menu.lisp
DEFINE-MODEL-MENU-COMMAND
Arguments: ()
Outputs:
Calls: None
Called by: None
Description: None

2.1.5.12 (COM-WRITE-TABLE-IND MENU-ACCELERATOR Write Table MENU-LEVEL COM-IND)

Type: >saf>interface>model-menu.lisp
DEFINE-MODEL-MENU-COMMAND
Arguments: ()
Outputs:
Calls: None
Called by: None
Description: None

2.1.5.13 (COM-UNDO-IND MENU-ACCELERATOR Undo MENU-LEVEL COM-IND)

Type: >saf>interface>model-menu.lisp
DEFINE-MODEL-MENU-COMMAND
Arguments: ()
Outputs:
Calls: None
Called by: None
Description: None

2.1.5.14 (COM-HULL MENU-ACCELERATOR Hull MENU-LEVEL PAR)

Type: >saf>interface>model-menu.lisp
DEFINE-MODEL-MENU-COMMAND
Arguments: ()
Outputs:
Calls: None
Called by: None
Description: None

2.1.5.15 (COM-STATIONARY MENU-ACCELERATOR Stationary MENU-LEVEL PAR)

Type: >saf>interface>model-menu.lisp
DEFINE-MODEL-MENU-COMMAND
Arguments: ()
Outputs:
Calls: None
Called by: None
Description: None

2.1.5.16 (COM-MOVING MENU-ACCELERATOR Moving MENU-LEVEL PAR)

Type: >saf>interface>model-menu.lisp
Arguments: DEFINE-MODEL-MENU-COMMAND
Outputs: ()
Calls: None
Called by: None
Description: None

2.1.5.17 (COM-CATASTROPHIC MENU-ACCELERATOR Catastrophic MENU-LEVEL PAR-1)

Type: >saf>interface>model-menu.lisp
Arguments: DEFINE-MODEL-MENU-COMMAND
Outputs: ()
Calls: None
Called by: None
Description: None

2.1.5.18 (COM-MOBILITY MENU-ACCELERATOR Mobility MENU-LEVEL PAR-1)

Type: >saf>interface>model-menu.lisp
Arguments: DEFINE-MODEL-MENU-COMMAND
Outputs: ()
Calls: None
Called by: None
Description: None

2.1.5.19 (COM-FIREPOWER MENU-ACCELERATOR Firepower MENU-LEVEL PAR-1)

Type: >saf>interface>model-menu.lisp
 Arguments: DEFINE-MODEL-MENU-COMMAND
 Outputs: ()
 Calls: None
 Called by: None
 Description: None

2.1.5.20 MAKE-GRAPH-GIVEN-POINTS

Type: >saf>interface>model-menu.lisp
 Arguments: Function
 Outputs: (LINE)
 Calls: POINT
 >saf>interface>model-menu.lisp
 LINE
 >saf>cm>line.lisp
 LINE
 >saf>cm>line.lisp
 LINE
 >saf>cm>line.lisp
 TABLE-TYPE
 >saf>interface>model-menu.lisp
 X-AXIS-LENGTH
 >saf>interface>model-menu.lisp
 X-ORIGIN-INCREMENT
 >saf>interface>model-menu.lisp
 DELTA-X
 >saf>interface>model-menu.lisp
 DELTA-Y
 >saf>interface>model-menu.lisp
 ABSOLUTE-ORIGIN
 >saf>interface>model-menu.lisp
 CALCULATE-POINT-DATA
 >saf>interface>model-menu.lisp
 POINT
 >saf>interface>model-menu.lisp
 Called by: SET-CURRENT-MODEL
 >saf>interface>model-menu.lisp
 CHANGE-MODE
 >saf>interface>model-menu.lisp
 Description: None

2.1.5.21 SWITCH-HIGHLIGHT

Type:	>saf>interface>model-menu.lisp
Arguments:	Function
Outputs:	()
Calls:	HIGHLIGHT-BUTTON-1 >saf>ui>frame-utils.lisp CONCAT >saf>sys>new-storage.lisp *OLD-SELECTION* >saf>interface>model-menu.lisp *MODE* >saf>interface>model-menu.lisp
Called by:	CHANGE-MODE >saf>interface>model-menu.lisp
Description:	None

2.1.5.22 CHANGE-MODE

Type:	>saf>interface>model-menu.lisp
Arguments:	Function
Outputs:	(TYPE WINDOW)
Calls:	*HULL-GRAPH* >saf>interface>model-menu.lisp *STATIONARY-GRAPH* >saf>interface>model-menu.lisp *MOVING-GRAPH* >saf>interface>model-menu.lisp *CATASTROPHIC-GRAPH* >saf>interface>model-menu.lisp *MOBILITY-GRAPH* >saf>interface>model-menu.lisp *FIREPOWER-GRAPH* >saf>interface>model-menu.lisp *HULL-POINTS* >saf>interface>model-menu.lisp *STATIONARY-POINTS* >saf>interface>model-menu.lisp *MOVING-POINTS* >saf>interface>model-menu.lisp *CATASTROPHIC-POINTS* >saf>interface>model-menu.lisp

MOBILITY-POINTS
 >saf>interface>model-menu.lisp
 FIREPOWER-POINTS
 >saf>interface>model-menu.lisp
 MODE
 >saf>interface>model-menu.lisp
 MAKE-GRAPH-GIVEN-POINTS
 >saf>interface>model-menu.lisp
 SWITCH-HIGHLIGHT
 >saf>interface>model-menu.lisp
 PRESENT-DATA
 >saf>interface>model-menu.lisp
 Called by: (METHOD COM-FIREPOWER-INTERNAL MODEL-MENU)
 No Source File Record
 (METHOD COM-MOBILITY-INTERNAL MODEL-MENU)
 No Source File Record
 (METHOD COM-CATASTROPHIC-INTERNAL MODEL-MENU)
 No Source File Record
 (METHOD COM-MOVING-INTERNAL MODEL-MENU)
 No Source File Record
 (METHOD COM-STATIONARY-INTERNAL MODEL-MENU)
 No Source File Record
 (METHOD COM-HULL-INTERNAL MODEL-MENU)
 No Source File Record
 Description: None

2.1.5.23 SELECT-HOST

Type: >saf>interface>model-menu.lisp
 Arguments: Function
 Outputs: ()
 Calls: MENU-CHOOSE
 >saf>sys>utilities.lisp
 HOST-FOR-CONFIG-DATA
 >saf>sys>interim-model.lisp
 GET-HOSTS-WITH-SIMNET-SERVICE
 >saf>network>vars.lisp
 CONCAT
 >saf>sys>new-storage.lisp

Called by: READ-CONFIGURATION-FILE
>saf>interface>formations.lisp
MAKE-HITMODELS-DICTIONARY
>saf>interface>model-menu.lisp
SELECT-TABLE
>saf>interface>model-menu.lisp
Description: None

2.1.5.24 SELECT-TABLE

Type: >saf>interface>model-menu.lisp
Function
Arguments: ()
Outputs:
Calls: *HOST-FOR-CONFIG-DATA*
>saf>sys>interim-model.lisp
SELECT-HOST
>saf>interface>model-menu.lisp
DISPATCH-TO-TABLE
>saf>interface>model-menu.lisp
Called by: (METHOD COM-READ-TABLE-IND-INTERNAL MODEL-
MENU)
MENU) No Source File Record
(METHOD COM-READ-TABLE-DIR-INTERNAL MODEL-
MENU) No Source File Record
(METHOD COM-READ-TABLE-INTERNAL MODEL-MENU)
Description: No Source File Record
None

2.1.5.25 DISPATCH-TO-TABLE

Type: >saf>interface>model-menu.lisp
Function
Arguments: ()
Outputs:
Calls: READ-HIT-DATA
>saf>interface>model-menu.lisp
READ-GROUND-DETECTION-DATA
>saf>interface>model-menu.lisp
READ-AIR-DETECTION-DATA
>saf>interface>model-menu.lisp
READ-INDIRECT-FIRE-DAMAGE-DATA
>saf>interface>model-menu.lisp
READ-DIRECT-FIRE-DAMAGE-DATA
>saf>interface>model-menu.lisp
Called by: SELECT-TABLE
>saf>interface>model-menu.lisp
Description: None

2.1.5.26 USER-SELECT-MODEL

Type:	>saf>interface>model-menu.lisp
Arguments:	Function
Outputs:	(WINDOW &OPTIONAL (TABLE-LIST *HIT-TABLES*))
Calls:	NAME
	>saf>sysdcl.lisp
	TABLE-TYPE
	>saf>interface>model-menu.lisp
	IND-FIR-TABLES
	>saf>interface>model-menu.lisp
	DIR-FIR-TABLES
	>saf>interface>model-menu.lisp
	HIT-TABLES
	>saf>interface>model-menu.lisp
	GET-TABLE
	>saf>interface>model-menu.lisp
	REMEMBER-TABLE
	>saf>interface>model-menu.lisp
	GET-NEW-WEAPON-NAME
	>saf>interface>model-menu.lisp
	SET-CURRENT-MODEL
	>saf>interface>model-menu.lisp
Called by:	(METHOD COM-SELECT-MODEL-IND-INTERNAL MODEL-
MENU)	
	No Source File Record
	(METHOD COM-SELECT-MODEL-DIR-INTERNAL MODEL-
MENU)	
	No Source File Record
	(METHOD COM-SELECT-MODEL-INTERNAL MODEL-MENU)
	No Source File Record
Description:	None

2.1.5.27 USER-SELECT-ASPECT

Type:	>saf>interface>model-menu.lisp
Arguments:	Function
Outputs:	(WINDOW)
Calls:	*PART* >saf>interface>model-menu.lisp *ASPECT* >saf>interface>model-menu.lisp *WEAPON-NAME* >saf>interface>model-menu.lisp *DIR-FIR-TABLES* >saf>interface>model-menu.lisp GET-TABLE >saf>interface>model-menu.lisp REMEMBER-TABLE >saf>interface>model-menu.lisp SET-CURRENT-MODEL >saf>interface>model-menu.lisp
Called by:	(METHOD COM-SELECT-ASPECT-DIR-INTERNAL MODEL-MENU)
Description:	No Source File Record None

2.1.6 Panes and their presentations.

[; ; PANES, their flavor definitions, and related code]

The panes within the MODEL-MENU frame are either system-provided or are specifically defined. This unit contains the code to define two special panes, as well as the presentations and presentation-actions that impart the rest of the user functionality. The two special panes are PARAMETER-DISPLAY-PANE, where the graph is displayed, and MS-PANE, where the numeric table representation is displayed.

The code is organized around each pane definition: a pane is first defined as a flavor, followed by presentation type definitions and the actions associated with the presentation.

In the PARAMETER-DISPLAY-PANE, the presentation-type POINT (there is also a flavor POINT, a different kind of object) has associated with it the presentation-action MOVE-POINT, which calls the top-level point-moving function SELECT-AND-DRAG-POINT (see above).

It is worth noticing that while Hit, and Detection points have two degrees of freedom of movement (up-down and left-right), Direct Damage points are given in terms of three fixed angles of incidence (30°, 60°, and 90°) and therefore can be moved only up-down. This is reflected in special presentation-types and actions for them.

In the MS-PANE, two presentation types are defined, PROBABILITY and RANGE, with their associated actions REPLACE-PROBABILITY and REPLACE-RANGE. This last action calls the function REPLACE-RANGE, which sends the message :XREPLACE-OBJECT-VALUE to the selected pane in the editor. This message is a rewrite of the symbolics dynamic-window method :REPLACE-OBJECT-VALUE, and is defined a few forms below where it is used, just after the comment

```
;;; Altered version, ...
```

Notice that the defmethod call is preceeded by "dw::", indicating that the method is to be created in the dynamic-windows package:

```
dw::
(defmethod (:xreplace-object-value dynamic-window) ... )
```

This unit also contains the cp:define-command definition for COM-MODELS, the command processor command to start the models editor. This command calls SELECT-MODEL-MENU, the top level function for the models editor. These forms appear under the following comment:

```
;;; Define command-processor command for the models editor
```

2.1.6.1 DYNAMIC-WINDOW-WITHOUT-SCROLL-BARS

	>saf>interface>model-menu.lisp
Type:	Flavor
Arguments:	()
Outputs:	
Calls:	None
Called by:	None
Description:	None

2.1.6.2 MS-DYNAMIC-WINDOW-PANE

	>saf>interface>model-menu.lisp
Type:	Flavor
Arguments:	()
Outputs:	
Calls:	None
Called by:	None
Description:	None

2.1.6.3 PARAMETER-DISPLAY-PANE

Type:	>saf>interface>model-menu.lisp
Arguments:	Flavor
Outputs:	()
Calls:	None
Called by:	None
Description:	None

2.1.6.4 POINT

Type:	>saf>interface>model-menu.lisp
Arguments:	DEFINE-PRESENTATION-TYPE
Outputs:	()
Calls:	None
Called by:	(PRESENTATION-MOUSE-HANDLER DELETE-POINT) No Source File Record (PRESENTATION-MOUSE-HANDLER MOVE-POINT) No Source File Record PRESENT-TABLE >saf>interface>model-menu.lisp (METHOD DRAG-UP-DOWN POINT) >saf>interface>model-menu.lisp (METHOD ERASE-POINT-AND-LINES POINT) >saf>interface>model-menu.lisp (METHOD DRAG POINT) >saf>interface>model-menu.lisp RECORD-NEW-POINT >saf>interface>model-menu.lisp RESCALE-POINT-LIST >saf>interface>model-menu.lisp FIND-SURROUNDING-POINTS >saf>interface>model-menu.lisp GET-POINT-LEFT >saf>interface>model-menu.lisp GET-POINT-RIGHT >saf>interface>model-menu.lisp DELETE-POINT-IF-THERE >saf>interface>model-menu.lisp PRESENT-GRAPH >saf>interface>model-menu.lisp

SORT-CMS

```
>saf>cm>overlay.lisp
(METHOD MOVE-CONTROL-MEASURE ZONE)
>saf>cm>zone.lisp
(METHOD COPY ZONE)
>saf>cm>zone.lisp
(METHOD MOVE-CONTROL-MEASURE AREA)
>saf>cm>area.lisp
(METHOD COPY AREA)
>saf>cm>area.lisp
(METHOD ORTHOGONALIZE GENERIC-AREA)
>saf>cm>generic-area.lisp
(METHOD INSERT-POINT-BEFORE GENERIC-AREA)
>saf>cm>generic-area.lisp
(METHOD INSERT-POINT-AFTER GENERIC-AREA)
>saf>cm>generic-area.lisp
(METHOD DELETE-POINT GENERIC-AREA)
>saf>cm>generic-area.lisp
(METHOD MOVE-POINT GENERIC-AREA)
>saf>cm>generic-area.lisp
(METHOD PAINT-NAME GENERIC-AREA)
>saf>cm>generic-area.lisp
(METHOD INITIALIZE-POINTS GENERIC-AREA)
>saf>cm>generic-area.lisp
(METHOD COPY LINE)
>saf>cm>line.lisp
(METHOD MOVE-CONTROL-MEASURE LINE)
>saf>cm>line.lisp
(METHOD INSERT-POINT-BEFORE LINE)
>saf>cm>line.lisp
(METHOD INSERT-POINT-AFTER LINE)
>saf>cm>line.lisp
(METHOD DELETE-POINT LINE)
>saf>cm>line.lisp
(METHOD MOVE-POINT LINE)
>saf>cm>line.lisp
(METHOD ORTHOGONALIZE LINE)
>saf>cm>line.lisp
(METHOD INITIALIZE-POINTS LINE)
>saf>cm>line.lisp
MAKE-POINT
>saf>cm>point.lisp
(PRESENTATION-MOUSE-HANDLER CM-POINT-
GESTURE)
No Source File Record
(METHOD DELETE-POINT CM-POINT)
>saf>cm>point.lisp
(METHOD MOVE-POINT CM-POINT)
>saf>cm>point.lisp
(METHOD COPY ROUTE)
>saf>cm>route.lisp
(METHOD INSERT-POINT-BEFORE ROUTE)
>saf>cm>route.lisp
(METHOD INSERT-POINT-AFTER ROUTE)
```

```
>saf>cm>route.lisp
(METHOD DELETE-POINT ROUTE)
>saf>cm>route.lisp
(METHOD MOVE-POINT ROUTE)
>saf>cm>route.lisp
(METHOD ORTHOGONALIZE ROUTE)
>saf>cm>route.lisp
(METHOD INITIALIZE-POINTS ROUTE)
>saf>cm>route.lisp
SKIRT-LAKE
>saf>cm>water-avoidance.lisp
FINAL-RELAX-POINTS
>saf>cm>water-avoidance.lisp
RELAX-POINTS-AUX
>saf>cm>water-avoidance.lisp
PRUNE-TO-POINT
>saf>cm>water-avoidance.lisp
MOUSE-ON-BRIDGE-APPROACH-POINT
>saf>cm>road-routes.lisp
GET-ROAD-SEGMENT-POINT
>saf>cm>road-routes.lisp
GET-ROAD-POINT
>saf>cm>road-routes.lisp
(PRESENTATION-MOUSE-HANDLER CONTROL-
MEASURE-GESTURE)
No Source File Record
(NCWHOPPER INSERT-POINT-BEFORE CONTROL-
MEASURE)
No Source File Record
(NCWHOPPER INSERT-POINT-AFTER CONTROL-
MEASURE)
No Source File Record
(NCWHOPPER DELETE-POINT CONTROL-MEASURE)
No Source File Record
(NCWHOPPER MOVE-POINT CONTROL-MEASURE)
No Source File Record
(METHOD INTERVENE SIMNET-AGENT LAND)
>saf>objects>intervention.lisp
(METHOD INTERVENE SIMNET-AGENT GO-TO-
LOCATION)
>saf>objects>intervention.lisp
FACE-DIRECTION
>saf>objects>simnet-agent.lisp
```

SEND-POINT
 >saf>network>commands.lisp
 BOMB-BUTTON
 >saf>network>commands.lisp
 PRESENT-TABLE
 >saf>interface>model-menu.lisp
 MAKE-GRAPH-GIVEN-POINTS
 >saf>interface>model-menu.lisp
 MAKE-COPY-OF-INSTANCE-POINT
 >saf>interface>model-menu.lisp
 ADD-NEW-POINT
 >saf>interface>model-menu.lisp
 PRESENT-GRAPH
 >saf>interface>model-menu.lisp
 (METHOD SEND-OVERLAY-TO-SIMHOST OVERLAY)
 >saf>cm>overlay.lisp
 CISS-FOR-CONTROL-MEASURE
 >saf>sys>interim-model.lisp
 Description: None

2.1.6.5 MOVE-POINT

Type: >saf>interface>model-menu.lisp
 Arguments: DEFINE-PRESENTATION-ACTION
 Outputs: ()
 Calls: None
 Called by: (PRESENTATION-MOUSE-HANDLER CM-POINT-GESTURE)
 No Source File Record
 (PRESENTATION-MOUSE-HANDLER CONTROL-MEASURE-GESTURE)
 No Source File Record
 Description: None

2.1.6.6 DELETE-POINT

Type: >saf>interface>model-menu.lisp
 Arguments: DEFINE-PRESENTATION-ACTION
 Outputs: ()
 Calls: None

Called by: (PRESENTATION-MOUSE-HANDLER CM-POINT-GESTURE)
No Source File Record

(PRESENTATION-MOUSE-HANDLER CONTROL-
MEASURE-GESTURE)
No Source File Record

Description: None

2.1.6.7 INSERT-POINT

Type: >saf>interface>model-menu.lisp
DEFINE-PRESENTATION-ACTION

Arguments: ()

Outputs:

Calls: None

Called by: None

Description: None

2.1.6.8 INSERT-POINT-1

Type: >saf>interface>model-menu.lisp
DEFINE-PRESENTATION-ACTION

Arguments: ()

Outputs:

Calls: None

Called by: None

Description: None

2.1.6.9 UP-DOWN-POINT

Type: >saf>interface>model-menu.lisp
DEFINE-PRESENTATION-TYPE

Arguments: ()

Outputs:

Calls: None

Called by: (PRESENTATION-MOUSE-HANDLER MOVE-UP-DOWN-
POINT)
No Source File Record
PRESENT-GRAPH
>saf>interface>model-menu.lisp

Description: None

2.1.6.10 MOVE-UP-DOWN-POINT

Type: >saf>interface>model-menu.lisp
Arguments: DEFINE-PRESENTATION-ACTION
Outputs: ()
Calls: None
Called by: None
Description: None

2.1.6.11 SELECT-AND-DRAG-UP-DOWN-POINT

Type: >saf>interface>model-menu.lisp
Arguments: Function
Outputs: (X Y WINDOW)
Calls: MKATOM
>saf>sys>new-storage.lisp
CONCAT
>saf>sys>new-storage.lisp
MODE
>saf>interface>model-menu.lisp
GIMME-VAR-NAME-OF-CURRENT-GRAPH
>saf>interface>model-menu.lisp
GET-CURRENT-GRAPH-POINTS
>saf>interface>model-menu.lisp
GET-POINT
>saf>interface>model-menu.lisp
Called by: (PRESENTATION-MOUSE-HANDLER MOVE-UP-DOWN-
POINT)
Description: No Source File Record
None

2.1.6.12 (METHOD DRAG-UP-DOWN POINT)

Type: >saf>interface>model-menu.lisp
Arguments: Method
Outputs: (WINDOW)
Calls: POINT
>saf>interface>model-menu.lisp
MKATOM
>saf>sys>new-storage.lisp

```

CONCAT
>saf>sys>new-storage.lisp
*MODE*
>saf>interface>model-menu.lisp
*PARAMETER-DISPLAY-PANE*
>saf>interface>model-menu.lisp
GIMME-VAR-NAME-OF-CURRENT-GRAPH
>saf>interface>model-menu.lisp
GET-CURRENT-GRAPH-POINTS
>saf>interface>model-menu.lisp
*DELTA-X*
>saf>interface>model-menu.lisp
*DELTA-Y*
>saf>interface>model-menu.lisp
*ABSOLUTE-ORIGIN*
>saf>interface>model-menu.lisp
CALCULATE-POINT-DATA-FROM-PIXEL-TO-REAL
>saf>interface>model-menu.lisp
DRAW-LINE-BETWEEN-POINTS
>saf>interface>model-menu.lisp
GET-POINT-RIGHT
>saf>interface>model-menu.lisp
GET-POINT-LEFT
>saf>interface>model-menu.lisp
UPDATE-POINT-LIST
>saf>interface>model-menu.lisp
POINT
>saf>interface>model-menu.lisp
PRESENT-DATA
>saf>interface>model-menu.lisp

```

Called by: None
 Description: None

2.1.6.13 MS-PANE

```

Type: >saf>interface>model-menu.lisp
Arguments: Flavor
Outputs: ()
Calls: None
Called by: None
Description: None

```

2.1.6.14 RANGE

Type: >saf>interface>model-menu.lisp
DEFINE-PRESENTATION-TYPE
Arguments: ()
Outputs:
Calls: None
Called by: (PRESENTATION-MOUSE-HANDLER REPLACE-RANGE)
No Source File Record
PRESENT-TABLE
>saf>interface>model-menu.lisp
(PROPERTY RANGE DEFTYPE)
No Source File Record
Description: None

2.1.6.15 PROBABILITY

Type: >saf>interface>model-menu.lisp
DEFINE-PRESENTATION-TYPE
Arguments: ()
Outputs:
Calls: None
Called by: (PRESENTATION-MOUSE-HANDLER REPLACE-
PROBABILITY)
No Source File Record
PRESENT-TABLE
>saf>interface>model-menu.lisp
(PROPERTY PROBABILITY DEFTYPE)
No Source File Record
Description: None

2.1.6.16 REPLACE-RANGE

Type: >saf>interface>model-menu.lisp
DEFINE-PRESENTATION-ACTION
Arguments: ()
Outputs:
Calls: None
Called by: (PRESENTATION-MOUSE-HANDLER REPLACE-RANGE)
No Source File Record
Description: None

2.1.6.17 REPLACE-RANGE

Type: >saf>interface>model-menu.lisp
Function
Arguments: (OLD-RANGE PRESENTATION WINDOW)
Outputs:
Calls: None
Called by: (PRESENTATION-MOUSE-HANDLER REPLACE-RANGE)
No Source File Record
Description: None

2.1.6.18 (METHOD ALTER-RANGE POINT)

Type: >saf>interface>model-menu.lisp
Method
Arguments: (NEW-RANGE)
Outputs:
Calls: MKATOM
>saf>sys>new-storage.lisp
CONCAT
>saf>sys>new-storage.lisp
MODE
>saf>interface>model-menu.lisp
PARAMETER-DISPLAY-PANE
>saf>interface>model-menu.lisp
GIMME-VAR-NAME-OF-CURRENT-GRAPH
>saf>interface>model-menu.lisp
GET-CURRENT-GRAPH-POINTS
>saf>interface>model-menu.lisp
DELTA-X
>saf>interface>model-menu.lisp
DELTA-Y
>saf>interface>model-menu.lisp
ABSOLUTE-ORIGIN
>saf>interface>model-menu.lisp
CALCULATE-POINT-DATA
>saf>interface>model-menu.lisp
UPDATE-POINT-LIST
>saf>interface>model-menu.lisp
PRESENT-DATA
>saf>interface>model-menu.lisp
Called by: None
Description: None

2.1.6.19 REPLACE-PROBABILITY

Type: >saf>interface>model-menu.lisp
 Arguments: DEFINE-PRESENTATION-ACTION
 Outputs: ()
 Calls: None
 Called by: (PRESENTATION-MOUSE-HANDLER REPLACE-PROBABILITY)
 No Source File Record
 Description: None

2.1.6.20 REPLACE-PROBABILITY

Type: >saf>interface>model-menu.lisp
 Arguments: Function
 Outputs: (PRESENTATION WINDOW)
 Calls: None
 Called by: (PRESENTATION-MOUSE-HANDLER REPLACE-PROBABILITY)
 No Source File Record
 Description: None

2.1.6.21 (METHOD ALTER-PROBABILITY POINT)

Type: >saf>interface>model-menu.lisp
 Arguments: Method
 Outputs: (NEW-PROBABILITY)
 Calls: MKATOM
 >saf>sys>new-storage.lisp
 CONCAT
 >saf>sys>new-storage.lisp
 MODE
 >saf>interface>model-menu.lisp
 PARAMETER-DISPLAY-PANE
 >saf>interface>model-menu.lisp
 GIMME-VAR-NAME-OF-CURRENT-GRAPH
 >saf>interface>model-menu.lisp
 GET-CURRENT-GRAPH-POINTS
 >saf>interface>model-menu.lisp
 DELTA-X
 >saf>interface>model-menu.lisp
 DELTA-Y
 >saf>interface>model-menu.lisp
 ABSOLUTE-ORIGIN
 >saf>interface>model-menu.lisp
 CALCULATE-POINT-DATA

Called by: >saf>interface>model-menu.lisp
Description: UPDATE-POINT-LIST
>saf>interface>model-menu.lisp
PRESENT-DATA
>saf>interface>model-menu.lisp
None
None

2.1.6.22 (METHOD XREPLACE-OBJECT-VALUE DYNAMIC-WINDOW)

Type: >saf>interface>model-menu.lisp
Arguments: Method
Outputs: (DISPLAYED-PRESENTATION)
Calls: None
Called by: None
Description: None

2.1.6.23 COM-MODELS

Type: >saf>interface>model-menu.lisp
Arguments: CP Command
Outputs: ()
Calls: SELECT-MODEL-MENU
>saf>interface>model-menu.lisp
Called by: None
Description: None

2.1.6.24 SELECT-MODEL-MENU

Type: >saf>interface>model-menu.lisp
Arguments: Function
Outputs: ()

Calls: *PARAMETER-DISPLAY-PANE*
 >saf>interface>model-menu.lisp
 PARAMETER-DISPLAY-PANE
 >saf>interface>model-menu.lisp
 MODEL-MENU
 >saf>interface>model-menu.lisp
Called by: COM-MODELS
 >saf>interface>model-menu.lisp
Description: None

2.1.7 File input/output and database preparation

[; ; FILE I/O and database preparation for internal use]

Since the various data are stored in different files in different formats there are specialized READ-... and a WRITE- functions for each: Hit, Detection, Indirect Damage, and Direct Damage.

The file data is organized into internal tables (e.g.: HIT-TABLE, IND-FIR-TABLE, and DIR-FIR-TABLE), and specialized functions to gather or extract chunks of data are provided.

Functions such as EXTRACT-RANGE-AND-DICE-NUMBERS are necessary because probability distributions are represented in different ways on the Simhost and in the model-menu code.

This unit also includes the function MAKE-HITMODELS-DICTIONARY, which creates an upper-case to mixed-case translation table so that upper-case Lisp symbols can be restored to their original mixed-case form before being written to the Simhost.

2.1.7.1 HIT-TABLE

Type: >saf>interface>model-menu.lisp
 DEFSTRUCT
Arguments: ()
Outputs:
Calls: None
Called by: BUILD-A-HIT-TABLE
 >saf>interface>model-menu.lisp
 HIT-TABLE-P
 >saf>interface>model-menu.lisp
 COPY-HIT-TABLE
 >saf>interface>model-menu.lisp
 MAKE-HIT-TABLE
 >saf>interface>model-menu.lisp
Description: None

2.1.7.2 IND-FIR-TABLE

Type: >saf>interface>model-menu.lisp
DEFSTRUCT
Arguments: ()
Outputs:
Calls: None
Called by: BUILD-A-IND-FIR-TABLE
>saf>interface>model-menu.lisp
IND-FIR-TABLE-P
>saf>interface>model-menu.lisp
COPY-IND-FIR-TABLE
>saf>interface>model-menu.lisp
MAKE-IND-FIR-TABLE
>saf>interface>model-menu.lisp
Description: None

2.1.7.3 DIR-FIR-TABLE

Type: >saf>interface>model-menu.lisp
DEFSTRUCT
Arguments: ()
Outputs:
Calls: None
Called by: BUILD-A-DIR-FIR-TABLE
>saf>interface>model-menu.lisp
DIR-FIR-TABLE-P
>saf>interface>model-menu.lisp
COPY-DIR-FIR-TABLE
>saf>interface>model-menu.lisp
MAKE-DIR-FIR-TABLE
>saf>interface>model-menu.lisp
Description: None

2.1.7.4 MAKE-SIMHOST-READTABLE

Type: >saf>interface>model-menu.lisp
Function
Arguments: ()
Outputs:
Calls: None

Called by: (METHOD REPLACE-ELEMENT OBJECT-MS-PANE)
>saf>interface>object-menu.lisp
(METHOD TOP-LEVEL OBJECT-MENU)
>saf>interface>object-menu.lisp
MAKE-OBJECT-FILE-DICTIONARY
>saf>interface>object-menu.lisp
REVERT-TO-FACTORY-FILE
>saf>interface>object-menu.lisp
MAKE-HITMODELS-DICTIONARY
>saf>interface>model-menu.lisp
Description: None

2.1.7.5 *SAVE-READTABLE*

Type: >saf>interface>model-menu.lisp
Parameter
Arguments: ()
Outputs:
Calls: None
Called by: None
Description: None

2.1.7.6 MAKE-HITMODELS-DICTIONARY

Type: >saf>interface>model-menu.lisp
Function
Arguments: ()
Outputs:
Calls: WITH-AUTOMATIC-LOGIN
>saf>sys>macros.lisp
WITH-OPEN-FILE-ON-BUTTERFLY
>saf>sys>macros.lisp
HANDLE-LOGIN
>saf>sys>macros.lisp
HOST-FOR-CONFIG-DATA
>saf>sys>interim-model.lisp
HIT-FILE-NAME
>saf>interface>model-menu.lisp
SELECT-HOST
>saf>interface>model-menu.lisp
MAKE-SIMHOST-READTABLE
>saf>interface>model-menu.lisp
ELIMINATE-DUPPLICATES
>saf>interface>model-menu.lisp

Called by: MAKE-DICTIONARY-PAIRS
>saf>interface>model-menu.lisp
(METHOD TOP-LEVEL MODEL-MENU)
>saf>interface>model-menu.lisp
Description: None

2.1.7.7 ELIMINATE-DUPLICATES

Type: >saf>interface>model-menu.lisp
Function
Arguments: (LIST-OF-PAIRS)
Outputs:
Calls: None
Called by: GET-FORMATION-CHOICES
>saf>interface>formations.lisp
GET-VEHICLE-CHOICES
>saf>interface>formations.lisp
MAKE-OBJECT-FILE-DICTIONARY
>saf>interface>object-menu.lisp
MAKE-HITMODELS-DICTIONARY
>saf>interface>model-menu.lisp
Description: None

2.1.7.8 MAKE-DICTIONARY-PAIRS

Type: >saf>interface>model-menu.lisp
Function
Arguments: (FORMS)
Outputs:
Calls: MKATOM
>saf>sys>new-storage.lisp
CONCAT
>saf>sys>new-storage.lisp
Called by: MAKE-DICTIONARY-PAIRS
>saf>interface>model-menu.lisp
MAKE-OBJECT-FILE-DICTIONARY
>saf>interface>object-menu.lisp
MAKE-DICTIONARY-PAIRS
>saf>interface>model-menu.lisp
MAKE-HITMODELS-DICTIONARY
>saf>interface>model-menu.lisp
Description: None

2.1.7.9 COPY-LIST-ALL-LEVELS

Type: >saf>interface>model-menu.lisp
Function
Arguments: (LST)
Outputs:
Calls: COPY-LIST-ALL-LEVELS
>saf>interface>model-menu.lisp
Called by: (METHOD TOP-LEVEL CONFIGURATION-MENU)
>saf>interface>formations.lisp
(METHOD TOP-LEVEL OBJECT-MENU)
>saf>interface>object-menu.lisp
REVERT-TO-FACTORY-FILE
>saf>interface>object-menu.lisp
COPY-LIST-ALL-LEVELS
>saf>interface>model-menu.lisp
Description: None

2.1.7.10 RESTORE-UPLOWCASE

Type: >saf>interface>model-menu.lisp
Function
Arguments: (FORMS DICTIONARY)
Outputs:
Calls: RESTORE-UPLOWCASE
>saf>interface>model-menu.lisp
LOOKUP-DICTIONARY
>saf>interface>model-menu.lisp
Called by: WRITE-HIT-DATA-FILE
>saf>interface>model-menu.lisp
RESTORE-UPLOWCASE
>saf>interface>model-menu.lisp
Description: None

2.1.7.11 LOOKUP-DICTIONARY

Type: >saf>interface>model-menu.lisp
Function
Arguments: (TOKEN DICTIONARY)
Outputs:
Calls: None
Called by: RESTORE-UPLOWCASE
>saf>interface>model-menu.lisp
Description: None

2.1.7.12 READ-HIT-DATA

Type:	>saf>interface>model-menu.lisp
Arguments:	Function
Outputs:	(&OPTIONAL (FILE-NAME NIL))
Calls:	 *HOST-FOR-CONFIG-DATA* >saf>sys>interim-model.lisp *TABLE-TYPE* >saf>interface>model-menu.lisp *WEAPON-NAME* >saf>interface>model-menu.lisp *PARAMETER-DISPLAY-PANE* >saf>interface>model-menu.lisp *HIT-FILE-NAME* >saf>interface>model-menu.lisp *HIT-TABLES* >saf>interface>model-menu.lisp READ-HIT-DATA-FILE >saf>interface>model-menu.lisp GET-TABLE >saf>interface>model-menu.lisp SET-CURRENT-MODEL >saf>interface>model-menu.lisp REVERT-TO-FACTORY-VERSION >saf>interface>model-menu.lisp DISPATCH-TO-TABLE >saf>interface>model-menu.lisp
Called by:	None
Description:	None

2.1.7.13 READ-HIT-DATA-FILE

Type:	>saf>interface>model-menu.lisp
Arguments:	Function
Outputs:	(FILE-NAME)

Calls: WITH-AUTOMATIC-LOGIN
>saf>sys>macros.lisp
WITH-OPEN-FILE-ON-BUTTERFLY
>saf>sys>macros.lisp
HANDLE-LOGIN
>saf>sys>macros.lisp
WEAPON-NAME
>saf>interface>model-menu.lisp
READ-HIT-DATA-FILE-AUX
>saf>interface>model-menu.lisp
Called by: READ-HIT-DATA
>saf>interface>model-menu.lisp
Description: read data file and return a list of data structures

2.1.7.14 READ-HIT-DATA-FILE-AUX

Type: >saf>interface>model-menu.lisp
Function
Arguments: (HIT-DATA)
Outputs:
Calls: READ-HIT-DATA-FILE-AUX
>saf>interface>model-menu.lisp
BUILD-A-HIT-TABLE
>saf>interface>model-menu.lisp
Called by: READ-HIT-DATA-FILE-AUX
>saf>interface>model-menu.lisp
READ-HIT-DATA-FILE
>saf>interface>model-menu.lisp
Description: None

2.1.7.15 READ-GROUND-DETECTION-DATA

Type: >saf>interface>model-menu.lisp
Function
Arguments: (&OPTIONAL (FILE-NAME NIL))
Outputs:
Calls: *HOST-FOR-CONFIG-DATA*
>saf>sys>interim-model.lisp
TABLE-TYPE
>saf>interface>model-menu.lisp
WEAPON-NAME
>saf>interface>model-menu.lisp
PARAMETER-DISPLAY-PANE
>saf>interface>model-menu.lisp

Called by: *DETECTION-FILE-NAME*
 >saf>interface>model-menu.lisp
 HIT-TABLES
 >saf>interface>model-menu.lisp
 READ-GROUND-DATA-FROM-FILE
 >saf>interface>model-menu.lisp
 GET-TABLE
 >saf>interface>model-menu.lisp
 SET-CURRENT-MODEL
 >saf>interface>model-menu.lisp
 REVERT-TO-FACTORY-VERSION
 >saf>interface>model-menu.lisp
 DISPATCH-TO-TABLE
 >saf>interface>model-menu.lisp
 Description: None

2.1.7.16 READ-GROUND-DATA-FROM-FILE

Type: >saf>interface>model-menu.lisp
 Arguments: Function
 Outputs: (FILE-NAME)
 Calls: WITH-AUTOMATIC-LOGIN
 >saf>sys>macros.lisp
 WITH-OPEN-FILE-ON-BUTTERFLY
 >saf>sys>macros.lisp
 HANDLE-LOGIN
 >saf>sys>macros.lisp
 WEAPON-NAME
 >saf>interface>model-menu.lisp
 BUILD-A-HIT-TABLE
 >saf>interface>model-menu.lisp
 Called by: READ-GROUND-DETECTION-DATA
 >saf>interface>model-menu.lisp
 Description: \D,#TD1Ps-T [Begin using 006 escapes]\1 0 (NIL 0)
 (NIL:BOLD NIL) "CPTFONTCB")read data file and return a
 list of data structures\0

2.1.7.17 READ-AIR-DETECTION-DATA

Type: >saf>interface>model-menu.lisp
 Function
 Arguments: (&OPTIONAL (FILE-NAME NIL))
 Outputs:
 Calls: *HOST-FOR-CONFIG-DATA*
 >saf>sys>interim-model.lisp
 TABLE-TYPE
 >saf>interface>model-menu.lisp
 WEAPON-NAME
 >saf>interface>model-menu.lisp
 PARAMETER-DISPLAY-PANE
 >saf>interface>model-menu.lisp
 DETECTION-FILE-NAME
 >saf>interface>model-menu.lisp
 HIT-TABLES
 >saf>interface>model-menu.lisp
 READ-AIR-DATA-FROM-FILE
 >saf>interface>model-menu.lisp
 GET-TABLE
 >saf>interface>model-menu.lisp
 SET-CURRENT-MODEL
 >saf>interface>model-menu.lisp
 REVERT-TO-FACTORY-VERSION
 >saf>interface>model-menu.lisp
 DISPATCH-TO-TABLE
 >saf>interface>model-menu.lisp
 Description: None

2.1.7.18 READ-AIR-DATA-FROM-FILE

Type: >saf>interface>model-menu.lisp
 Function
 Arguments: (FILE-NAME)
 Outputs:
 Calls: WITH-AUTOMATIC-LOGIN
 >saf>sys>macros.lisp
 WITH-OPEN-FILE-ON-BUTTERFLY
 >saf>sys>macros.lisp
 HANDLE-LOGIN
 >saf>sys>macros.lisp
 WEAPON-NAME
 >saf>interface>model-menu.lisp
 BUILD-A-HIT-TABLE
 >saf>interface>model-menu.lisp
 READ-AIR-DETECTION-DATA
 >saf>interface>model-menu.lisp
 Called by:
 Description: \read data file and return a list of data structures\0

2.1.7.19 READ-INDIRECT-FIRE-DAMAGE-DATA

Type:	>saf>interface>model-menu.lisp
Arguments:	Function
Outputs:	(&OPTIONAL (FILE-NAME NIL))
Calls:	WITH-AUTOMATIC-LOGIN >saf>sys>macros.lisp WITH-OPEN-FILE-ON-BUTTERFLY >saf>sys>macros.lisp HANDLE-LOGIN >saf>sys>macros.lisp *HOST-FOR-CONFIG-DATA* >saf>sys>interim-model.lisp CONCAT >saf>sys>new-storage.lisp *TABLE-TYPE* >saf>interface>model-menu.lisp *WEAPON-NAME* >saf>interface>model-menu.lisp *PARAMETER-DISPLAY-PANE* >saf>interface>model-menu.lisp *INDIRECT-FIRE-DAMAGE-FILE-NAME* >saf>interface>model-menu.lisp *IF-VEHICLE* >saf>interface>model-menu.lisp *IND-FIR-TABLES* >saf>interface>model-menu.lisp *INDIRECT-FIRE-DAMAGE-DATA* >saf>interface>model-menu.lisp FETCH-IF-DATA >saf>interface>model-menu.lisp GET-TABLE >saf>interface>model-menu.lisp SET-CURRENT-MODEL >saf>interface>model-menu.lisp REVERT-TO-FACTORY-VERSION >saf>interface>model-menu.lisp DISPATCH-TO-TABLE >saf>interface>model-menu.lisp
Called by:	None
Description:	None

2.1.7.20 FETCH-IF-DATA

Type: >saf>interface>model-menu.lisp
Function
Arguments: (VEHICLE-TYPE)
Outputs:
Calls: CONCAT
>saf>sys>new-storage.lisp
INDIRECT-FIRE-DAMAGE-DATA
>saf>interface>model-menu.lisp
BUILD-A-IND-FIR-TABLE
Called by: >saf>interface>model-menu.lisp
READ-INDIRECT-FIRE-DAMAGE-DATA
>saf>interface>model-menu.lisp
Description: None

2.1.7.21 READ-INDIRECT-FIRE-DAMAGE-DATA-FROM-FILE

Type: >saf>interface>model-menu.lisp
Function
Arguments: (FILE-NAME)
Outputs:
Calls: WITH-AUTOMATIC-LOGIN
>saf>sys>macros.lisp
WITH-OPEN-FILE-ON-BUTTERFLY
>saf>sys>macros.lisp
HANDLE-LOGIN
>saf>sys>macros.lisp
WEAPON-NAME
>saf>interface>model-menu.lisp
BUILD-A-IND-FIR-TABLE
>saf>interface>model-menu.lisp
Called by: None
Description: \read data file and return a list of data structures\0

2.1.7.22 READ-DIRECT-FIRE-DAMAGE-DATA

Type:	>saf>interface>model-menu.lisp
Arguments:	Function
Outputs:	(&OPTIONAL (FILE-NAME NIL))
Calls:	WITH-AUTOMATIC-LOGIN >saf>sys>macros.lisp WITH-OPEN-FILE-ON-BUTTERFLY >saf>sys>macros.lisp HANDLE-LOGIN >saf>sys>macros.lisp *HOST-FOR-CONFIG-DATA* >saf>sys>interim-model.lisp CONCAT >saf>sys>new-storage.lisp *PART* >saf>interface>model-menu.lisp *ASPECT* >saf>interface>model-menu.lisp *TABLE-TYPE* >saf>interface>model-menu.lisp *WEAPON-NAME* >saf>interface>model-menu.lisp *PARAMETER-DISPLAY-PANE* >saf>interface>model-menu.lisp *DIRECT-FIRE-DAMAGE-FILE-NAME* >saf>interface>model-menu.lisp *DF-VEHICLE* >saf>interface>model-menu.lisp *DIR-FIR-TABLES* >saf>interface>model-menu.lisp *DIRECT-FIRE-DAMAGE-DATA* >saf>interface>model-menu.lisp FETCH-DF-DATA >saf>interface>model-menu.lisp GET-TABLE >saf>interface>model-menu.lisp SET-CURRENT-MODEL >saf>interface>model-menu.lisp REVERT-TO-FACTORY-VERSION >saf>interface>model-menu.lisp DISPATCH-TO-TABLE >saf>interface>model-menu.lisp
Called by:	
Description:	None

2.1.7.23 FETCH-DF-DATA

Type: >saf>interface>model-menu.lisp
Function
Arguments: (VEHICLE-TYPE)
Outputs:
Calls: CONCAT
>saf>sys>new-storage.lisp
DIRECT-FIRE-DAMAGE-DATA
>saf>interface>model-menu.lisp
BUILD-A-DIR-FIR-TABLE
>saf>interface>model-menu.lisp
Called by: READ-DIRECT-FIRE-DAMAGE-DATA
>saf>interface>model-menu.lisp
Description: None

2.1.7.24 READ-DIRECT-FIRE-DAMAGE-DATA-FROM-FILE

Type: >saf>interface>model-menu.lisp
Function
Arguments: (FILE-NAME)
Outputs:
Calls: WITH-AUTOMATIC-LOGIN
>saf>sys>macros.lisp
WITH-OPEN-FILE-ON-BUTTERFLY
>saf>sys>macros.lisp
HANDLE-LOGIN
>saf>sys>macros.lisp
BUILD-A-DIR-FIR-TABLE
>saf>interface>model-menu.lisp
Called by: None
Description: \read data file and return a list of data structures\0

2.1.7.25 BUILD-A-HIT-TABLE

Type: >saf>interface>model-menu.lisp
Function
Arguments: (HIT-DATA)
Outputs:
Calls: HIT-TABLE
>saf>interface>model-menu.lisp
Called by: READ-AIR-DATA-FROM-FILE
>saf>interface>model-menu.lisp
READ-GROUND-DATA-FROM-FILE
>saf>interface>model-menu.lisp
READ-HIT-DATA-FILE-AUX
>saf>interface>model-menu.lisp
Description: translates butterfly format of hit data to a hit-table structure

2.1.7.26 BUILD-A-DIR-FIR-TABLE

Type: >saf>interface>model-menu.lisp
Function
Arguments: (DIR-FIR-DATA PART ROUND)
Outputs:
Calls: DIR-FIR-TABLE
Called by: >saf>interface>model-menu.lisp
READ-DIRECT-FIRE-DAMAGE-DATA-FROM-FILE
>saf>interface>model-menu.lisp
FETCH-DF-DATA
Description: >saf>interface>model-menu.lisp
structure translates butterfly format of direct fire damage data to a dir-fir-table

2.1.7.27 BUILD-A-IND-FIR-TABLE

Type: >saf>interface>model-menu.lisp
Function
Arguments: (IND-FIR-DATA)
Outputs:
Calls: IND-FIR-TABLE
Called by: >saf>interface>model-menu.lisp
READ-INDIRECT-FIRE-DAMAGE-DATA-FROM-FILE
>saf>interface>model-menu.lisp
FETCH-IF-DATA
Description: >saf>interface>model-menu.lisp
table structure translates butterfly format of indirect fire damage data to a ind-fir-

2.1.7.28 WRITE-CORRESPONDING-TABLE

Type: >saf>interface>model-menu.lisp
Function
Arguments: ()
Outputs:
Calls: *TABLE-TYPE*
>saf>interface>model-menu.lisp
WRITE-HIT-DATA
>saf>interface>model-menu.lisp
WRITE-DETECTION-DATA
>saf>interface>model-menu.lisp
WRITE-INDIRECT-FIRE-DAMAGE-DATA
>saf>interface>model-menu.lisp
WRITE-DIRECT-FIRE-DAMAGE-DATA
>saf>interface>model-menu.lisp
Called by: (METHOD COM-WRITE-TABLE-IND-INTERNAL MODEL-MENU)
No Source File Record

MENU) (METHOD COM-WRITE-TABLE-DIR-INTERNAL MODEL-
 No Source File Record
 (METHOD COM-WRITE-TABLE-INTERNAL MODEL-MENU)
 No Source File Record
 Description: None

2.1.7.29 *DEVELOPERS-FLG*

Type: >saf>interface>model-menu.lisp
 Parameter
 Arguments: ()
 Outputs:
 Calls: None
 Called by: WRITE-CONFIGURATION-FILE
 >saf>interface>formations.lisp
 WRITE-OBJECT-FILE
 >saf>interface>object-menu.lisp
 WRITE-DIRECT-FIRE-DAMAGE-DATA-FILE
 >saf>interface>model-menu.lisp
 WRITE-INDIRECT-FIRE-DAMAGE-DATA-FILE
 >saf>interface>model-menu.lisp
 WRITE-DETECTION-DATA-FILE
 >saf>interface>model-menu.lisp
 WRITE-HIT-DATA-FILE
 >saf>interface>model-menu.lisp
 Description: None

2.1.7.30 WRITE-HIT-DATA

Type: >saf>interface>model-menu.lisp
 Function
 Arguments: ()
 Outputs:
 Calls: *HOST-FOR-CONFIG-DATA*
 >saf>sys>interim-model.lisp
 HIT-FILE-NAME
 >saf>interface>model-menu.lisp
 WRITE-HIT-DATA-FILE
 >saf>interface>model-menu.lisp
 Called by: WRITE-CORRESPONDING-TABLE
 >saf>interface>model-menu.lisp
 Description: None

2.1.7.31 WRITE-HIT-DATA-FILE

Type: >saf>interface>model-menu.lisp
Function
Arguments: (FILE-NAME)
Outputs:
Calls: WITH-AUTOMATIC-LOGIN
>saf>sys>macros.lisp
WITH-OPEN-FILE-ON-BUTTERFLY
>saf>sys>macros.lisp
HANDLE-LOGIN
>saf>sys>macros.lisp
HIT-TABLES
>saf>interface>model-menu.lisp
HITMODELS-DICTIONARY
>saf>interface>model-menu.lisp
RESTORE-UPLOWCASE
>saf>interface>model-menu.lisp
DEVELOPERS-FLG
>saf>interface>model-menu.lisp
GATHER-DATABASE
>saf>interface>model-menu.lisp
REMEMBER-CURRENT-HIT-TABLE
>saf>interface>model-menu.lisp
Called by: WRITE-HIT-DATA
Description: >saf>interface>model-menu.lisp
write all hit-tables to the passed file

2.1.7.32 WRITE-DETECTION-DATA

Type: >saf>interface>model-menu.lisp
Function
Arguments: ()
Output:
Calls: *HOST-FOR-CONFIG-DATA*
>saf>sys>interim-model.lisp
DETECTION-FILE-NAME
>saf>interface>model-menu.lisp
WRITE-DETECTION-DATA-FILE
>saf>interface>model-menu.lisp
Called by: WRITE-CORRESPONDING-TABLE
>saf>interface>model-menu.lisp
Description: None

2.1.7.33 WRITE-DETECTION-DATA-FILE

Type: >saf>interface>model-menu.lisp
Function
Arguments: (FILE-NAME)
Outputs:
Calls: WITH-AUTOMATIC-LOGIN
>saf>sys>macros.lisp
WITH-OPEN-FILE-ON-BUTTERFLY
>saf>sys>macros.lisp
HANDLE-LOGIN
>saf>sys>macros.lisp
TABLE-TYPE
>saf>interface>model-menu.lisp
HIT-TABLES
>saf>interface>model-menu.lisp
DEVELOPERS-FLG
>saf>interface>model-menu.lisp
GATHER-DATABASE
>saf>interface>model-menu.lisp
REMEMBER-CURRENT-HIT-TABLE
>saf>interface>model-menu.lisp
Called by: WRITE-DETECTION-DATA
>saf>interface>model-menu.lisp
Description: None

2.1.7.34 WRITE-INDIRECT-FIRE-DAMAGE-DATA

Type: >saf>interface>model-menu.lisp
Function
Arguments: ()
Outputs:
Calls: *HOST-FOR-CONFIG-DATA*
>saf>sys>interim-model.lisp
INDIRECT-FIRE-DAMAGE-FILE-NAME
>saf>interface>model-menu.lisp
WRITE-INDIRECT-FIRE-DAMAGE-DATA-FILE
>saf>interface>model-menu.lisp
Called by: WRITE-CORRESPONDING-TABLE
>saf>interface>model-menu.lisp
Description: None

2.1.7.35 WRITE-INDIRECT-FIRE-DAMAGE-DATA-FILE

Type: >saf>interface>model-menu.lisp
 Function
 Arguments: (FILE-NAME)
 Outputs:
 Calls: WITH-AUTOMATIC-LOGIN
 >saf>sys>macros.lisp
 WITH-OPEN-FILE-ON-BUTTERFLY
 >saf>sys>macros.lisp
 HANDLE-LOGIN
 >saf>sys>macros.lisp
 MKATOM
 >saf>sys>new-storage.lisp
 IF-VEHICLE
 >saf>interface>model-menu.lisp
 IND-FIR-TABLES
 >saf>interface>model-menu.lisp
 INDIRECT-FIRE-DAMAGE-DATA
 >saf>interface>model-menu.lisp
 DEVELOPERS-FLG
 >saf>interface>model-menu.lisp
 GATHER-DATABASE
 >saf>interface>model-menu.lisp
 REMEMBER-TABLE
 >saf>interface>model-menu.lisp
 Called by: WRITE-INDIRECT-FIRE-DAMAGE-DATA
 >saf>interface>model-menu.lisp
 Description: None

2.1.7.36 WRITE-DIRECT-FIRE-DAMAGE-DATA

Type: >saf>interface>model-menu.lisp
 Function
 Arguments: ()
 Outputs:
 Calls: *HOST-FOR-CONFIG-DATA*
 >saf>sys>interim-model.lisp
 DIRECT-FIRE-DAMAGE-FILE-NAME
 >saf>interface>model-menu.lisp
 WRITE-DIRECT-FIRE-DAMAGE-DATA-FILE
 >saf>interface>model-menu.lisp
 Called by: WRITE-CORRESPONDING-TABLE
 >saf>interface>model-menu.lisp
 Description: None

2.1.7.37 WRITE-DIRECT-FIRE-DAMAGE-DATA-FILE

Type: >saf>interface>model-menu.lisp
Function
Arguments: (FILE-NAME)
Outputs:
Calls: WITH-AUTOMATIC-LOGIN
>saf>sys>macros.lisp
WITH-OPEN-FILE-ON-BUTTERFLY
>saf>sys>macros.lisp
HANDLE-LOGIN
>saf>sys>macros.lisp
MKATOM
>saf>sys>new-storage.lisp
DF-VEHICLE
>saf>interface>model-menu.lisp
DIR-FIR-TABLES
>saf>interface>model-menu.lisp
DIRECT-FIRE-DAMAGE-DATA
>saf>interface>model-menu.lisp
DEVELOPERS-FLG
>saf>interface>model-menu.lisp
GATHER-DATABASE
>saf>interface>model-menu.lisp
REMEMBER-TABLE
>saf>interface>model-menu.lisp
Called by: WRITE-DIRECT-FIRE-DAMAGE-DATA
>saf>interface>model-menu.lisp
Description: None

2.1.7.38 GATHER-DATABASE

Type: >saf>interface>model-menu.lisp
Function
Arguments: (LIST-OF-TABLES)
Outputs:
Calls: *IND-FIR-TABLES*
>saf>interface>model-menu.lisp
DIR-FIR-TABLES
>saf>interface>model-menu.lisp
GATHER-INDIRECT-FIRE-DATABASE
>saf>interface>model-menu.lisp
GATHER-DIRECT-FIRE-DATABASE
>saf>interface>model-menu.lisp

Called by: WRITE-DIRECT-FIRE-DAMAGE-DATA-FILE
>saf>interface>model-menu.lisp
WRITE-INDIRECT-FIRE-DAMAGE-DATA-FILE
>saf>interface>model-menu.lisp
WRITE-DETECTION-DATA-FILE
>saf>interface>model-menu.lisp
WRITE-HTT-DATA-FILE
>saf>interface>model-menu.lisp
Description: None

2.1.7.39 GATHER-INDIRECT-FIRE-DATABASE

Type: >saf>interface>model-menu.lisp
Function
Arguments: ()
Outputs:
Calls: *IND-FIR-TABLES*
>saf>interface>model-menu.lisp
EXTRACT-RANGE-AND-DICE-NUMBERS
>saf>interface>model-menu.lisp
Called by: GATHER-DATABASE
>saf>interface>model-menu.lisp
Description: None

2.1.7.40 EXTRACT-RANGE-AND-DICE-NUMBERS

Type: >saf>interface>model-menu.lisp
Function
Arguments: (TABLE)
Outputs:
Calls: None
Called by: GATHER-INDIRECT-FIRE-DATABASE
>saf>interface>model-menu.lisp
Description: None

2.1.7.41 GATHER-DIRECT-FIRE-DATABASE

Type: >saf>interface>model-menu.lisp
Function
Arguments: ()
Outputs:
Calls: *DIR-FIR-TABLES*
>saf>interface>model-menu.lisp
EXTRACT-DICE-NUMBERS
>saf>interface>model-menu.lisp
Called by: GATHER-DATABASE
>saf>interface>model-menu.lisp
Description: None

2.1.7.42 EXTRACT-DICE-NUMBERS

Type: >saf>interface>model-menu.lisp
 Arguments: Function
 Outputs: (TABLE)
 Calls: None
 Called by: GATHER-DIRECT-FIRE-DATABASE
 Description: >saf>interface>model-menu.lisp
 Makes roll-of-the-dice numbers for the simhost config file

2.1.7.43 GET-TABLE

Type: >saf>interface>model-menu.lisp
 Arguments: Function
 Outputs: (NAME &OPTIONAL (TABLE-LIST *HIT-TABLES*))
 Calls: *PART*
 >saf>interface>model-menu.lisp
 ASPECT
 >saf>interface>model-menu.lisp
 TABLE-TYPE
 >saf>interface>model-menu.lisp
 HIT-TABLES
 Called by: >saf>interface>model-menu.lisp
 REMEMBER-TABLE
 >saf>interface>model-menu.lisp
 READ-DIRECT-FIRE-DAMAGE-DATA
 >saf>interface>model-menu.lisp
 READ-INDIRECT-FIRE-DAMAGE-DATA
 >saf>interface>model-menu.lisp
 READ-AIR-DETECTION-DATA
 >saf>interface>model-menu.lisp
 READ-GROUND-DETECTION-DATA
 >saf>interface>model-menu.lisp
 READ-HIT-DATA
 >saf>interface>model-menu.lisp
 USER-SELECT-ASPECT
 >saf>interface>model-menu.lisp
 USER-SELECT-MODEL
 >saf>interface>model-menu.lisp
 GRAPH-UNDO
 Description: >saf>interface>model-menu.lisp
 None

2.1.7.44 GET-HIT-TABLE

Type: >saf>interface>model-menu.lisp
 Function
 Arguments: (NAME &KEY (SHOULD-FIND NIL) (HIT-TABLES *HIT-
 TABLES*))
 Outputs:
 Calls: *HIT-TABLES*
 >saf>interface>model-menu.lisp
 GET-HIT-TABLE
 >saf>interface>model-menu.lisp
 Called by: GET-NEW-WEAPON-NAME
 >saf>interface>model-menu.lisp
 REMEMBER-CURRENT-HIT-TABLE
 >saf>interface>model-menu.lisp
 GET-HIT-TABLE
 >saf>interface>model-menu.lisp
 Description: None

2.1.7.45 REMEMBER-TABLE

Type: >saf>interface>model-menu.lisp
 Function
 Arguments: (&OPTIONAL (CURRENT-TABLE-LIST *HIT-TABLES*))
 Outputs:
 Calls: *HULL-POINTS*
 >saf>interface>model-menu.lisp
 STATIONARY-POINTS
 >saf>interface>model-menu.lisp
 MOVING-POINTS
 >saf>interface>model-menu.lisp
 CATASTROPHIC-POINTS
 >saf>interface>model-menu.lisp
 MOBILITY-POINTS
 >saf>interface>model-menu.lisp
 FIREPOWER-POINTS
 >saf>interface>model-menu.lisp
 TABLE-TYPE
 >saf>interface>model-menu.lisp
 WEAPON-NAME
 >saf>interface>model-menu.lisp
 HIT-TABLES
 >saf>interface>model-menu.lisp
 GET-TABLE
 >saf>interface>model-menu.lisp
 MAKE-UNDOTTED-SUBLISTS
 >saf>interface>model-menu.lisp
 Called by: WRITE-DIRECT-FIRE-DAMAGE-DATA-FILE
 >saf>interface>model-menu.lisp
 WRITE-INDIRECT-FIRE-DAMAGE-DATA-FILE
 >saf>interface>model-menu.lisp

USER-SELECT-ASPECT
>saf>interface>model-menu.lisp
USER-SELECT-MODEL
>saf>interface>model-menu.lisp
Description: None

2.1.7.46 REMEMBER-CURRENT-HIT-TABLE

Type: >saf>interface>model-menu.lisp
Function
Arguments: ()
Outputs:
Calls: *HULL-POINTS*
>saf>interface>model-menu.lisp
STATIONARY-POINTS
>saf>interface>model-menu.lisp
MOVING-POINTS
>saf>interface>model-menu.lisp
WEAPON-NAME
>saf>interface>model-menu.lisp
GET-HIT-TABLE
>saf>interface>model-menu.lisp
MAKE-UNDOTTED-SUBLISTS
>saf>interface>model-menu.lisp
Called by: WRITE-DETECTION-DATA-FILE
>saf>interface>model-menu.lisp
WRITE-HIT-DATA-FILE
>saf>interface>model-menu.lisp
Description: None

2.1.7.47 MAKE-DOTTED-SUBLISTS

Type: >saf>interface>model-menu.lisp
Function
Arguments: (LIST)
Outputs:
Calls: MAKE-DOTTED-SUBLISTS
>saf>interface>model-menu.lisp
Called by: SET-CURRENT-MODEL
>saf>interface>model-menu.lisp
MAKE-DOTTED-SUBLISTS
>saf>interface>model-menu.lisp
Description: None

2.1.7.48 MAKE-UNDOTTED-SUBLISTS

Type:	>saf>interface>model-menu.lisp
Arguments:	Function
Outputs:	(LIST)
Calls:	MAKE-UNDOTTED-SUBLISTS
Called by:	>saf>interface>model-menu.lisp MAKE-UNDOTTED-SUBLISTS >saf>interface>model-menu.lisp REMEMBER-CURRENT-HIT-TABLE >saf>interface>model-menu.lisp REMEMBER-TABLE >saf>interface>model-menu.lisp
Description:	None

2.1.7.49 QUERY-WINDOW

Type:	>saf>interface>model-menu.lisp
Arguments:	Flavor
Outputs:	()
Calls:	None
Called by:	None
Description:	None

2.1.7.50 *DEFAULT-FILE-NAME*

Type:	>saf>interface>model-menu.lisp
Arguments:	Variable
Outputs:	()
Calls:	None
Called by:	QUERY-FOR-FILENAME
Description:	>saf>interface>model-menu.lisp None

2.1.7.51 *QUERY-WINDOW*

Type: >saf>interface>model-menu.lisp
Parameter
Arguments: ()
Outputs:
Calls: None
Called by: GET-NEW-WEAPON-NAME
>saf>interface>model-menu.lisp
QUERY-FOR-FILENAME
>saf>interface>model-menu.lisp
QUERY-USER
>saf>interface>model-menu.lisp
Description: None

2.1.7.52 QUERY-USER

Type: >saf>interface>model-menu.lisp
Macro
Arguments: (DATA-TYPE &REST KEYWORD-VALUE-PAIRS)
Outputs:
Calls: *QUERY-WINDOW*
>saf>interface>model-menu.lisp
QUERY-USER
>saf>interface>model-menu.lisp
Called by: GET-NEW-WEAPON-NAME
>saf>interface>model-menu.lisp
QUERY-FOR-FILENAME
>saf>interface>model-menu.lisp
QUERY-USER
>saf>interface>model-menu.lisp
Description: None

2.1.7.53 QUERY-FOR-FILENAME

Type: >saf>interface>model-menu.lisp
Function
Arguments: (PROMPT &OPTIONAL (DEFAULT-FILE *DEFAULT-FILE-NAME*))
Outputs:
Calls: *DEFAULT-FILE-NAME*
>saf>interface>model-menu.lisp
QUERY-WINDOW
>saf>interface>model-menu.lisp

QUERY-USER
 >saf>interface>model-menu.lisp
 Called by: None
 Description: None

2.1.7.54 GET-NEW-WEAPON-NAME

>saf>interface>model-menu.lisp
 Type: Function
 Arguments: ()
 Outputs:
 Calls: GET-HIT-TABLE
 >saf>interface>model-menu.lisp
 QUERY-WINDOW
 >saf>interface>model-menu.lisp
 QUERY-USER
 >saf>interface>model-menu.lisp
 Called by: USER-SELECT-MODEL
 >saf>interface>model-menu.lisp
 Description: get weapon system name from user

2.1.8 High level functions

[; ; ; High level functions]

The function SET-CURRENT-MODEL sets up the basic lists *NAME-points* and *NAME-graph* (defined in the Parameters CSU), sets up the appropriate frame configuration, and calls PRESENT-DATA to do the work.

PRESENT-DATA calls CLEAN-UP-AXIS-AND-REDRAW and PRESENT-GRAPH to plot the probability data, and then constructs and displays the appropriate title in the black title pane. Finally, it calls PRESENT-TABLE to display the probability data in tabular form.

PRESENT-TABLE relies on system-provided macros to format the data in tabular form.

2.1.8.1 SET-CURRENT-MODEL

>saf>interface>model-menu.lisp
 Type: Function
 Arguments: (WINDOW TABLE)
 Outputs:
 Calls: *OLD-SELECTION*
 >saf>interface>model-menu.lisp
 HULL-GRAPH
 >saf>interface>model-menu.lisp
 STATIONARY-GRAPH
 >saf>interface>model-menu.lisp
 MOVING-GRAPH

```
>saf>interface>model-menu.lisp
*CATASTROPHIC-GRAPH*
>saf>interface>model-menu.lisp
*MOBILITY-GRAPH*
>saf>interface>model-menu.lisp
*FIREPOWER-GRAPH*
>saf>interface>model-menu.lisp
*HULL-POINTS*
>saf>interface>model-menu.lisp
*STATIONARY-POINTS*
>saf>interface>model-menu.lisp
*MOVING-POINTS*
>saf>interface>model-menu.lisp
*CATASTROPHIC-POINTS*
>saf>interface>model-menu.lisp
*MOBILITY-POINTS*
>saf>interface>model-menu.lisp
*FIREPOWER-POINTS*
>saf>interface>model-menu.lisp
*TABLE-TYPE*
>saf>interface>model-menu.lisp
*MODE*
>saf>interface>model-menu.lisp
*WEAPON-NAME*
>saf>interface>model-menu.lisp
*GRAPH-BEFORE-LAST-STEP*
>saf>interface>model-menu.lisp
MAKE-GRAPH-GIVEN-POINTS
>saf>interface>model-menu.lisp
MAKE-DOTTED-SUBLISTS
>saf>interface>model-menu.lisp
HIGHLIGHT-SELECTION
>saf>interface>model-menu.lisp
PRESENT-DATA
>saf>interface>model-menu.lisp
Called by: READ-DIRECT-FIRE-DAMAGE-DATA
>saf>interface>model-menu.lisp
READ-INDIRECT-FIRE-DAMAGE-DATA
>saf>interface>model-menu.lisp
READ-AIR-DETECTION-DATA
>saf>interface>model-menu.lisp
READ-GROUND-DETECTION-DATA
>saf>interface>model-menu.lisp
READ-HIT-DATA
>saf>interface>model-menu.lisp
USER-SELECT-ASPECT
>saf>interface>model-menu.lisp
USER-SELECT-MODEL
>saf>interface>model-menu.lisp
GRAPH-UNDO
>saf>interface>model-menu.lisp
Description: None
```

2.1.8.2 HIGHLIGHT-SELECTION

Type: >saf>interface>model-menu.lisp
 Arguments: Function
 Outputs: ()
 Calls: HIGHLIGHT-BUTTON-1
 >saf>ui>frame-utils.lisp
 OLD-SELECTION
 >saf>interface>model-menu.lisp
 Called by: SET-CURRENT-MODEL
 >saf>interface>model-menu.lisp
 Description: None

2.1.8.3 PRESENT-DATA

Type: >saf>interface>model-menu.lisp
 Arguments: Function
 Outputs: (GRAPH WINDOW &OPTIONAL (CLEAR-WINDOW NIL))
 Calls: *PART*
 >saf>interface>model-menu.lisp
 ASPECT
 >saf>interface>model-menu.lisp
 TABLE-TYPE
 >saf>interface>model-menu.lisp
 WEAPON-NAME
 >saf>interface>model-menu.lisp
 TABULAR-DISPLAY-PANE
 >saf>interface>model-menu.lisp
 DISPLAY-LOGO-PANE
 >saf>interface>model-menu.lisp
 TABULAR-LOGO-PANE
 >saf>interface>model-menu.lisp
 INDIRECT-FIRE-DAMAGE-FILE-NAME
 >saf>interface>model-menu.lisp
 DIRECT-FIRE-DAMAGE-FILE-NAME
 >saf>interface>model-menu.lisp
 DRAW-SCALE-BAR
 >saf>interface>model-menu.lisp
 DRAW-X-TICKS
 >saf>interface>model-menu.lisp
 DRAW-Y-TICKS
 >saf>interface>model-menu.lisp
 DRAW-TICKS
 >saf>interface>model-menu.lisp
 CLEAN-UP-AXIS-AND-REDRAW
 >saf>interface>model-menu.lisp
 PRESENT-GRAPH
 >saf>interface>model-menu.lisp
 GET-PLATFORM

Called by: >saf>interface>model-menu.lisp
PRESENT-TABLE
>saf>interface>model-menu.lisp
(PRESENTATION-MOUSE-HANDLER INSERT-POINT-1)
No Source File Record
(PRESENTATION-MOUSE-HANDLER INSERT-POINT)
No Source File Record
SET-CURRENT-MODEL
>saf>interface>model-menu.lisp
(METHOD ALTER-PROBABILITY POINT)
>saf>interface>model-menu.lisp
(METHOD ALTER-RANGE POINT)
>saf>interface>model-menu.lisp
(METHOD DRAG-UP-DOWN POINT)
>saf>interface>model-menu.lisp
CHANGE-MODE
>saf>interface>model-menu.lisp
GO-BACK-TO-PREVIOUS-STEP
>saf>interface>model-menu.lisp
(METHOD DRAG POINT)
>saf>interface>model-menu.lisp
Description: graph and tabular display

2.1.8.4 GET-PLATFORM

Type: >saf>interface>model-menu.lisp
Function
Arguments: (FILE-NAME)
Outputs:
Calls: *DF-VEHICLE*
>saf>interface>model-menu.lisp
IF-VEHICLE
>saf>interface>model-menu.lisp
Called by: PRESENT-DATA
>saf>interface>model-menu.lisp
Description: None

2.1.8.5 PRESENT-TABLE

Type:	>saf>interface>model-menu.lisp
Arguments:	Function
Outputs:	(GRAPH WINDOW)
Calls:	POINT
	>saf>interface>model-menu.lisp
	POINT
	>saf>interface>model-menu.lisp
	TABLE-TYPE
	>saf>interface>model-menu.lisp
	POINT
	>saf>interface>model-menu.lisp
	POINT
	>saf>interface>model-menu.lisp
	RANGE
	>saf>interface>model-menu.lisp
	PROBABILITY
Called by:	>saf>interface>model-menu.lisp
	PRESENT-DATA
	>saf>interface>model-menu.lisp
	ADD-CORRESPONDING-NEW-POINT-IN-POINT-LIST
	>saf>interface>model-menu.lisp
	DELETE-POINT-IF-THERE
	>saf>interface>model-menu.lisp
Description:	None

2.2. WEAPONS SYSTEMS EDITOR CSC

This CSC contains the code which allows an operator to edit weapons systems parameters. Weapons systems are vehicles, guns, missiles, etc. The guns and missiles are separated from the vehicles to allow the user to interchange weapons from different vehicles. The weapons system editor does not guarantee a valid result with this feature but the restrictions are fairly simple. The primary purpose of the editor is to allow users to change parameters such as weapon ranges, vehicle speeds, etc. It also allows the comparison of different weapons.

The software for this editor is found in the file `saf>interface>object-menu.lisp`.

This CSC edits the Simhost file:

/usr/saf/config/config.lisp.

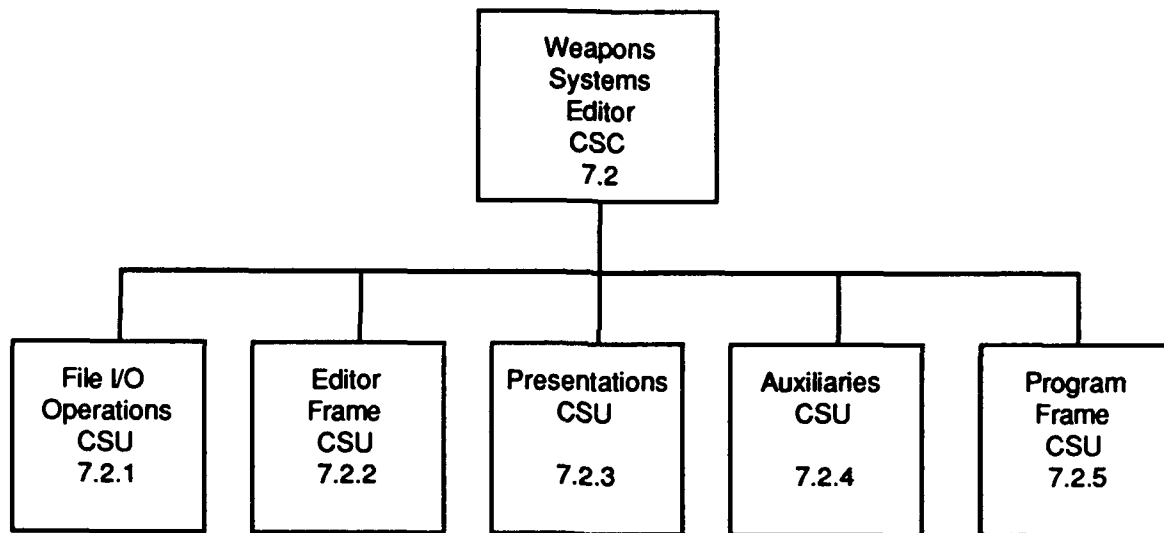


Figure 2.2-1 Weapons Systems Editor CSC Structure

2.2.1. File I/O Operations CSU

This CSU contains the functions that perform input-output operations between the Simhost file containing the weapon systems parameter data and the editor.

Function READ-OBJECT-FILE:

The purpose of this function is to read the Simhost file */saf/config/config.lisp* into the editor. The file contains parameter setting for all weapon systems used by the semi-automated forces.

The procedure that this function follows is to first construct a file access string using the Simnet Host to which the battlemaster is connected. When the NIL argument is passed, the default */usr/saf/config/config.lisp* file is used. The operator may pass as an argument a pathway string to an alternative data file. The function then uses WITH-OPEN-FILE-ON-BUTTERFLY to open the file and input its contents into the editor as a list structure, setting the global *OBJECTS-FORMS* to this list structure.

Function WRITE-OBJECT-FILE:

The purpose of this function is to write the edited version of the *config.lisp* file back into the Simhost at */saf/config/config.lisp*.

The Lisp code used by the editors ignores upper-case/lower-case differences. The procedure that this function follows is to first call the function RESTORE-UPLOWCASE to restore all parameter name and symbol values to their original multicase form. The restored data is then written into the SIMHOST using the system macro WITH-OPEN-FILE. This macro requires a password. However, setting the global *DEVELOPERS-FLG* to T writes the file using WITH-OPEN-FILE-ON-BUTTERFLY instead, bypassing the password request.

Function PREPARE-OBJECT-DATA-FOR-MENU-USE:

The purpose of this function is to prepare the weapon systems data for usage by the editor.

The procedure that this function follows is to first set the global *OBJECT-NAMES* to a list of names of all weapons systems in the file. Next it sets the global *OBJECT-TO-WINDOW-ASSOC* to a list of pairs, each consisting of the name of the weapon system being edited and the window pane on which it is displayed. Finally, it sets the global *OBJECT-DESCRIPTIONS* to the list of descriptors of the various data items in the *config.lisp* file.

2.2.2. Editor Frame CSU

This CSU contains the functions that construct and maintain the editor frame. The editor frame is an assemblage of windows constrained to fit aspanes within a rectangular frame, sharing a common input-output buffer.

Flavor OBJECT-GLOBAL-COMMAND-PANE:

This is a window pane flavor built on top of flavor OBJECT-MS-PANE. It has a single instance variable, PANE-NAME, and sets the default character style to datum (:fix :bold :normal).

Flavor TABLE-COMMAND-PANE:

This is a window pane flavor built on top of flavor OBJECT-MS-PANE. It has a single instance variable, PANE-NAME, and sets the default character style to datum (:fix :bold :normal).

Flavor OBJECT-MS-PANE:

This is the basic window pane flavor for the editor. It is built on top of MS-PANE (defined in Model Editor CSC. It has three instance variables; COLUMN-WIDTH, ITEM-LIST, and LIST-OF-CHANGES. COLUMN-WIDTH, which is initially set to the global variable *COLUMN-WIDTH*, controls the width of each of the columns of alphanumeric data into which the panel is divided. ITEM-LIST is a list of all the parameter name value pairs that define the weapon system. LIST-OF-CHANGES keeps track of all the parameter pairs that have changed with the editor. The default character style is set to datum (:fix :bold :small).

Method :REVERT-PANE Of The Flavor OBJECT-MS-PANE:

This method restores a weapon system data to the conditions that existed when it was last saved. This is invoked by an operator when a weapon system data currently being modified has deviated sharply from the desired results.

The procedure that this method follows is to first clear the window pane by invoking the Lisp system method :CLEAR-HISTORY, initialize the instance variables ITEM-LIST and LIST-OF-CHANGES, and then send the STUFF-OBJECT-INTO-PANE message to the constraining frame (which is the value returned by the message (send self :superior)), with the values of CATEGORY-NAME and of the global variable *WORKED-ON-OBJECT-FORMS* as arguments.

Method :SAVE-ITEM-LIST Of The Flavor OBJECT-MS-PANE:

The purpose of this method is to keep a copy of the original (unedited) list of parameter name value pairs.

The procedure that this method uses is a standard COLLECT loop.

Method :REPLACE-ITEM Of The Flavor OBJECT-MS-PANE:

The purpose of this method is to replace the value of a parameter.

The procedure that this method uses is to search ITEM-LIST for a parameter name value pair with a name equal to FIELD (an argument set to the parameter name that is to be replaced). It then replaces that parameter's value with NEW-VALUE (a second parameter containing the replacement value).

Method :REDISPLAY-PANE Of The Flavor OBJECT-MS-PANE:

The purpose of this method is to erase and display anew the weapon systems' parameter data being edited.

The procedure that this method uses is first blank the pane to be redisplayed by invoking the system's :clear-history method. The method next invokes the extensive system macro FORMATTING-TABLE, with its attendant FORMATTING ROWS macros. The first call to FORMATTING-TABLE displays the weapons system's name in the top center of the pane. The second call invokes a loop where parameter name value pairs are taken four at a time and displayed in four columns in a single row (each row has four pairs). The first element in a pair, the parameter name is displayed as a FIELD-ELEMENT presentation using the system function PRESENT. The second element, the parameter's value, is displayed as a NUMBER-ELEMENT presentation if it is a number, or as a defaulty presentation if not. If the name and/or the value are alphanumeric, the function TRUNCATE-IF-NECESSARY is called to ensure they fit within the allotted column width.

Method :EXPOSE Of The Flavor TABLE-COMMAND-PANE:

The purpose of this method is to construct the mouse-sensitive commands that enable the editor to "Save" a weapon system's parameter set, "Undo" the last change made, "Undo All" changes, and close a pane.

This method uses the system function PRESENT after the pane has been exposed, but before it is displayed, to display the appropriate mouse-sensitive presentation. The four commands mentioned in the previous paragraph are evenly spread in the center 80% of the pane's width with the help of the system method :SET-CURSPOS.

Method :EXPOSE Of The Flavor OBJECT-GLOBAL-COMMAND-PANE:

The purpose of this method is to construct the mouse-sensitive commands that enable the editor to "Write" a new Simhost *config.lisp* file, "Revert" to a factory version of the *config.lisp* file and "Reset".

This method uses the system function PRESENT after the pane has been exposed, but before it is displayed, to display the appropriate mouse-sensitive presentation. The three commands mentioned in the previous paragraph are evenly spread in the center 75% of the pane's width with the help of the system method :SET-CURSPOS.

2.2.3. Presentations CSU

This CSU documents the functions that deal with mouse-sensitive presentations that enable most of this editor's functionality. It comprises the definitions of presentation types, actions, and their associated functions. These are further separated into groups corresponding to their presentation pane.

Presentation Type WEAPON-SYSTEM:

This is a presentation type definition where the definition of a type is not required. Otherwise it contains standard settings. It is used in the pane named MODEL-COMMAND-PANE of the flavor OBJECT-MS-PANE.

Presentation Action SELECT-WEAPON-SYSTEM:

This is a presentation action of the WEAPON-SYSTEM presentation type. When the mouse is within a displayed weapon-system presentation, the mouse documentation line at the bottom of the screen displays the character string "L: Select this weapon system". Clicking the left mouse button invokes the action, which is executed if a test is passed. (The test checks to see if the pane on which the click took place is named the MODEL-COMMAND-PANE.) The action invoked upon passing the test is to send the messages :PUT-OBJECT-IN-TABLE and :PUT-OTHER-STUFF-BACK-TO-WHERE-THEY-WERE to the frame.

Function PRESENT-WEAPON-NAMES:

The purpose of this function is to display the various weapon system names as mouse-sensitive presentations so that they can be used to select a weapons system to edit.

The procedure for this function is to first locally bind N-COLUMNS and COLUMN-WIDTH and then clear the window. Next the weapon system names in *OBJECT-NAMES* are displayed in two rows as presentations of types WEAPON-SYSTEM. The :SET-CURSORPOS message insures that the names are properly spaced and that the second row is started before running out of space.

Presentation Type FIELD-ELEMENT:

This is a presentation type definition where the definition of a TYPE is not required. Otherwise, the presentation type uses standard settings. It is used in the panes named TABLE1, TABLE2, and TABLE3 of the flavor OBJECT-MS-PANE.

Presentation Action DESCRIBE-FIELD:

This is a presentation action of the FIELD-ELEMENT presentation type. When the mouse position on screen is within a displayed field-element presentation, the mouse documentation line at the bottom of the screen displays the string retrieved by the function GET-FIELD-DESCRIPTOR. No action is triggered by clicking the mouse button while on this presentation; the only effect is the appearance of descriptive information in the mouse documentation line.

Function GET-FIELD-DESCRIPTOR:

The purpose of this function is to retrieve the descriptor string corresponding to the name of the argument passed (FIELD-NAME), and to compose it for display in the mouse document line.

The function calls the function FIND-FIELD-DESCRIPTOR in order to retrieve the descriptor string and binds the result to the local variable DESCRIPTOR. If no descriptor is found, it returns the string "No descriptor in file". If DESCRIPTOR is not null, it uses the truth value of the second element of DESCRIPTOR to append the string "Changeable by User: yes (no)" to the actual descriptor string which is the third element of DESCRIPTOR.

Function FIND-FIELD-DESCRIPTOR:

The purpose of this function is to retrieve the descriptor string corresponding to the name of the argument passed (FIELD-NAME) from the descriptors in *OBJECT-DESCRIPTIONS*.

This is a presentation action of the UNDO-COMMAND presentation type. When the mouse is within a display "UNDO" command presentation, the mouse documentation line at the bottom of the screen displays the string "L: Undoes last change". Clicking the left mouse button invokes the action, calling the function UNDO-LAST-CHANGE.

Presentation Type NUMBER-ELEMENT:

This is a presentation type definition where the TYPE definition is expanded to be a NUMBER. This enables inheritance of all presentation actions of the NUMBER type. It is used in the panes named TABLE1, TABLE2, and TABLE3 of the flavor OBJECT-MS-PANES.

Presentation Action REPLACE-NUMBER-ELEMENT:

This is a presentation action of the NUMBER-ELEMENT presentation type. When the mouse is within a display number-element presentation, the mouse documentation line at the bottom of the screen displays the string "L: Replace this number". Clicking the left mouse button invokes the action, sending the message :REPLACE-NUMBER.

Method :REPLACE-NUMBER Of The Flavor OBJECT-PANE:

The purpose of this method is to replace the numerical value of a weapons system parameter with a new value specified by the user.

The method first binds the local variables NEW-NUMBER, SUPERIOR, and FIELD-NAME. It then sends the message :REPLACE-ITEM to update the pane's ITEM-LIST and adds information useful to undo the replacement of the push down list in LIST-OF-CHANGES. It then sets NEW-NUMBER to the value of :XREPLACE-OBJECT-VALUE, a system method that handles types inputs on numeric presentations.

Presentation Type SAVE-COMMAND:

This is a presentation type definition that does not require the definition of a TYPE. It otherwise uses standard settings. It is used in the panes TABLE1-MENU, TABLE2-MENU, and TABLE3-MENU of the flavor SAVE-OBJECT.

Presentation Action EXECUTE-SAVE-COMMAND:

This is a presentation action of the SAVE-COMMAND presentation type. When the mouse is within a display number-element presentation, the mouse documentation line at the bottom of the screen displays the string "L: Preserves changed data". Clicking the left mouse button invokes the action, sending the message :SAVE-OBJECT.

Function SAVE-OBJECT:

The purpose of this function is to save the newly edited parameter values of a weapons system.

The procedure is to send the message :SAVE-ALL-OBJECT-INFORMATION to the pane.

Method :SAVE-ALL-OBJECT-INFORMATION Of The Flavor OBJECT-MS-PANE:

The purpose of this method is to save the set of parameters of the given weapons system in the list bound to the global variable *WORKED-ON-OBJECT-FORMS*.

The procedure is to first check if the instance variable LIST-OF-CHANGES of the pane does not contain any changes to preserve. If this is the case, the pane's interaction panel flashes (beeps) and displays the string "There aren't any changes to preserve". If there are changes to preserve, the local variable CATEGORY is first bound to the name of the weapons system under edit. Next in the event of changes, the value of ITEM-LIST, the entire set of parameter names and values of the weapon system, substitutes for the old set in *WORKED-ON-OBJECT-FORMS*. The substitution is made in the system function SUBST.

Presentation Type UNDO-COMMAND:

This is a presentation type definition that does not require the definition of a TYPE. It otherwise uses standard settings. It is used in the panes TABLE1-MENU, TABLE2-MENU, and TABLE3-MENU of the flavor TABLE-COMMAND_FRAME.

Presentation Action EXECUTE-UNDO-COMMAND:

This is a presentation action of the UNDO-COMMAND presentation type. When the mouse is within a display "UNDO" command presentation, the mouse documentation line at the bottom of the screen displays the string "L: Undoes last change". Clicking the left mouse button invokes the action, calling the function UNDO-LAST-CHANGE.

Function UNDO-LAST-CHANGE:

The purpose of this function is to undo the last change made to a weapon system parameter value.

The function binds the arguments needed by UNDO-LAST-CHANGE-INTERNAL , a call to which does the work.

Method :UNDO-LAST-CHANGE-INTERNAL Of The Flavor OBJECT-MS-PANE:

The purpose of this method is to undo the last change made to a weapons system parameter value.

The method first binds the local variable ITEM-CHANGED to the old value of the parameter last changed by calling the function POP-CHANGE-LIST. It next binds the local variable CATEGORY to the name of the weapon system being edited by searching through the contents of *OBJECT-TO-WINDOW-ASSOC*, which is a list of pairs of the form pane-name weapon-system-name. Then, if there is no ITEM-CHANGED it beeps the interaction pane. Otherwise, it replaces the new value given in the parameter with the old one and redisplay the pane.

Presentation Action EXECUTE-UNDO-ALL-COMMAND:

This is a presentation action of the UNDO-COMMAND presentation type. When the mouse is within a display "UNDO" command presentation, the mouse documentation line at the bottom of the screen displays the string "L: Undoes last change". Clicking the left mouse button invokes the action, calling the function UNDO-LAST-CHANGE.

Function UNDO-ALL-CHANGES:

The purpose of this function is to undo all changes made to a weapons system parameter values since it was last saved.

The function first binds the local variables PANE-NAME, PANE, CATEGORY-NAME, OLD-DATA, and NEW-DATA. It then sets CATEGORY-NAME to the name of the weapon system being edited. OLD-DATA is set to the old values of the weapon system parameters, extracted from the value of the global variable *OBJECT-FORMS*. NEW-DATA is set to the current (edited) values of the weapon system parameters, extracted from the global variable *WORKED-ON-OBJECT-FORMS*. Then the old values are made to substitute for the new values and the pane is sent the :REVERT-PANE message.

Presentation Type CLOSE-COMMAND:

This is a presentation type definition that does not require the definition of a TYPE. It otherwise uses standard settings. It is used in the panes TABLE1-MENU, TABLE2-MENU, and TABLE3-MENU of the flavor TABLE-COMMAND-PANE.

Presentation Action EXECUTE-CLOSE-COMMAND:

This is a presentation action of the CLOSE-COMMAND presentation type. When the mouse is within a display "Close" command presentation, the mouse documentation line at the bottom of the screen displays the string "L: Get rid of this pane. Changes will be lost unless SAVED". Clicking the left mouse button invokes the action, calling the function CLOSE-OBJECT-PANE.

Function CLOSE-OBJECT-PANE:

The purpose of this function is to close the pane where the weapon system is displayed. The frame readjusts to display the remaining open panes.

The function first binds the local variables PANE-NAME and PANE. It then nulls the instance variable ITEM-LIST, calls DELETE-TABLE-OBJECT-ASSOCIATION, and readjusts the frame by calling :REPUT-STUFF-BACK-INTO-EDITOR-IN-A-DIFFERENT-PLACE-IF-NECESSARY.

Function PRESENT-PANE-COMMANDS:

The purpose of this function is to make and display mouse-sensitive presentations which when activated can issue editor commands for saving, undoing, and closing.

This function binds the local variable COMMAND-PANES to the list of command panes in the frame and then calls PRESENT-PANE-COMMANDS-1 on each of them to do the work.

Function PRESENT-PANE-COMMANDS-1:

The purpose of this function is to make and display mouse-sensitive presentations for the editor commands for "Save", "Undo", "Undo All", and "Close".

The function first clears the pane. It then uses the system function PRESENT to display the appropriate mouse-sensitive presentation. The four commands, listed above, are evenly spread in the central 80% of the pane's width with the help of the system method :SET-CURSORPOS.

Presentation Type WRITE-COMMAND:

This is a presentation type definition that does not require the definition of a TYPE. It otherwise uses standard settings. It is used in the pane COMMAND-PANE of the flavor OBJECT-GLOBAL-COMMAND-PANE.

Presentation Action EXECUTE-WRITE-COMMAND:

This is a presentation action of the WRITE-COMMAND presentation type. When the mouse is within a display "Write" command presentation, the mouse documentation line at the bottom of the screen displays the string "L: Store all data in SIMHOST file". Clicking the left mouse button invokes the action, calling the function WRITE-OBJECT-FILE.

Presentation Type REVERT-COMMAND:

This is a presentation type definition that does not require the definition of a TYPE. It otherwise uses standard settings. It is used in the pane COMMAND-PANE of the flavor OBJECT-GLOBAL-COMMAND-PANE.

Presentation Action EXECUTE-REVERT-COMMAND:

This is a presentation action of the REVERT-COMMAND presentation type. When the mouse is within a display "Revert" command presentation, the mouse documentation line at the bottom of the screen displays the string "L: Reads the original FACTORY version of the SIMHOST file". Clicking the left mouse button invokes the action, calling the function REVERT-TO-FACTORY-FILE.

Function REVERT-TO-FACTORY-FILE:

The purpose of this function is to read and display weapons system parameters stored in the original factory version of the *CONFIG.LISP* file. This provides a way to revert to the factory set parameters for all weapons.

The function calls the function READ-OBJECT-FILE with a path segment pointing to the factory file, and initializes the editor.

Presentation Type RESET-COMMAND:

This is a presentation type definition that does not require the definition of a TYPE. It otherwise uses standard settings. It is used in the pane COMMAND-PANE of the flavor OBJECT-GLOBAL-COMMAND-PANE.

Presentation Action EXECUTE-RESET-COMMAND:

This is a presentation action of the RESET-COMMAND presentation type. When the mouse is within a display "Reset" command presentation, the mouse documentation line at the bottom of the screen displays the string "L: Clean slate and start over by re-reading SIMHOST file". Clicking the left mouse button invokes the action, calling the function RESET-OBJECT-MENU.

Function RESET-OBJECT-MENU:

The purpose of this function is to create a new editor frame.

This function initializes the editor by nulling the global variables *WORKED-ON-OBJECT-FORMS*, *OBJECT-FORMS*, and *OBJECT-FRAME*. Then it calls the function SELECT-OBJECT-EDITOR and sends a :KILL message to the frame (obtained by sending WINDOW the :SUPERIOR message).

Function SELECT-OBJECT-EDITOR:

The purpose of this function is to select a screen with the weapons systems editor if one exists, and to create one if no screen exists.

This function determines if a frame exists by determining that the global variable *OBJECT-FRAME* is not null. It then checks to see that the frame is active and that there are panes in it. If that is the case, it selects the first pane, TABLE1. Otherwise, it creates a screen with the system function TV::RUN-SELECT-KEY-FORM. In either case, it waits until the frame or pane is exposed before returning to caller.

2.2.4 Auxiliaries CSU

This CSU contains the utility code for this editor. These consist of predicates, macros, in-flavor functions, and methods of general use.

Function TABLE-MENU?:

The purpose of this function is to predicate to test if the given pane could be displaying a weapons system's parameter.

The procedure for this function is to check for the presence of the substring "lisp" or "menu" in the pane's name.

Defsubst-in-flavor GARCON-GET-ME-A-TABLE Of The Flavor DW::PROGRAM-FRAME:

The purpose of this defsubst is to either find the pane displaying the weapon system's name given in the argument, or to come up with the first available empty one.

The defsubst accomplishes this search using the pane to weapon system association list kept in the global variable *OBJECT-TO-WINDOW-ASSOC*.

Defun-in-flavor BOOKEEPING-OF-PUTTING-OBJECT-INTO-NEW-WINDOW Of The Flavor DW::PROGRAM-FRAME:

The purpose of this function is to assign a pane to a new weapon system to be edited and to update the association list that tracks the correspondence between panes and weapon systems.

The procedure to accomplish this is to bind the local variable FIRST-WINDOW-FREE to the value returned by the call to GARCON-GET-ME-A-TABLE.

Function TRUNCATE-IF-NECESSARY:

The purpose of this function is to truncate excessively long parameter names or alphanumeric values so that the pane width is not exceeded.

The procedure that this function uses is for the STRING passed as a argument to be trimmed to LENGTH characters, if the STRING is longer. The characters "..." are substituted for last three characters in the STRING if this is the case. LENGTH may either be a second argument passed or, if no argument is passed, 18 characters.

Method :STUFF-OBJECT-INTO-PANE Of The Flavor DW::PROGRAM-FRAME:

The purpose of this method is to display a weapon system's parameters in a pane.

The procedure used is a straightforward association.

Inline Function COUNT-WINDOWS-USED:

The purpose of this function is to count the number of weapon system panes in use.

The procedure to accomplish this inline function is a count loop using the association list bound to the global variable *OBJECT-TO-WINDOW-ASSOC*.

Inline Function TRANSLATE-CONFIG-TO-NUM-TABLES-USED:

The purpose of this function is to come up with the number of panes into which a given frame configuration can fit.

The procedure used is a straightforward association.

Inline Function GET-TABLE-POSITION-IN-LIST:

The purpose of this function is to come up with an ordinal number for the given pane.

The procedure used is a straightforward association.

Inline Function UP-OR-LOWER-CONFIGURATION-IF-NECESSARY In The Flavor DW::PROGRAM-FRAME:

The purpose of this function is to change up or down the number of panes in the frame, if necessary.

The procedure that this inline function uses is to first bind the local variable NUMBERS-USED to the number of the pane being currently used and to bind the local variable NUMBERS-AVAILABLE to the number of panes that can be accommodated in the current frame configuration. Then it resets the frame configuration accordingly, increasing or decreasing the number of panes as needed.

Method :PUT-OBJECT-IN-TABLE Of The Flavor DW::PROGRAM-FRAME:

The purpose of this method is to set up the frame configuration and to display a weapon system's parameters.

The procedure is to call UP-OR-LOWER-CONFIGURATION-IF-NECESSARY and :PUT-OBJECT-IN-TABLE1 to do the work.

Method :PUT-OBJECT-IN-TABLE-1 Of The Flavor DW::PROGRAM-FRAME:

The purpose of this method is to display a weapon system's parameters for the first time.

The procedure of this method is to first bind the local variables TABLE and TABLE-OBJECT to a pane obtained via a call to :BOOKKEEPING-OF-PUTTING-OBJECT-INTO-NEW-WINDOW. It then clears the pane and calls :STUFF-OBJECT-INTO-PANE to fill it.

Method :PUT-OTHER-STUFF-BACK-TO-WHERE-THEY-WERE Of The Flavor
DW::PROGRAM FRAME:

The purpose of this method is to redisplay weapon systems parameters after a configuration change.

The procedure of this method is to check the list of pairs (weapon system name, pane) in the global variable *OBJECT-TO-WINDOW-ASSOC* to either send the :REDISPLAY-PANE message or the :STUFF-OBJECT-INTO-PANE message, as required.

2.2.5 Program Frame CSU

This CSU defines the frame object that underlies the Vehicle and Weapon System Parameter Editor. It consists of the definition of the panes and of the configurations (arrangement of panes) of the frame, as well as some top-level methods and functions.

Program-Framework OBJECT-MENU:

This defines the kinds of panes and configurations that the Weapons Systems Editor uses.

The Weapons Systems Editor uses six kinds of panes: LOGO-PANE, MODEL-COMMAND-PANE, COMMAND-PANE, TABLE pane type, TABLE-MENU pane type, and LISP-PANE.

The LOGO-PANE displays the "Weapons System Editor" identifier at the top of the screen. It is of a type built into the system that is specified by the keyword :TITLE. It is displayed in reverse video.

The MODEL-COMMAND-PANE displays the names of all the weapon systems as mouse-sensitive presentations. Its type is specified by the flavor OBJECT-MS-PANE.

The COMMAND-PANE displays the three presentations for the commands "Write", "Reset", and "Revert". Its type is specified by the flavor OBJECT-GLOBAL-COMMAND-PANE.

The member of the TABLE pane type (TABLE1, TABLE2, and TABLE3) display weapon system parameter sets in columns of name/value pairs. Their type is specified by the flavor OBJECT-MS-PANE.

The members of the TABLE-MENU pane type (TABLE1-MENU, TABLE2-MENU, and TABLE3-MENU) fit underneath their corresponding TABLE panes. They contain the four presentations for the command "Close", "Undo", "Undo All", and "Save" commands. Their type is specified in the flavor TABLE-COMMAND-PANE.

The LISP-PANE is the interaction pane of pre-defined type :LISTENER.

The Weapons Systems Editor frame uses three configurations, to accommodate as many as three weapon system parameter sets on the screen. The three configurations are named MAIN (for the minimum of one parameter set), MAIN-WITH-2 (for two sets), and MAIN-WITH-3 (for three sets). Each configuration specification consists of a layout specification, where panes are given in top-to-bottom and left-to-right order, and a size specification, given in screen percentages or in number of lines. The following table shows the panes contained in each frame configuration.

CONFIGURATION PANE	MAIN	MAIN+2	MAIN+3
LOGO-PANE	●	●	●
MODEL-COMMAND-PANE	●	●	●
COMMAND-PANE	●	●	●
TABLE1	●	●	●
TABLE2		●	●
TABLE3			●
TABLE1-MENU	●	●	●
TABLE2-MENU		●	●
TABLE3-MENU			●
LISP-PANE	●	●	●

● Indicates Configuration Contains This Pane

Table 2.2.5-1 Panes Contained in Each Frame Configuration

Method :TOP-LEVEL Of The Flavor OBJECT-MENU:

The purpose of this method is to perform initialization. The procedure involved is to set a number of global variables, read in data via a call to READ-OBJECT-FILE , display initial presentation, and launch the Lisp listener that handles user inputs.

Method :SET-TO-LOWER-CONFIGURATION Of The Flavor DW::PROGRAM-FRAME:

The purpose of this method is to decrease the number of weapons systems shown on the screen. The procedure involved is to send itself the :CONFIGURATION message to determine the current configuration, and then to choose the one with one less table pane in it.

Method :SET-TO-HIGHER-CONFIGURATION Of The Flavor DW::PROGRAM-FRAME:

The purpose of this method is to increase the number of weapons systems shown on the screen. The procedure involved is to send itself the :CONFIGURATION message to determine the current configuration, and then to choose the one with one less table pane in it.

2.3 FORMATIONS EDITOR CSC

The FORMATIONS EDITOR allows users to edit the positions (x, y and z coordinates) of each of the elements that configure a military unit. For example, a NATO tank platoon in Line formation has its tanks 90 meters apart, on a straight line, heading 90° from the line. With the help of the editor, a user can alter any formation in a natural visual-oriented manner.

The resulting formation changes can then be evaluated for their tactical effects.

The software for this editor can be found in the file saf>interface>formations.lisp.

This CSC edits the Simhost files saf/config/blue_forms.lisp and saf/config/red_forms.lisp.

The formations editor consists of the following CSU's:

- Parameters and Globals
- File Input/Output
- Formations object
- Panes and Frame
- Presentations and commands
- Drawing Routines

A description of each of these units follows. Each unit is located in the file following a comment header; an abbreviated form of this header is included below, in brackets, at the beginning of each section.

2.3.1 Parameters and Globals CSU

[;;; PARAMETERS and other GLOBALS]

This is just a list of variable definitions used globally (mostly within the Formations editor code). Their names are enclosed in asterisks (*), their initial values are specified, and most have their purpose explained by a brief documentation string.

2.3.2 File Input/Output CSU

[;;; FILE I/O FUNCTIONS]

This unit contains two functions that read the SIMHOST files into the Editor, and write it back onto the SIMHOST when the editing is finished. The entire SIMHOST file is read as a list which is the value of *CONFIGURATION-FORMS*. For all practical purposes this variable represents the entire Formations database.

2.3.3 Formations object CSU

[;;; FORMATIONS]

This unit contains the code that defines FORMATION-OBJECT (the data-structures the editor deals with, representing each vehicle within a formation). FORMATION-OBJECT has Instance-Variables (fields) representing relative position, the name of the formation it belongs to, the kind of vehicle it is, and what kind of unit it is a part of.

The unit also contains the Methods (accessor functions) for reading and writing the values of these fields.

The position Instance-Variables (x, y, and z) are location pointers into the database (i.e.: the value of *CONFIGURATION-FORMS*). GET and SET methods perform the required pointer setting so that any changes to the position variables of a FORMATION-OBJECT instance will be reflected in the database.

The function CREATE-FORMATION-OBJECTS uses the database as a pattern to make instances of FORMATION-OBJECTs, while GATHER-FORMATION-OBJECTS rounds up those belonging to any echelon, vehicle-type or formation (e.g.: all FORMATION-OBJECTs that are Tanks, belong to a Platoon, and are in Line formation)

2.3.4 Panes and Frame CSU

[; ; ; PANES and FRAME]

This CSU contains the code that defines the Editor Frame, as well as the individual panes that appear within it, as seen on the screen.

Each pane is defined via a flavor definition. :AFTER :EXPOSE methods are responsible for presenting the command labels (Grid, Zoom, Write, etc) that will be activated when clicked by the user.

The choice pane near the bottom of the screen is serviced by the :UPDATE-CHOICES method, with the help of the functions GET-VEHICLE-CHOICES and GET-FORMATION-CHOICES.. Together they constitute the machinery for selecting the desired formation by clicking on any one of the Echelon, Vehicle type, Formation, or Tactics choices.

Finally, the frame itself is defined as a program framework with two configurations: MAIN, which provides an XY plan view used for ground vehicles, and MAIN-FOR-ALTITUDE-DISPLAY, which adds a YZ vertical view for air units.

2.3.5 Presentations and commands CSU

[; ; ; PRESENTATIONS]

This unit contains code (i.e.: DEFINE-PRESENTATION-TYPE) to define the mouse sensitive presentations (command buttons, vehicle icons, coordinate values) that users will activate to effect the changes they want. It also contains code to implement the functionality associated with the presentations (DEFINE-PRESENTATION-ACTION and associated functions and methods)

The software is grouped by commands, as they appear in the editor frame (e.g.: Grid, Zoom, etc). Within groups, presentation-types go first, followed by presentation-actions and their associated code.

2.3.6 Drawing Routines CSU

[; ; ; Code for drawing formation vehicles]

This unit contains code for drawing tank, helicopter, and aircraft icons on the screen, and code for drawing the distance grids on the display panes. Many constants governing the appearance of these images are coded right into the forms, rather than stored in named symbols.

APPENDIX A: SAF COMMAND PROTOCOL

A.1	Introduction.....	A-3
A.1.1	Overview.....	A-3
A.1.2	Data Types.....	A-3
A.1.3	Data Elements of a SAF Packet	A-4
A.1.3.1	Header Data	A-4
A.1.3.2	Message Data	A-4
A.2	Workstation to Simhost Message Descriptions.....	A-4
A.2.1	Set-Up Messages	A-4
A.2.1.1	Create Message	A-5
A.2.1.2	Reset Message	A-6
A.2.1.3	Vehicle Reinitialize Message	A-6
A.2.1.4	Disconnect Message.....	A-7
A.2.1.5	Attach Stealth Message	A-7
A.2.2	Ask for Data Messages.....	A-8
A.2.2.1	Poll Message.....	A-8
A.2.2.2	Query Sub State Message	A-8
A.2.3	IVIS Messages	A-9
A.2.3.1	IVIS Xmit Transmit Message.....	A-9
A.2.3.2	IVIS Parameters Message	A-9
A.2.4	Overlay Messages.....	A-10
A.2.4.1	Point Message	A-10
A.2.4.2	Area Message	A-11
A.2.4.3	Zone Message.....	A-11
A.2.4.4	Line Message	A-12
A.2.4.5	Route Message.....	A-13
A.2.4.6	Delete Overlay Message	A-13
A.2.4.7	Execute Overlay Message	A-14
A.2.4.8	Delete Control Measure Message.....	A-14
A.2.5	Immediate Intervention Messages.....	A-15
A.2.5.1	Rejoin Unit Message.....	A-15
A.2.5.2	Halt Message.....	A-15
A.2.5.3	Resume Message	A-16
A.2.5.4	Speed Message	A-16
A.2.5.5	Formation Message	A-16
A.2.5.6	Follow Vehicle Message.....	A-17
A.2.5.7	Simulator In Command Message.....	A-17
A.2.5.8	Go To Point Message.....	A-18
A.2.5.9	Face Direction Message.....	A-18

A.2.5.10	Hold Message.....	A-19
A.2.5.11	Altitude Message	A-19
A.2.5.12	Land Message.....	A-20
A.2.5.13	Attack Message	A-20
A.2.5.14	Resupply Message	A-21
A.2.5.15	Targeting Message	A-21
A.2.6	Misc Messages	A-22
A.2.6.1	Artillery Message.....	A-22
A.3	Simhost to Workstation Message Descriptions.....	A-23
A.3.1	Set-Up Messages	A-23
A.3.1.1	Reset Message	A-23
A.3.2	Combat Messages.....	A-23
A.3.2.1	Ground Impact Message.....	A-23
A.3.2.2	Vehicle Impact Message	A-24
A.3.2.3	Indirect Fire Message	A-24
A.3.3	Simhost Status Messages	A-25
A.3.3.1	Vehicle PAE Message.....	A-25
A.3.3.2	Vehicle Position Message	A-25
A.3.3.3	Vehicle Load Message	A-26
A.3.3.4	Vehicle Position Poll Completed Message	A-26
A.3.3.5	Vehicle Appearance Message	A-27
A.3.3.6	Vehicle Echelon Message.....	A-27
A.3.4	Misc Messages	A-28
A.3.4.1	Generic Message	A-28
A.3.4.2	Stealth Position Message	A-28
A.3.4.3	Machine Status Message.....	A-29

APPENDIX A: SAF COMMAND PROTOCOL

A.1 Introduction

A.1.1 Overview

Semi-Automated Forces (SAF) runs on two different kinds of computers: workstation computers and simulation (simhost) computers. The SAF workstations allow a user (the SAF commander) to interact with the SAF system by way of an extensive list of actions and options for monitoring and controlling the SAF units under his command. The simhost computers are in charge of simulating the motion and activities of the SAF vehicles. They are responsible for making SAF vehicles behave reasonably (e.g., tanks stay on the ground, avoid obstacles, deplete supplies, and much more) and for generating up-to-date packets of data to represent the state (terrain position, motion, firing, turret motion, and so on) of each and all of the active SAF vehicles.

The SAF Command Protocol provides a communications framework between the Symbolics workstation and the Mips simhost. SAF uses the User Datagram Protocol (UDP), which does not guarantee ordered reception of individual data packets, to build a Reliable User Datagram Protocol (RUDP), which does guarantee ordered reception of individual data packets. Within this framework, the workstation and simhost are able to send messages to each other.

This document describes the layout of the messages passed between the workstation and the simhost. The messages that are passed between the workstation and simhost are described in the following source files: /simnet/include/saf/src/message.h on the simhost and >saf>network>packet-layouts.lisp on the workstation. Messages described in the source files but not identified in this document are not used in SAF version 3.9.10.

A.1.2 Data Types

The following data types are used in the descriptions of the SAF Command Protocol data:

char - 8 bits, 1-byte capable of holding one character

short - 16 bits, 2-bytes capable of holding one integer

int - 32 bits, 4-bytes capable of holding one integer

float - 32 bits, 4-bytes capable of holding one single-precision floating point number

double - 64 bits, 8-bytes capable of holding one double-precision floating point number

When a data type is described as unsigned, the leftmost bit will not contain a sign. When an array is described, the number of elements of the specified type will be enclosed in brackets following the item name. For example, the description, "char formation[36]", means that the item, formation, is an array of 36 elements and each element is a 1-byte character.

A.1.3 Data Elements of a SAF Packet

The packets that are passed between the workstation and simhost are made up of two components: a header plus message data. Every packet has a header that can identify a unit (or vehicle) and a message type. Most packets contain message data that is specific to the message type. A few types of packets do not require any message data.

A.1.3.1 Header Data

Description: The header contains two fields, an identifier and a message type. The identifier contains relevant vehicle or unit identifier data. Usually this identifier specifies the unit or vehicle that the message applies to.

Simhost ID: hdr

Workstation ID: opfor-header

Content:

<u>ITEM</u>	<u>TYPE</u>	<u>COMMENT</u>
id	unsigned short	unit or vehicle ID
id	unsigned short	message type

A.1.3.2 Message Data

See the next two chapters for descriptions of message data.

A.2 Workstation to Simhost Message Descriptions

A.2.1 Set-Up Messages

A.2.1.1 Create Message

Description: The create message is sent each time the user creates a force via the Create Unit command on the battlemaster screen. When an exercise is restored via the Restore Exercise command on the battlemaster screen, a create message is automatically generated and sent for each created force that was saved in the exercise file. The create message causes a unit and all of its inferior units and vehicles to be instantiated by the simhost. See "The SIMNET Network and Protocols", BBN report number 7102, July 31, 1989 or the files mentioned in the field descriptions below for additional information.

Message Type: 1

Simhost ID: OPFOR_MSG_CREATE

Workstation ID: create-request

Content:

<u>ITEM</u>	<u>TYPE</u>	<u>COMMENT</u>
forceID	unsigned char	belongs to US or USSR force
countryD	unsigned char	country distinguished vehicle guise
countryO	unsigned char	country other vehicle guise
tactics	unsigned char	Warsaw Pact or NATO tactics
echelon	unsigned char	code value from the mappings.lisp file. For example, company, platoon, vehicle etc.
echelon_type	unsigned char	code value from the mappings.lisp file. For example, tank, mech, ada etc.
sbx_uniq_id	unsigned char	a unique Symbolics-generated sequence number. Each time the user places a force via the Select Unit command on the battlemaster screen, the sequence number is incremented by 1. The sequence number is reset to 0 whenever the workstation code is booted.
padding	unsigned char	
formation[36]	char	one of the valid formation names. See red_forms.lisp and blue_forms.lisp
heading	double	heading in mils
position[3]	double	x,y,z coordinates
battalion	unsigned short	battalion number of this workstation
company	unsigned char	company assignment
platoon	unsigned char	platoon assignment
percent_ammo	unsigned char	this percent applies to each type of ammunition carried
percent_fuel	float	percent of fuel on board

A.2.1.2 Reset Message

Description: The reset message is sent each time the user deletes his forces via the Clear Units command or via the Clear Units and Overlays command. Both of these commands appear on the Command Menu on the color screen. Additionally, there is a 'Clear' type in command. In SAF version 3.9.10, global, not selective, reset is performed. The reset is the only message that travels in both directions. A reset from the workstation instructs the simhost to delete the indicated vehicles and to return the reset message as confirmation.

Message Type: 2

Simhost ID: OPFOR_MSG_RESET

Workstation ID: reset-request

Content:

<u>ITEM</u>	<u>TYPE</u>	<u>COMMENT</u>
flags	int	when set at 0x0001, the message means reset all vehicles
unit_count	int	count of relevant elements in the unit_list array, unused
unit_list[32]	unsigned int	delete these units or vehicles, unused

A.2.1.3 Vehicle Reinitialize Message

Description: When an exercise is restored via the Restore Exercise command on the battlemaster screen, a vehicle reinitialize message is automatically generated and sent for each vehicle that was saved in the exercise file. The vehicle reinitialize message restores a vehicle to its state at the time the user saved the exercise with the Save Exercise button on the commander screen.

Message Type: 5

Simhost ID: OPFOR_MSG_VEHICLE_REINIT

Workstation ID: vehicle-reinit-request

Content:

<u>ITEM</u>	<u>TYPE</u>	<u>COMMENT</u>
loads[4]	short	quantity of each type of ammo
fuel	int	quantity of fuel
status	char	mobility kill, catastrophic kill, firepower kill, dug-in. See vehicle.c.
padding[3]	char	
x	float	reinitialize vehicle at this x-coordinate
y	float	reinitialize vehicle at this y-coordinate
bearing	float	reinitialize vehicle with this heading

A.2.1.4 Disconnect Message

Description: Sent when the user selects the Connected To command in the battlemaster screen. This command closes the connection between the Symbolics and the Mips.

Message Type: 17

Simhost ID: OPFOR_MESSAGE_DISCONNECT

Workstation ID: disconnect-request

Content: 0 bytes; only a header element is needed.

A.2.1.5 Attach Stealth Message

Description: Sent when the user selects the Out-the-Window View command from the Display Menu. This message sets the site and host number of stealth to enable the out-the-window-view option.

Message Type: 39

Simhost ID: OPFOR_MSG_ATTACH_STEALTH

Workstation ID: attach-stealth-request

Content:

<u>ITEM</u>	<u>TYPE</u>	<u>COMMENT</u>
site	short	site number of the stealth
host	short	host number of the stealth

A.2.2 Ask for Data Messages

A.2.2.1 Poll Message

Description: This message polls the simhost for data. The polling time is set at fifteen seconds by default, but can be changed by the operator through the "robo cop control" command.

Message Type: 14

Simhost ID OPFOR_MSG_POLL

Workstation ID: poll-request

Content:

<u>ITEM</u>	<u>TYPE</u>	<u>COMMENT</u>
request	int	There are three types of request: GODS EYE VIEW = 0, means send all data NON GODS EYE VIEW = 1, means send only data that the unit named in header knows about COMMANDER'S EYE VIEW = 2, means send everything my troops can see

A.2.2.2 Query Sub State Message

Description: The query sub state message is sent each time the user asks for status data via the Status button on the commander screen or via the Information About command on the Display Menu. This message asks the simhost for data about the vehicle or unit (composite) named in the header. The simhost will both compose and format the message and then send it back via the generic message.

Message Type: 18

Simhost ID: OPFOR_MESSAGE_QUERY_SUB_STATE

Content: 0 bytes; only a header element is needed.

A.2.3 IVIS Messages

A.2.3.1 IVIS Xmit Transmit Message

Description: The IVIS Xmit Transmit message is sent when the user changes reporting requirements via the Set Reporting Requirements command on the Display Menu. This message indicates which IVIS packets are transmitted to the SIMNET network and which IVIS reports are sent to the Symbolics workstation.

Message Type: 19

Simhost ID: OPFOR_MSG_IVIS_XMIT_MODES

Workstation ID ivis-control-request

Content:

<u>ITEM</u>	<u>TYPE</u>	<u>COMMENT</u>
xmit_simnet_p	int	contains a code to indicate which reports to send to the SIMNET network. The five transmit codes are: SENDALL = -1 SENDNONE = 0 SENDCONTACT = 1 SENDSPOT = 2 SENDSHELL = 4
xmit_sbx_p	int	one of the transmit codes as shown above to indicate which reports are sent to the Symbolics workstation.

A.2.3.2 IVIS Parameters Message

Description: This message is sent when the operator sets the IVIS Fine Control Options accessed through the "robo cop control" command. This message allows the operator at the workstation to change key parameters for certain ivis reports.

Message Type: 19

Simhost ID: OPFOR_MSG_IVIS_PARAMETERS

Workstation ID ivis-fine control-request

Content:

<u>ITEM</u>	<u>TYPE</u>	<u>COMMENT</u>
cluster_distance	unsigned int	base the clustering of enemy vehicles on this distance in meters
decluster_distance	unsigned int	distance at which a cluster of vehicles will be broken into separate clusters; decluster distance is 120% of cluster distance
spot_rep_range_threshold	unsigned int	send a spot report when an observed enemy vehicle moves more than this distance in meters
report_monitor_time_msec	unsigned int	default time for issuing spot and contact reports
max_reappear_latency_msec	unsigned int	send a contact report if an observed enemy vehicle reappears after this latency period

A.2.4 Overlay Messages**A.2.4.1 Point Message**

Description: A user can create or edit a point control measure on the workstations's color monitor. When the workstation code determines that the simhost needs to be told that a control measure was created or edited, it sets a send bit for the control measure. This message is sent when the workstation calls for it via the send bit setting.

Message Type: 23

Simhost ID: OPFOR_MSG_POINT

Workstation ID: point-request

<u>ITEM</u>	<u>TYPE</u>	<u>COMMENT</u>
(overlay) name[20]	char	user-generated name of the point's overlay
(overlay) id	int	unique Symbolics-generated id of point's overlay
(point) name[20]	char	user-generated name of the point control measure
(point) id	int	unique Symbolics-generated id of the point
x	float	x-coordinate of the point
y		y-coordinate of the point
(route) name[20]	char	user-generated name of the route
(route) id	int	unique Symbolics-generated id of route
speed	float	assign this speed (km/hr) at this control measure
cis[40];	char	assign this combat instruction set at this point
report	short	send a report (yes or no) at this control measure
num_units	short	number of units in the applies_to array
applies_to[32]	short	apply this control measure to these units

A.2.4.2 Area Message

Description: A user can create or edit an area control measure on the workstations's color monitor. When the workstation code determines that the simhost needs to be told that a control measure was created or edited, it sets a send bit for the control measure. This message is sent when the workstation calls for it via the send bit setting.

Message Type: 24

Simhost ID: OPFOR_MSG_AREA

Workstation ID: area-request

Content::

<u>ITEM</u>	<u>TYPE</u>	<u>COMMENT</u>
(overlay) name[20]	char	user-generated name of the area's overlay
(overlay) id	int	unique Symbolics-generated id of area's overlay
(area) name[20]	char	user-generated name of the area control measure
(area) id	int	unique Symbolics-generated id of the area
(waypoint) count	int	number of waypoints in the area perimeter
waypoints[20]	float, float	x, y-coordinates of the area waypoints
type[20]	char	assembly area or battle position
speed	float	not used
cis[40]	char	assign this combat instruction set at this control measure
report	short	send a report (yes or no) at this control measure
num_units	short	number of units in the applies_to array
applies_to[32]	short	apply this control measure to these units

A.2.4.3 Zone Message

Description: A user can create or edit a zonecontrol measure on the workstations's color monitor. When the workstation code determines that the simhost needs to be told that a control measure was created or edited, it sets a send bit for the control measure. This message is sent when the workstation calls for it via the send bit setting.

Message Type: 25

Simhost ID: OPFOR_MSG_ZONE

Workstation ID: zone-request

Content:

<u>ITEM</u>	<u>TYPE</u>	<u>COMMENT</u>
(overlay) name[20]	char	user-generated name of the zone's overlay
(overlay) id	int	unique Symbolics-generated id of zone's overlay
(area) name[20]	char	user-generated name of the zone control measure
(area) id	int	unique Symbolics-generated id of the zone
(waypoint) count	int	number of waypoints in the zone perimeter
waypoints[20]	float, float	x, y-coordinates of the zone waypoints
type[20]	char	recon zone or no-fire zone, not used
speed	float	not used
cis[40]	char	assign this combat instruction set at this zone
report	short	send a report (yes or no) at this control measure
num_units	short	number of units in the applies_to array
applies_to[32]	short	apply this control measure to these units

A.2.4.4 Line Message

Description: A user can create or edit a line control measure on the workstations's color monitor. When the workstation code determines that the simhost needs to be told that a control measure was created or edited, it sets a send bit for the control measure. This message is sent when the workstation calls for it via the send bit setting.

Message Type: 26

Simhost ID: OPFOR_MSG_LINE

Workstation ID: line-request

Content:

<u>ITEM</u>	<u>TYPE</u>	<u>COMMENT</u>
(overlay) name[20]	char	user-generated name of the line's overlay
(overlay) id	int	unique Symbolics-generated id of line's overlay
(line) name[20]	char	user-generated name of the line control measure
(line) id	int	unique Symbolics-generated id of the line
(waypoint) count	int	number of waypoints in the line
waypoints[20]	float, float	x, y-coordinates of the line waypoints
speed	float	assign this speed (km/hr) at this control measure
cis[40];	char	assign this combat instruction set at this line
report	short	send a report (yes or no) at this control measure
num_units	short	number of units in the applies_to array
applies_to[32]	short	apply this control measure to these units

A.2.4.5 Route Message

Description: A user can create or edit a route control measure on the workstations's color monitor. When the workstation code determines that the simhost needs to be told that a control measure was created or edited, it sets a send bit for the control measure. This message is sent when the workstation calls for it via the send bit setting.

Message Type: 27

Simhost ID: OPFOR_MSG_ROUTE

Workstation ID: route-request

Content:

<u>ITEM</u>	<u>TYPE</u>	<u>COMMENT</u>
(overlay) name[20]	char	user-generated name of the route's overlay
(overlay) id	int	unique Symbolics-generated id of route's overlay
(line) name[20]	char	user-generated name of the route control measure
(line) id	int	unique Symbolics-generated id of the route
(waypoint) count	int	number of waypoints in the route
waypoints[100]	int, int, int	a route waypoint can be either a terrain point or a road point. When the waypoint is on the terrain , the tuple is an even-number id, x-coord, y-coord of the route waypoint. When the waypoint is on a road the tuple is an odd-number id, road segment id, direction.
num_units	short	number of units in the applies_to array
applies_to[32]	short	apply this control measure to these units

A.2.4.6 Delete Overlay Message

Description: Sent when the user v deletes his overlays via the Clear Units and Overlays command on the color screen's Command Menu or the Clear command on the Overlay Operations Menu of the commander's screen. This message tells the simhost to eliminate this overlay.

Message Type: 27

Simhost ID: OPFOR_MSG_DELETE_OVERLAY

Workstation ID: delete-overlay-request

Content:

<u>ITEM</u>	<u>TYPE</u>	<u>COMMENT</u>
(overlay) name[20]	char	user-generated name of the overlay to delete
(overlay) id	int	unique Symbolics-generated id of overlay

A.2.4.7 Execute Overlay Message

Description: Sent when the user issues an order via the Execute Overlay or Issue Frag Order command on the Subordinate Unit Tasking Display on the commander's screen. This message tells the simhost to execute this overlay. If this is the first time that this overlay is being asked to execute, all the control measures in the overlay will also be sent to the simhost.

Message Type: 29

Simhost ID: OPFOR_MSG_EXECUTE_OVERLAY

Workstation ID: execute-overlay-request

Content:

<u>ITEM</u>	<u>TYPE</u>	<u>COMMENT</u>
(overlay) name[20]	char	user-generated name of the overlay to use
(overlay) id	int	unique Symbolics-generated id of overlay
(route) name[20]	char	user-generated name of the route to follow
(route) id	int	unique Symbolics-generated id of route
initial_cis[40]	char	initial combat instruction set to execute

A.2.4.8 Delete Control Measure Message

Description: Sent when the user deletes the control measures of an overlay via the Clear command on the Overlay Operations Menu of the commander's screen. This message tells the simhost to eliminate this control measure.

Message Type: 33

Simhost ID: OPFOR_MSG_DELETE_CM

Workstation ID: delete-cm-request

Content:

<u>ITEM</u>	<u>TYPE</u>	<u>COMMENT</u>
(overlay) name[20]	char	user-generated name of the overlay to delete
(overlay) id	int	unique Symbolics-generated id of overlay
(control measure) name[20]	char	user-generated name of the control measure to delete
(control measure) id	int	unique Symbolics-generated id of control measure

A.2.5 Immediate Intervention Messages

A.2.5.1 Rejoin Unit Message

Description: Sent when the user issues a Rejoin Unit Immediate Intervention. Cancels an order that was issued to a subordinate of a unit and have the subordinate rejoin its unit. Reforms the parent unit by placing under parent command the subordinate unit that was separated from its parent. An example is the cancelling of a platoon order issued to a platoon that is part of a company. The platoon stops executing its order and returns to following its company commander's order.

Message Type: 44
Simhost ID: OPFOR_MSG_REJOIN_UNIT
Workstation ID: rejoin-unit-request
Content:: 0 data; header only.

A.2.5.2 Halt Message

Description: Sent when the user issues a Halt Immediate Intervention. Has the vehicles stop in place.

Message Type: 30
Simhost ID: OPFOR_MSG_HALT
Workstation ID: halt-request
Content:: 0 data; header only.

A.2.5.3 Resume Message

Description: Sent when the user issues a Resume or a Resume-All-Subordinates Immediate Intervention. Continues an interrupted order. If a unit has been moved off of its previously-assigned route, it returns via the shortest legal path and then resumes the behavior assigned by the overlay under which it was operating.

Message Type: 36
Simhost ID: OPFOR_MSG_RESUME_MISSION
Workstation ID: resume-request
Content:: 0 data; header only.

A.2.5.4 Speed Message

Description: Sent when the user issues a Speed Immediate Intervention. Changes speed to this value.

Message Type: 31
Simhost ID: OPFOR_MSG_CHANGE_SPEED
Workstation ID: change-speed-request
Content:

<u>ITEM</u>	<u>TYPE</u>	<u>COMMENT</u>
speed	int	new speed in km/hr

A.2.5.5 Formation Message

Description: Sent when the user issues a Formation Immediate Intervention. Changes formation to this value.

Message Type: 32
Simhost ID: OPFOR_MSG_CHANGE_FORMATION
Workstation ID: change-formation-request

Content:

<u>ITEM</u>	<u>TYPE</u>	<u>COMMENT</u>
formation[20]	char	name of a SAF formation

A.2.5.6 Follow Vehicle Message

Description: Sent when the user issues a Follow-Vehicle Immediate Intervention. Has unit follow a selected (leader) vehicle.

Message Type: 34

Simhost ID: OPFOR_MSG_FOLLOW_VEHICLE

Workstation ID: follow-vehicle-request

Content:

<u>ITEM</u>	<u>TYPE</u>	<u>COMMENT</u>
xoff	float	x-offset in leader's coordinate system
yoff	float	y-offset in leader's coordinate system
leadid	unsigned short	id of the vehicle to follow
padding	unsigned short	

A.2.5.7 Simulator In Command Message

Description: Sent when the user issues a Command-From-Simulator Immediate Intervention. Has unit take direction from a manned simulator.

Message Type: 43

Simhost ID: OPFOR_MSG_SIMULATOR_IN_COMMAND

Workstation ID: simulator-in-command-request

Content:

<u>ITEM</u>	<u>TYPE</u>	<u>COMMENT</u>
leadid	unsigned short	id of the commander vehicle
padding	unsigned short	

A.2.5.8 Go To Point Message

Description: Sent when the user issues a Go-To-Location Immediate Intervention. Has unit travel to the given location.

Message Type: 35

Simhost ID: OPFOR_MSG_GOTO_POINT

Workstation ID: go-to-point-request

Content:

<u>ITEM</u>	<u>TYPE</u>	<u>COMMENT</u>
x	float	x-coordinate of the location to travel to
y	float	y-coordinate of the location to travel to
backwardp	int	travel in forward or in reverse
type	int	type of hold for air vehicles to use when they arrive at the specified location There are four codes for the hold type: HOVER_HOLD = 0 ORBIT_HOLD = 1 RACETRACK_HOLD = 2 LAND_HOLD = 3

A.2.5.9 Face Direction Message

Description: Sent when the user issues a Face-Direction Immediate Intervention. Has unit face the direction specified.

Message Type: 37

Simhost ID: OPFOR_MSG_FACE_DIRECTION

Workstation ID: face-direction-request

Content:

<u>ITEM</u>	<u>TYPE</u>	<u>COMMENT</u>
mathradians	float	the direction the unit is to face

A.2.5.10 Hold Message

Description: Sent when the user issues a Hold Immediate Intervention. Has unit hold in the specified hold type.

Message Type: 40

Simhost ID: OPFOR_MSG_HOLD

Workstation ID: hold-request

Content:

<u>ITEM</u>	<u>TYPE</u>	<u>COMMENT</u>
type	int	type of hold for air vehicles to use. There are four codes for the hold type: HOVER_HOLD = 0 ORBIT_HOLD = 1 RACETRACK_HOLD = 2 LAND_HOLD = 3

A.2.5.11 Altitude Message

Description: Sent when the user issues an Altitude Immediate Intervention. Has unit change altitude.

Message Type: 41

Simhost ID: OPFOR_MSG_ALTITUDE

Workstation ID: change-altitude-request

Content:

<u>ITEM</u>	<u>TYPE</u>	<u>COMMENT</u>
altitude	float	change altitude to this value in meters.
type	int	type of hold for air vehicles to use. There are four codes for the hold type: ABS_ALTITUDE = 0 REL_ALTITUDE = 1.

A.2.5.12 Land Message

Description: Sent when the user issues an Altitude Immediate Intervention. Has the air unit land at the specified location.

Message Type: 45

Simhost ID: OPFOR_MSG_LAND

Workstation ID: land-request

<u>ITEM</u>	<u>TYPE</u>	<u>COMMENT</u>
x	float	x-coordinate of landing location
y	float	y-coordinate of landing location

A.2.5.13 Attack Message

Description: Sent when the user issues an Attack Immediate Intervention. Has the air unit attack according to the attack type type. Currently there is only one attack profile that both codes map to. The air vehicle travels to the start firing location. It begins to fire when it arrives at the start firing locations. It fires for ten seconds at targets nearest to the target location then reverses direction and travels about 1 km.

Message Type: 46

Simhost ID: OPFOR_MSG_ATTACK

Workstation ID: attack-request

Content:

<u>ITEM</u>	<u>TYPE</u>	<u>COMMENT</u>
attack_type	int	There are two codes for the attack type: ATTACK_RUNNING_FIRE = 1 ATTACK_HOVER_FIRE = 2.
start_x	float	x-coordinate of start firing location
start_y	float	y-coordinate of start firing location
target_x	float	x-coordinate of target location
target_y	float	y-coordinate of target location

A.2.5.14 Resupply Message

Description: Sent when the user issues a Resupply Immediate Intervention. Causes a vehicle to be refueled or reloaded from a resupply vehicle.

Message Type: 8

Simhost ID: OPFOR_MSG_RESUPPLY

Workstation ID resupply-request

Content:

<u>ITEM</u>	<u>TYPE</u>	<u>COMMENT</u>
resupply_vehicle	int	vehicle named in header is to be resupplied by this vehicle

A.2.5.15 Targeting Message

Description: Sent when the user issues a Rules-of-Engagement Immediate Intervention. Sets the unit's targeting parameters.

Message Type: 38

Simhost ID: OPFOR_MSG_SET_TARGETING

Workstation ID: targeting-request

Content:

<u>ITEM</u>	<u>TYPE</u>	<u>COMMENT</u>
firestatus	int	There are three types currently implemented: HOLD_FIRE FIRE_AT_WILL FIRE_AT_POSITION
max_engagement_range	int	can only shoot at vehicles within this range in meters
marksmanship	int	the probability of hit multiplier
position_x	float	when firestatus is set to FIRE_AT_POSITION, fire at this X-coordinate
position_y	float	when firestatus is set to FIRE_AT_POSITION, fire at this Y-coordinate
radius	float	fire at all targets within the specified radius in meters, when firestatus is set to FIRE_AT_POSITION
targets[16]	unsigned short	not used

A.2.6 Misc Messages

A.2.6.1 Artillery Message

Description: This message instructs the simhost to generate MCC-like artillery. See `prosim.c` and `sbx.c`. This message can only be requested by a special "bomb-button" command that is not generally made known to users since artillery in SAF normally is MCC-generated.

Message Type: 3

Simhost ID: OPFOR_MSG_ARTY

Workstation ID: arty-request

Content:

<u>ITEM</u>	<u>TYPE</u>	<u>COMMENT</u>
type	int	There are three types currently implemented: GROUND - generate an artillery hit on the ground at the location named in the position field. VEHICLE - generate an artillery hit near the vehicle named in the header. DEATH - generate an artillery hit to the most vulnerable part of the vehicle named in the header.
ammo	int	see definitions in <code>prosim.c</code>
fuze	int	see definitions in <code>prosim.c</code>
count	int	number of explosions
spread	float	place explosions randomly inside a square whose side is twice the value of spread
position[3]	float	for a ground explosion, use this as center of mass of explosion

A.3 Simhost to Workstation Message Descriptions

A.3.1 Set-Up Messages

A.3.1.1 Reset Message

Description: The reset is the only message that travels in both directions. A reset from the Simhost instructs the Symbolics to delete the indicated vehicles. In SAF version 3.9.10, global, not selective, reset is performed.

Message Type: 2

Simhost ID: OPFOR_MSG_RESET

Workstation ID: reset-request

Content:

<u>ITEM</u>	<u>TYPE</u>	<u>COMMENT</u>
flags	int	when set at 0x0001, the message means reset all vehicles.
unit_count	int	count of relevant elements in the unit_list array, unused.
unit_list[32]	unsigned int	delete these units or vehicles, unused.

A.3.2 Combat Messages

A.3.2.1 Ground Impact Message

Description: Sent in response to a direct fire explosion that does not impact a vehicle. The unit identifier in the header will contain the identifier of the attacker vehicle to enable the workstation color screen to display which vehicle did the shooting (indicated via the absence of solid circle at the end of the firing ray) and which location was impacted (indicated by the presence of the a solid white circle at the end of the firing ray).

Message Type: 102

Simhost ID: GROUND_IMPACT_MSG

Workstation ID: ground-impact

Content:

<u>ITEM</u>	<u>TYPE</u>	<u>COMMENT</u>
ammunition	unsigned char	type of ammo in the impact. Set at 0.
quantity	unsigned char	
padding	short	
locx	float	x coordinate of the impact
locy	float	y coordinate of the impact

A.3.2.2 Vehicle Impact Message

Description: Sent in response to a direct fire explosion on a vehicle. The unit identifier in the header will contain the identifier of the attacker vehicle to enable the workstation color screen to display which vehicle did the shooting (indicated via the absence of a solid circle at the end of the firing ray) and which vehicle was impacted (indicated by the presence of the yellow solid circle at the end of the firing ray).

Message Type: 103

Simhost ID: VEHICLE_IMPACT_MSG

Workstation ID: vehicle-impact

Content:

<u>ITEM</u>	<u>TYPE</u>	<u>COMMENT</u>
target_id	int	id of the vehicle hit
round_type	int	type of round used in the impact. Set at 0.
burst_length	int	impact burst quantity

A.3.2.3 Indirect Fire Message

Description: Used to display indirect fire bursts on the workstation color screen.

Message Type: 108

Simhost ID: INDIRECT_FIRE_MSG

Workstation ID: indirect-fire

Content:

<u>ITEM</u>	<u>TYPE</u>	<u>COMMENT</u>
ammunition	unsigned char	type of ammo. Set at 0 except set at 1 for mines.
fuze	unsigned char	fuze description. Set at 1
quantity	unsigned char	indirect fire burst quantity
rate	unsigned char	indirect fire burst rate
indirect_fire_burst[quantity]	float,float,float, short,short	the array element contains five items: x-coord of burst, y-coord of burst, z-coord of burst, time before next burst, padding

A.3.3 Simhost Status Messages

A.3.3.1 Vehicle PAE Message

Description: Sent in response to creation on the simhost of each vehicle and composite. It provides all information (position, appearance, and echelon) about a vehicle or composite.

Message Type: 120

Simhost ID: VEHICLE_PAE_MSG

Workstation ID: vehicle-pae

Content:

<u>ITEM</u>	<u>TYPE</u>	<u>COMMENT</u>
x	float	SIMNET x position
y	float	SIMNET y position
direction_angle	float	computed by simhost
turret_angle	float	computed by simhost
icon	unsigned char	appearance computed by simhost
status	unsigned char	SAF status data: immobile, catastrophic kill, etc
force	unsigned char	SIMNET force data
tactics	unsigned char	SAF tactics data: Warsaw or NATO
marking[11]	char	bumper number
port_number	int	port number for the workstation at which this SAF vehicle was created (for example, 2054)
echelon	unsigned char	code value from mappings.lisp file. For example, company, platoon, vehicle, etc.
job_desc	unsigned char	set to zero
superior_id	unsigned short	ID for superior, if any
sbx_uniq_id	unsigned char	Symbolics unique ID
top_superior_uniq_id	unsigned char	ID for top superior
relative_id	unsigned char	relative position in echelon
inf_count	unsigned char	for composites, number of inferiors
inferiors[8]	unsigned short	for composites, the inferiors

A.3.3.2 Vehicle Position Message

Description: Sent as the normal reply to a polling request from the workstation. A vehicle position message for each vehicle is a very frequent message.

Message Type: 121

Simhost ID: VEHICLE_POSITION_MSG

Workstation ID: vehicle-position

Content:

<u>ITEM</u>	<u>TYPE</u>	<u>COMMENT</u>
x	float	SIMNET x position
y	float	SIMNET y position
direction_angle	float	computed by simhost
turret_angle	float	computed by simhost

A.3.3.3 Vehicle Load Message

Description: Sent frequently to checkpoint a vehicle's ammo and fuel loads. When an exercise is saved with the Save Exercise workstation command, the most recently returned ammo and fuel loads of the exercise vehicles are also saved.

Message Type: 127

Workstation ID: vehicle-load

Simhost ID: VEHICLE_LOAD_MSG

Content:

<u>ITEM</u>	<u>TYPE</u>	<u>COMMENT</u>
loads[4]	short	one level for each weapon type
fuel	int	fuel level

A.3.3.4 Vehicle Position Poll Completed Message

Description: Sent to indicate the completion of the sending of all the vehicle position messages for this polling cycle.

Message Type: 122

Simhost ID: VEHICLE_POSITION_POLL_COMPLETED_MSG

Workstation ID: vehicle-position-poll-completed

Content: 0 bytes; only a header element is needed.

A.3.3.5 Vehicle Appearance Message

Description: Sent when a vehicle gets damaged. This is a medium frequency message used to set a vehicle's appearance.

Message Type: 123

Simhost ID: VEHICLE_APPEARANCE_MSG

Workstation ID: vehicle-appearance

<u>ITEM</u>	<u>TYPE</u>	<u>COMMENT</u>
icon	unsigned char	appearance computed by simhost
status	unsigned char	SAF status data: immobile, catastrophic kill, etc
force	unsigned char	SIMNET force data
tactis	unsigned char	SAF tactics data: Warsaw or NATO
marking[11]	char	bumper number

A.3.3.6 Vehicle Echelon Message

Description: Sent by a composite when an inferior gets killed. This is a very infrequent message used to set a vehicle's echelon information.

Message Type: 124

Simhost ID: OPFOR_MSG_VEHICLE_ECHELON

Workstation ID: Vehicle-echelon

Content:

<u>ITEM</u>	<u>TYPE</u>	<u>COMMENT</u>
port_number	int	port number for the workstation at which this SAF vehicle was created (for example, 2054)
echelon	unsigned char	code value from mappings.lisp file. For example, company, platoon, vehicle, etc.
job_desc	unsigned char	set to zero
superior_id	unsigned short	ID for superior, if any
sbx_uniq_id	unsigned char	Symbolics unique ID
top_superior_uniq_id	unsigned char	ID for top superior
relative_id	unsigned char	relative position in echelon
inf_count	unsigned char	for composites, number of inferiors
inferiors[8]	unsigned short	for composites, the inferiors

A.3.4 Misc Messages

A.3.4.1 Generic Message

Description: Used to print an arbitrary message at the workstation. The simhost will both compose and format the message and then send it to the workstation for display to the user.

Message Type: 125

Simhost ID: GENERIC_MESSAGE_MSG

Content:

<u>ITEM</u>	<u>TYPE</u>	<u>COMMENT</u>
type	int	message types, used to determine the font on the message display, are : GM_RADIO_MESSAGE = 0, means display in normal way GM_ERROR_MESSAGE = 1, means display in boldface GM_RADIO_ALERT_MESSAGE = 2, means display in boldface and also beep the workstation. This code is used for issuing a user warning, such as the simhost overload message.
message[1024]	char	the message, his string is null terminated.

A.3.4.2 Stealth Position Message

Description: Sends a stealth vehicle's position information so that a large yellow arrow can be drawn on the workstation's color monitor.

Message Type: 126

Simhost ID: STEALTH_POSITION_MSG

Content:

<u>ITEM</u>	<u>TYPE</u>	<u>COMMENT</u>
x	float	SIMNET x position
y	float	SIMNET y position
direction_angle	float	computed by simhost
turret_angle	float	computed by simhost

A.3.4.3 Machine Status Message

Description: Sent when the workstation connects to the simhost. It synchronizes the workstation time with the simhost time.

Message Type: 114

Simhost ID: MACHINE_STATUS_MSG

Content:

<u>ITEM</u>	<u>TYPE</u>	<u>COMMENT</u>
real_time_clock_value	unsigned int	current simhost time

APPENDIX B: SAF PARAMETER FILES

B.1	Introduction.....	B-2
B.2	Configuration Files	B-2
B.3	Echelon Files	B-5
	B.3.1 Red Echelon Files.....	B-5
	B.3.2 Blue Echelon Files.....	B-7
B.4	Formation Files	B-11
	B.4.1 Red Formations File.....	B-11
	B.4.2 Blue Formations File.....	B-14
B.5	Combat Instruction Set Files	B-17
	B.5.1 Red CIS File.....	B-19
	B.5.2 Blue CIS File.....	B-20
B.6	Probability of Detection files	B-21
B.7	Probability of Hit Files.....	B-22
B.8	Probability of Damage from Direct Fire Files	B-22
B.9	Probability of Damage from Indirect Fire Files	B-24
B.10	Mapping Files	B-24
B.11	Macro Substitution Files	B-24

APPENDIX B: SAF PARAMETER FILES

B.1 INTRODUCTION

This document describes the parameter files of the Semi-Automated Forces (SAF) segment of the SIMNET system. The SAF segment is divided into three Computer Software Configuration Items (CSCIs): a Workstation CSCI, a Simulation Host (Simhost) CSCI, and a Parameter Editors CSCI. The SAF parameter files are accessed by all three CSCIs. During an exercise, the parameter files are read by both the Workstation CSCI and the Simhost CSCI. Offline, some of the parameter files can be edited by a non-programmer using the editors of the Parameter Editors CSCI.

The SAF hardware is composed of a MIPS M2000 simulation host (simhost) which generates, simulates, and projects semi-automated vehicles onto the SIMNET network and a Symbolics workstation used by the SAF commander to control his semi-automated forces. The parameter files reside on the Mips simhost.

B.2 CONFIGURATION FILES

SAF replicates outward behavior of all vehicle and weapon systems to a level of realism sufficient for training and combat development. The kinematics and dynamics of the SAF vehicles are indistinguishable by soldiers in manned simulators from those expected of manned simulators. Similarly, weapons systems exhibit realistic rates of fire and realistic trajectories to the same degree as those modelled by manned weapons systems. The parameter file, `config.lisp`, contains modeling data for the vehicles and weapons simulated by the SAF Simhost CSCI. It also contains a description of these modeling parameters.

Some configuration file parameters can be edited via the Weapons Systems Editor of the Parameter Editors CSCI. The Weapons Systems Editor allows the user to examine and modify data in tabular form on the Symbolics workstation. In this way, a SAF client can insert classified data and can modify the system for combat development purposes.

The configuration file is divided into five sections:

<u>SECTION</u>	<u>COMMENT</u>
USSR VEHICLES	This section contains a set of parameter names and values for the Soviet vehicles modeled in SAF. SAF version 3.9.10 models the following Soviet ground vehicles: the t72 tank, the bmp2, the bmpsa7, the sa9, the zsu23-4, and the arcv. For air vehicles, SAF version 3.9.10 models the hind and havoc helicopters, and the su25 fixed wing aircraft.
US VEHICLES	This section contains a set of parameter names and values for the US vehicles modeled in SAF. SAF version 3.9.10 models the following US ground vehicles: the m1 tank, the m2 infantry fighting vehicle, a hummv-stinger vehicle, and an adats vehicle. For air vehicles, SAF version 3.9.10 models the oh58d scout helicopter, the ah64 Apache helicopter, and the a10 fixed wing aircraft.
USSR WEAPONS	This section contains a set of parameter names and values for the Soviet weapons modeled in SAF. SAF version 3.9.10 models the following Soviet weapons: 125mm HEAT, 125mm APDS, 30mm gun, 30mm cannon, 23mm AA, 57mm rocket, Spiral missile, Spandrel missile, Grail missile, and Gaskin missile.
US WEAPONS	This section contains a set of parameter names and values for the American weapons modeled in SAF. SAF version 3.9.10 models the following US weapons: 105mm HEAT, 105mm APDS, 25mm gun, TOW missile, ADATS missile, and Stinger missile.

**PARAMETER
DESCRIPTIONS**

This section contains the set of configuration file parameters arranged in alphabetical order. In SAF version 3.9.10, each description is represented as a tuple with four elements:

parameter name - The parameter name maps to one of the parameters appearing in the sections described above.

edit indicator - The edit indicator has the value, nil, when this parameter value can't be changed by the Weapons Systems Editor. The edit indicator has the value, t, when this parameter value can be changed by the Weapons Systems Editor.

comment - The comment appears enclosed in double quotes. This is the parameter description that will be shown to the Weapons Systems Editor user.

values indicator - The values indicator can contain either the value, nil, or a list of possible parameter values to be offered to the Weapons Systems Editor user.

B.3 ECHELON FILES

The composition parameters for setting up the echelon configuration for the various opfor (enemy) units are contained in the file named, `red_echelons.lisp`. The composition parameters for setting up the echelon configuration for the various bluefor (friendly) units are contained in the file named, `blue_echelons.lisp`.

Each echelon file organizes groups of vehicles into units, referred to as composites. This allows a single command or change to be given to entire units or groups of vehicles, simplifying the task of the person running the vehicles from the workstation. Composites "move" around the database, taking their sub composites and vehicles with them. (The composite location will be the lead vehicle of that composite, or the lead vehicle of the lead composite, etc).

In SAF, a unit consists of an echelon/unit-type pair (for example, platoon/ ada). Each of these echelon/unit-type pairs has a set of vehicles and composites. Each vehicle in a set of vehicles is described by a tuple with four elements: vehicle type, least significant bumper digit(s), job, and rank. The vehicle type values map to the vehicles described in the configuration files. The least significant bumper digit(s) is used when a bumper number is generated for a vehicle. The job values correspond to the job values in the formation files, thereby specifying where each vehicle of a unit is positioned relative to the other vehicles in its unit for each valid formation. Rank is not currently used.

The example below shows a vehicle tuple for the vehicle/tank unit. The vehicle type is the M1 tank, the bumper digit is 1, the job is `tp-pc` (platoon commander), and the rank is 3.

(m1 1 tp-pc 3)

B.3.1 Red Echelon Files

For the red force, the valid echelon types are: vehicle, platoon, company, flight-of-2, flight-of-3 and flight-of-4. The red ground forces make use of the vehicle, platoon, and company echelons. The red air forces make use of the vehicle, flight-of-2, flight-of-3 and flight-of-4 echelons.

For the red ground vehicle echelon, the valid unit-types are: tank, motorized-rifle, and ada. The vehicle/tank unit contains a `t72` vehicle with a job of platoon commander (`tp-pc`) and bumper 1. The vehicle/motorized-rifle unit contains a `bmp2` vehicle with a job of platoon commander (`mrp-pc`) and bumper 1. The vehicle/ada unit contains a `sa9` vehicle with a job of platoon commander (`ad-pc`) and bumper 1.

For the red platoon echelon, the valid units are: platoon/tank, platoon/motorized-rifle, and platoon/ada. The vehicles in a red tank platoon (`tp`), motorized-rifle platoon (`mrp`), and ada platoon (`ad`) have the following jobs: platoon commander (`pc`), wing 1 (`w1`), wing 2 (`w2`), and wing 3 (`w3`).

The tank platoon contains the following vehicles:

- a `t72` vehicle for the platoon commander (`tp-pc`) with bumper 1
- a `t72` vehicle for the wing 1 job (`tp-w1`) with bumper 2
- a `t72` vehicle for the wing 2 job (`tp-w2`) with bumper 3
- a `t72` vehicle for the wing 3 job (`tp-w3`) with bumper 4.

The motorized-rifle platoon contains the following vehicles:

- a bmp2 vehicle for the platoon commander (mrp-pc) with bumper 1
- a bmp2 vehicle for the wing 1 job (mrp-w1) with bumper 2
- a bmp2 vehicle for the wing 2 job (mrp-w2) with bumper 3
- a bmp2 vehicle for the wing 3 job (mrp-w3) with bumper 4.

The ada platoon contains the following vehicles:

- an ada vehicle for the platoon commander (ad-pc) with bumper 1
- an ada vehicle for the wing 1 job (ad-w1) with bumper 2
- an ada vehicle for the wing 2 job (ad-w2) with bumper 3
- an ada vehicle for the wing 3 job (ad-w3) with bumper 4.

For the red company echelon, the valid units are: company/motorized-rifle-reinforced, company/motorized-rifle, and company/tank.

The motorized-rifle-reinforced company contains the following vehicles:

- a bmp2 vehicle for the company commander with bumper 66
- a zsu23-4 vehicle for the zsu-23 job with bumper 00
- a bmpsa7 vehicle for the aa-bmp job with bumper 00
- one motorized-rifle platoon (mrp1) containing:
 - a bmp2 vehicle for platoon commander (mrp1-pc) with bumper 11
 - a bmp2 vehicle for the wing 1 job (mrp1-w1) with bumper 12
 - a bmp2 vehicle for the wing 2 job (mrp1-w2) with bumper 13
- one motorized-rifle platoon (mrp2) containing:
 - a bmp2 vehicle for platoon commander (mrp2-pc) with bumper 21
 - a bmp2 vehicle for the wing 1 job (mrp2-w1) with bumper 22
 - a bmp2 vehicle for the wing 2 job (mrp2-w2) with bumper 23
- one motorized-rifle platoon (mrp3) containing:
 - a bmp2 vehicle for platoon commander (mrp3-pc) with bumper 31
 - a bmp2 vehicle for the wing 1 job (mrp3-w1) with bumper 32
 - a bmp2 vehicle for the wing 2 job (mrp3-w2) with bumper 33
- one tank platoon containing:
 - a t72 vehicle for the platoon commander (tp-pc) with bumper 41
 - a t72 vehicle for the wing 1 job (tp-w1) with bumper 42
 - a t72 vehicle for the wing 2 job (tp-w2) with bumper 43
 - a t72 vehicle for the wing 2 job (tp-w1) with bumper 43.

The motorized-rifle company contains the following vehicles:

- a bmp2 vehicle for the company commander (co) with bumper 66
- one motorized-rifle platoon (mrp1) containing:
 - a bmp2 vehicle for platoon commander (mrp1-pc) with bumper 11
 - a bmp2 vehicle for the wing 1 job (mrp1-w1) with bumper 12
 - a bmp2 vehicle for the wing 2 job (mrp1-w2) with bumper 13
- one motorized-rifle platoon (mrp2) containing:
 - a bmp2 vehicle for platoon commander (mrp2-pc) with bumper 21
 - a bmp2 vehicle for the wing 1 job (mrp2-w1) with bumper 22
 - a bmp2 vehicle for the wing 2 job (mrp2-w2) with bumper 23.
- one motorized-rifle platoon (mrp3) containing:
 - a bmp2 vehicle for platoon commander (mrp3-pc) with bumper 31
 - a bmp2 vehicle for the wing 1 job (mrp3-w1) with bumper 32
 - a bmp2 vehicle for the wing 2 job (mrp3-w2) with bumper 33.

The tank company contains the following vehicles:

- a t72 vehicle for the company commander (tc-co) with bumper 66

- one tank platoon containing:

 - a t72 vehicle for the platoon commander (tp1-pc) with bumper 11

 - a t72 vehicle for the wing 1 job (tp1-w1) with bumper 12

 - a t72 vehicle for the wing 2 job (tp1-w2) with bumper 13

- one tank platoon containing:

 - a t72 vehicle for the platoon commander (tp2-pc) with bumper 21

 - a t72 vehicle for the wing 1 job (tp2-w1) with bumper 22

 - a t72 vehicle for the wing 2 job (tp2-w2) with bumper 23

- one tank platoon containing:

 - a t72 vehicle for the platoon commander (tp3-pc) with bumper 31

 - a t72 vehicle for the wing 1 job (tp3-w1) with bumper 32

 - a t72 vehicle for the wing 2 job (tp3-w2) with bumper 33.

For SAF red aircraft, the echelon/unit-type pairs are: vehicle/fwa, vehicle/havoc, vehicle/hind, flight-of-2/fwa, flight-of-2/havoc, flight-of-2/hind, flight-of-3/havoc, flight-of-3/hind, flight-of-3/fwa, flight-of-4/havoc, flight-of-4/hind, and flight-of-4/fwa.

For the red air vehicle echelon, the valid unit-types are: fwa, hind, and havoc. The vehicle/fwa unit contains a su25 vehicle with a job of leader and bumper 1. The vehicle/hind unit contains a hind vehicle with a job of leader and bumper 1. The vehicle/havoc contains a havoc vehicle with a job of leader and bumper 1.

The flight-of-2/fwa unit contains the vehicle/fwa vehicle plus an su25 vehicle for the wing1 job with bumper 2. The flight-of-2/havoc unit contains the vehicle/fwa vehicle plus a havoc vehicle for the wing1 job with bumper 2. The flight-of-2/hind unit contains the vehicle/hind vehicle plus a hind vehicle for the wing1 job with bumper 2.

The flight-of-3/fwa unit contains the flight-of-2/fwa vehicles plus an su25 vehicle for the wing2 job with bumper 3. The flight-of-3/havoc unit contains the flight-of-2/havoc vehicles plus a havoc vehicle for the wing2 job with bumper 3. The flight-of-3/hind unit contains the flight-of-3/hind vehicles plus a hind vehicle for the wing2 job with bumper 3.

The flight-of-4/fwa unit contains the flight-of-3/fwa vehicles plus an su25 vehicle for the wing3 job with bumper 4. The flight-of-4/havoc unit contains the flight-of-3/havoc vehicles plus a havoc vehicle for the wing3 job with bumper 4. The flight-of-4/hind unit contains the flight-of-4/hind vehicles plus a hind vehicle for the wing3 job with bumper 4.

B.3.2 Blue Echelon Files

For the blue echelons, the valid echelon types are: vehicle, platoon, company, flight-of-2, flight-of-3 and flight-of-4. The blue ground forces make use of the vehicle, platoon, and company echelons. The blue air forces make use of the vehicle, flight-of-2, flight-of-3 and flight-of-4 echelons.

For the blue ground vehicle echelon, the valid units are: vehicle/tank, vehicle/mechanized-infantry, and vehicle/ada. The vehicle/tank unit contains an m1 vehicle with a job of platoon commander (tp-pc) and bumper 1. The vehicle/mechanized-infantry unit contains an m2 vehicle with a job of platoon commander (mp-pc) and bumper 1. The vehicle/ada unit contains an adats vehicle with a job of platoon commander (ad-pc) and bumper 1.

For the blue platoon echelon, the valid units are: platoon/tank, platoon/mechanized-infantry, platoon/ada. The vehicles in a blue tank platoon (tp), mechanized-infantry platoon (mp), and an ada platoon (ad), have the following jobs: platoon commander (pc), wing 1 (w1), wing 2 (w2), and wing 3 (w3).

The tank platoon contains the following vehicles:

- an m1 vehicle for the platoon commander (tp-pc) with bumper 1
- an m1 vehicle for the wing 1 job (tp-w1) with bumper 2
- an m1 vehicle for the wing 2 job (tp-w2) with bumper 3
- an m1 vehicle for the wing 3 job (tp-w3) with bumper 4.

The mechanized-infantry platoon contains the following vehicles:

- an m2 vehicle for the platoon commander (mp-pc) with bumper 1
- an m2 vehicle for the wing 1 job (mp-w1) with bumper 2
- an m2 vehicle for the wing 2 job (mp-w2) with bumper 3
- an m2 vehicle for the wing 3 job (mp-w3) with bumper 4.

The ada platoon contains the following vehicles:

- an adats vehicle for the platoon commander (ad-pc) with bumper 1
- an adats vehicle for the wing 1 job (ad-w1) with bumper 2
- an adats vehicle for the wing 2 job (ad-w2) with bumper 3
- an adats vehicle for the wing 3 job (ad-w3) with bumper 4.

For the blue company echelon, the valid units are: company/mechanized-infantry-heavy, company/mechanized-infantry, company/tank, and company/tank-heavy.

The mechanized-infantry-heavy company contains the following:

- an m2 vehicle for the company commander (mc-co) with bumper 66
- an m2 vehicle for the executing officer (mc-xo) with bumper 65
- one tank platoon (tp1) containing:
 - an m1 vehicle for platoon commander job (tp1-pc) with bumper 11
 - an m1 vehicle for the wing 1 job (tp1-w1) with bumper 12
 - an m1 vehicle for the wing 2 job (tp1-w2) with bumper 13
 - an m1 vehicle for the wing 3 job (tp1-w3) with bumper 14
- one mechanized infantry platoon (mp2) containing:
 - an m2 vehicle for the platoon commander (mp2-pc) with bumper 21
 - an m2 vehicle for the wing 1 job (mp2-w1) with bumper 22
 - an m2 vehicle for the wing 2 job (mp2-w2) with bumper 23
 - an m2 vehicle for the wing 3 job (mp2-w3) with bumper 24
- one mechanized infantry platoon (mp3) containing:
 - an m2 vehicle for the platoon commander (mp3-pc) with bumper 31
 - an m2 vehicle for the wing 1 job (mp3-w1) with bumper 32
 - an m2 vehicle for the wing 2 job (mp3-w2) with bumper 33
 - an m2 vehicle for the wing 3 (mp3-w3) with bumper 34.

The mechanized-infantry company contains the following:

- an m2 vehicle for the company commander (mc-co) with bumper 66
- an m2 vehicle for the executing officer (mc-xo) with bumper 65
- one mechanized infantry platoon (mp1) containing:
 - an m2 vehicle for the platoon commander (mp1-pc) with bumper 11
 - an m2 vehicle for the wing 1 job (mp1-w1) with bumper 12
 - an m2 vehicle for the wing 2 job (mp1-w2) with bumper 13
 - an m2 vehicle for the wing 3 job (mp1-w3) with bumper 14
- one mechanized infantry platoon (mp2) containing:
 - an m2 vehicle for the platoon commander (mp2-pc) with bumper 21
 - an m2 vehicle for the wing 1 job (mp2-w1) with bumper 22
 - an m2 vehicle for the wing 2 job (mp2-w2) with bumper 23
 - an m2 vehicle for the wing 3 job (mp2-w3) with bumper 24
- one mechanized infantry platoon (mp3) containing:
 - an m2 vehicle for the platoon commander (mp3-pc) with bumper 31
 - an m2 vehicle for the wing 1 job (mp3-w1) with bumper 32
 - an m2 vehicle for the wing 2 job (mp3-w2) with bumper 33
 - an m2 vehicle for the wing 3 job (mp3-w3) with bumper 34.

The tank company contains the following:

- an m1 vehicle for the company commander (tc-co) with bumper 66
- an m1 vehicle for the executing officer (tc-xo) with bumper 65
- one tank platoon (tp1) containing:
 - an m1 vehicle for the platoon commander (tp1-pc) with bumper 11
 - an m1 vehicle for the wing 1 job (tp1-w1) with bumper 12
 - an m1 vehicle for the wing 2 job (tp1-w2) with bumper 13
 - an m1 vehicle for the wing 3 job (tp1-w3) with bumper 14
- one tank platoon (tp2) containing:
 - an m1 vehicle for the platoon commander (tp2-pc) with bumper 21
 - an m1 vehicle for the wing 1 job (tp2-w1) with bumper 22
 - an m1 vehicle for the wing 2 job (tp2-w2) with bumper 23
 - an m1 vehicle for the wing 3 job (tp2-w3) with bumper 24
- one tank platoon (tp3) containing:
 - an m1 vehicle for the platoon commander (tp3-pc) with bumper 31
 - an m1 vehicle for the wing 1 job (tp3-w1) with bumper 32
 - an m1 vehicle for the wing 2 job (tp3-w2) with bumper 33
 - an m1 vehicle for the wing 3 job (tp3-w3) with bumper 34.

The tank-heavy company contains the following:

- an m1 vehicle for the company commander (tc-co) with bumper 66
- an m1 vehicle for the executing officer (tc-xo) with bumper 65
- one mechanized-infantry platoon (mp1) containing:
 - an m2 vehicle for the platoon commander (mp1-pc) with bumper 11
 - an m2 vehicle for the wing 1 job (mp1-w1) with bumper 12
 - an m2 vehicle for the wing 2 job (mp1-w2) with bumper 13
 - an m2 vehicle for the wing 3 job (mp1-w3) with bumper 14
- one tank platoon (tp2) containing:
 - an m1 vehicle for the platoon commander (tp2-pc) with bumper 21
 - an m1 vehicle for the wing 1 job (tp2-w1) with bumper 22
 - an m1 vehicle for the wing 2 job (tp2-w2) with bumper 23
 - an m1 vehicle for the wing 3 job (tp2-w3) with bumper 24

one tank platoon (tp3) containing:

- an m1 vehicle for the platoon commander (tp3-pc) with bumper 31
- an m1 vehicle for the wing 1 job (tp3-w1) with bumper 32
- an m1 vehicle for the wing 2 job (tp3-w2) with bumper 33
- an m1 vehicle for the wing 3 job (tp3-w3) with bumper 34.

For SAF blue aircraft, the echelon/unit-type pairs are: vehicle/fwa, vehicle/attack-rwa, vehicle/scout-rwa, flight-of-2/fwa, flight-of-2/attack-rwa, flight-of-2/scout-rwa, flight-of-3/fwa, flight-of-3/attack-rwa, flight-of-3/scout-rwa, flight-of-4/fwa, flight-of-4/attack-rwa, and flight-of-4/scout-rwa.

For the blue air vehicle echelon, the valid units are: vehicle/fwa, vehicle/scout-rwa and vehicle/attack-rwa. The vehicle/fwa unit contains an a10 vehicle with a job of leader and bumper 1. The vehicle/attack-rwa unit contains an ah64 vehicle with a job of leader and bumper 1. The vehicle/scout-rwa contains an Oh58d vehicle with a job of leader and bumper 1.

The flight-of-2/fwa unit contains the vehicle/fwa vehicle plus an a10 vehicle for the wing1 job with bumper 2. The flight-of-2/attack-rwa unit contains the vehicle/attack-rwa vehicle plus an ah64 vehicle for the wing1 job with bumper 2. The flight-of-2/scout-rwa unit contains the vehicle/scout-rwa vehicle plus an Oh58d vehicle for the wing1 job with bumper 2.

The flight-of-3/fwa unit contains the flight-of-2/fwa vehicles plus an a10 vehicle for the wing2 job with bumper 3. The flight-of-3/attack-rwa unit contains the flight-of-2/attack-rwa vehicles plus an ah64 vehicle for the wing2 job with bumper 3. The flight-of-3/scout-rwa unit contains the flight-of-2/scout-rwa vehicles plus an oh58d vehicle for the wing2 job with bumper 3.

The flight-of-4/fwa unit contains the flight-of-3/fwa vehicles plus an a10 vehicle for the wing3 job with bumper 4. The flight-of-4/attack-rwa unit contains the flight-of-3/attack-rwa vehicles plus an ah64 vehicle for the wing3 job with bumper 4. The flight-of-4/scout-rwa unit contains the flight-of-3/scout-rwa vehicles plus a oh58d vehicle for the wing3 job with bumper 4.

B.4 FORMATION FILES

The positioning parameters of the various red force units are contained in the file named, `red_forms.lisp`. The positioning parameters of the blue force formations are contained in the file named, `blue_forms.lisp`.

Vehicles are set up in positions relative to other vehicles in the composite and sub composite structure through the use of the formation files. The Simhost CSCI uses the formation data (for example, who will follow who in formation keeping, spacing) to place the vehicles of a unit into their proper position. Note that who a vehicle follows is not usually the vehicle it takes orders from.

In SAF, a unit consists of an echelon/unit-type pair (for example, platoon/ada). Each of these units has a set of valid formations. Each valid formation contains a table of positioning tuples for each vehicle in the unit. Each tuple consists of three elements: a vehicle job, an x coordinate, and a y coordinate. The example below shows a table for the road formation of an ada platoon. The jobs are `ad-pc` (platoon commander), `ad-w1` (wingman 1), `ad-w2` (wingman 2), and `ad-w3` (wingman 3). These are the same jobs found in the echelon file.

Job	x	y

<code>ad-w3</code>	0	225
<code>ad-pc</code>	0	150
<code>ad-w1</code>	0	75
<code>ad-w2</code>	0	0

The positioning values (x,y) of the formation tables can easily be changed by using the Formations Editor component of the Parameter Editors CSCI. The Formations Editor allows the user to examine and modify data in both tabular and graphical form on the Symbolics workstation. In this way, a SAF client can insert classified data and can modify the formations for combat development purposes.

B.4.1 Red Formations File

In SAF version 3.9.10, the echelon names are vehicle, platoon, flight-of-2, flight-of-3, flight-of-4 and company. The vehicle echelon implies a single vehicle and therefore does not map to any formations. Formations are provided for the other echelons in the formations file, `red_forms.lisp`.

The red formations file is divided by echelon name into five sections:

- 1 Platoon Formations - This section contains a set of formations (road, line, echelon-right, echelon-left, wedge, vee, and air raid) for the following unit-types:

Ada- An ada platoon's only job in SAF version 3.9.10 is to be the single platoon of the platoon/ada unit. Therefore, only one table of positioning tuples is needed for each on the platoon formations.

Tank- It is possible for a platoon of a company to operate independently. Therefore, in SAF version 3.9.10 there are four tables (one each for `tp`, `tp1`, `tp2`, and `tp3`) of positioning tuples for each of the platoon formations.

Motorized-rifle - It is possible for a platoon of a company to operate independently. Therefore, in SAF version 3.9.10 there are four tables (one each for mrp, mrp1, mrp2, and mrp3) of positioning tuples for each of the platoon formations.

- 2 **Company Formations** - This section contains a set of formations (attack, road, line, wedge, vee, march-internal, march-rear, battle-position, occupy-area, and air raid) for the following unit-types:

Motorized-rifle-reinforced (MRR) - One table of positioning tuples is provided for each on the company formations.

Motorized-rifle (MR) - One table of positioning tuples is provided for each on the company formations.

Tank - One table of positioning tuples is provided for each on the company formations.

- 3 **Flight-of-2 Formations** - This section contains a set of formations for the following unit types: fwa, havoc, and hind. Figure 4-1 shows the valid formations for the flight-of-2 units.
- 4 **Flight-of-3 Formations** - This section contains a set of formations for the following unit types: fwa, havoc, and hind. Figure 4-1 shows the valid formations for the flight-of-3 units.
- 5 **Flight-of-4 Formations** - This section contains a set of formations for the following unit types: fwa, havoc, and hind. Figure 4-1 shows the valid formations for the flight-of-4 units.

Red Flight-of-2 Formations			
formation	fwa	havoc	hind
fighting-wing	X	X	X
line-abreast	X	O	O
bearing	X	O	O
trail	X	X	X
combat-line	O	X	X
extended-combat-line	O	X	X
line	O	X	X
Red Flight-of-3 Formations			
formation	fwa	havoc	hind
fighting-wing	X	O	O
line-abreast	X	O	O
bearing	X	O	O
trail	X	X	X
combat-line	O	X	X
extended-combat-line	O	X	X
vic	X	O	O
wedge	O	X	X
echelon-left	O	X	X
echelon-right	O	X	X
line	O	X	X
Red Flight-of-4 Formations			
formation	fwa	havoc	hind
fighting-wing	X	O	O
lineabreast	X	O	O
bearing	X	O	O
trail	X	X	X
scatter	X	X	X
combat-line	O	X	X
extended-combat-line	O	X	X
box	X	O	O
wedge	O	X	X
echelon-left	O	X	X
echelon-right	O	X	X
line	O	X	X
staggered-trail	O	X	X
reverse-combat-wedge	O	X	X

Figure 4-1: Red Air Formations

B.4.2 Blue Formations File

In SAF version 3.9.10, the echelon names are vehicle, platoon, flight-of-2, flight-of-3, flight-of-4 and company. The vehicle echelon implies a single vehicle and therefore does not map to any formations. Formations are provided for the other echelons in the file, blue_forms.lisp.

The blue formations file is divided by echelon name into five sections:

- 1 Platoon Formations - This section contains a set of formations for the following unit types:

Ada - An ada platoon's only job in SAF version 3.9.10 is to be the single platoon of the platoon/ada unit. Therefore, only one table of positioning tuples is needed for each on the following formations: road, staggered-column, line, echelon-right, echelon-left, wedge, vee, and air raid.

Tank - It is possible for a platoon of a company to operate independently. Therefore, in SAF version 3.9.10 there are four tables (one each for tp, tp1, tp2, and tp3) of positioning tuples for each of the following formations: road, staggered-column, line, echelon-right, echelon-left, wedge, vee, and air raid.

Mechanized-infantry - It is possible for a platoon of a company to operate independently. Therefore, in SAF version 3.9.10 there are four tables (one each for mp, mp1, mp2, and mp3) of positioning tuples for each of the following formations: road, staggered-column, line, echelon-right, echelon-left, wedge, vee, scout, and air raid.

- 2 Company Formations - This section contains a set of formations (attack, road, line, wedge, vee, march-internal, march-rear, battle-position, occupy-area, and air raid) for the following unit types:

Mechanized-infantry - One table of positioning tuples is provided for each on the company formations listed above.

Mechanized-infantry-heavy - One table of positioning tuples is provided for each on the company formations listed above.

Tank - One table of positioning tuples is provided for each on the company formations listed above.

Tank-heavy - One table of positioning tuples is provided for each on the company formations listed above.

- 3 Flight-of-2 Formations - This section contains a set of formations for the following unit types: fwa, scout-rwa, and attack-rwa. Figure 4-2 shows the valid formations for the flight-of-2 units.
- 4 Flight-of-3 Formations - This section contains a set of formations for the following unit types: fwa, scout-rwa, and attack-rwa. Figure 4-2 shows the valid formations for the flight-of-3 units.

- 5 **Flight-of-4 Formations** - This section contains a set of formations for the following unit types: fwa, scout-rwa, and attack-rwa. Figure 4-2 shows the valid formations for the flight-of-4 units.

Blue Flight-of-2 Formations			
formation	fwa	scout-rwa	attack-rwa
fighting-wing	X	X	X
line-abreast	X	X	X
bearing	X	O	O
trail	X	X	X
scatter	X	X	X
combat-line	O	X	X
extended-combat-line	O	X	X
Blue Flight-of-3 Formations			
formation	fwa	scout-rwa	attack-rwa
fighting-wing	X	O	O
line-abreast	X	O	O
trail	X	X	X
scatter	X	X	X
combat-line	O	X	X
extended-combat-line	O	X	X
vic	X	O	O
wedge	O	X	X
echelon-left	O	X	X
echelon-right	O	X	X
line	O	X	X
Blue Flight-of-4 Formations			
formation	fwa	scout-rwa	attack-rwa
fighting-wing	X	O	O
line-abreast	X	O	O
bearing	X	O	O
trail	X	X	X
scatter	X	X	X
combat-line	O	X	X
extended-combat-line	O	X	X
box	X	O	O
wedge	O	X	X
echelon-left	O	X	X
echelon-right	O	X	X
line	O	X	X
staggered-trail	O	X	X
reverse-combat-wedge	O	X	X

Figure 4-2: Blue Air Formations

B.5 COMBAT INSTRUCTION SET FILES

A military operation for the units under command is specified via a Combat Instruction Set (CIS), which corresponds to a battle drill since it defines a standardized form of behavior. The parameters for the combat instruction sets of the red force units are contained in the file named, `red_cis.lisp`. The parameters for the combat instruction sets of the blue force units are contained in the file named, `blue_cis.lisp`.

Aggregate levels, from platoons up to companies, are controlled by doctrinal tactics involving CIS necessary to carry out the fundamental tasks of move, shoot, communicate. Via the workstation, the SAF commander is able to have a CIS executed. Default values are provided by the SAF system for parameters, these values being drawn from standard military doctrine. However these default values are open to modification by the SAF commander.

During the planning process, a workstation commander can tag a control measure with a CIS so that a unit executes that CIS upon reaching the tagged control measure. With a CIS, the SAF system will take charge and generate all the detailed instructions needed to make the units involved, down to the last vehicle, behave in accordance to the CIS. Thus, for a tank company attack CIS, appropriate formation and speeds will be passed along all the way down to individual vehicles. In this way, all of the company units will act in a coordinated manner in support of the CIS.

A CIS usually begins intentionally; in other words, it begins when the SAF commander tells it to via subordinate unit tasking or when the route being travelled by a unit crosses a control measure that is both tagged with that CIS and assigned to that unit. Some CIS are spontaneous which means they are generated automatically through what is called a situational interrupt. For example, when hit by a surprise attack, vehicles in a unit seek the high ground between themselves and the enemy without the SAF commander having to micro-manage the platoons or individual vehicles.

The following are examples of situational interrupts:

For companies:

A ground enemy engagement can force a Hasty-Attack CIS - If a threat of attack occurs (four or more forward or advancing enemy ground vehicles within 2000 meters), the company moves into an attack position (line) and stops advancing at the high ground between itself and the enemy. It then awaits further orders.

An air strike forces a React-to-Air-Raid CIS - If an air strike occurs while a company is in one of the march or prebattle combat instruction sets, stagger the formation by having the vehicles move 100 meters to either side of its centerline (air-raid formation) and then engage aircraft. After three minutes of no firing, the vehicles resume the original CIS. The React-To-Air-Raid CIS is the only CIS that actually ends on its own. It ends when three minutes elapse without an air strike.

At that time the SAF system automatically transitions the unit back to its current CIS (the CIS it was executing before it needed to jump into the React-To-Air-Raid CIS)

For platoons:

Action Drill Needed - If a platoon is engaged by 2 or more advancing or forward enemy ground vehicles within 2000 meters then it moves into an attack position (line) facing the enemy center of mass and stops advancing when it gets to the high ground in front of the enemy. It remains there engaging enemy and awaits further orders.

Each CIS contains a default speed and formation. An immediate intervention or a control measure intersection can cause a speed change and can cause a unit to transition to one of its valid formations. Even though a unit does stop moving when it reaches the end of a route, it will still be in a state of executing the current CIS and will therefore need to check for situational interrupts.

The example below shows a sample CIS entry (named action-drill) and explains how the Simhost CSCI interprets it.

```
(action-drill)
  (parameters)
  (control-measures)
  (situational t)
  (resumable t)
  (formation line)
  (speed 25)
  (move high-ground)
  (communicate "Commencing action drill")
  (enabled-predicates))
```

The action-drill CIS directs the platoon to assume the line formation, adjust its speed to 25 kph, move to the high ground between itself and the enemy, and then stop and turn to face the enemy center of mass. The message, 'Commencing action drill', will be sent to the workstation to confirm the performance of this CIS. This CIS has no enabled-predicates so the CIS ends when the unit receives another task via subordinate unit tasking. The tuple, (resumable t), indicates that this CIS is resumable. This means that if the user issues a RESUME immediate intervention, the unit will continue to travel along the route it was following before transitioning to the action-drill CIS. When the route is resumed, the unit advances to the waypoint it was heading for when the situational interrupt occurred.

In SAF version 3.9.10, each CIS contains nine entries:

- 1 parameters - currently unused. Later versions of SAF intend to give the user more ability to change default CIS parameter values (such as speed).
- 2 control-measures - possible values are derived from the set of control measures that this CIS can be tagged to. Currently the set includes the following: point, line, zone and area.
- 3 situational - possible value is t or nil. A value of t indicates that this CIS can be transitioned to automatically by one of the situational interrupts described above. A value of nil indicates that this CIS is only executed by direct intent of the user.

- 4 resumable - possible value is t or nil. The tuple, (resumable t), indicates that this CIS is resumable. This means that if the user issues a RESUME immediate intervention, the unit will continue to travel along the route it was following when the situational interrupt occurred. The tuple, (resumable nil), indicates that this CIS is not resumable. This means that if the user issues a RESUME immediate intervention, the unit will ignore it.
- 5 formation - possible value is one of the formation names given in the formation file. The unit is directed to assume this formation when it executes this CIS.
- 6 speed - possible value is a speed in kph to indicate the speed the unit is to assume when executing this CIS.
- 7 move - possible values are one of the following:
 - halt* to indicate that the vehicles are to stop moving
 - withdrawing* to indicate that the vehicles are to travel along the withdraw route with their hulls at 0 mils relative to the route and their turrets at 3200 mils relative to their hulls.
 - high-ground* to indicate that the vehicles are to advance to the highest point between themselves and the enemy
 - along-route* to indicate that the vehicles are to continue on their route.
- 8 communicate - value is a string that will be displayed in the Message Log Pane when the unit begins executing this CIS.
- 9 enabled-predicates - value is the name of any tests that need to be performed while the unit is executing this CIS.

B.5.1 Red CIS File

In SAF version 3.9.10, the echelon names are vehicle, platoon, company, flight-of-2, flight-of-3, and flight-of-4. The vehicle echelon implies a single vehicle and therefore does not need any CIS. CIS are provided for the other echelons in the file, red_cis.lisp.

The red CIS file is divided by echelon name into five sections:

- 1 Company CIS - In SAF version 3.9.10, there are twelve implemented Red Company CIS: occupy-assembly-area, roadmarch, march-as-internal, march-as-rear, pre-battle-wedge, pre-battle-line, pre-battle-vee, attack, occupy-battle-position, withdraw, hasty-attack, and react to air-raid.
- 2 Platoon CIS - In SAF version 3.9.10, there are six implemented Red Platoon CIS: roadmarch, assault, occupy-assembly-area, occupy-battle-position, withdraw, and action-drill.
- 3 Flight-of-2 CIS - In SAF version 3.9.10, there are no implemented air CIS.
- 4 Flight-of-3 CIS - In SAF version 3.9.10, there are no implemented air CIS.
- 5 Flight-of-4 CIS - In SAF version 3.9.10, there are no implemented air CIS.

B.5.2 Blue CIS File

In SAF version 3.9.10, the echelon names are vehicle, platoon, company, flight-of-2, flight-of-3, and flight-of-4. The vehicle echelon implies a single vehicle and therefore does not need any CIS. CIS are provided for the other echelons in the file, blue_cis.lisp.

The blue CIS file is divided by echelon name into five sections:

- 1 Company CIS - In SAF version 3.9.10, there are eight implemented Blue Company CIS: occupy-assembly-area, roadmarch, move-in-wedge, move-in-vee, attack, occupy-battle-position, hasty-attack, and react to air-raid.
- 2 Platoon CIS - In SAF version 3.9.10, there are twelve implemented Blue Platoon CIS: roadmarch, combat-formation-column, combat-formation-line, combat-formation-wedge, combat-formation-vee, combat-formation-echelon-left, combat-formation-echelon-right, assault, occupy-assembly-area, occupy-battle-position, withdraw, and action-drill.
- 3 Flight-of-2 CIS - In SAF version 3.9.10, there are no implemented air CIS.
- 4 Flight-of-3 CIS - In SAF version 3.9.10, there are no implemented air CIS.
- 5 Flight-of-4 CIS - In SAF version 3.9.10, there are no implemented air CIS.

B.6 PROBABILITY OF DETECTION FILES

The probability of detection tables for both the red and blue force are located in the file named detection.lisp.

The intervisibility algorithm is based on the idea of running a ray from an eye point to a target and checking whether the ray encounters any obstacles (can we see from the ray start point to the target). The algorithm will detect both terrain and structural obstacles. A target is fully visible when there are no obstacles between the eye point and the bottom of the target. A target is partly visible when there are obstacles between the eye point and the bottom of the target but no obstacles between the eye point and the top of the target, for example a hull down tank. A target is invisible when there are obstacles between the eye point and the top of the target. If there is no line of sight between the target and the observer, then the probability of detection is zero.

Probability of detection tables vary according to the state of the target (target aspect), range, and arc of observation. The states of the target are: partially visible (hull defilade), fully visible stationary (stationary) and fully visible moving (moving). If the target is partially visible, for example hull down or occluded by terrain, then the probability of detection is reduced from that for a fully visible target. The observation and surveillance capabilities of the crew are aggregated for simplicity by the concept of arcs of observation. The primary arc of observation (120 degrees) is centered on the armament or main axis of the platform. If the target is not within the primary arc, a secondary arc detection probability table is used.

The detection file contains four sections:

- 1 Primary Arc Probability of Detection for an Air Vehicle
- 2 Secondary Arc Probability of Detection for an Air Vehicle
- 3 Primary Arc Probability of Detection for a Ground Vehicle
- 4 Secondary Arc Probability of Detection for a Ground Vehicle.

Each section contains three probability tables, one for each target aspect (stationary, moving, or hull defilade). The table consists of a set of tuples with two elements: a range and a probability of detection at that range.

The range and probability values of the detection tables used by the SAF Simhost CSCI can easily be changed by using the Model Editor component of the Parameter Editors CSCI. The Model Editor allows the user to examine and modify data in both tabular and graphical form on the Symbolics workstation. In this way, a SAF client can insert classified data and can modify the system for combat development purposes.

B.7 PROBABILITY OF HIT FILES

The parameter file, `hitmodels.lisp`, contains the probability of hit tables for the various munitions used by the SAF Simhost CSCI.

During combat, the simhost determines if a SAF weapon hits its target. When a SAF weapon fires at a target the probability of hit is determined by the range, weapon and ammunition type, and state (target aspect). The states of the target are: partially visible (hull defilade), fully visible stationary (stationary) and fully visible moving (moving).

The probability of hit file contains five tables:

- 1 Probability of Hitting All Targets for 25mm Gun
- 2 Probability of Hitting All Targets for 105mm Gun
- 3 Probability of Hitting All Targets for a Missile
- 4 Probability of Hitting All Targets for a 23mm AA Missile
- 5 Probability of Hitting All Targets for a 57mm Rocket.

The range and probability values of the hit tables used by the SAF Simhost CSCI can easily be changed by using the Model Editor component of the Parameter Editors CSCI. The Model Editor allows the user to examine and modify data in both tabular and graphical form on the Symbolics workstation. In this way, a SAF client can insert classified data and can modify the system for combat development purposes.

B.8 PROBABILITY OF DAMAGE FROM DIRECT FIRE FILES

The SAF vehicles and weapons systems are subject to catastrophic, mobility, and firepower combat damage according to the type of weapon used, the impacted area, and the angle of incidence. The probability of damage from direct fire tables are located in the file named `df_damage.lisp`.

In SAF version 3.9.10 the file, `df_damg.lisp`, is divided into four sections: tank, ifv, helo (helicopter) and airplane. Each section contains a probability of direct fire damage table for each type of attacking weapon. The types of attacking weapons are: heat105, sabot105, heat25, sabot25, tow_missile, hellfire_missile, antiaircraft_missile, small arms, mine, and bomb. The file, `damage_map.lisp`, maps these weapons to all those present in the SIMNET protocol.

Each of these direct fire damage tables contains probability entries for eight impact areas. The impact area is determined by both type of hit and location of hit. The type of hit is either a hull hit or a turret hit. The location of hit is either front, side, back, or top. Therefore, the eight impact areas are: hull front, hull side, hull back, hull top, turret front, turret side, turret back, and turret top.

Each probability entry consists of three probability tuples, one for each of the following angle of incidence grouping: 0 to 30 degrees, 30 to 60 degrees, and 60 to 90 degrees. The angle of incidence, determined from the direction of the incoming round, varies from 0 to 90 degrees as the incoming round moves from being tangential to perpendicular to the impacted side. Each probability tuple contains three elements: a probability of catastrophic damage, a probability of mobility damage, and a probability of firepower damage.

Each configuration file vehicle contains a parameter, `df_damage_model`, whose value indicates which section of `df_damg.lisp` applies to the vehicle. In SAF version 3.9.10, both the `t72` and the `m1` vehicles are directed to use the tank direct fire damage tables. The `bmp2`, `bmps7`, `sa9`, `zsu23-4`, `arcv`, `m2`, `hummv-stinger`, and `adat` vehicles are directed to use the `ifv` direct fire damage tables. The `hind`, `havoc`, `ah64`, and `oh58d` vehicles are directed to use the `helo` direct fire damage tables. The `su25` and `a10` vehicles are directed to use the airplane direct fire damage tables.

The probability values of the direct fire damage tables used by the SAF Simhost CSCI can easily be changed by using the Model Editor component of the Parameter Editors CSCI. The Model Editor allows the user to examine and modify data in both tabular and graphical form on the Symbolics workstation. In this way, a SAF client can insert classified data and can modify the system for combat development purposes.

B.9 PROBABILITY OF DAMAGE FROM INDIRECT FIRE FILES

The SAF vehicles and weapons are subject to catastrophic, mobility, and firepower combat damage according to the type of weapon used and the range. The probability of damage tables for indirect fire are located in the file named `if_damage.lisp`.

In SAF version 3.9.10 the file, `if_damg.lisp`, is divided into two sections: tank and ifv. Each section contains a probability of direct fire damage table for each type of indirect fire weapon. The types of indirect fire weapons are: bomb, pd155, vt155, pd107, vt107, and mine.

Each configuration file vehicle contains a parameter, `if_damage_model`, whose value indicates which section of `if_damg.lisp` applies to the vehicle. In SAF version 3.9.10, both the t72 and the m1 vehicles are directed to use the tank indirect fire damage tables. The bmp2, bmpsa7, sa9, zsu23-4, arcv, m2, hummv-stinger, adats, hind, havoc, ah64, oh58d, su25, and a10 vehicles are directed to use the ifv indirect fire damage tables.

The probability values of the indirect fire damage tables used by the SAF Simhost CSCI can easily be changed by using the Model Editor component of the Parameter Editors CSCI. The Model Editor allows the user to examine and modify data in both tabular and graphical form on the Symbolics workstation. In this way, a SAF client can insert classified data and can modify the system for combat development purposes.

B.10 MAPPING FILES

The file, `mappings.lisp`, contains four sections and each section is headed with a comment describing its rules for use. The first section maps numbers to icons used by the workstation. The second section maps numbers to the echelons described in the echelon files (`red_echelons.lisp` and `blue_echelons.lisp`). The third section maps numbers to the echelon-type (also referred to as the unit-type). The fourth section maps the icon names of the first section to the echelon-types of the third section.

The file, `damage_map.lisp`, maps the type of attacking weapon used in the combat damage tables to all those present in the SIMNET protocol. In SAF 3.9.10 the types of attacking weapons are: heat105, sabot105, heat25, sabot25, tow_missile, hellfire_missile, antiaircraft_missile, small arms, mine, and bomb.

B.11 MACRO SUBSTITUTION FILES

Every constant in the SIMNET 6.6 protocol has an entry in the file, `simnet.mac`. At simhost runtime, this file is read to convert values in the configuration files to the appropriate SIMNET values. The name, `simnet.mac`, (an abbreviation for SIMNET macro) is used because this process is similar to a macro substitution.

Every constant in the SIMNET 6.0 protocol has an entry in the file, `simnet6-0.mac`. When the 6.0 protocol is specified at simhost runtime, this file is read to convert values in the configuration files to the appropriate SIMNET values. The name, `simnet6-0.mac`, (an abbreviation for SIMNET6-0.mac) is used because this process is similar to a macro substitution.

APPENDIX C: SAF TERRAIN FILES

C.1	Introduction.....	C-2
C.2	Polygons File.....	C-3
C.3	Quadtree Files.....	C-4
C.4	Nets File	C-9
	C.4.1 Segment Data.....	C-10
	C.4.2 Intersection Data	C-10
	C.4.3 Bridge Data	C-12
	C.4.4 Rail Data	C-12
C.5	Model File	C-12
C.6	Ground File.....	C-13
C.7	Contours File.....	C-13

APPENDIX C: SAF TERRAIN FILES

C.1 INTRODUCTION

Within the SAF project there is a requirement for both a polygonal and a quadtree representation of terrain. The low level terrain needed for SAF vehicle simulation is represented by the polygonal data structure. The higher level terrain needed for displaying terrain to the commander and for terrain reasoning is represented by a quadtree data structure optimized for these activities.

The simulation of the SAF vehicles uses a terrain carpet built from a large number of small polygons, in a format similar to that used in the SIMNET Computer Image Generator (CIG). The runtime terrain database used by the manned simulators consists of a large number of polygons, of two types, which are optimized to display quickly when used with the CIG hardware. The terrain polygons, which have different elevations at the vertices, are used to display the underlying terrain. Terrain objects, which are made up of large numbers of individual polygons, are used to display the terrain specific objects on the underlying terrain. The individual runtime objects are made up of a large number of polygons for two reasons. First, the runtime database is divided into load modules to limit the number of polygons that need to be handled at any one time by the CIG hardware. This forces objects to be broken up at these runtime module boundaries. Second, objects are broken up so that they lie flat on the underlying terrain polygons. This large number of object polygons make drawing of the terrain objects very slow. Also, reasoning on these objects is extremely difficult because there is no connection between the individual polygons of the same object.

To create a diversified data structure database that works well for both display and reasoning, the terrain model in the SAF program processes the runtime database to store the terrain as feature arrays with a quadtree structure of feature indices into these feature arrays. The quadtree terrain database is object oriented, with individual terrain objects identified as such and appropriate information about those objects stored with them. Also, connections between objects are identified. For example, roads are stored as individual road segments, with attributes such as road width and distance. Also, road network information is stored to determine which road segments meet at the intersections. The combat instruction set and route planning routines need to know about the terrain in order to make intelligent decisions about where to go and how to perform. For fast map drawing, the terrain database needs to have the least amount of items to be drawn. These items are identified as to what type of object they are, in order to draw them appropriately. The display routines use the quadtree representation to paint a map, at various scales, onto the workstation display.

Several types of terrain objects are stored and each type is stored in a data structure which is optimized for that object type. The four files that hold the terrain objects in the SAF version 3.9.10 terrain database are named:

- **nets** - This file stores data for road segments, road intersections, river segments, river intersections, bridges and rails.
- **models** - This file stores data for trees, tree lines, canopies, and buildings.
- **contours** - This file stores data for contour lines.
- **ground** - This file stores data for lakes, ocean, or other ground soil types that are not standard hard-packed dirt.

The following types are used to describe the data of the SAF terrain files:

- **INT8** - 8 bits, 1-byte capable of holding one character
- **INT16** - 16 bits, 2-bytes capable of holding one short integer
- **INT32** - 32 bits, 4-bytes capable of holding one integer

When an array is described, the number of elements of the specified type will be enclosed in brackets following the item name. For example, the description, "points[num_points]", means that the item, points, is an array of elements. The number of elements in the array is specified by the value of the item, num_points.

C.2 POLYGONS FILE

The polygons file represents the terrain as a carpet of polygons which have different elevations at the vertices. The polygonal data structure is used to determine the support polygon of the ground vehicles from which can be derived the maximal accelerations and the orientation of the vehicles. Additionally, the polygons file is used by the intervisibility code to determine if two vehicles have a line of sight between them.

C.3 QUADTREE FILES

There are two quadtree files: a quadtree full file and a quadtree leaf file. Each file contains a quadtree structure which divides the terrain database into 500 square meter terrain patches. In the quadtree full file, each node of the quadtree has an index into what is called a feature array. The feature array contains pointers into the data structures for the actual terrain features within that quadnode's terrain patch. In the quadtree leaf file, only leaf nodes of the quadtree contain indices into the arrays data structures holding the actual terrain features.

These two quadtree files allow representation of objects at different levels of fidelity so that reasoning can be performed quickly over large areas of the terrain and then focused in on smaller areas for more exact reasoning. Also, large areas of the terrain can be displayed at a lower fidelity to speed up drawing time. Using the quadtree full file, abstract data structures can be stored at higher levels of the quadtree since a terrain feature that spans multiple nodes is stored only with the one node that is high enough in the quadtree to contain the entire feature. In the quadtree leaf file, when a feature is present in more than one quadnode, the index is duplicated in those quadnodes.

The quadtree representation provides the data organization to limit the search space. Given an area of interest, only the feature indices of the quadtree nodes that are partially or totally within that area need to be accessed.

Figure 1 shows the structure of the quadtree. Each parent node has four child nodes, one each to represent the division of the terrain into a NorthEast, SouthEast, SouthWest, and NorthWest quadrant. Each node of the quadtree has a field to hold an index into the feature array for that node. Each node also has a field which serves as a child node pointer; it contains a zero when this node is a leaf node or else it contains an index to the node's first (NorthEast quadrant) child node.

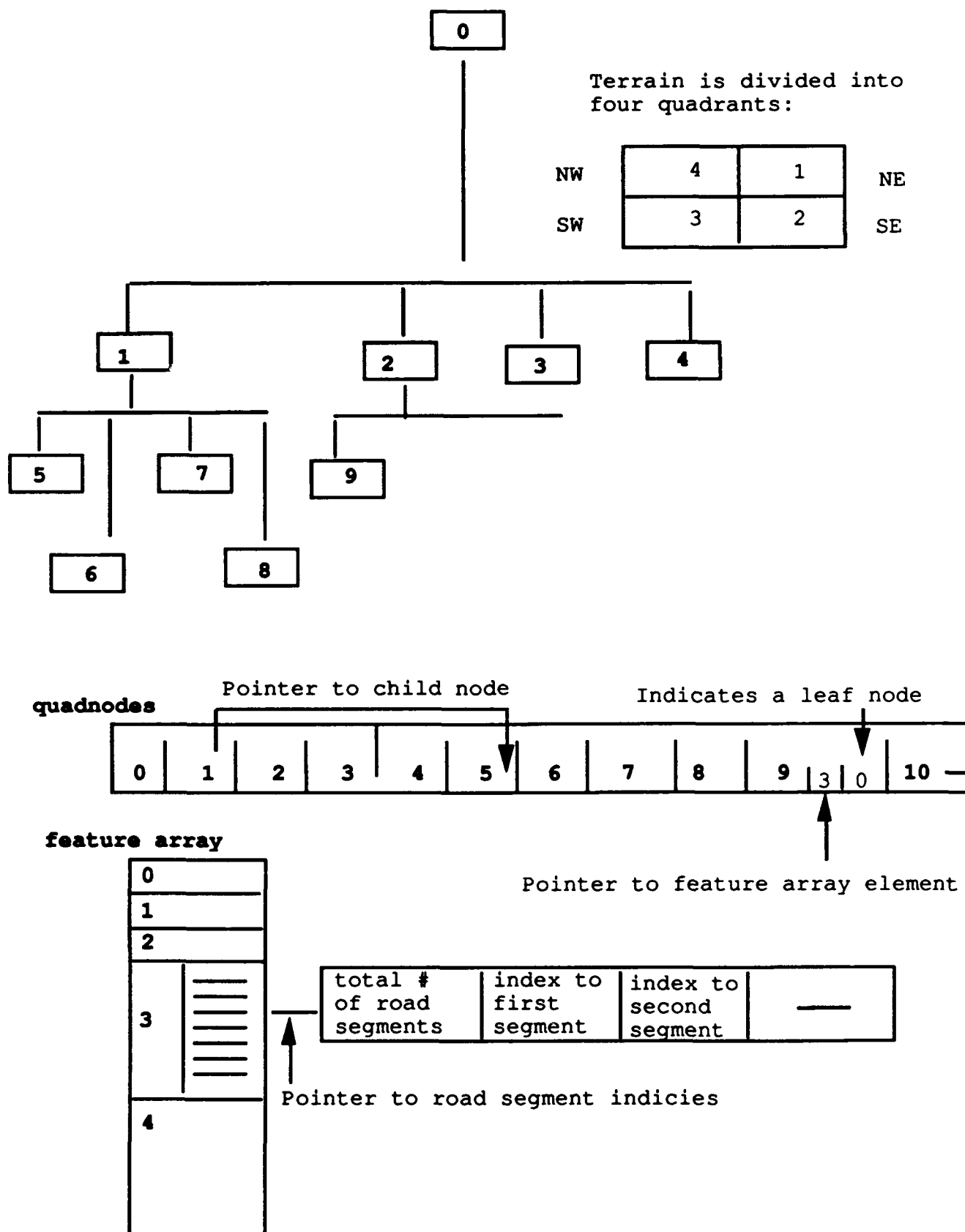


Figure 1: Quadtree Structure

The header of the quadtree file contains the following items:

<u>ITEM</u>	<u>TYPE</u>	<u>COMMENT</u>
size	INT32	the size in meters of the largest quadnode in the file
resolution	INT32	the size in meters of the smallest quadnode in the file
x	INT32	X coordinate of SW corner of this terrain database
y	INT32	Y coordinate of SW corner of this terrain database
num_nodes	INT32	number of quadtree nodes in this file
major_version	INT32	major version number of this database, for example, 0311
max_x	INT32	X coordinate of NE corner of this terrain database
max_y	INT32	Y coordinate of NE corner of this terrain database
fill_type	INT8	a value of 1 indicates that this is a leaf filled quadtree and a value of 0 indicates that this is a full filled quadtree
padding	INT8	unused
num_chars_name	INT32	number of characters making up name
num_chars_UTM_SW	INT32	number of characters making up UTM_SW
num_chars_UTM_NE	INT32	number of characters making up UTM_NE
name	null-terminated string, padded to a short boundary	the name of this database, for example, Knox
UTM_SW	null-terminated string, padded to a short boundary	the UTM coordinate of the SW corner of the database, for example, ES450550 for Knox database
UTM_NE	null-terminated string, padded to a short boundary	the UTM coordinate of the NE corner of the database, for example, FT200050 for Knox database

In the quadtree representation, each patch of terrain is represented by a quadnode containing the following items:

<u>ITEM</u>	<u>TYPE</u>	<u>COMMENT</u>
min_x	INT32	X coordinate of SW corner of this quadnode
min_y	INT32	Y coordinate of SW corner of this quadnode
max_x	INT32	X coordinate of NE corner of this quadnode
max_y	INT32	Y coordinate of NE corner of this quadnode
feature_index	INT32	index to the feature array which contains pointers to the objects contained in this patch of terrain
child_index	INT32	index to the child node or 0 for a leaf node

Each element in the feature array contains the following pointers:

<u>ITEM</u>	<u>TYPE</u>	<u>COMMENT</u>
water_indicies	array of INT32	the list of water segment indices of this quadtree node. The first element in the list tells the total number of indices in the list. Water segment data is stored in the Nets File.
water_int_indicies	array of INT32	the list of water intersection indices of this quadtree node. The first element in the list tells the total number of indices in the list. Water intersection data is stored in the Nets File.
road_indicies	array of INT32	the list of road segment indices of this quadtree node. The first element in the list tells the total number of indices in the list. Road segment data is stored in the Nets File.
road_int_indicies	array of INT32	the list of road intersection indices of this quadtree node. The first element in the list tells the total number of indices in the list. Road intersection data is stored in the Nets File.
bridge_indicies	array of INT32	the list of bridge indices of this quadtree node. The first element in the list tells the total number of indices in the list. Bridge data is stored in the Nets File.
rail_indicies	array of INT32	the list of bridge indices of this quadtree node. The first element in the list tells the total number of indices in the list. Bridge data is stored in the Nets File.
lake_indicies	array of INT32	the list of lake indices of this quadtree node. The first element in the list tells the total number of indices in the list. Lake data is stored in the Ground File.
canopy_indicies	array of INT32	the list of tree canopy indices of this quadtree node. The first element in the list tells the total number of indices in the list. Tree canopy data is stored in the Models File.
tree_indicies	array of INT32	the list of tree indices of this quadtree node. The first element in the list tells the total number of indices in the list. Tree data is stored in the Models File.

building_indicies	array of INT32	the list of building indices of this quadtree node. The first element in the list tells the total number of indices in the list. Building data is stored in the Models File.
contour_indicies	array of INT32	the list of contour line indices of this quadtree node. The first element in the list tells the total number of indices in the list. Contour line data is stored in the Contours File.

C.4 NETS FILE

The nets file is used for road and water networks. In the road network, the road segment objects are the segments that run between the road intersection objects. Each road segment object contains a list of points specifying the midline endpoints of each leg within that segment. The width of each road segment and the distance along each road segment are also stored in the segment objects. The road intersection data maps to road segment objects via a list of indices identifying the road segments connected to the intersection. The road network can be used to find the minimum distance paths between two points, and the midline points can be used to specify the route along those road segments. Likewise, the array of water segment objects identifies the sections of each river that run between the water intersections.

Figure 2 shows some segment and intersection objects. Segment 1, the stretch of road between intersection 3 and intersection 1, has 3 points to it. The coordinates of those points are labeled $(X1,Y1)$, $(X2,Y2)$, and $(X3,Y3)$.

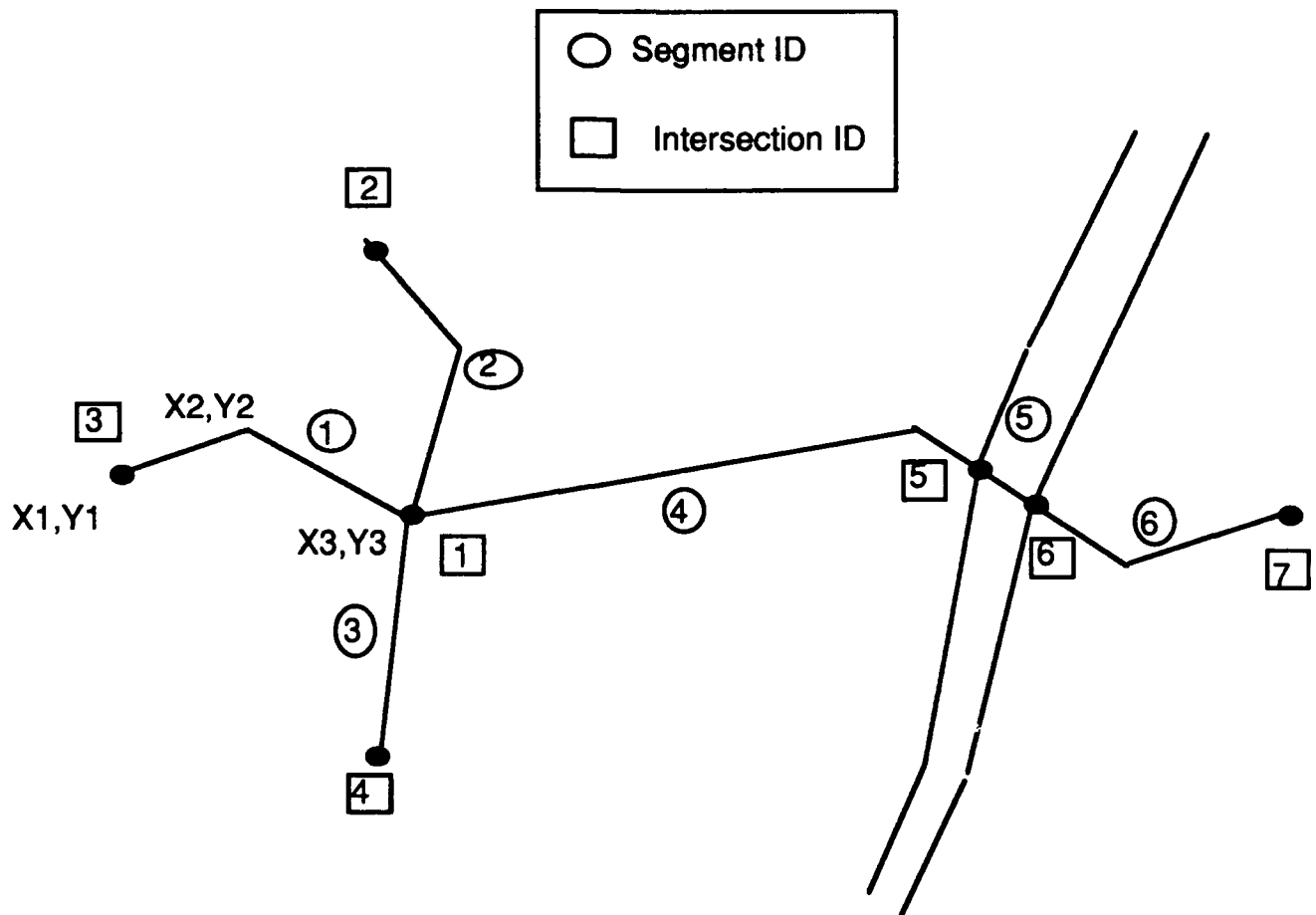


Figure 2: Network Objects

Road segments are broken at intersection points. There is always an intersection at the end of a road segment even when the road doesn't connect to another segment. Bridges are always their own segment, so there is always an intersection at each end of a bridge, as shown by segment 5 in Figure 2.

The nets file contains arrays of road segment, road intersection, water segment, water intersection, bridge and rail data. The format of each of these arrays is described below.

C.4.1 Segment Data

Each segment (road segment or river segment) element contains the following items:

<u>ITEM</u>	<u>TYPE</u>	<u>COMMENT</u>
num_pts	INT16	number of points that make up the segment
coordinates[num_pts]	INT32	the list of coordinates for the points of the segment
width	INT16	width of the segment
width_end1	INT16	used to provide the tapered end of a water segment
width_end2	INT16	used to provide the tapered end of a water segment
int1	INT16	index to an intersection point at one end of the segment
int2	INT16	index to an intersection point at opposite end of the segment
x_min	INT32	X coordinate of point that is farthest West
y_min	INT32	Y coordinate of point that is farthest South
x_max	INT32	X coordinate of point that is farthest East
y_max	INT32	Y coordinate of point that is farthest North
fordable	INT8	for a water segment, is segment fordable?, yes or no
bridge	INT8	for a road segment, is segment a bridge?, yes or no

C.4.2 Intersection Data

Figure 3 shows the intersection data of a road network. Each intersection (road intersection or river intersection) has at least one intersection-segment index pair associated with it. These identify the segments and intersections that connect to the intersection point. Intersection 1 connects to intersection 2 via segment 2, to intersection 3 via segment 1, to intersection 5 via segment 4, and to intersection 4 via segment 3.

Each intersection (road intersection or river intersection) element contains the following items:

<u>ITEM</u>	<u>TYPE</u>	<u>COMMENT</u>
num_pairs	INT16	number of intersection-segment index pairs
pairs[num_pairs]	INT16	the list of intersection-segment index pairs
x	INT32	X coordinate of intersection point
y	INT32	Y coordinate of intersection point
bridge	INT16	index to bridge object if this intersection is at one end of a bridge

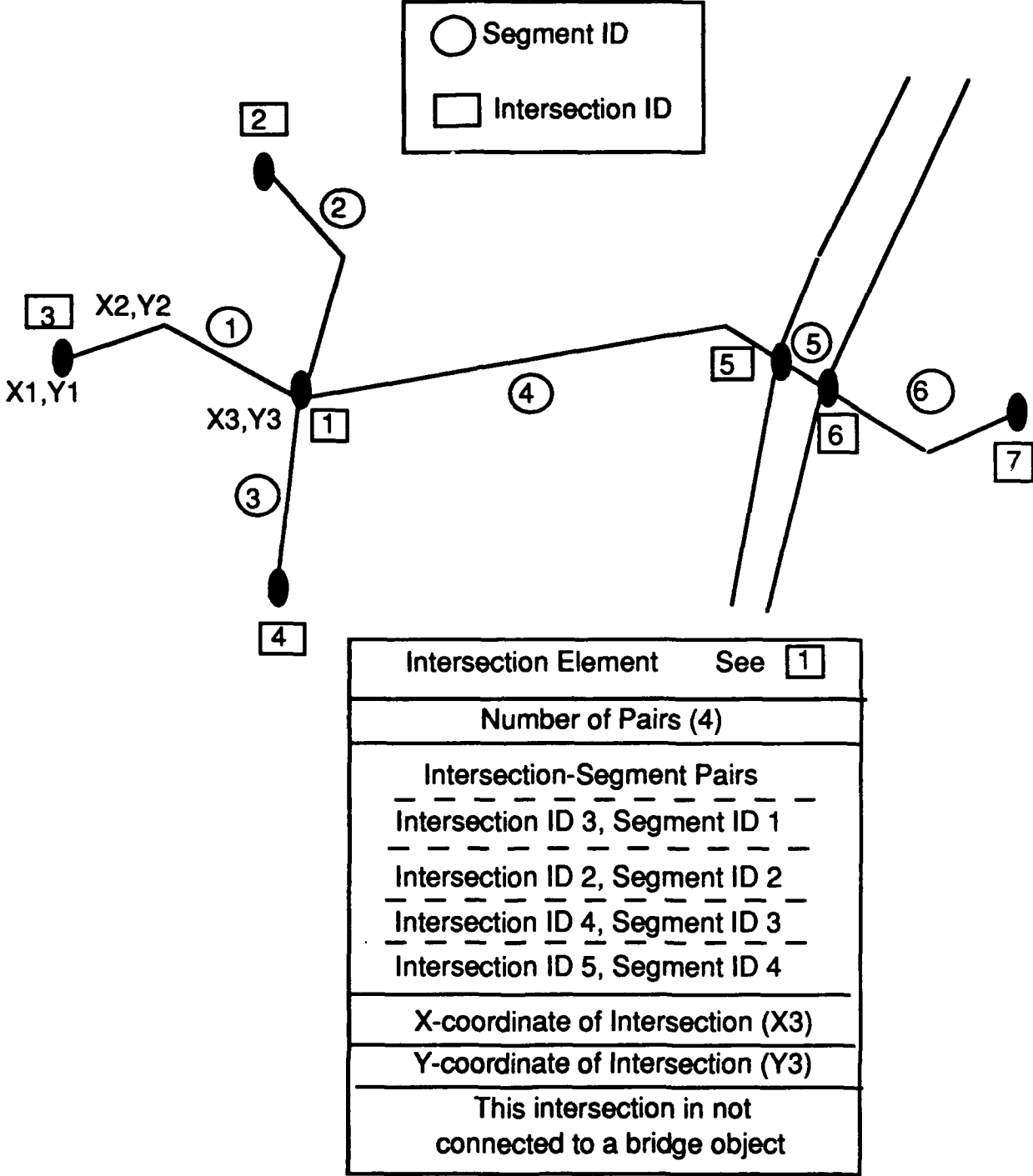


Figure 3: Intersection Data

C.4.3 Bridge Data

Each bridge is identified as a separate object to allow more precise road following routines to be run when a vehicle is crossing a bridge, as well as to allow bridges to be identified as a targets. Each element of the bridge array contains the following items:

<u>ITEM</u>	<u>TYPE</u>	<u>COMMENT</u>
num_points	INT16	number of points making up bridge
points[num_points]	INT32	the coordinates of the bridge points
x	INT32	X coordinate of the center of the bridge
y	INT32	Y coordinate of the center of the bridge
width	INT16	width of bridge in meters

C.4.4 Rail Data

Currently railroad tracks are used only for display. Each element of the rail array contains the following items:

<u>ITEM</u>	<u>TYPE</u>	<u>COMMENT</u>
num_points	INT16	number of points making up the railroad tracks
points[num_points]	INT32	the coordinates of the points making up the railroad tracks
width	INT16	width of tracks in meters

C.5 MODEL FILE

The model file is an array for trees, tree lines, tree canopies, and cultural objects (such as buildings). Tree lines are simply lists of points along the tree line or a single point to specify a single tree. Tree canopies are stored as polygons. Cultural objects are stored as bounding boxes, where each object has a location, width, and breadth.

Each element of the array contains the following items:

<u>ITEM</u>	<u>TYPE</u>	<u>COMMENT</u>
num_points	INT16	number of points making up the model
points[num_points]	INT32	the coordinates of the model points
x_min	INT32	X coordinate of point farthest West
y_min	INT32	Y coordinate of point farthest South
x_max	INT32	X coordinate of point farthest East
y_max	INT32	Y coordinate of point farthest North
model_type	INT16	identifies the model as a tree, treeline, tree canopy, or building
height	INT16	height in meters
impenetrable	INT8	can this object be driven through? yes or no

C.6 GROUND FILE

The ground file is an array for lakes, oceans, or areas of soil that are not standard hard-packed dirt. Each element of the array contains the following items:

<u>ITEM</u>	<u>TYPE</u>	<u>COMMENT</u>
num_points	INT16	number of points making up the area
points[num_points]	INT32	the coordinates of the area points
x_min	INT32	X coordinate of point farthest West
y_min	INT32	Y coordinate of point farthest South
x_max	INT32	X coordinate of point farthest East
y_max	INT32	Y coordinate of point farthest North
level	INT16	identifies level of nested area. The level is 0 for the outer boundary, 1 for an area nested inside a level 0 area, and 2 for an area nested inside a level 1, etc. Therefore, an island in a lake would be a level 1, and a pond on that island would be level 2.
poly_type	INT16	used to identify soil type of area (water, muck, hard soil etc.)

C.7 CONTOURS FILE

Contour lines are lists of points with an elevation value, which are simply drawn as lines. Contour lines can be obtained for a range of intervals. The interval of the contour lines displayed depends on the scale of the map being displayed, so that for larger scales the contour interval is greater than for smaller scales.

The contour file is an array for contour lines. Each element in the array contains the following items:

<u>ITEM</u>	<u>TYPE</u>	<u>COMMENT</u>
num_points	INT16	number of points making up the contour line
points[num_points]	INT32	the coordinates of the contour line points
x_min	INT32	X coordinate of point farthest West
y_min	INT32	Y coordinate of point farthest South
x_max	INT32	X coordinate of point farthest East
y_max	INT32	Y coordinate of point farthest North
type	INT16	identifies the object as a contour line
height	INT16	elevation in meters above sea level
impenetrable	INT16	not used

INDEX BY SECTION NUMBER

(COM-CATASTROPHIC MENU-ACCELERATOR Catastrophic MENU-LEVEL PAR-1)	2.1.5.17
(COM-FIREPOWER MENU-ACCELERATOR Firepower MENU-LEVEL PAR-1)	2.1.5.19
(COM-HULL MENU-ACCELERATOR Hull MENU-LEVEL PAR)	2.1.5.14
(COM-MOBILITY MENU-ACCELERATOR Mobility MENU-LEVEL PAR-1)	2.1.5.18
(COM-MOVING MENU-ACCELERATOR Moving MENU-LEVEL PAR)	2.1.5.16
(COM-READ-TABLE MENU-ACCELERATOR Read Table MENU-LEVEL COM)	2.1.5.2
(COM-READ-TABLE-DIR MENU-ACCELERATOR Read Table MENU-LEVEL COM-DIR)	2.1.5.7
(COM-READ-TABLE-IND MENU-ACCELERATOR Read Table MENU-LEVEL COM-IND)	2.1.5.11
(COM-SELECT-ASPECT-DIR MENU-ACCELERATOR Select Aspect MENU-LEVEL COM-DIR)	2.1.5.6
(COM-SELECT-MODEL MENU-ACCELERATOR Select Model MENU-LEVEL COM)	2.1.5.1
(COM-SELECT-MODEL-DIR MENU-ACCELERATOR Select Model MENU-LEVEL COM-DIR)	2.1.5.5
(COM-SELECT-MODEL-IND MENU-ACCELERATOR Select Model MENU-LEVEL COM-IND)	2.1.5.10
(COM-STATIONARY MENU-ACCELERATOR Stationary MENU-LEVEL PAR)	2.1.5.15
(COM-UNDO MENU-ACCELERATOR Undo MENU-LEVEL COM)	2.1.5.4
(COM-UNDO-DIR MENU-ACCELERATOR Undo MENU-LEVEL COM-DIR)	2.1.5.9
(COM-UNDO-IND MENU-ACCELERATOR Undo MENU-LEVEL COM-IND)	2.1.5.13
(COM-WRITE-TABLE MENU-ACCELERATOR Write Table MENU-LEVEL COM)	2.1.5.3
(COM-WRITE-TABLE-DIR MENU-ACCELERATOR Write Table MENU-LEVEL COM-DIR)	2.1.5.8
(COM-WRITE-TABLE-IND MENU-ACCELERATOR Write Table MENU-LEVEL COM-IND)	2.1.5.12
(METHOD ALTER-PROBABILITY POINT)	2.1.6.21
(METHOD ALTER-RANGE POINT)	2.1.6.18

(METHOD DRAG POINT)	2.1.3.15
(METHOD DRAG-UP-DOWN POINT)	2.1.6.12
(METHOD DRAW POINT)	2.1.2.22
(METHOD DRAW-WITH-CHECK POINT)	2.1.2.23
(METHOD ERASE POINT)	2.1.2.25
(METHOD ERASE-POINT-AND-LINES POINT)	2.1.3.16
(METHOD EXPUNGE POINT)	2.1.2.26
(METHOD MAKE-INSTANCE MODEL-MENU AFTER)	2.1.4.2
(METHOD ON-POINT? POINT)	2.1.2.24
(METHOD TOP-LEVEL MODEL-MENU)	2.1.4.5
(METHOD XREPLACE-OBJECT-VALUE DYNAMIC-WINDOW)	2.1.6.22
ABSOLUTE-ORIGIN	2.1.2.8
ABSOLUTE-X-ORIGIN	2.1.2.9
ABSOLUTE-Y-ORIGIN	2.1.2.10
ASPECT	2.1.1.15
CATASTROPHIC-GRAPH	2.1.1.5
CATASTROPHIC-POINTS	2.1.1.11
DEFAULT-FILE-NAME	2.1.7.50
DELTA-X	2.1.2.6
DELTA-Y	2.1.2.7
DETECTION-FILE-NAME	2.1.1.25
DEVELOPERS-FLG	2.1.7.29
DF-VEHICLE	2.1.1.28
DIR-FIR-TABLES	2.1.1.31
DIRECT-FIRE-DAMAGE-DATA	2.1.1.33
DIRECT-FIRE-DAMAGE-FILE-NAME	2.1.1.27
DISPLAY-LOGO-PANE	2.1.1.22
FIREPOWER-GRAPH	2.1.1.7
GRAPH-BEFORE-LAST-STEP	2.1.3.23
HIT-FILE-NAME	2.1.1.24
HIT-TABLES	2.1.1.32
HITMODELS-DICTIONARY	2.1.4.4
HULL-GRAPH	2.1.1.2
HULL-POINTS	2.1.1.8
IF-VEHICLE	2.1.1.29

IND-FIR-TABLES	2.1.1.30
INDIRECT-FIRE-DAMAGE-DATA	2.1.1.34
INDIRECT-FIRE-DAMAGE-FILE-NAME	2.1.1.26
MOBILITY-GRAPH	2.1.1.6
MOBILITY-POINTS	2.1.1.12
MODE	2.1.1.17
MODEL-FRAME	2.1.1.21
MODEL-IO	2.1.4.3
MOVING-GRAPH	2.1.1.4
MOVING-POINTS	2.1.1.10
OLD-SELECTION	2.1.1.1
PARAMETER-DISPLAY-PANE	2.1.1.19
PART	2.1.1.14
QUERY-WINDOW	2.1.7.51
SAVE-READTABLE	2.1.7.5
STATIONARY-GRAPH	2.1.1.3
STATIONARY-POINTS	2.1.1.9
TABLE-TYPE	2.1.1.16
TABULAR-DISPLAY-PANE	2.1.1.20
TABULAR-LOGO-PANE	2.1.1.23
TOTAL-X-RANGE	2.1.2.4
TOTAL-Y-RANGE	2.1.2.5
WEAPON-NAME	2.1.1.18
X-AXIS-LENGTH	2.1.2.1
X-ORIGIN-INCREMENT	2.1.2.3
Y-AXIS-LENGTH	2.1.2.2
ADD-CORRESPONDING-NEW-POINT-IN-POINT-LIST	2.1.3.12
ADD-NEW-POINT	2.1.3.14
Auxiliaries CSU	2.2.4
Background	1.1
BUILD-A-DIR-FIR-TABLE	2.1.7.26
BUILD-A-HIT-TABLE	2.1.7.25
BUILD-A-IND-FIR-TABLE	2.1.7.27
CALCULATE-POINT-DATA	2.1.2.19
CALCULATE-POINT-DATA-FROM-PIXEL-TO-REAL	2.1.2.20
CHANGE-MODE	2.1.5.22

CLEAN-UP-AXIS-AND-REDRAW	2.1.2.17
COM-MODELS	2.1.6.23
Command processor commands	2.1.5
Configuration and Configuration Management	1.4
COPY-LIST-ALL-LEVELS	2.1.7.9
COPY-POINT-LIST	2.1.3.25
CREATE-MAKE-INSTANCE-FORM-ARGS	2.1.3.21
CSC Descriptions	2
DELETE-POINT	2.1.6.6
DELETE-POINT-IF-THERE	2.1.3.1
DIR-FIR-TABLE	2.1.7.3
DISPATCH-TO-TABLE	2.1.5.25
DRAW-LINE-BETWEEN-POINTS	2.1.3.4
DRAW-SCALE-BAR	2.1.2.13
DRAW-TICKS	2.1.2.16
DRAW-X-SCALE-BAR	2.1.2.11
DRAW-X-TICKS	2.1.2.14
DRAW-Y-SCALE-BAR	2.1.2.12
DRAW-Y-TICKS	2.1.2.15
Drawing Routines CSU	2.3.6
DYNAMIC-WINDOW-WITHOUT-SCROLL-BARS	2.1.6.1
Editor Frame CSU	2.2.2
ELIMINATE-DUPPLICATES	2.1.7.7
External Communications	1.2
EXTRACT-DICE-NUMBERS	2.1.7.42
EXTRACT-FILE-NAME	2.1.3.28
EXTRACT-RANGE-AND-DICE-NUMBERS	2.1.7.40
FETCH-DF-DATA	2.1.7.23
FETCH-IF-DATA	2.1.7.20
File I/O Operations CSU	2.2.1
File input/output and database preparation	2.1.7
File Input/Output CSU	2.3.2
FIND-SURROUNDING-POINTS	2.1.3.7
FORMATIONS EDITOR CSC	2.3
Formations object CSU	2.3.3
Frame and screen configurations.	2.1.4

GATHER-DATABASE	2.1.7.38
GATHER-DIRECT-FIRE-DATABASE	2.1.7.41
GATHER-INDIRECT-FIRE-DATABASE	2.1.7.39
GET-CURRENT-GRAPH-POINTS	2.1.1.37
GET-CURRENT-LINE-POINTS	2.1.1.39
GET-HIT-TABLE	2.1.7.44
GET-NEW-WEAPON-NAME	2.1.7.54
GET-PLATFORM	2.1.8.4
GET-POINT	2.1.3.17
GET-POINT-LEFT	2.1.3.6
GET-POINT-RIGHT	2.1.3.5
GET-TABLE	2.1.7.43
GIMME-VAR-NAME-OF-CURRENT-GRAPH	2.1.1.36
GIMME-VAR-NAME-OF-CURRENT-POINTS	2.1.1.38
GO-BACK-TO-PREVIOUS-STEP	2.1.3.26
GRAPH-UNDO	2.1.3.29
Graphics manipulation functions	2.1.3
Graphing code	2.1.2
High level functions	2.1.8
HIGHLIGHT-SELECTION	2.1.8.2
HIT-TABLE	2.1.7.1
IND-FIR-TABLE	2.1.7.2
INSERT-POINT	2.1.6.7
INSERT-POINT-1	2.1.6.8
Internal Structure	1.3
Introduction : SAF Parameter Editor CSCI Description	1
LAST-CAR	2.1.3.2
LIST-INSTANCE-VARIABLE-VALUES	2.1.3.20
LIST-INSTANCE-VARIABLES	2.1.3.19
LOOKUP-DICTIONARY	2.1.7.11
MAKE-COPY-OF-INSTANCE-POINT	2.1.3.22
MAKE-DICTIONARY-PAIRS	2.1.7.8
MAKE-DOCUMENTATION	2.1.4.6
MAKE-DOTTED-SUBLISTS	2.1.7.47
MAKE-GAP-BETWEEN-POINTS	2.1.3.3
MAKE-GRAPH-GIVEN-POINTS	2.1.5.20

MAKE-HITMODELS-DICTIONARY	2.1.7.6
MAKE-SIMHOST-READTABLE	2.1.7.4
MAKE-UNDOTTED-SUBLISTS	2.1.7.48
MODEL EDITOR CSC	2.1
MODEL-MENU	2.1.4.1
MOVE-POINT	2.1.6.5
MOVE-UP-DOWN-POINT	2.1.6.10
MS-DYNAMIC-WINDOW-PANE	2.1.6.2
MS-PANE	2.1.6.13
Panes and Frame CSU	2.3.4
Panes and their presentations.	2.1.6
PARAMETER-DISPLAY-PANE	2.1.6.3
Parameters and Globals CSU	2.3.1
Parameters, Globals, and Macros	2.1.1
POINT	2.1.2.21
POINT	2.1.6.4
PRESENT-DATA	2.1.8.3
PRESENT-GRAPH	2.1.2.18
PRESENT-TABLE	2.1.8.5
Presentations and commands CSU	2.3.5
Presentations CSU	2.2.3
PROBABILITY	2.1.6.15
Program Frame CSU	2.2.5
QUERY-FOR-FILENAME	2.1.7.53
QUERY-USER	2.1.7.52
QUERY-WINDOW	2.1.7.49
RANGE	2.1.6.14
READ-AIR-DATA-FROM-FILE	2.1.7.18
READ-AIR-DETECTION-DATA	2.1.7.17
READ-DIRECT-FIRE-DAMAGE-DATA	2.1.7.22
READ-DIRECT-FIRE-DAMAGE-DATA-FROM-FILE	2.1.7.24
READ-GROUND-DATA-FROM-FILE	2.1.7.16
READ-GROUND-DETECTION-DATA	2.1.7.15
READ-HIT-DATA	2.1.7.12
READ-HIT-DATA-FILE	2.1.7.13
READ-HIT-DATA-FILE-AUX	2.1.7.14

READ-INDIRECT-FIRE-DAMAGE-DATA	2.1.7.19
READ-INDIRECT-FIRE-DAMAGE-DATA-FROM-FILE	2.1.7.21
RECORD-NEW-POINT	2.1.3.13
REMEMBER-CURRENT-HIT-TABLE	2.1.7.46
REMEMBER-TABLE	2.1.7.45
REPLACE-PROBABILITY	2.1.6.19
REPLACE-PROBABILITY	2.1.6.20
REPLACE-RANGE	2.1.6.16
REPLACE-RANGE	2.1.6.17
RESCALE-POINT-LIST	2.1.3.11
RESTORE-UPLOWCASE	2.1.7.10
REVERT-TO-FACTORY-VERSION	2.1.3.27
SELECT-AND-DRAG-POINT	2.1.3.18
SELECT-AND-DRAG-UP-DOWN-POINT	2.1.6.11
SELECT-HOST	2.1.5.23
SELECT-MODEL-MENU	2.1.6.24
SELECT-TABLE	2.1.5.24
SET-CURRENT-MODEL	2.1.8.1
SPLICE-IN-LIST-AFTER	2.1.3.8
SPLICE-IN-LIST-AT-POSITION	2.1.3.9
SWITCH-HIGHLIGHT	2.1.5.21
Terminology and Documentation	1.5
TRUNCATE-TO-N-POSITIONS	2.1.1.35
UP-DOWN-POINT	2.1.6.9
UPDATE-BACKTRACKING-CAPABILITY	2.1.3.24
UPDATE-POINT-LIST	2.1.3.10
USER-SELECT-ASPECT	2.1.5.27
USER-SELECT-MODEL	2.1.5.26
WEAPONS SYSTEMS EDITOR CSC	2.2
WRITE-CORRESPONDING-TABLE	2.1.7.28
WRITE-DETECTION-DATA	2.1.7.32
WRITE-DETECTION-DATA-FILE	2.1.7.33
WRITE-DIRECT-FIRE-DAMAGE-DATA	2.1.7.36
WRITE-DIRECT-FIRE-DAMAGE-DATA-FILE	2.1.7.37
WRITE-HIT-DATA	2.1.7.30
WRITE-HIT-DATA-FILE	2.1.7.31

WRITE-INDIRECT-FIRE-DAMAGE-DATA

2.1.7.34

WRITE-INDIRECT-FIRE-DAMAGE-DATA-FILE

2.1.7.35