AD-A244 929



1

SOFTWARE DESIGN DOCUMENT
MCC CSCI (1)

Volume 2 of 2 Sections 2.18.1 - 2.22

DTIC
ELECTE
JAN 09 1992
S D
D

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

92-00263



92 1 6 071

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	3. REPORT TYPE AND DATES COVERED
	June 1991	Software Design Document

4. TITLE AND SUBTITLE

Software Design Document MCC CSCI (1)

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 #:
9104

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 Management, Command, and Control (MCC) Computer Software Configuration Item (CSCI number 1) of the SIMNET hardware and software training system for vehicle crew training and operational training.

14. SUBJECT TERMS

SIMNET Software Design Document for the MCC CSCI (CSCI 1).

15. NUMBER OF PAGES

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	Same as report.

SOFTWARE DESIGN DOCUMENT
MCC CSCI (1)

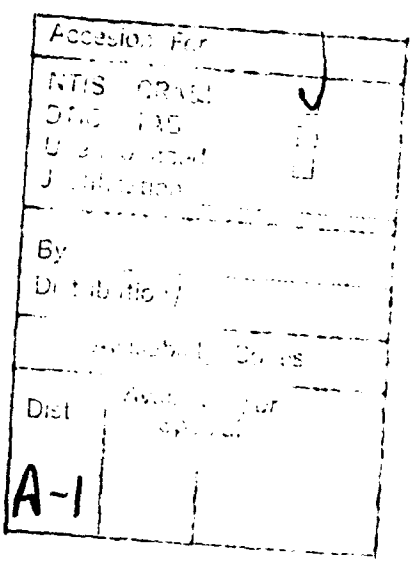
Volume 2 of 2 Sections 2.18.1 - 2.22

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

APPROVED FOR PUBLIC RELEASE
DISTRIBUTION UNLIMITED

2.18.1 CAS Console Definitions

2.18.1.1 CAS.h
Development:SIMNET:MCC:CAS:CAS.h

CAS.h defines the representation of information communicated between the CAS Macintosh application and the MCC host.

2.18.1.2 CASMac.h
Development:SIMNET:MCC:CAS:CASMac.h

CASMac.h defines types, constants and routines used within the CAS Macintosh application. Table 2.18-1 describes the variables used by CASMac.h.

Variables		
Variable	**Type**	**Where Typedef Declared**
scheduleDialog	extern pointer to DialogState	Development:SIMNET:libmac:dialog.h
scheduleField	extern ScrollTableFieldDefn	Development:SIMNET:libmac:scroll.h
onStationDialogDefn	extern DialogDefn	Development:SIMNET:libmac:dialog.h
bingoFuelDialogDefn	extern DialogDefn	Development:SIMNET:libmac:dialog.h
undoField	extern PushButtonFieldDefn	Development:SIMNET:libmac:dialog.h
cancelField	extern PushButtonFieldDefn	Development:SIMNET:libmac:dialog.h

Table 2.18-1 CASMac.h Variable Information.

2.18.1.3 version.h
Development:SIMNET:MCC:CAS:version.h

version.h defines constants that determine which version of the CAS application is compiled.

2.18.1.4 data.c
Development:SIMNET:MCC:CAS:data.c

data.c defines various data structures used by the CAS application such as tables of information about CAS missions. Table 2.18-2 describes the variables used by data.c.

Variables		
Variable	**Type**	**Where Typedef Declared**
application	pointer to char	Standard C type.
authors	pointer to char	Standard C type.
copyright	pointer to char	Standard C type.
undoField	extern PushButtonFieldDefn	Development:SIMNET:libmac:dialog.h
cancelField	extern PushButtonFieldDefn	Development:SIMNET:libmac:dialog.h
descriptionStrings	pointer to char	Standard C type.

Table 2.18-2 data.c Variable Information.

2.18.1.5 resource.h
Development:SIMNET:MCC:CAS:resource.h

resource.h defines the numbers of the resources present in the CAS Pictures resource file.

2.18.1.6 ditl.h
Development:SIMNET:MCC:CAS:ditl.h

ditl.h defines the numbers of items within CAS application dialogs.

2.18.2 CAS Console Software

2.18.2.1 condition.c
Development:SIMNET:MCC:CAS:condition.c

condition.c implements dialogs reporting the situations of aircraft arriving on station and aircraft reaching the bingo fuel condition. Table 2.18-3 describes the variables used by condition.c.

Variables		
Variable	**Type**	**Where Typedef Declared**
conditionDialog	pointer to DialogState	Development:SIMNET:libmac: dialog.h
currentCondition	MissionHandle	Development:SIMNET:MCC: CAS:CASMac.h
conditionTime	unsigned long	Standard C type.
conditionBuffe:	MissionDescriptor	Development:SIMNET:MCC: CAS:CASMac.h
conditionCallSignField	TextFieldDefn	Development:SIMNET:libmac: dialog.h
conditionLocationField	CoordinatesFieldDefn	Development:SIMNET:libmac: dialog.h
conditionHeadingField	NumberFieldDefn	Development:SIMNET:libmac: dialog.h
conditionFieldList	pointer to array of FieldDefn	Development:SIMNET:libmac: dialog.h
bingoFuelDialogDefn	DialogDefn	Development:SIMNET:libmac: dialog.h
onStationDialogDefn	DialogDefn	Development:SIMNET:libmac: dialog.h

Table 2.18-3 condition.c Variable Information.

2.18.2.1.1 SetUpCondition

SetUpCondition initializes the condition dialogs. The function call is SetUpCondition(). Table 2.18-4 describes the functions called using this function.

Calls	
Function	**Where Described**
NewPtr	Standard Memory Manager function for Macintosh.
Called By	
Function	**Where Described**
SetUp	See 2.18.2.8.1.

Table 2.18-4 SetUpCondition Information.

2.18.2.1.2 CheckForCondition

CheckForCondition checks for an aircraft arriving on station or reaching its bingo fuel condition. The function call is CheckForCondition(). Table 2.18-5 describes the internal variables used and functions called using this function.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
now	unsigned long	Standard C type.
time	unsigned long	Standard C type.
m	register MissionHandle	Development:SIMNET:MCC: CAS:CASMac.h

Calls	
Function	**Where Described**
GetDateTime	Standard Operating System Utility function for Macintosh.
UpdateMission	See Section 2.18.2.6.2.
CancelMission	See Section 2.18.2.6.4.
ThrowDialog	See Section 2.22.1.11.3.
ClearDialogsForMission	See Section 2.18.2.2.3.
RaiseCondition	See Section 2.18.2.1.3.
Called By	
Function	**Where Described**
MainEventLoop	See Section 2.18.2.5.2.

Table 2.18-5 CheckForCondition Information.

2.18.2.1.3 RaiseCondition

RaiseCondition interrupts the user with a notice of an aircraft on station or a bingo fuel condition. The function call is RaiseCondition(m, dialog). Table 2.18-6 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
m	MissionHandle	Development:SIMNET:MCC:CAS:CASMac.h
dialog	pointer to DialogDefn	Development:SIMNET:libmac:dialog.h
Calls		
Function	**Where Described**	
ClearDialogsForMission	See Section 2.18.2.2.3.	
ShowDialog	See Section 2.22.1.11.1.	
ShowWindow	Standard Window Manager function for Macintosh.	
SysBeep	Standard Operating System Utility function for Macintosh.	
GetDateTime	Standard Operating System Utility function for Macintosh.	
Called By		
Function	**Where Described**	
CheckForCondition	See Section 2.18.2.1.2.	
MainEventLoop	See Section 2.18.2.5.2.	

Table 2.18-6 RaiseCondition Information.

2.18.2.1.4 ConditionFetch

ConditionFetch reads the current values of the station and bingo fuel condition data structures into the Condition dialogs. The function call is ConditionFetch(dialog). Table 2.18-7 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
dialog	pointer to DialogState	Development:SIMNET:libmac: dialog.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
theItem	Handle	Development:THINK C: Mac #includes:MacTypes.h
i	int	Standard C type.
box	Rect	Development:THINK C: Mac #includes:MacTypes.h
Calls		
Function	**Where Described**	
GetDItem	Standard Dialog Manager function for Macintosh.	
HideControl	Standard Control Manager function for Macintosh.	
StuffHeldMission	See Section 2.18.2.2.9.	

Table 2.18-7 ConditionFetch Information.

2.18.2.1.5 ConditionEvent

ConditionEvent handles events in the Condition dialog. The function call is ConditionEvent(dialog, itemNo). Table 2.18-8 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
dialog	pointer to DialogState	Development:SIMNET:libmac: dialog.h
itemNo	int	Standard C type.
Calls		
Function	**Where Described**	
showhelp	See Section 2.22.1.19.4.	
UpdateDialog	See Section 2.22.1.11.2.	
CancelMission	See Section 2.18.2.6.4.	
ThrowDialog	See Section 2.22.1.11.3.	
CheckMandatoryFields	See Section 2.22.1.13.1.	
UpdateMission	See Section 2.18.2.6.2.	
ClearHot	See Section 2.18.2.6.5.	
HiliteMission	See Section 2.18.2.6.3.	

Table 2.18-8 ConditionEvent Information.

2.18.2.2 dialog.c

Development:SIMNET:MCC:CAS:dialog.c

dialog.c implements dialogs for creating and viewing the status of CAS missions. Table 2.18-9 describes the variables used by dialog.c.

Variables		
Variable	**Type**	**Where Typedef Declared**
preplannedDialogDefn	extern DialogDefn	Development:SIMNET:libmac: dialog.h
onCallDialogDefn	extern DialogDefn	Development:SIMNET:libmac: dialog.h
heldDialgoDefn	extern DialogDefn	Development:SIMNET:libmac: dialog.h
pastDialogDefn	extern DialogDefn	Development:SIMNET:libmac: dialog.h
missionDialog	pointer to DialogState	Development:SIMNET:libmac: dialog.h
currentMission	MissionHandle	Development:SIMNET:MCC: CAS:CASMac.h
missionBuffer	MissionDescriptor	Development:SIMNET:MCC: CAS:CASMac.h
futureTOTField	DTGFieldDefn	Development:SIMNET:libmac: dialog.h
futureCallSignField	TextFieldDefn	Development:SIMNET:libmac: dialog.h
futureLocationField	CoordinatesFieldDefn	Development:SIMNET:libmac: dialog.h
futureHeadingField	NumberFieldDefn	Development:SIMNET:libmac: dialog.h
futureTargetFields	RBFieldDefn	Development:SIMNET:libmac: dialog.h
futureSortiesField	NumberFieldDefn	Development:SIMNET:libmac: dialog.h
futureFieldList	pointer to array of FieldDefn	Development:SIMNET:libmac: dialog.h
preplannedDialogDefn	DialogDefn	Development:SIMNET:libmac: dialog.h
onCallDialogDefn	DialogDefn	Development:SIMNET:libmac: dialog.h
heldCallSignField	TextFieldDefn	Development:SIMNET:libmac: dialog.h
heldLocationField	CoordinatesFieldDefn	Development:SIMNET:libmac: dialog.h
heldHeadingField	NumberFieldDefn	Development:SIMNET:libmac: dialog.h
heldFieldList	pointer to array of FieldDefn	Development:SIMNET:libmac: dialog.h
heldDialogDefn	DialogDefn	Development:SIMNET:libmac: dialog.h
past ResultsField	TextFieldDefn	Development:SIMNET:libmac: dialog.h
pastFieldList	pointer to array of FieldDefn	Development:SIMNET:libmac: dialog.h
pastDialogDefn	DialogDefn	Development:SIMNET:libmac: dialog.h

Table 2.18-9 dialog.c Variable Information.

2.18.2.2.1 SetUpDialog

SetUpDialog initializes the mission dialog data structures. The function call is SetUpDialog(). Table 2.18-10 describes the functions called using this function.

Calls	
Function	**Where Described**
NewPtr	Standard Memory Manager function for Macintosh.
Called By	
Function	**Where Described**
SetUp	See Section 2.18.2.8.1.

Table 2.18-10 SetUpDialog Information.

2.18.2.2.2 ShowMission

ShowMission puts up a dialog box, *box*, for a mission, *m*, according to the status of the mission. The function call is ShowMission(m, box). Table 2.18-11 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
m	MissionHandle	Development:SIMNET:MCC:CAS:CASMac.h
box	pointer to Rect	Development:THINK C:Mac #includes:MacTypes.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
dialog	pointer to DialogDefn	Development:SIMNET:libmac:dialog.h
Calls		
Function	**Where Described**	
ShowDialog	See Section 2.22.1.11.1.	
ZoomToWindow	See Section 2.22.1.47.4.	
ShowWindow	Standard Window Manager function for Macintosh.	
Called By		
Function	**Where Described**	
ScheduleSelect	See Section 2.18.2.7.3.	
ScheduleEvent	See Section 2.18.2.7.12.	

Table 2.18-11 ShowMission Information.

2.18.2.2.3 ClearDialogsForMission

ClearDialogsForMission throws away the frontmost dialogs for help, and alerts any dialogs related to the indicated mission, *m*. The function call is ClearDialogsForMission(m). Table 2.18-12 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
m	MissionHandle	Development:SIMNET:MCC:CAS:CASMac.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
w	WindowPtr	Development:THINK C:Mac #includes:WindowMgr.h
newFront	WindowPtr	Development:THINK C:Mac #includes:WindowMgr.h
Calls		
Function	**Where Described**	
ThrowDialog	See Section 2.22.1.11.3.	
FrontWindow	Standard Window Manager function for Macintosh.	
DisposDialog	Standard Dialog Manager function for Macintosh.	
Called By		
Function	**Where Described**	
RaiseCondition	See Section 2.18.2.1.3.	
CheckForCondition	See Section 2.18.2.1.2.	

Table 2.18-12 ClearDialogsForMission Information.

2.18.2.2.4 StuffTimeOnTarget

StuffTimeOnTarget fills time on target data into the specified dialog item. The function call is StuffTimeOnTarget (dialog, itemNo, m). Table 2.18-13 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
dialog	pointer to DialogState	Development:SIMNET:libmac:dialog.h
itemNo	int	Standard C type.
m	MissionHandle	Development:SIMNET:MCC:CAS:CASMac.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
str	array of 20 char	Standard C type.
Calls		
Function	**Where Described**	
DTGToString	See Section 2.22.1.15.1.	
SetText	See Section 2.22.1.11.8.	

Called By	
Function	**Where Described**
OnCallFetch	See Section 2.18.2.2.6.
StuffHeldMission	See Section 2.18.2.2.9.
PastFetch	See Section 2.18.2.2.12.

Table 2.18-13 StuffTimeOnTarget Information.

2.18.2.2.5 ZoomMissionDown

ZoomMissionDown zooms between an entry in the Scheduled Missions Table and its pop-up Mission dialog. The function call is ZoomMissionDown (). Table 2.18-14 describes the internal variables used and functions called using this function.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
row	int	Standard C type.
box	Rect	Development:THINK C: Mac #includes:MacTypes.h
Calls		
Function	**Where Described**	
ScrollTableEntryToRow	See Section 2.22.1.34.1.	
ScrollTableRowRect	See Section 2.22.1.34.3.	
HideWindow	Standard Window Manager function for Macintosh.	
ZoomToWindow	See Section 2.22.1.47.4.	
Called By		
Function	**Where Described**	
FutureEvent	See Section 2.18.2.2.7.	

Table 2.18-14 ZoomMissionDown Information.

2.18.2.2.6 OnCallFetch

OnCallFetch reads current mission data into the On Call Mission dialog. The function call is OnCallFetch(dialog). Table 2.18-15 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
dialog	pointer to DialogState	Development:SIMNET:libmac: dialog.h
Calls		
Function	**Where Described**	
StuffTimeOnTarget	See Section 2.18.2.2.4.	

Table 2.18-15 OnCallFetch Information.

2.18.2.2.7 FutureEvent

FutureEvent handles events in the Preplanned Missions dialog. The function call is FutureEvent(dialog, itemNo). Table 2.18-16 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
dialog	pointer to DialogState	Development:SIMNET:libmac:dialog.h
itemNo	int	Standard C type.
Calls		
Function	**Where Described**	
showhelp	See Section 2.22.1.19.4.	
UpdateDialog	See Section 2.22.1.11.2.	
CancelMission	See Section 2.18.2.6.4.	
ThrowDialog	See Section 2.22.1.11.3.	
CheckMandatoryFields	See Section 2.22.1.13.1.	
CheckSortieLimits	See Section 2.18.2.3.4.	
UpdateMission	See Section 2.18.2.6.2.	
HiliteMission	See Section 2.18.2.6.3.	
ZoomMissionDown	See Section 2.18.2.2.5.	

Table 2.18-16 FutureEvent Information.

2.18.2.2.8 ValidTOT

ValidTOT determines if the time on target is valid. The time on target must be at least 25 minutes from now and on the same day, or it is not valid. The function call is ValidTOT(dialog, fp, str). Table 2.18-17 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
dialog	pointer to DialogState	Development:SIMNET:libmac:dialog.h
fp	pointer to TypeInFieldDefn	Development:SIMNET:libmac:dialog.h
str	pointer to char	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
now	unsigned long	Standard C type.
today	DateTimeRec	Development:THINK C Mac #includes:OSUtil.h
Return Values		
Return Value	**Type**	**Meaning**
0	int	Time not valid.
1	int	Time valid.

Calls	
Function	**Where Described**
GetDateTime	Standard Operating System Utility function for Macintosh.
ShowCaution	See Section 2.22.1.4.1.
GetTime	Standard Operating System Utility function for Macintosh.
SameDay	Macro defined in Development:SIMNET:MCC:CAS:CASMac.h.

Table 2.18-17 ValidTOT Information.

2.18.2.2.9 StuffHeldMission

StuffHeldMission fills in the fields of a dialog, *dialog*, for a mission, *m*, with a "held" status. The function call is StuffHeldMission(dialog, m). Table 2.18-18 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
dialog	pointer to DialogState	Development:SIMNET:libmac:dialog.h
m	MissionHandle	Development:SIMNET:MCC:CAS:CASMac.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
descriptionStrings	extern pointer to array of char	Standard C type.
str	array of 10 char	Standard C type.
Calls		
Function	**Where Described**	
SetText	See Section 2.22.1.11.8.	
StuffTimeOnTarget	See Section 2.18.2.2.4.	
Called By		
Function	**Where Described**	
HeldFetch	See Section 2.18.2.2.10.	
ConditionFetch	See Section 2.18.2.1.4.	

Table 2.18-18 StuffHeldMission Information.

2.18.2.2.10 HeldFetch

HeldFetch reads the current values of the "held" missions into the Held Missions dialog. The function call is HeldFetch(dialog). Table 2.18-19 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
dialog	pointer to DialogState	Development:SIMNET:libmac:dialog.h

Calls	
Function	**Where Described**
StuffHeldMission	See Section 2.18.2.2.9.

Table 2.18-19 HeldFetch Information.

2.18.2.2.11 HeldEvent

HeldEvent handles events in the Held Missions dialog. The function call is HeldEvent(dialog, itemNo). Table 2.18-20 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
dialog	pointer to DialogState	Development:SIMNET:libmac:dialog.h
itemNo	int	Standard C type.
Calls		
Function	**Where Described**	
showhelp	See Section 2.22.1.19.4.	
UpdateDialog	See Section 2.22.1.11.2.	
CancelMission	See Section 2.18.2.6.4.	
ThrowDialog	See Section 2.22.1.11.3.	
CheckMandatoryFields	See Section 2.22.1.13.1.	
UpdateMission	See Section 2.18.2.6.2.	
ClearHot	See Section 2.18.2.6.5.	

Table 2.18-20 HeldEvent Information.

2.18.2.2.12 PastFetch

PastFetch reads the values of past missions into the Past Missions dialog. The function call is PastFetch(dialog). Table 2.18-21 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
dialog	pointer to DialogState	Development:SIMNET:libmac:dialog.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
descriptionStrings	extern pointer to array of char	Standard C type.
str	array of 10 char	Standard C type
Calls		
Function	**Where Described**	
SetText	See Section 2.22.1.11.8.	
StuffTimeOnTarget	See Section 2.18.2.2.4.	

Table 2.18-21 PastFetch Information.

2.18.2.2.13 PastEvent

PastEvent handles events in the Past Missions dialog. The function call is PastEvent(dialog, itemNo). Table 2.18-22 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
dialog	pointer to DialogState	Development:SIMNET:libmac: dialog.h
itemNo	int	Standard C type.
Calls		
Function	**Where Described**	
showhelp	See Section 2.22.1.19.4.	
UpdateDialog	See Section 2.22.1.11.2.	
UpdateMission	See Section 2.18.2.6.2.	
ThrowDialog	See Section 2.22.1.11.3.	

Table 2.18-22 PastEvent Information.

2.18.2.3 limits.c

Development:SIMNET:MCC:CAS:limits.c

limits.c contains routines that implement and display the limits of preplanned and on-call CAS sorties. Table 2.18-23 describes the variables used by limits.c.

Variables		
Variable	**Type**	**Where Typedef Declared**
totalSorties	int	Standard C type.
preplannedSorties	int	Standard C type.
daySortiesAllotted	DateTimeRec	Development:THINK C: Mac #includes:OSUtil.h
monthNames	pointer to array of char	Standard C type.

Table 2.18-23 limits.c Variable Information.

2.18.2.3.1 SetSortieLimits

SetSortieLimits sets the total and per-day sortie limits. *totall* is the total number of sorties and *preplanned* is the number of sorties for today. The function call is SetSortieLimits(total, preplanned). Table 2.18-24 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
total	int	Standard C type.
preplanned	int	Standard C type.
Calls		
Function	**Where Described**	
GetTime	Standard Operating System Utility function for Macintosh.	

Called By	
Function	**Where Described**
LoadCannedLimits	See Section 2.18.2.4.2.
ProcessRequest	See Section 2.18.2.5.3.

Table 2.18-24 SetSortieLimits Information.

2.18.2.3.2 AdditionalSorties

AdditionalSorties is called when more sorties are added to the number allotted for today. *extra* is the number of additional sorties requested. These sorties may only be used for on-call missions. The function call is AdditionalSorties(extra). Table 2.18-25 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
extra	int	Standard C type.
Called By		
Function	**Where Described**	
MainEventLoop	See Section 2.18.2.5.2.	

Table 2.18-25 AdditionalSorties Information.

2.18.2.3.3 ShowSummary

ShowSummary puts up a dialog box summarizing the sortie limits and number of sorties used so far. The function call is ShowSummary(). Table 2.18-26 describes the internal variables used and functions called using this function.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
dialog	register DialogPtr	Development:THINK C: Mac #includes:DialogMgr.h
i	int	Standard C type.
totalSortiesUsed	int	Standard C type.
preplannedSortiesUsed	int	Standard C type.
totalSortiesRemaining	int	Standard C type.
theItem	Handle	Development:THINK C: Mac #includes:MacTypes.h
box	Rect	Development:THINK C: Mac #includes:MacTypes.h
str	array of 256 char	Standard C type.

Calls	
Function	**Where Described**
ShowSimpleDialog	See Section 2.22.1.40.1.
SetWTitle	Standard Window Manager function for Macintosh.
GetDItem	Standard Dialog Manager function for Macintosh.
SetIText	Standard Dialog Manager function for Macintosh.
SortiesScheduled	See Section 2.18.2.3.5.
NumToString	Standard Binary to Decimal Conversion Package function for Macintosh.
ShowWindow	Standard Window Manager function for Macintosh.
Called By	
Function	**Where Described**
ScheduleEvent	See Section 2.18.2.7.12.

Table 2.18-26 ShowSummary Information.

2.18.2.3.4 CheckSortieLimits

CheckSortieLimits checks to see that the sortie limits are not exceeded. The function call is CheckSortieLimits(currentMission, b). Table 2.18-27 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
currentMission	MissionHandle	Development:SIMNET:MCC:CAS:CASMac.h
b	register pointer to MissionDescriptor	Development:SIMNET:MCC:CAS:CASMac.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
m	register MissionHandle	Development:SIMNET:MCC:CAS:CASMac.h
totalSortiesUsed	int	Standard C type.
preplannedSortiesUsed	int	Standard C type.
str	array of char	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
0	int	Sortie limit will be exceeded.
1	int	Sortie limit will not be exceeded.
Calls		
Function	**Where Described**	
SameDay	Macro defined in Development:SIMNET:MCC:CAS:CASMac.h.	
ShowCaution	See Section 2.22.1.4.1.	
SortiesScheduled	See Section 2.18.2.3.5.	

Called By	
Function	**Where Described**
FutureEvent	See Section 2.18.2.2.7.

Table 2.18-27 CheckSortieLimits Information.

2.18.2.3.5 SortiesScheduled

SortiesScheduled computes the number of sorties scheduled for a given day, *date*. The function call is SortiesScheduled(date, totalSortiesUsed, preplannedsortiesUsed). Table 2.18-28 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
date	register pointer to DateTimeRec	Development:THINK C: Mac #includes:OSUtil.h
totalSortiesUsed	pointer to int	Standard C type.
preplannedSortiesUsed	pointer to int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
m	register MissionHandle	Development:SIMNET:MCC: CAS:CASMac.h
Calls		
Function	**Where Described**	
SameDay	Macro defined in Development:SIMNET:MCC:CAS:CASMac.h.	
Called By		
Function	**Where Described**	
CheckSortieLimits	See Section 2.18.2.3.4.	
ShowSummary	See Section 2.18.2.3.3.	

Table 2.18-28 SortiesScheduled Information.

2.18.2.4 load.c

Development:SIMNET:MCC:CAS:load.c

load.c loads canned data into application data structures, in the standalone demo version of the CAS application. The routines in this file are only compiled if VERSION is FULDAGAP. Table 2.18-29 describes the variables used by load.c.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
cannedMissions	array of CannedMission	Development:SIMNET:MCC: CAS:load.c

Table 2.18-29 load.c Variable Information.

2.18.2.4.1 LoadCannedTerrainMap

LoadCannedTerrainMap loads a canned terrain map. The function call is LoadCannedTerrainMap(). Table 2.18-30 describes the function calling this function.

Called By	
Function	**Where Described**
main	See Section 2.18.2.5.1.

Table 2.18-30 LoadCannedTerrainMap Information.

2.18.2.4.2 LoadCannedLimits

LoadCannedLimits loads canned sortie limits. The function call is LoadCannedLimits(). Table 2.18-31 describes the functions called using this function.

Calls	
Function	**Where Described**
SetSortieLimits	See Section 2.18.2.3.1.
Called By	
Function	**Where Described**
main	See Section 2.18.2.5.1.

Table 2.18-31 LoadCannedLimits Information.

2.18.2.4.3 LoadCannedSchedule

LoadCannedSchedule populates the mission schedule with canned data. The function call is LoadCannedSchedule(). Table 2.18-32 describes the internal variables used and functions called using this function.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
sortiesRemaining	extern array of int	Standard C type.
m	MissionHandle	Development:SIMNET:MCC: CAS:CASMac.h
cm	pointer to CannedMission	Development:SIMNET:MCC: CAS:load.c
now	unsigned long	Standard C type.
Calls		
Function	**Where Described**	
GetDateTime	Standard Operating system Utility function for Macintosh.	
NewMission	See Section 2.18.2.6.1.	
HLock	Standard Memory Manager function for Macintosh.	
StringToMapCoordinates	See Section 2.22.1.26.1.	
DTGElapsed	See Section 2.22.1.15.1.	
HUnlock	Standard Memory Manager function for Macintosh.	
UpdateMission	See Section 2.18.2.6.2.	

Called By	
Function	**Where Described**
main	See Section 2.18.2.5.1.

Table 2.18-32 LoadCannedSchedule Information.

2.18.2.5 main.c

Development:SIMNET:MCC:CAS:main.c

main.c contains the CAS application's program entry point and main event loop.

2.18.2.5.1 main

main is the program entry point. The function call is main(). Table 2.18-33 describes the functions called using this function.

Calls	
Function	**Where Described**
SetUp	See Section 2.18.2.8.1.
ProcessRequest	See Section 2.18.2.5.3.
SetUpAppleTalk	See Section 2.22.1.3.1.
DownloadTerrainMap	See Section 2.22.1.3.4.
MenuBarTitle	See Section 2.22.1.43.1.
InstallClock	See Section 2.22.1.6.4.
ShowSchedule	See Section 2.18.2.7.2.
LooadCannedTerrainMap	See Section 2.18.2.4.1.
LoadCannedLimits	See Section 2.18.2.4.2.
LoadCannedSchedule	See Section 2.18.2.4.3.
MainEventLoop	See Section 2.18.2.5.2.

Table 2.18-33 main Information.

2.18.2.5.2 MainEventLoop

MainEventLoop constantly fields incoming events. The function call is MainEventLoop(). Table 2.18-34 describes the parameters used and functions called using this function.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
myEvent	EventRecord	Development:THINK C: Mac #includes:EventMgr.h

Calls	
Function	**Where Described**
CheckForCondition	See Section 2.18.2.1.2.
UpdateClock	See Section 2.22.1.6.2.
SystemTask	Standard Desk Manager function for Macintosh.
GetNextEvent	Standard Toolbox Event Manager function for Macintosh.
AdditionalSorties	See Section 2.18.2.3.2.
RaiseCondition	See Section 2.18.2.1.3.
ATPCloseSocket	Standard Appletalk Manager function for Macintosh.
ExitToShell	Standard Segment Loader function for Macintosh.
ShowVersions	See Section 2.22.1.44.1.
WindowEvent	See Section 2.22.1.46.2.
NetworkEventHandler	See Section 2.22.1.3.5.
Called By	
Function	**Where Described**
main	See Section 2.18.2.5.1.

Table 2.18-34 MainEventLoop Information.

2.18.2.5.3 ProcessRequest

ProcessRequest processes a request received from the MCC host. This function is only compiled if VERSION is APPLETALK. The function call is ProcessRequest(hdl). Table 2.18-35 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
hdl	register Handle	Development:THINK C: Mac #includes:MacTypes.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
p	register Ptr	Development:THINK C: Mac #includes:MacTypes.h
errCode	int	Standard C type.
Calls		
Function	**Where Described**	
SetDateTime	Standard Operating System Utility function for Macintosh.	
SetSortieLimits	See Section 2.18.2.3.1.	
ATPResponse	Standard Appletalk Manager function for Macintosh.	
Restart	Standard Operating System Utility function for Macintosh.	
ATPError	See Section 2.22.1.3.6.	
Called By		
Function	**Where Described**	
main	See Section 2.18.2.5.1.	

Table 2.18-35 ProcessRequest Information.

2.18.2.6 mission.c
Development:SIMNET:MCC:CAS:mission.c

mission.c contains routines for creating and updating data structures describing CAS missions.

2.18.2.6.1 NewMission

NewMission allocates a record for a new mission. The function call is NewMission(type). Table 2.18-36 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
type	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
m	register MissionHandle	Development:SIMNET:MCC:CAS:CASMac.h
now	unsigned long	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
m	MissionHandle	Record for new mission.
Calls		
Function	**Where Described**	
NewHandle	Standard Memory Manager function for Macintosh.	
GetDateTime	Standard Operating System Utility function for Macintosh.	
DTGElapsed	See Section 2.22.1.15.3.	
Called By		
Function	**Where Described**	
LoadCannedSchedule	See Section 2.18.2.4.3.	
ScheduleEvent	See Section 2.18.2.7.12.	

Table 2.18-36 NewMission Information.

2.18.2.6.2 UpdateMission

UpdateMission updates a mission, *m*, in the schedule. The function call is UpdateMission(m). Table 2.18-37 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
m	MissionHandle	Development:SIMNET:MCC:CAS:CASMac.h
Calls		
Function	**Where Described**	
InstallScrollTableEntry	See Section 2.22.1.21.1.	

Called By	
Function	**Where Described**
CheckForCondition	See Section 2.18.2.1.2.
ConditionEvent	See Section 2.18.2.1.5.
FutureEvent	See Section 2.18.2.2.7.
HeldEvent	See Section 2.18.2.2.11.
PastEvent	See Section 2.18.2.2.13.
LoadCannedSchedule	See Section 2.18.2.4.3.
ClearHot	See Section 2.18.2.6.5.

Table 2.18-37 UpdateMission Information.

2.18.2.6.3 HiliteMission

HiliteMission hilites a mission, *m*, in the schedule table. The function call is HiliteMission(m). Table 2.18-38 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
m	MissionHandle	Development:SIMNET:MCC:CAS:CASMac.h
Calls		
Function	**Where Described**	
SetScrollTableSelection	See Section 2.22.1.37.2.	
Called By		
Function	**Where Described**	
ConditionEvent	See Section 2.18.2.1.5.	
FutureEvent	See Section 2.18.2.2.7.	

Table 2.18-38 HiliteMission Information.

2.18.2.6.4 CancelMission

CancelMission removes a mission, *m*, from the schedule and discards it. The function call is CancelMission(m). Table 2.18-39 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
m	MissionHandle	Development:SIMNET:MCC:CAS:CASMac.h
Calls		
Function	**Where Described**	
RemoveScrollTableEntry	See Section 2.22.1.28.1.	

Called By	
Function	**Where Described**
CheckForCondition	See Section 2.18.2.1.2.
ConditionEvent	See Section 2.18.2.1.5.
FutureEvent	See Section 2.18.2.2.7.
HeldEvent	See Section 2.18.2.2.11.

Table 2.18-39 CancelMission Information.

2.18.2.6.5 ClearHot

ClearHot is called when a mission, *m*, is cleared hot. The function call is ClearHot(m). Table 2.18-40 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
m	MissionHandle	Development:SIMNET:MCC:CAS:CASMac.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
now	unsigned long	Standard C type.
Calls		
Function	**Where Described**	
GetDateTime	Standard Operating System Utility function for Macintosh.	
DTGElapsed	See Section 2.22.1.15.3.	
UpdateMission	See Section 2.18.2.6.2.	
NewPtr	Standard Memory Manager function for Macintosh	
NewHandle	Standard Memory Manager function for Macintosh.	
SetUpATPRequest	See Section 2.22.1.3.2.	
ATPError	See Section 2.22.1.3.6.	
ATPReqeust	Standard Appletalk Manager function for Macintosh.	
Called By		
Function	**Where Described**	
ConditionEvent	See Section 2.18.2.1.5.	
HeldEvent	See Section 2.18.2.2.11.	

Table 2.18-40 ClearHot Information.

2.18.2.7 schedule.c

Development:SIMNET:MCC:CAS:schedule.c

schedule.c implements a dialog displaying a schedule of CAS missions. Table 2.18-41 describes the variables used by schedule.c.

Variables		
Variable	**Type**	**Where Typedef Declared**
scheduleColumns	extern array of ScrollTableColumnDefn	Development:SIMNET:libmac: scroll.h
scheduleDialogDefn	extern DialogDefn	Development:SIMNET:libmac: dialog.h
scheduleDialog	pointer to DialogState	Development:SIMNET:libmac: dialog.h
scheduleColumns	array of ScrollTableColumnDefn	Development:SIMNET:libmac: scroll.h
scheduleField	ScrollTableFieldDefn	Development:SIMNET:libmac: scroll.h
scheduleFieldList	pointer to FieldDefn	Development:SIMNET:libmac: dialog.h
scheduleDialogDefn	DialogDefn	Development:SIMNET:libmac: dialog.h

Table 2.18-41 schedule.c Variable Information.

2.18.2.7.1 SetUpSchedule

SetUpSchedule initializes the mission schedule. The function call is SetUpSchedule(). Table 2.18-42 describes the internal variables used and functions called using this function.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
port	GrafPort	Development:THINK C: Mac #includes:Quickdraw.h
Calls		
Function	**Where Described**	
NewPtr	Standard Memory Manager function for Macintosh.	
OpenPort	Standard Quickdraw function for Macintosh.	
TextFont	Standard Quickdraw function for Macintosh.	
TextSize	Standard Quickdraw function for Macintosh.	
StringWidth	Standard Quickdraw function for Macintosh.	
CharWidth	Standard Quickdraw function for Macintosh.	
ClosePort	Standard Quickdraw function for Macintosh.	
Called By		
Function	**Where Described**	
SetUp	See Section 2.18.2.8.1.	

Table 2.18-42 SetUpSchedule Information.

2.18.2.7.2 ShowSchedule

ShowSchedule brings up the MissionSchedule dialog. The function call is ShowSchedule(). Table 2.18-43 describes the functions called using this function.

Calls	
Function	**Where Described**
ShowDialog	See Section 2.22.1.11.1.
ShowWindow	Standard Window Manager function for Macintosh.
Called By	
Function	**Where Described**
main	See Section 2.18.2.5.1.

Table 2.18-43 ShowSchedule Information.

2.18.2.7.3 ScheduleSelect

ScheduleSelect checks the validity of the seleted item and brings up the next dialog describing the selection. The function call is ScheduleSelect(defn, row, box, event). Table 2.18-44 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
defn	pointer to ScrollTableFieldDefn	Development:SIMNET:libmac: scroll.h
row	int	Standard C type.
box	pointer to Rect	Development:THINK C: Mac #includes:MacTypes.h
event	pointer to EventRecord	Development:THINK C: Mac #includes:EventMgr.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
entry	ScrollTableEntryHandle	Development:SIMNET:libmac: scroll.h
Calls		
Function	**Where Described**	
ScrollTableRowToEntry	See Section 2.22.1.34.1.	
SetScrollTableSelection	See Section 2.22.1.37.2.	
ShowMission	See Section 2.18.2.2.2.	

Table 2.18-44 ScheduleSelect Information.

2.18.2.7.4 ScheduleHilite

ScheduleHilite highlights an entry in the Schedule Table. The function call is ScheduleHilite(defn, entry, box). Table 2.18-45 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
defn	pointer to ScrollTableFieldDefn	Development:SIMNET:libmac: scroll.h
entry	ScrollTableEntryHandle	Development:SIMNET:libmac: scroll.h
box	pointer to Rect	Development:THINK C: Mac #includes:MacTypes.h
Calls		
Function	**Where Described**	
InvertRect	Standard Quickdraw function for Macintosh.	

Table 2.18-45 ScheduleHilite Information.

2.18.2.7.5 ScheduleDrawType

ScheduleDrawType fills in the Type field of the specified column in the Schedule Table. The function call is ScheduleDrawType(defn, col, entry). Table 2.18-46 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
defn	poindter to ScrollTableFieldDefn	Development:SIMNET:libmac: scroll.h
col	pointer to ScrollTableColumnDefn	Development:SIMNET:libmac: scroll.h
entry	register ScrollTableEntryHandle	Development:SIMNET:libmac: scroll.h
Calls		
Function	**Where Described**	
Move	Standard Quickdraw function for Macintosh.	
CharWidth	Standard Quickdraw function for Macintosh.	
DrawString	Standard Quickdraw function for Macintosh.	

Table 2.18-46 ScheduleDrawType Information.

2.18.2.7.6 ScheduleDrawTOT

ScheduleDrawTOT fills in the TOT field of the specified column in the Schedule Table. The function call is ScheduleDrawTOT(defn, col, entry). Table 2.18-47 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
defn	poindter to ScrollTableFieldDefn	Development:SIMNET:libmac: scroll.h
col	pointer to ScrollTableColumnDefn	Development:SIMNET:libmac: scroll.h
entry	register ScrollTableEntryHandle	Development:SIMNET:libmac: scroll.h

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
str	array of 20 char	Standard C type.
Calls		
Function	**Where Described**	
DTGToString	See Section 2.22.1.15.1.	
DrawText	Standard Quickdraw function for Macintosh.	

Table 2.18-47 ScheduleDrawTOT Information.

2.18.2.7.7 ScheduleDrawLocation

ScheduleDrawLocation fills in the Location field of the specified column in the Schedule Table. The function call is ScheduleDrawLocation(defn, col, entry). Table 2.18-48 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
defn	poindter to ScrollTableFieldDefn	Development:SIMNET:libmac: scroll.h
col	pointer to ScrollTableColumnDefn	Development:SIMNET:libmac: scroll.h
entry	register ScrollTableEntryHandle	Development:SIMNET:libmac: scroll.h
Calls		
Function	**Where Described**	
DrawText	Standard Quickdraw function for Macintosh.	

Table 2.18-48 ScheduleDrawLocation Information.

2.18.2.7.8 ScheduleDrawDescription

ScheduleDrawDescription fills in the Description field of the specified column in the Schedule Table. The function call is ScheduleDrawDescription(defn, col, entry). Table 2.18-49 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
defn	pointer to ScrollTableFieldDefn	Development:SIMNET:libmac: scroll.h
col	pointer to ScrollTableColumnDefn	Development:SIMNET:libmac: scroll.h
entry	register ScrollTableEntryHandle	Development:SIMNET:libmac: scroll.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
descriptionStrings	extern pointer to array of char	Standard C type.
cp	register pointer to char	Standard C type.

Calls	
Function	**Where Described**
DrawText	Standard Quickdraw function for Macintosh.

Table 2.18-49 ScheduleDrawDescription Information.

2.18.2.7.9 ScheduleDrawSorties

ScheduleDrawSorties fills in the Sorties field of the specified column in the Schedule Table. The function call is ScheduleDrawSorties(defn, col, entry). Table 2.18-50 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
defn	poindter to ScrollTableFieldDefn	Development:SIMNET:libmac: scroll.h
col	pointer to ScrollTableColumnDefn	Development:SIMNET:libmac: scroll.h
entry	register ScrollTableEntryHandle	Development:SIMNET:libmac: scroll.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
str	array of 10 char	Standard C type.
Calls		
Function	**Where Described**	
NumToString	Standard Binary to Decimal Conversion Package function for Macintosh.	
Move	Standard Quickdraw function for Macintosh.	
CharWidth	Standard Quickdraw function for Macintosh.	
StringWidth	Standard Quickdraw function for Macintosh.	
DrawString	Standard Quickdraw function for Macintosh.	

Table 2.18-50 ScheduleDrawSorties Information.

2.18.2.7.10 ScheduleDrawResults

ScheduleDrawResults fills in the Results field of the specified column in the Schedule Table. The function call is ScheduleDrawResults(defn, col, entry). Table 2.18-51 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
defn	poindter to ScrollTableFieldDefn	Development:SIMNET:libmac: scroll.h
col	pointer to ScrollTableColumnDefn	Development:SIMNET:libmac: scroll.h
entry	register ScrollTableEntryHandle	Development:SIMNET:libmac: scroll.h

Calls	
Function	**Where Described**
DrawText	Standard Quickdraw function for Macintosh.

Table 2.18-51 ScheduleDrawResults Information.

2.18.2.7.11 ScheduleDrawStatus

ScheduleDrawStatus fills in the Status field of the specified column in the Schedule Table. The function call is ScheduleDrawStatus(defn, col, entry). Table 2.18-52 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
defn	poindter to ScrollTableFieldDefn	Development:SIMNET:libmac: scroll.h
col	pointer to ScrollTableColumnDefn	Development:SIMNET:libmac: scroll.h
entry	register ScrollTableEntryHandle	Development:SIMNET:libmac: scroll.h
Calls		
Function	**Where Described**	
DrawString	Standard Quickdraw function for Macintosh.	
TextFace	Standard Quickdraw function for Macintosh.	

Table 2.18-52 ScheduleDrawStatus Information.

2.18.2.7.12 ScheduleEvent

ScheduleEvent handles events for the Schedule dialog. The function call is ScheduleEvent(dialog, itemNo). Table 2.18-53 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
dialog	pointer to DialogState	Development:SIMNET:libmac: dialog.h
itemNo	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
m	MissionHandle	Development:SIMNET:MCC: CAS:CASMac.h
theItem	Handle	Development:THINK C: Mac #includes:MacTypes.h
i	int	Standard C type.
box	Rect	Development:THINK C: Mac #includes:MacTypes.h

Calls	
Function	**Where Described**
showhelp	See Section 2.22.1.19.4.
ShowSummary	See Section 2.18.2.3.3.
NewMission	See Section 2.18.2.6.1.
GetDItem	Standard Dialog Manager function for Macintosh.
ShowMission	See Section 2.18.2.2.2.

Table 2.18-53 ScheduleEvent Information.

2.18.2.8 setup.c

Development:SIMNET:MCC:CAS:setup.c

setup.c initializes the CAS application at start-up.

2.18.2.8.1 SetUp

SetUp initializes the SIMNET Close Air Support application. The toolbox is intialized, the SIMNET Resources and CAS Pictures files are opened, the Help Manager is intialized, and the dialogs are initialized. The function call is SetUp(). Table 2.18-54 describes the internal variables used and functions called using this function.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
errCode	int	Standard C type.
finalTicks	long	Standard C type.
Calls		
Function	**Where Described**	
InitToolbox	See Section 2.22.1.20.1.	
ZoomInit	See Section 2.22.1.47.1.	
OpenResFile	Standard Resource Manager function for Macintosh.	
MenuBarTitle	See Section 2.22.1.43.1.	
Delay	Standard Operating System Utility function for Macintosh.	
ExitToShell	Standard Segment Loader function for Macintosh.	
DeepShit	See Section 2.22.1.8.2.	
helpinit	See Section 2.22.1.19.1.	
SystemFailure	See Section 2.22.1.8.1.	
SetUpCondition	See Section 2.18.2.1.1.	
SetUpDialog	See Section 2.18.2.2.1.	
SetUpSchedule	See Section 2.18.2.7.1.	
Called By		
Function	**Where Described**	
main	See Section 2.18.2.5.1.	

Table 2.18-54 SetUp Information.

2.18.3 Appearance of CAS Console User Interface

2.18.3.1 CAS Pictures
Development:SIMNET:MCC:CAS:CAS Pictures

CAS Pictures contains resources that determine the appearance of the CAS application's user interface.

2.19 The CEW (Combat Engineering Workstation) Console

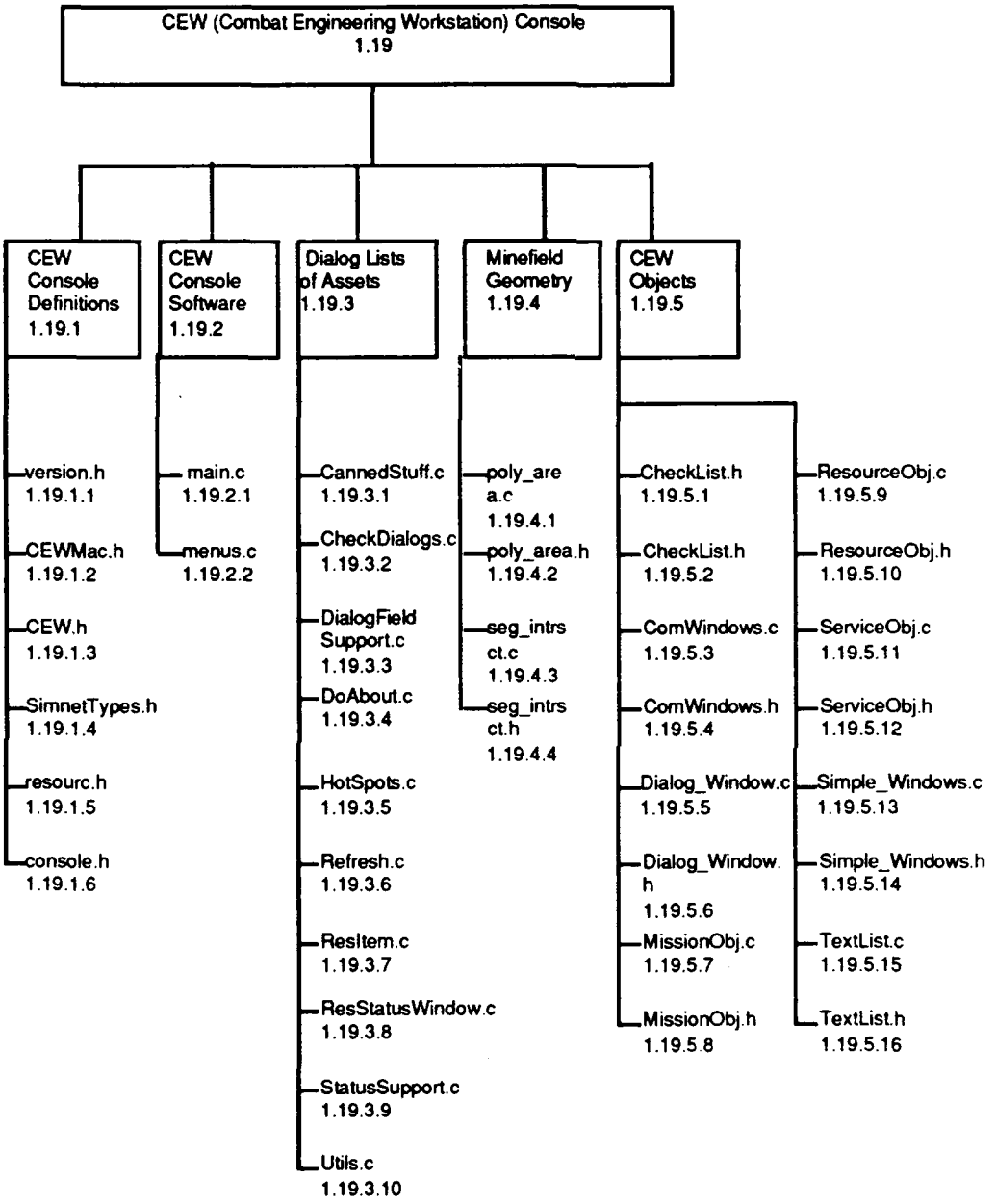


Figure 2.19-1: CEW (Combat Engineering Workstation) Console Structure.

2.19.1 CEW Console Definitions

2.19.1.1 version.h
Development:SIMNET:MCC:CEW:version.h

version.h defines the version of the CEW Macintosh application to be compiled.

2.19.1.2 CEWMac.h
Development:SIMNET:MCC:CEW:CEWMac.h

CEWMac.h is the header file used by all of the CEW source code files in the CEW Macintosh application. Table 2.19-1 describes the variables used by CEWMac.h.

Variables		
Variable	**Type**	**Where Typedef Declared**
totalResources	extern short	Standard C type.
terrainMap	extern TerrainMap	Development:SIMNET:libmac:map.h
legalGEMSSDensities	extern array of float	Standard C type.
emplacementPointArray	extern array of short	Standard C type.
breach_availList	extern pointer to CheckList	Development:SIMNET:CEW:CheckList.h
move_availList	extern pointer to CheckList	Development:SIMNET:CEW:CheckList.h
emplace_availList	extern pointer to CheckList	Development:SIMNET:CEW:CheckList.h
stat_emplace_availList	extern pointer to CheckList	Development:SIMNET:CEW:CheckList.h
stat_breach_availList	extern pointer to CheckList	Development:SIMNET:CEW:CheckList.h
stat_move_availList	extern pointer to CheckList	Development:SIMNET:CEW:CheckList.h
cewResources	extern pointer to array of ResourceObj	Development:SIMNET:CEW:ResourceObj.h
review_dlog	extern pointer to DialogWindow	Development:SIMNET:CEW:Dialog_Windows.h
emplace_dlog	extern pointer to DialogWindow	Development:SIMNET:CEW:Dialog_Windows.h
main_window	extern pointer to ComWindow	Development:SIMNET:CEW:ComWindows.h
cewMission	extern pointer to array of MissionObj	Development:SIMNET:CEW:MissionObj.h

Table 2.19-1 CEWMac.h Variable Information.

2.19.1.3 CEW.h
Development:SIMNET:MCC:CEW:CEW.h

CEW.h defines the MCC SIMNET Combat Engineering Workstation (CEW) parameters and interfaces.

2.19.1.4 SimnetTypes.h
Development:SIMNET:MCC:CEW:SimnetTypes.h

SimnetTypes.h contains a type definition of the UTMCoordinates array. Table 2.19-2 describes the variables used by SimnetTypes.h.

Variables		
Variable	**Type**	**Where Typedef Declared**
UTMCoordinates	unsigned array of 9 char	Standard C type.

Table 2.19-2 SimnetTypes.h Variable Information.

2.19.1.5 resource.h
Development:SIMNET:MCC:CEW:resource.h

resource.h contains the Mac resource IDs for the CEW.

2.19.1.6 console.h
Development:SIMNET:MCC:CEW:console.h

console.h defines things relevant to communication between the Masscomp host and the Macintosh console. Table 2.19-3 describes the variables used by console.h.

Variables		
Variable	**Type**	**Where Typedef Declared**
hostSocket	extern long	Standard C type.

Table 2.19-3 console.h Variable Information.

2.19.2 CEW Console Software

2.19.2.1 main.c
Development:SIMNET:MCC:CEW:main.c

main.c contains the CEW application's program entry point and main event loop. Table 2.19-4 describes the variables used by main.c.

Variables		
Variable	**Type**	**Where Typedef Declared**
main_window	pointer to ComWindow	Development:SIMNET:CEW: ComWindows.h
emplace_dlog	pointer to DialogWindow	Development:SIMNET:CEW: Dialog_Windows.h
select_dlog	pointer to DialogWindow	Development:SIMNET:CEW: Dialog_Windows.h
breach_dlog	pointer to DialogWindow	Development:SIMNET:CEW: Dialog_Windows.h
move_dlog	pointer to DialogWindow	Development:SIMNET:CEW: Dialog_Windows.h
stat_emplace	pointer to DialogWindow	Development:SIMNET:CEW: Dialog_Windows.h
stat_breach	pointer to DialogWindow	Development:SIMNET:CEW: Dialog_Windows.h
stat_move	pointer to DialogWindow	Development:SIMNET:CEW: Dialog_Windows.h
review_dlog	pointer to DialogWindow	Development:SIMNET:CEW: Dialog_Windows.h
missionList	pointer to TextList	Development:SIMNET:CEW: TextList.h
cewMission	pointer to array of NUM_OF_MISSION MissionObj	Development:SIMNET:CEW: MissionObj.h
cewResources	pointer to array of NUM_OF_RESOURCES ResourceObj	Development:SIMNET:CEW: ResourceObj.h
serviceq	pointer to ServiceObj	Development:SIMNET:CEW: ServiceObj.h
nextMission	short	Standard C type.
totalResources	short	Standard C type.
authors	pointer to char	Standard C type.
version	pointer to char	Standard C type.
copyright	pointer to char	Standard C type.
application	pointer to char	Standard C type.
atpSocket	extern short	Standard C type.

Table 2.19-4 main.c Variable Information.

2.19.2.1.1 main

main is the program entry point. The function call is main(). Table 2.19-5 describes the internal variables used and functions called using this function.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
r	Rect	Development:THINK C: Mac #includes:MacTypes.h
r1	Rect	Development:THINK C: Mac #includes:MacTypes.h
i	short	Standard C type.
tempstr	array of 255 char	Standard C type
testmp	MapCoordinates	Development:SIMNET:libmac map.h

Calls	
Function	**Where Described**
OpenResFile	Standard Resource Manager function for Macintosh.
Notify	See Section 2.19.3.4.4.
ExitToShell	Standard Segment Loader function for Macintosh.
MacInits	See Section 2.19.2.1.3.
ProcessRequest	See Section 2.19.2.1.5.
SetUpAppleTalk	See Section 2.22.1.3.1.
DownloadTerrainMap	See Section 2.22.1.3.4.
LoadCannedTerrainMap	See Section 2.19.3.1.1.
LoadCannedResources	See Section 2.19.3.1.2.
ComWindow::initial	See Section 2.19.5.3.3.
ComWindow::show	See Section 2.19.5.3.1.
SetRect	Standard Quickdraw function for Macintosh.
TextList::init	See Section 2.19.5.15.1.
ShowWait	See Section 2.22.1.45.1.
InstallClock	See Section 2.22.1.6.4.
MainEventLoop	See Section 2.19.2.1.2.

Table 2.19-5 main Information.

2.19.2.1.2 MainEventLoop

MainEventLoop processes incoming events. The function call is MainEventLoop(). Table 2.19-6 describes the internal variables used and functions called using this function.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
myEvent	EventRecord	Development:THINK C: Mac #includes:EventMgr.h
eventpt	Point	Development:THINK C: Mac #includes:MacTypes.h
c	char	Standard C type.
astr	Str255	Development:THINK C: Mac #includes:MacTypes.h
i	short	Standard C type.
r	Rect	Development:THINK C: Mac #includes:MacTypes.h
r1	Rect	Development:THINK C: Mac #includes:MacTypes.h
wwindow	WindowPtr	Development:THINK C: Mac #includes:WindowMgr.h
windowcode	short	Standard C type.

Calls	
Function	**Where Described**
SystemTask	Standard Desk Manager function for Macintosh.
UpdateClock	See Section 2.22.1.6.2.
ServiceObj::UpdateQs	See Section 2.19.5.11.6.
GetNextEvent	Standard Toolbox Event Manager function for Macintosh.
DoAbout	See Section 2.19.3.4.2.
MacQuit	See Section 2.19.2.1.4.
DoMenu	See Section 2.19.2.2.2.
MenuKey	Standard Menu Manager function for Macintosh.
HiliteMenu	Standard Menu Manager function for Macintosh.
DoResourceStatus	See Section 2.19.3.8.1.
FindWindow	Standard Window Manager function for Macintosh.
MenuSelect	Standard Menu Manager function for Macintosh.
SystemClick	Standard Desk Manager function for Macintosh.
HiliteWindow	Standard Window Manager function for Macintosh.
SelectWindow	Standard Window Manager function for Macintosh.
SetPort	Standard Quickdraw function for Macintosh.
GlobalToLocal	Standard Quickdraw function for Macintosh.
Swindow::equal	See Section 2.19.5.13.8.
ComWindow::CheckHotSpot	See Section 2.19.5.3.4.
TextList::List1Click	See Section 2.19.5.15.12.
Swindow::GetWindowPtr	See Section 2.19.5.13.10.
TextList::GetCellRect	See Section 2.19.5.15.4.
ZoomRect	See Section 2.22.1.47.2.
ReviewSelection	See Section 2.19.3.2.15.
NetworkEventHandler	See Section 2.22.1.3.5.
Called By	
Function	**Where Described**
main	See Section 2.19.2.1.1.

Table 2.19-6 MainEventLoop Information.

2.19.2.1.3 MacInits

MacInits intializes the SIMNET Combat Engineer Workstation application. The master pointers for handles in the Memory Manager are made, the system heap is grown, the event queue is flushed, and the quickdraw is intialized. The random number generator is seeded, the pull-down menus are set up, the service queue is intialized, and the dialogs are set up. The function call is MacInits(). Table 2.19-7 describes the internal variables used and functions called using this function.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	short	Standard C type.
r	Rect	Development:THINK C: Mac #includes:MacTypes.h

Calls	
Function	**Where Described**
MoreMasters	Standard Memory Manager function for Macintosh.
MaxApplZone	Standard Memory Manager function for Macintosh.
FlushEvents	Standard Operating System Memory Manager function for Macintosh.
InitGraf	Standard Quickdraw function for Macintosh.
InitFonts	Standard Font Manager function for Macintosh.
InitWindows	Standard Window Manager function for Macintosh.
InitMenus	Standard Menu Manager function for Macintosh.
TEInit	Standard TextEdit function for Macintosh.
InitDialogs	Standard Dialog Manager function for Macintosh.
InitCursor	Standard Quickdraw function for Macintosh.
ZoomInit	See Section 2.22.1.47.1.
helpinit	See Section 2.22.1.19.1.
Notify	See Section 2.19.3.4.4.
GetDateTime	Standard Operating System Utility function for Macintosh.
SetUpMenus	See Section 2.19.2.2.1.
ServiceObj::init	See Section 2.19.5.11.1.
DialogWindow::init	See Section 2.19.5.5.1.
DialogWindow::SetUserItem	See Section 2.19.5.5.13.
Called By	
Function	**Where Described**
main	See Section 2.19.2.1.1.

Table 2.19-7 MacInits Information.

2.19.2.1.4 MacQuit

MacQuit quits the SIMNET Combat Engineer Workstation application. The function call is MacQuit(). Table 2.19-8 describes the internal variables used and functions called using this function.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	short	Standard C type.
Calls		
Function	**Where Described**	
TextList::ListDispose	See Section 2.19.5.15.3.	
Swindow::quit	See Section 2.19.5.13.3.	
ATPCloseSocket	Standard AppleTalk Manager function for Macintosh.	
ExitToShell	Standard Segment Loader function for Macintosh.	
Called By		
Function	**Where Described**	
MainEventLoop	See Section 2.19.2.1.2.	
DoMenu	See Section 2.19.2.2.2.	

Table 2.19-8 MacQuit Information.

2.19.2.1.5 ProcessRequest

ProcessRequest processes a request received from the MCC host. *hdl* is the handle of the request from the host. The function call is ProcessRequest(hdl). Table 2.19-9 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
hdl	register Handle	Development:THINK C: Mac #includes:MacTypes.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
p	register Ptr	Development:THINK C: Mac #includes:MacTypes.h
errCode	short	Standard C type.
buf	array of 100 char	Standard C type.
tempname	array of 255 char	Standard C type.
kind	long	Standard C type.
speed	short	Standard C type.
ccv	long	Standard C type.
i	short	Standard C type.
foos	Str255	Development:THINK C: Mac #includes:MacTypes.h
warnStr	Str255	Development:THINK C: Mac #includes:MacTypes.h
nameStr	Str255	Development:THINK C: Mac #includes:MacTypes.h
miss	short	Standard C type.
Calls		
Function	**Where Described**	
ATPError	See Section 2.22.1.3.6.	
ATPResponse	Standard AppleTalk Manager function for Macintosh.	
SetDateTime	Standard Operating System Utility function for Macintosh.	
ThrowWait	See Section 2.22.1.45.2.	
ResourceObj::init	See Section 2.19.5.9.1.	
ResourceObj::SetPosition	See Section 2.19.5.9.4.	
ResourceObj::GetCCVNumber	See Section 2.19.5.9.3.	
Notify	See Section 2.19.3.4.4.	
ResourceObj::SetStatus	See Section 2.19.5.9.11.	
ResourceObj::GetName	See Section 2.19.5.9.10.	
ResourceObj::GetMission	See Section 2.19.5.9.5.	
MissionObj::GetStatus	See Section 2.19.5.7.21.	
MissionObj::ResetMissionTime	See Section 2.19.5.7.17.	
Restart	Standard Operating System Utility function for Macintosh.	

Called By	
Function	**Where Described**
main	See Section 2.19.2.1.1.

Table 2.19-9 ProcessRequest Information.

2.19.2.1.6 GetTerrainBounds

GetTerrainBounds sends a message to the MCC host requesting information about the boundary limits of the CEW terrain. The function call is GetTerrainBounds(). Table 2.19-10 describes the internal variables used and functions called using this function.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
ab	register ABRecHandle	Development:THINK C: Mac #includes:Appletalk.h
n	char	Standard C type.
errCode	short	Standard C type.
theBounds	CEWBoundsResponse	Development:SIMNET:CEW: CEW.h
tempStr	Str255	Development:THINK C: Mac #includes:MacTypes.h
Calls		
Function	**Where Described**	
NewHandle	Standard Memory Manager function for Macintosh.	
SetUpATPRequest	See Section 2.22.1.3.2.	
ATPError	See Section 2.22.1.3.6.	
ATPResponse	Standard Appletalk Manager function for Macintosh.	
DisposHandle	Standard Memory Manager function for Macintosh.	
Notify	See Section 2.19.3.4.4.	

Table 2.19-10 GetTerrainBounds Information.

2.19.2.2 menus.c

Development:SIMNET:MCC:CEW:menus.c

menus.c contains routines that implement the menus for the CEW Macintosh console. Table 2.19-11 describes the variables used by menus.c.

Variables		
Variable	**Type**	**Where Typedef Declared**
comMenu	array of 5 MenuHandle	Development:THINK C: Mac #includes:MenuMgr.h
main_window	extern pointer to ComWindow	Development:SIMNET:CEW: ComWindows.h

Table 2.19-11 menus.c Variable Information.

2.19.2.2.1 SetUpMenus

SetUpMenus sets up the menu bar for the CEW console. The function call is SetUpMenus(). Table 2.19-12 describes the functions called using this function.

Calls	
Function	**Where Described**
NewMenu	Standard Menu Manager function for Macintosh.
AppendMenu	Standard Menu Manager function for Macintosh.
AddResMenu	Standard Menu Manager function for Macintosh.
InsertMenu	Standard Menu Manager function for Macintosh.
GetMenu	Standard Menu Manager function for Macintosh.
DrawMenuBar	Standard Menu Manager function for Macintosh.
Called By	
Function	**Where Described**
MacInits	See Section 2.19.2.1.3.

Table 2.19-12 SetUpMenus Information.

2.19.2.2.2 DoMenu

DoMenu handles menu selections. *menuresult* contains the specifiec menu ID, and the menu item. The function call is DoMenu(). Table 2.19-13 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
menuresult	long	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
menuid	short	Standard C type.
item	short	Standard C type.
daname	array of 255 char	Standard C type.
cport	GrafPtr	Development:THINK C: Mac #includes:Quickdraw.h
refnum	short	Standard C type.
Calls		
Function	**Where Described**	
HiWord	Standard Toolbox Utility function for Macintosh.	
LoWord	Standard Toolbox Utility function for Macintosh.	
GetItem	Standard Menu Manager function for Macintosh.	
GetPort	Standard Quickdraw function for Macintosh.	
OpenDeskAcc	Standard Desk Manager function for Macintosh.	
SetPort	Standard Quickdraw function for Macintosh.	
HiliteMenu	Standard Menu Manager function for Macintosh.	
DoAbout	See Section 2.19.3.4.2.	
MacQuit	See Section 2.19.2.1.4.	
SysBeep	Standard Operating System Utility function for Macintosh.	
NotImpDialog	See Section 2.19.3.4.3.	

Called By	
Function	**Where Described**
MainEventLoop	See Section 2.19.2.1.2.

Table 2.19-13 DoMenu Information.

2.19.3 Dialog Lists of Assets

2.19.3.1 CannedStuff.c

Development:SIMNET:MCC:CEW:CannedStuff.c

2.19.3.1.1 LoadCannedTerrainMap

LoadCannedTerrainMap loads in a canned terrain map for the demo version of the CEW. The function call is LoadCannedTerrainMap(). Table 2.19-14 describes the functions which call this function.

Called By	
Function	**Where Described**
main	See Section 2.19.2.1.1.

Table 2.19-14 LoadCannedTerrainMap Information.

2.19.3.1.2 LoadCannedResources

LoadCannedResources loads in a canned set of CEW assets for the demo version. The function call is LoadCannedResources(). Table 2.19-15 describes the internal variables used and functions called using this function.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	short	Standard C type.
point	LongPt	Development:SIMNET:libmac:longpt.h
Calls		
Function	**Where Described**	
ResourceObj::init	See Section 2.19.5.9.1.	
ResourceObj::SetPosition	See Section 2.19.5.9.4.	
ResourceObj::SetStatus	See Section 2.19.5.9.11.	
Called By		
Function	**Where Described**	
main	See Section 2.19.2.1.1.	

Table 2.19-15 LoadCannedResources Information.

2.19.3.2 CheckDialogs.c
Development:SIMNET:MCC:CEW:CheckDialogs.c

CheckDialogs.c contains routines for implementing dialogs. Table 2.19-16 describes the variables used by CheckDialogs.c.

Variables		
Variable	**Type**	**Where Typedef Declared**
serviceq	extern pointer to ServiceObj	Development:SIMNET:CEW: ServiceObj.h
missionList	extern pointer to TextList	Development:SIMNET:CEW: TextList.h
emplacementPointArray	array of short	Standard C type.
legalGEMSSDensities	array of float	Standard C type.
time_modified	short	Standard C type.

Table 2.19-16 CheckDialogs.c Variable Information.

2.19.3.2.1 isLegalGEMSSDensity

isLegalGEMSSDensity determines if a density is a legal value for GEMSS vehicles. The function call is isLegalGEMSSDensity(dens). Table 2.19-17 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
dens	float	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	short	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
TRUE	long	Legal density value.
FALSE	long	Illegal density value.
Called By		
Function	**Where Described**	
CheckEmplacementDialog	See Section 2.19.3.2.5.	

Table 2.19-17 isLegalGEMSSDensity Information.

2.19.3.2.2 isLegalMinefield

isLegalMinefield determines if a minefield is of a legal shape. The function call is isLegalMinefield(). Table 2.19-18 describes the internal variables used and functions called using this function.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	short	Standard C type.
pts	array of LongPt	Development:SIMNET:libmac: longpt.h
string	Str255	Development:THINK C: Mac #includes:MacTypes.h
Return Values		
Return Value	**Type**	**Meaning**
TRUE	short	Legal minefield.
FALSE	short	Illegal minefield.
Calls		
Function	**Where Described**	
DialogWindow::GetText	See Section 2.19.5.5.7.	
UTMToLongPt	See Section 2.19.3.3.5.	
seg_intrsct	See Section 2.19.4.3.1.	
Notify	See Section 2.19.3.4.4.	
Called By		
Function	**Where Described**	
CheckEmplacementDialog	See Section 2.19.3.2.5.	
DetermineMines	See Section 2.19.3.2.14.	

Table 2.19-18 isLegalMinefield Information.

2.19.3.2.3 ReviewDialogHandler

ReviewDialogHandler is a template for handling events in the Review dialogs. The routine is not called in the application. Table 2.19-19 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
dlog	pointer to DialogWindow	Development:SIMNET:CEW: Dialog_Windows.h
event	EventRecord	Development:THINK C: Mac #includes:EventMgr.h

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
itemhit	short	Standard C type.
done	short	Standard C type.
lastitem	short	Standard C type.
c	char	Standard C type.
type	short	Standard C type.
ltype	short	Standard C type.
h_item	Handle	Development:THINK C: Mac #includes:MacTypes.h
lh_item	Handle	Development:THINK C: Mac #includes:MacTypes.h
box	Rect	Development:THINK C: Mac #includes:MacTypes.h
lbox	Rect	Development:THINK C: Mac #includes:MacTypes.h
reset	Boolean	Development:THINK C: Mac #includes:MacTypes.h
Return Values		
Return Value	**Type**	**Meaning**
START_MOVEMENT_NOW	short	Start the movement.
END_MISSION_NOW	short	End the mission.
START_ACTIVITY_NOW	short	Start the activity.
0	short	Do nothing; event was not a "hot key".
Calls		
Function	**Where Described**	
SysBeep	Standard Operating System Utility function for Macintosh.	

Table 2.19-19 ReviewDialogHandler Information.

2.19.3.2.4 setEmplacementTime

setEmplacementTime calculates the amount of time it will take to emplace the minefield, given the assets information entered in the dialog specified by the *dialog* and *item* parameters. The function call is setEmplacementTime(dialogp, item). Table 2.19-20 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
dialogp	pointer to DialogWindow	Development:SIMNET:CEW: Dialog_Windows.h
item	short	Standard C type.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
item_time	DateTimeGroup	Development:SIMNET:libmac:dtg.h
gemss_number	short	Standard C type.
cewPlatoon_number	short	Standard C type.
m57_number	short	Standard C type.
move_time	float	Standard C type.
activity_time	float	Standard C type.
per_hr	float	Standard C type.
at	long	Standard C type.
apf	long	Standard C type.
apb	long	Standard C type.
i	short	Standard C type.
x	short	Standard C type.
retval	long	Standard C type.
string	Str255	Development:THINK C: Mac #includes:MacTypes.h
string2	Str255	Development:THINK C: Mac #includes:MacTypes.h
sm_time	long	Standard C type.
sa_time	long	Standard C type.
f_time	long	Standard C type.

Return Values		
Return Value	**Type**	**Meaning**
retval	long	The amount time it will take to emplace the minefield.

Calls	
Function	**Where Described**
TextList::GetCount	See Section 2.19.5.15.5.
ResourceObj::GetKind	See Section 2.19.5.9.2.
DialogWindow::GetText	See Section 2.19.5.5.7.
CheckList::GetState	See Section 2.19.5.1.6.
StringToDTG	See Section 2.22.1.15.2.
DTGElapsed	See Section 2.22.1.15.3.
DTGToString	See Section 2.22.1.15.1.
DialogWindow::SetText	See Section 2.19.5.5.8.

Called By	
Function	**Where Described**
CheckEmplacementDialog	See Section 2.19.3.2.5.

Table 2.19-20 setEmplacementTime Information.

2.19.3.2.5 CheckEmplacementDialog

CheckEmplacementDialog checks the validity of data entered in the Emplacement Dialog fields. The shape of the minefield is checked, and the selected assets are checked. In addition, the routine performs calculations of the mission time and mine requirements. The function call is CheckEmplacementDialog(dialogp, item, to_reset, theEvent). Table 2.19-21 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
dialogp	pointer to DialogWindow	Development:SIMNET:CEW: Dialog_Windows.h
item	short	Standard C type.
to_reset	pointer to Boolean	Development:THINK C: Mac #includes:MacTypes.h
theEvent	pointer to EventRecord	Development:THINK C: Mac #includes:EventMgr.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
dtg	DateTimeGroup	Development:SIMNET:libmac: dtg.h
current_time	DateTimeGroup	Development:SIMNET:libmac: dtg.h
wp	DialogPtr	Development:THINK C: Mac #includes:DialogMgr.h
h_item	Handle	Development:THINK C: Mac #includes:MacTypes.h
box	Rect	Development:THINK C: Mac #includes:MacTypes.h
string	Str255	Development:THINK C: Mac #includes:MacTypes.h
string2	Str255	Development:THINK C: Mac #includes:MacTypes.h
type	short	Standard C type.
just_gemss	short	Standard C type.
x	short	Standard C type.
i	short	Standard C type.
ts	Str255	Development:THINK C: Mac #includes:MacTypes.h
Return Values		
Return Value	**Type**	**Meaning**
TRUE	short	Data is accepted; user is allowed to continue.
FALSE	short	Data is not accepted; the dialog cannot be brought down.

Calls	
Function	**Where Described**
DialogWindow::GetDialogPtr	See Section 2.19.5.5.16.
GetDItem	Standard Dialog Manager function for Macintosh.
DialogWindow::GetText	See Section 2.19.5.5.7.
Notify	See Section 2.19.3.4.4.
DialogWindow::GetValue	See Section 2.19.5.5.10.
isLegalMinefield	See Section 2.19.3.2.2.
TextList::GetCount	See Section 2.19.5.15.5.
CheckList::GetState	See Section 2.19.5.1.6.
ResourceObj::GetKind	See Section 2.19.5.9.2.
isLegalGEMSSDensity	See Section 2.19.3.2.1.
DetermineMines	See Section 2.19.3.2.14.
Get_DTG	See Section 2.22.1.15.4.
DTGToString	See Section 2.22.1.15.1.
DialogWindow::SetText	See Section 2.19.5.5.8.
setEmplacementTime	See Section 2.19.3.2.4.
CheckvsCurrentTimeStr	See Section 2.19.3.3.4.
DialogWindow::SetValue	See Section 2.19.5.5.11.
CheckList_Handler	See Section 2.19.3.7.2.
CheckUTMString	See Section 2.19.3.3.1.
HiliteWindow	Standard Window Manager function for Macintosh.
SelectWindow	Standard Window Manager function for Macintosh.
SetPort	Standard Quickdraw function for Macintosh.
DialogWindow::SelectText	See Section 2.19.5.5.9.
ToUpper	Macro defined in Development:SIMNET:CEW:CheckDialogs.c.
CheckDTGString	See Section 2.19.3.3.2.
NotImpDialog	See Section 2.19.3.4.3.

Table 2.19-21 CheckEmplacementDialog Information.

2.19.3.2.6 setBreachTime

serBreachTime calculates the amount of time it will take to breach a lane, given the assets information entered in the dialog by the user. The dialog is specified by the *dialog* and *item* parameters. The function call is setBreachTime(dialogp, item). Table 2.19-22 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
dialogp	pointer to DialogWindow	Development:SIMNET:CEW: Dialog_Windows.h
item	short	Standard C type.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
end_pt	LongPt	Development:SIMNET:libmac:longpt.h
start_pt	LongPt	Development:SIMNET:libmac:longpt.h
lineCharge_number	short	Standard C type.
area	long	Standard C type.
move_time	float	Standard C type.
activity_time	float	Standard C type.
i	short	Standard C type.
retval	long	Standard C type.
length	long	Standard C type.
string	Str255	Development:THINK C:Mac #includes:MacTypes.h
string2	Str255	Development:THINK C:Mac #includes:MacTypes.h
item_time	DateTimeGroup	Development:SIMNET:libmac:dtg.h
sm_time	long	Standard C type.
sa_time	long	Standard C type.
f_time	long	Standard C type.
charges	float	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
retval	long	The amount of time it will take to breach the lane.
Calls		
Function	**Where Described**	
TextList::GetCount	See Section 2.19.5.15.5.	
CheckList::GetState	See Section 2.19.5.1.6.	
ResourceObj::GetKind	See Section 2.19.5.9.2.	
DialogWindow::GetText	See Section 2.19.5.5.7.	
DialogWindow::SetText	See Section 2.19.5.5.8.	
UTMToLongPt	See Section 2.19.3.3.5.	
DistBetween2Pts	See Section 2.22.1.23.2.	
NumToString	Standard Binary to Decimal Conversion Package function for Macintosh.	
ResourceObj::DistanceToSecs	See Section 2.19.5.9.9.	
StringToDTG	See Section 2.22.1.15.2.	
DTGElapsed	See Section 2.22.1.15.3.	
DTGToString	See Section 2.22.1.15.1.	
Notify	See Section 2.19.3.4.4.	
Called By		
Function	**Where Described**	
CheckBreachDialog	See Section 2.19.3.2.7.	

Table 2.19-22 setBreachTime Information.

2.19.3.2.7 CheckBreachDialog

CheckBreachDialog checks the validity of data entered in the Breach Dialog fields. In addition, the routine performs calculations of mission time requirements. The function call is CheckBreachDialog(dialogp, item, to_reset, theEvent). Table 2.19-23 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
dialogp	pointer to DialogWindow	Development:SIMNET:CEW: Dialog_Windows.h
item	short	Standard C type.
to_reset	pointer to Boolean	Development:THINK C: Mac #includes:MacTypes.h
theEvent	pointer to EventRecord	Development:THINK C: Mac #includes:EventMgr.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
dtg	DateTimeGroup	Development:SIMNET:libmac: dtg.h
current_time	DateTimeGroup	Development:SIMNET:libmac: dtg.h
wp	DialogPtr	Development:THINK C: Mac #includes:DialogMgr.h
end_pt	LongPt	Development:SIMNET:libmac: longpt.h
start_pt	LongPt	Development:SIMNET:libmac: longpt.h
h_item	Handle	Development:THINK C: Mac #includes:MacTypes.h
box	Rect	Development:THINK C: Mac #includes:MacTypes.h
string	Str255	Development:THINK C: Mac #includes:MacTypes.h
string2	Str255	Development:THINK C: Mac #includes:MacTypes.h
x	short	Standard C type.
i	short	Standard C type.
lineCharge_number	short	Standard C type.
type	short	Standard C type.
time_diff	long	Standard C type.
move_time	unsigned long	Standard C type.
activity_time	unsigned long	Standard C type.
ts	Str255	Development:THINK C: Mac #includes:MacTypes.h
Return Values		
Return Value	**Type**	**Meaning**
TRUE	short	Data is accepted; user is allowed to continue.
FALSE	short	Data is not accepted; the dialog cannot be brought down.

Calls	
Function	**Where Described**
DialogWindow::GetDialogPtr	See Section 2.19.5.5.16.
GetDItem	Standard Dialog Manager function for Macintosh.
DialogWindow::GetText	See Section 2.19.5.5.7.
UTMToLongPt	See Section 2.19.3.3.5.
Notify	See Section 2.19.3.4.4.
TextList::GetCount	See Section 2.19.5.15.5.
CheckList::GetState	See Section 2.19.5.1.6.
ResourceObj::GetKind	See Section 2.19.5.9.2.
Get_DTG	See Section 2.22.1.15.4.
DTGToString	See Section 2.22.1.15.1.
DialogWindow::SetText	See Section 2.19.5.5.8.
setBreachTime	See Section 2.19.3.2.6.
CheckvsCurrentTimeStr	See Section 2.19.3.3.4.
DialogWindow::SetValue	See Section 2.19.5.5.11.
CheckList_Handler	See Section 2.19.3.7.2.
CheckUTMString	See Section 2.19.3.3.1.
HiliteWindow	Standard Window Manager function for Macintosh.
SelectWindow	Standard Window Manager function for Macintosh.
SetPort	Standard Quickdraw function for Macintosh.
SelIText	Standard Dialog Manager function for Macintosh.
ToUpper	Macro defined in Development:SIMNET:CEW:CheckDialogs.c.
CheckDTGString	See Section 2.19.3.3.2.
NotImpDialog	See Section 2.19.3.4.3.

Table 2.19-23 CheckBreachDialog Information.

2.19.3.2.8 CheckMoveDialog

CheckMoveDialog checks the validity of the data entered in the Move Mission Dialog fields. In addition, the routine performs calculations of mission time requirements (the length of time it takes to move to the given point). The function call is CheckMoveDialog(dialogp, item, to_reset, theEvent). Table 2.19-24 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
dialogp	pointer to DialogWindow	Development:SIMNET:CEW: Dialog_Windows.h
item	short	Standard C type.
to_reset	pointer to Boolean	Development:THINK C: Mac #includes:MacTypes.h
theEvent	pointer to EventRecord	Development:THINK C: Mac #includes:EventMgr.h

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
type	short	Standard C type.
h_item	Handle	Development:THINK C: Mac #includes:MacTypes.h
box	Rect	Development:THINK C: Mac #includes:MacTypes.h
string	Str255	Development:THINK C: Mac #includes:MacTypes.h
string2	Str255	Development:THINK C: Mac #includes:MacTypes.h
x	short	Standard C type.
i	short	Standard C type.
wp	DialogPtr	Development:THINK C: Mac #includes:DialogMgr.h
xlong	LongPt	Development:SIMNET:libmac: longpt.h
move_time	unsigned long	Standard C type.
dtg	DateTimeGroup	Development:SIMNET:libmac: dtg.h
time_diff	long	Standard C type.
ts	Str255	Development:THINK C: Mac #includes:MacTypes.h
current_time	DateTimeGroup	Development:SIMNET:libmac: dtg.h

Return Values		
Return Value	**Type**	**Meaning**
TRUE	short	Data is accepted; user is allowed to continue.
FALSE	short	Data is not accepted; the dialog cannot be brought down.

Calls	
Function	**Where Described**
DialogWindow::GetDialogPtr	See Section 2.19.5.5.16.
GetDItem	Standard Dialog Manager function for Macintosh.
DialogWindow::GetText	See Section 2.19.5.5.7.
UTMToLongPt	See Section 2.19.3.3.5.
Notify	See Section 2.19.3.4.4.
TextList::GetCount	See Section 2.19.5.15.5.
CheckList::GetState	See Section 2.19.5.1.6.
Get_DTG	See Section 2.22.1.15.4.
DTGToString	See Section 2.22.1.15.1.
DialogWindow::SetText	See Section 2.19.5.5.8.
DTGElapsed	See Section 2.22.1.15.3.
CheckvsCurrentTimeStr	See Section 2.19.3.3.4.
CheckUTMString	See Section 2.19.3.3.1.
HiliteWindow	Standard Window Manager function for Macintosh.
SelectWindow	Standard Window Manager function for Macintosh.
SetPort	Standard Quickdraw function for Macintosh.
SelIText	Standard Dialog Manager function for Macintosh.
ToUpper	Macro defined in Development:SIMNET:CEW:CheckDialogs.c.
ResourceObj::DistanceToSecs	See Section 2.19.5.9.9.
StringToDTG	See Section 2.22.1.15.2.
CheckList_Handler	See Section 2.19.3.7.2.
CheckDTGString	See Section 2.19.3.3.2.
NotImpDialog	See Section 2.19.3.4.3.

Table 2.19-24 CheckMoveDialog Information.

2.19.3.2.9 CheckMissionTypeDialog

CheckMissionTypeDialog checks for validity of user input in the Mission Type Dialog. The OK button, Emplace Button, Breach button, and Move button are enabled. The Mission Type dialog allows the user to choose whether the mission is an Emplacement, Breach, or Move Mission. The function call is CheckMissionTypeDialog(dialogp, item, to_reset, theEvent). Table 2.19-25 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
dialogp	pointer to DialogWindow	Development:SIMNET:CEW: Dialog_Windows.h
item	short	Standard C type.
to_reset	pointer to Boolean	Development:THINK C: Mac #includes:MacTypes.h
theEvent	pointer to EventRecord	Development:THINK C: Mac #includes:EventMgr.h

Return Values		
Return Value	**Type**	**Meaning**
TRUE	short	Data is accepted; user is allowed to continue.
FALSE	short	Data is not accepted; the dialog cannot be brought down.

Table 2.19-25 CheckMissionTypeDialog Information.

2.19.3.2.10 CheckStatEmplacementDialog

CheckStatEmplacementDialog checks for changes made in the Emplacement dialog. This function is not currently called by the application. The function call is CheckStatEmplacementDialog(dialogp, item, to_reset, theEvent). Table 2.19-26 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
dialogp	pointer to DialogWindow	Development:SIMNET:CEW: Dialog_Windows.h
item	short	Standard C type.
to_reset	pointer to Boolean	Development:THINK C: Mac #includes:MacTypes.h
theEvent	pointer to EventRecord	Development:THINK C: Mac #includes:EventMgr.h
Return Values		
Return Value	**Type**	**Meaning**
TRUE	short	The changes are valid.
FALSE	short	The changes are not valid.
Calls		
Function	**Where Described**	
CheckList_Handler	See Section 2.19.3.7.2.	
NotImpDialog	See Section 2.19.3.4.3.	

Table 2.19-26 CheckStatEmplacementDialog Information.

2.19.3.2.11 CheckStatBreachDialog

CheckStatBreachDialog checks for changes made in the Breach dialog. This function is not currently called by the application. The function call is CheckStatBreachDialog(dialogp, item, to_reset, theEvent). Table 2.19-27 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
dialogp	pointer to DialogWindow	Development:SIMNET:CEW: Dialog_Windows.h
item	short	Standard C type.
to_reset	pointer to Boolean	Development:THINK C: Mac #includes:MacTypes.h
theEvent	pointer to EventRecord	Development:THINK C: Mac #includes:EventMgr.h
Return Values		
Return Value	**Type**	**Meaning**
TRUE	short	The changes are valid.
FALSE	short	The changes are invalid.
Calls		
Function	**Where Described**	
CheckList_Handler	See Section 2.19.3.7.2.	
NotImpDialog	See Section 2.19.3.4.3.	

Table 2.19-27 CheckStatBreachDialog Information.

2.19.3.2.12 CheckStatMoveDialog

CheckStatMoveDialog checks for changes in the Move Mission dialog. This function is not currently called by the application. The function call is CheckStatMoveDialog(dialogp, item, to_reset, theEvent). Table 2.19-28 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
dialogp	pointer to DialogWindow	Development:SIMNET:CEW: Dialog_Windows.h
item	short	Standard C type.
to_reset	pointer to Boolean	Development:THINK C: Mac #includes:MacTypes.h
theEvent	pointer to EventRecord	Development:THINK C: Mac #includes:EventMgr.h
Return Values		
Return Value	**Type**	**Meaning**
TRUE	short	The changes are valid.
FALSE	short	The changes are invalid.

Calls	
Function	**Where Described**
CheckList_Handler	See Section 2.19.3.7.2.
NotImpDialog	See Section 2.19.3.4.3.

Table 2.19-28 CheckStatMoveDialog Information.

2.19.3.2.13 CheckReviewDialog

CheckReviewDialog checks the validity of events in the Review dialogs. This function is not currently called by the application. The function call is CheckReviewDialog(). Table 2.19-29 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
dialogp	pointer to DialogWindow	Development:SIMNET:CEW: Dialog_Windows.h
item	short	Standard C type.
to_reset	pointer to Boolean	Development:THINK C: Mac #includes:MacTypes.h
theEvent	pointer to EventRecord	Development:THINK C: Mac #includes:EventMgr.h
Return Values		
Return Value	**Type**	**Meaning**
TRUE	short	The event is valid
FALSE	short	The event is not valid
Calls		
Function	**Where Described**	
DialogWindow::SetValue	See Section 2.19.5.5.11.	

Table 2.19-29 CheckReviewDialog Information.

2.19.3.2.14 DetermineMines

DetermineMines determines the number of mines it would take to make the given density of minefield that the user entered. The user entered data is located in the dialog specified by the parameters, *dialogp* and *item*. The function call is DetermineMines(dialogp, item). Table 2.19-30 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
dialogp	pointer to DialogWindow	Development:SIMNET:CEW: Dialog_Windows.h
item	short	Standard C type.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
string	Str255	Development:THINK C: Mac #includes:MacTypes.h
string2	Str255	Development:THINK C: Mac #includes:MacTypes.h
den	array of 3 float	Standard C type.
area	unsigned long	Standard C type.
dis	unsigned long	Standard C type.
apf	unsigned long	Standard C type.
apb	unsigned long	Standard C type.
at	unsigned long	Standard C type.
dis1	unsigned long	Standard C type.
i	short	Standard C type.
width	short	Standard C type.
count	short	Standard C type.
pt	array of 5 LongPt	Development:SIMNET:libmac: longpt.h
index	short	Standard C type.
dlogitem	short	Standard C type.
d_area	double	Standard C type.
Calls		
Function	**Where Described**	
isLegalMinefield	See Section 2.19.3.2.2.	
DialogWindow::GetValue	See Section 2.19.5.5.10.	
DialogWindow::GetText	See Section 2.19.5.5.7.	
UTMToLongPt	See Section 2.19.3.3.5.	
DistBetween2Pts	See Section 2.22.1.23.2.	
poly_area	See Section 2.19.4.1.1.	
NumToString	Standard Binary to Decimal Conversion Package function for Macintosh.	
DialogWindow::SetText	See Section 2.19.5.5.8.	
Called By		
Function	**Where Described**	
CheckEmplacementDialog	See Section 2.19.3.2.5.	

Table 2.19-30 DetermineMines Information.

2.19.3.2.15 ReviewSelection

ReviewSelection sets up the Review dialogs. The review dialogs show a mission (specified by the mission number *num*) to the user. The routine allows the user access to the mission data based on the status of the mission. If the mission is in progress, the user may only show the mission for review. If the mission is not in progress, the user may either show the mission, edit the mission, or abort the mission. The routine also handles the use of "hot keys" to expedite missions. TRUE is returned if the selection was successfully handled. FALSE is returned if the user attempts an illegal event, i.e. trying to force an aborted mission to execute. The function call is ReviewSelection(num). Table 2.19-31 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
num	short	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
lastButton	short	Standard C type.
thisMission	short	Standard C type.
status	short	Standard C type.
current_time	DateTimeGroup	Development:SIMNET:libmac:dtg.h
old_start_time	unsigned long	Standard C type.
old_end_time	unsigned long	Standard C type.
new_end_time	unsigned long	Standard C type.
diff	long	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
TRUE	short	Successful.
FALSE	short	Unsuccessful.
Calls		
Function	**Where Described**	
MissionObj::GetStatus	See Section 2.19.5.7.21.	
DialogWindow::SetValue	See Section 2.19.5.5.11.	
DialogWindow::HiliteItem	See Section 2.19.5.5.12.	
DialogWindow::activate	See Section 2.19.5.5.3.	
Swindow::update	See Section 2.19.5.13.4.	
DialogWindow::GetValue	See Section 2.19.5.5.10.	
MissionObj::show	See Section 2.19.5.7.4.	
MissionObj::edit	See Section 2.19.5.7.2.	
MissionObj::SetStatus	See Section 2.19.5.7.20.	
MissionObj::GetStartMoveTime	See Section 2.19.5.7.6.	
MissionObj::GetEndTime	See Section 2.19.5.7.8.	
Get_DTG	See Section 2.22.1.15.4.	
MissionObj::SetStartMoveTime	See Section 2.19.5.7.9.	
MissionObj::SetEndTime	See Section 2.19.5.7.11.	
MissionObj::UpdateMissionString	See Section 2.19.5.7.5.	
ServiceObj::CheckQs	See Section 2.19.5.11.4.	
Notify	See Section 2.19.3.4.4.	
MissionObj::GetStartActivityTime	See Section 2.19.5.7.7.	
MissionObj::SetStartActivityTime	See Section 2.19.5.7.10.	
Called By		
Function	**Where Described**	
MainEventLoop	See Section 2.19.2.1.2.	

Table 2.19-31 ReviewSelection Information.

2.19.3.3 DialogFieldSupport.c
Development:SIMNET:MCC:CEW:DialogFieldSupport.c

This file provides routines used to check dialog fields data.

2.19.3.3.1 CheckUTMString

CheckUTMString checks the validity of the UTM coordinate passed in the parameter, *utmstring*. The function call is CheckUTMString(utmstring). Table 2.19-32 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
utmstring	UTMCoordinates	Development:SIMNET:CEW: SimnetTypes.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
cStr	pointer to char	Standard C type.
map	MapCoordinates	Development:SIMNET:libmac: map.h
tempStr	Str255	Development:THINK C: Mac #includes:MacTypes.h
outStr	Str255	Development:THINK C: Mac #includes:MacTypes.h
result	short	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
TRUE	short	Valid UTM string.
FALSE	short	Invalid UTM string.
Calls		
Function	**Where Described**	
stpcpy	See Section 2.19.3.10.3.	
Notify	See Section 2.19.3.4.4.	
StringToMapCoordinates	See Section 2.22.1.26.1.	
Called By		
Function	**Where Described**	
CheckEmplacementDialog	See Section 2.19.3.2.5.	
CheckBreachDialog	See Section 2.19.3.2.7.	
CheckMoveDialog	See Section 2.19.3.2.8.	

Table 2.19-32 CheckUTMString Information.

2.19.3.3.2 CheckDTGString

CheckDTGString checks the validity of the DTG string passed in the parameter, *dtgstring*. The function call is CheckDTGString(dtgstring). Table 2.19-33 describes the parameters used, errors returned and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
dtgstring	pointer to char	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
dtg	DateTimeGroup	Development:SIMNET:libmac:dtg.h
tempStr	array of 255 char	Standard C type.
cStr	char	Standard C type.
result	short	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
TRUE	short	Valid DTG string.
FALSE	short	Invalid DTG string.
Errors		
Error Name	**Reason for Error**	
DTG_SUCCESS	Valid DTG string.	
DTG_ERROR	Incorrect date time group.	
DTG_BAD_DAY	Wrong day.	
DTG_BAD_HOUR	Wrong hour.	
DTG_BAD_MINUTE	Wrong minute.	
Calls		
Function	**Where Described**	
StringToDTG	See Section 2.22.1.15.2.	
Notify	See Section 2.19.3.4.4.	
Called By		
Function	**Where Described**	
CheckEmplacementDialog	See Section 2.19.3.2.5.	
CheckBreachDialog	See Section 2.19.3.2.7.	
CheckMoveDialog	See Section 2.19.3.2.8.	

Table 2.19-33 CheckDTGString Information.

2.19.3.3.3 CheckvsCurrentTime

CheckvsCurrentTime compares the time passed in the parameter, *dtg*, with the current time. If *dtg* is later than or equal to the current time, TRUE is returned. If *dtg* is earlier than the current time, FALSE is returned. The function call is CheckvsCurrentTime(dtg). Table 2.19-34 describes the parameters used, errors returned and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
dtg	DateTimeGroup	Development:SIMNET:libmac:dtg.h

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
now	unsigned long	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
TRUE	short	*dtg* is later than or equal to the current time.
FALSE	short	*dtg* is earlier than the current time.
Calls		
Function	**Where Described**	
GetDateTime	Standard Operating System Utility function for Macintosh.	
Called By		
Function	**Where Described**	
CheckvsCurrentTimeStr	See Section 2.19.3.3.4.	

Table 2.19-34 CheckvsCurrentTime Information.

2.19.3.3.4 CheckvsCurrentTimeStr

CheckvsCurrentTimeStr compares the time string passed in the parameter, *dtgstring*, with the current time. If *dtgstring* is later than or equal to the current time, TRUE is returned. If *dtgstring* is earlier than the current time, FALSE is returned. The function call is CheckvsCurrentTimeStr(dtgstring). Table 2.19-35 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
dtgstring	pointer to char	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
dtg	DateTimeGroup	Development:SIMNET:libmac:dtg.h
tempStr	array of 255 char	Standard C type.
cStr	pointer to char	Standard C type.
result	short	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
CheckvsCurrentTime(dtg)	short	If TRUE, *dtgstring* is later than or equal to the current time; If FALSE, *dtgstring* is earlier than the current time.
Calls		
Function	**Where Described**	
StringToDTG	See Section 2.22.1.15.2.	
CheckvsCurrentTime	See Section 2.19.3.3.3.	

Called By	
Function	**Where Described**
CheckEmplacementDialog	See Section 2.19.3.2.5.
CheckBreachDialog	See Section 2.19.3.2.7.
CheckMoveDialog	See Section 2.19.3.2.8.

Table 2.19-35 CheckvsCurrentTimeStr Information.

2.19.3.3.5 UTMToLongPt

UTMToLongPt converts a UTM coordinate, *utmstring*, to a LongPt coordinate. The parameter, *pt*, is filled in with the long (x,y) point. The function call is UTMToLongPt(utmstring, pt). Table 2.19-36 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
utmstring	UTMCoordinates	Development:SIMNET:CEW: SimnetTypes.h
pt	pointer to LongPt	Development:SIMNET:libmac: longpt.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
map	MapCoordinates	Development:SIMNET:libmac: map.h
cStr	pointer to char	Standard C type.
Calls		
Function	**Where Described**	
StringToMapCoordinates	See Section 2.22.1.26.1.	
SetLongPt	See Section 2.22.1.23.5.	
Called By		
Function	**Where Described**	
isLegalMinefield	See Section 2.19.3.2.2.	
setBreachTime	See Section 2.19.3.2.6.	
CheckBreachDialog	See Section 2.19.3.2.7.	
CheckMoveDialog	See Section 2.19.3.2.8.	
DetermineMines	See Section 2.19.3.2.14.	
GetDisTime	See Section 2.19.3.3.6.	
MissionObj::ResetMission Time	See Section 2.19.5.7.18.	
MissionObj::SendEmplace ment	See Section 2.19.5.7.27.	
MissionObj::SendBreach	See Section 2.19.5.7.28.	
ResourceObj::DistanceToSec s	See Section 2.19.5.9.9.	

Table 2.19-36 UTMToLongPt Information.

2.19.3.3.6 GetDisTime

GetDisTime calculates the amount of time it takes to travel between two given points, *start_pt* and *end_pt*, at a given speed, *kph*. The function call is GetDisTime(start_c, end_c, kph). Table 2.19-37 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
start_c	pointer to char	Standard C type.
end_c	pointer to char	Standard C type.
kph	short	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
start_pt	LongPt	Development:SIMNET:libmac:longpt.h
end_pt	LongPt	Development:SIMNET:libmac:longpt.h
speed	float	Standard C type.
dis	long	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
dis	long	The amount of time it takes to travel the given distance at the given speed.
Calls		
Function	**Where Described**	
UTMToLongPt	See Section 2.19.3.3.5.	
DistBetween2Pts	See Section 2.22.1.23.2.	

Table 2.19-37 GetDisTime Information.

2.19.3.4 DoAbout.c

Development:SIMNET:MCC:CEW:DoAbout.c

This file contains routines for handling alerts.

2.19.3.4.1 SetParamText

SetParamText sets the text fields for an alert. The parameter, *strNum*, is the index to a standard text string. The function call is SetParamText(strNum). Table 2.19-38 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
strNum	short	Standard C type.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
h	StringHandle	Development:THINK C: Mac #includes:MacTypes.h
Calls		
Function	**Where Described**	
GetString	Standard Toolbox Utility function for Macintosh.	
HLock	Standard Memory Manager function for Macintosh.	
ParamText	Standard Dialog Manager function for Macintosh.	
HUnlock	Standard Memory Manager function for Macintosh.	
Called By		
Function	**Where Described**	
DoAbout	See Section 2.19.3.4.2.	
NotImpDialog	See Section 2.19.3.4.3.	

Table 2.19-38 SetParamText Information.

2.19.3.4.2 DoAbout

DoAbout implements the "About..." selection in the Macintosh apple menu. The function call is DoAbout(). Table 2.19-39 describes the functions called using this function.

Calls	
Function	**Where Described**
SetParamText	See Section 2.19.3.4.1.
Alert	Standard Dialog Manager function for Macintosh.
Called By	
Function	**Where Described**
MainEventLoop	See Section 2.19.2.1.2.
DoMenu	See Section 2.19.2.2.2.

Table 2.19-39 DoAbout Information.

2.19.3.4.3 NotImpDialog

NotImpDialog implements the "This function not implemented" dialog. The function call is NotImpDialog(). Table 2.19-40 describes the functions called using this function.

Calls	
Function	**Where Described**
SetParamText	See Section 2.19.3.4.1.
Alert	Standard Dialog Manager function for Macintosh.

Called By	
Function	**Where Described**
DoMenu	See Section 2.19.2.2.2.
CheckEmplacementDialog	See Section 2.19.3.2.5.
CheckBreachDialog	See Section 2.19.3.2.7.
CheckMoveDialog	See Section 2.19.3.2.8.
CheckStatEmplacementDialog	See Section 2.19.3.2.10.
CheckStatBreachDialog	See Section 2.19.3.2.11.
CheckStatMoveDialog	See Section 2.19.3.2.12.
MainHelpButton	See Section 2.19.3.5.5.
HelpStatus	See Section 2.19.3.8.6.

Table 2.19-40 NotImpDialog Information.

2.19.3.4.4 Notify

Notify fills in an alert and notifies the user of a problem. The alert is filled in with the string passed in *str*. The function call is Notify(str). Table 2.19-41 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
str	Ptr	Development:THINK C: Mac #includes:MacTypes.h
Calls		
Function	**Where Described**	
ParamText	Standard Dialog Manager function for Macintosh.	
Alert	Standard Dialog Manager function for Macintosh.	

Called By	
Function	**Where Described**
main	See Section 2.19.2.1.1.
MacInits	See Section 2.19.2.1.3.
ProcessRequest	See Section 2.19.2.1.5.
GetTerrainBounds	See Section 2.19.2.1.6.
isLegalMinefield	See Section 2.19.3.2.2.
CheckEmplacementDialog	See Section 2.19.3.2.5.
setBreachTime	See Section 2.19.3.2.6.
CheckBreachDialog	See Section 2.19.3.2.7.
CheckMoveDialog	See Section 2.19.3.2.8.
ReviewSelection	See Section 2.19.3.2.15.
CheckUTMString	See Section 2.19.3.3.1.
CheckDTGString	See Section 2.19.3.3.2.
poly_area	See Section 2.19.4.1.1.
poly_area_loc	See Section 2.19.4.1.2.
inside_poly	See Section 2.19.4.1.3.
MissionObj::NewMission	See Section 2.19.5.7.3.
MissionObj::show	See Section 2.19.5.7.4.
MissionObj::ResetMission Time	See Section 2.19.5.7.18.
MissionObj::SetPosition	See Section 2.19.5.9.4.
ServiceObj::CheckQs	See Section 2.19.5.11.4.

Table 2.19-41 Notify Information.

2.19.3.4.5 MySound

MySound makes a noise to alert the user. *num* is the index of the sound to make. The function call is MySound(num). Table 2.19-42 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
num	short	Standard C type.
Calls		
Function	**Where Described**	
SysBeep	Standard Operating System Utility function for Macintosh.	
Called By		
Function	**Where Described**	
ServiceObj::CheckQs	See Section 2.19.5.11.4.	

Table 2.19-42 MySound Information.

2.19.3.5 HotSpots.c

Development:SIMNET:MCC:CEW:HotSpots.c

This file contains button handling routines. Table 2.19-43 describes the variables used by HotSpots.c.

Variables		
Variable	**Type**	**Where Typedef Declared**
cewMission	extern pointer to array of MissionObj	Development:SIMNET:CEW: MissionObj.h
nextMission	extern short	Standard C type.
select_dlog	extern pointer to DialogWindow	Development:SIMNET:CEW: Dialog_Windows.h
main_window	extern pointer to ComWindow	Development:SIMNET:CEW: ComWindows.h
serviceq	extern pointer to ServiceObj	Development:SIMNET:CEW: ServiceObj.h

Table 2.19-43 HotSpots.c variables Information.

2.19.3.5.1 indone

indone is the prototype routine (or template) used to model other button handling routines in this file.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
cw	pointer to ComWindow	Development:SIMNET:CEW: ComWindows.h
xpoint	Point	Development:THINK C: Mac #includes:MacTypes.h
r	Rect	Development:THINK C: Mac #includes:MacTypes.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
aevent	EventRecord	Development:THINK C: Mac #includes:EventMgr.h
eventpt	Point	Development:THINK C: Mac #includes:MacTypes.h
Return Values		
Return Value	**Type**	**Meaning**
TRUE	short	The user has requested an action
FALSE	short	The user has not requested an action.
Calls		
Function	**Where Described**	
PtInRect	Standard Quickdraw function for Macintosh.	
InvertRect	Standard Quickdraw function for Macintosh.	
GetNextEvent	Standard Toolbox Event Manager function for Macintosh.	
GlobalToLocal	Standard Quickdraw function for Macintosh.	

Table 2.19-44 indone Information.

2.19.3.5.2 MissionButton

MissionButton is called when the user clicks the mouse in the New Mission button. The new mission series is brought up. The function call is MissionButton(cw, r, xpoint). Table 2.19-45 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
cw	pointer to ComWindow	Development:SIMNET:CEW: ComWindows.h
xpoint	Point	Development:THINK C: Mac #includes:MacTypes.h
r	Rect	Development:THINK C: Mac #includes:MacTypes.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
r1	Rect	Development:THINK C: Mac #includes:MacTypes.h
aevent	EventRecord	Development:THINK C: Mac #includes:EventMgr.h
eventpt	Point	Development:THINK C: Mac #includes:MacTypes.h
Return Values		
Return Value	**Type**	**Meaning**
TRUE	short	The user requested the new mission dialog.
FALSE	short	The user did not request the new mission dialog.
Calls		
Function	**Where Described**	
PtInRect	Standard Quickdraw function for Macintosh.	
InvertRect	Standard Quickdraw function for Macintosh.	
GetNextEvent	Standard Toolbox Event Manager function for Macintosh.	
GlobalToLocal	Standard Quickdraw function for Macintosh.	
SetRect	Standard Quickdraw function for Macintosh.	
LocalRectToGlobal	See Section 2.19.3.10.1.	
ZoomRect	See Section 2.22.1.47.2.	
MissionSelection	See Section 2.19.3.5.3.	

Table 2.19-45 MissionButton Information.

2.19.3.5.3 MissionSelection

MissionSelection handles the user's selection of the mission type. The choices of mission types are the Emplacement Mission, Breach Mission, or Move Mission. The function call is MissionSelection(). Table 2.19-46 describes the internal variables used and functions called using this function.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
lastButton	short	Standard C type.
thisMission	short	Standard C type.
status	short	Standard C type.
missionType	short	Standard C type.
Calls		
Function	**Where Described**	
DialogWindow::activate	See Section 2.19.5.5.3.	
Swindow::update	See Section 2.19.5.13.4.	
MissionObj::init	See Section 2.19.5.7.1.	
MissionObj::NewMission	See Section 2.19.5.7.3.	
Called By		
Function	**Where Described**	
MissionButton	See Section 2.19.3.5.2.	

Table 2.19-46 MissionSelection Information.

2.19.3.5.4 StatusButton

StatusButton is called when the user clicks the mouse in the Asset Status button. The list of assets and their status is brought up. The function call is StatusButton(cw, r, xpoint). Table 2.19-47 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
cw	pointer to ComWindow	Development:SIMNET:CEW: ComWindows.h
xpoint	Point	Development:THINK C: Mac #includes:MacTypes.h
r	Rect	Development:THINK C: Mac #includes:MacTypes.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
r1	Rect	Development:THINK C: Mac #includes:MacTypes.h
aevent	EventRecord	Development:THINK C: Mac #includes:EventMgr.h
eventpt	Point	Development:THINK C: Mac #includes:MacTypes.h
Return Values		
Return Value	**Type**	**Meaning**
TRUE	short	The user requested the Asset Status list.
FALSE	short	The user did not request the Asset Status list.

Calls	
Function	**Where Described**
PtInRect	Standard Quickdraw function for Macintosh.
InvertRect	Standard Quickdraw function for Macintosh.
GetNextEvent	Standard Toolbox Event Manager function for Macintosh.
GlobalToLocal	Standard Quickdraw function for Macintosh.
DoResourceStatus	See Section 2.19.3.8.1.

Table 2.19-47 StatusButton Information.

2.19.3.5.5 MainHelpButton

MainHelpButton is called when the user clicks the mouse in the Main Help button. Since the Main Help dialog is not implemented, the "This dialog not implemented" dialog is brought up. The function call is MainHelpButton(cw, r, xpoint). Table 2.19-48 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
cw	pointer to ComWindow	Development:SIMNET:CEW: ComWindows.h
xpoint	Point	Development:THINK C: Mac #includes:MacTypes.h
r	Rect	Development:THINK C: Mac #includes:MacTypes.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
r1	Rect	Development:THINK C: Mac #includes:MacTypes.h
aevent	EventRecord	Development:THINK C: Mac #includes:EventMgr.h
eventpt	Point	Development:THINK C: Mac #includes:MacTypes.h
Return Values		
Return Value	**Type**	**Meaning**
TRUE	short	The user requested the main help dialog.
FALSE	short	The user did not request the main help dialog.
Calls		
Function	**Where Described**	
PtInRect	Standard Quickdraw function for Macintosh.	
InvertRect	Standard Quickdraw function for Macintosh.	
GetNextEvent	Standard Toolbox Event Manager function for Macintosh.	
GlobalToLocal	Standard Quickdraw function for Macintosh.	
NotImpDialog	See Section 2.19.3.4.3.	

Table 2.19-48 MainHelpButton Information.

2.19.3.6 Refresh.c
Development:SIMNET:MCC:CEW:Refresh.c

This file contains a routine which implements the initial window in the CEW Console.

Variables		
Variable	**Type**	**Where Typedef Declared**
missionList	extern pointer to TextList	Development:SIMNET:CEW: TextList.h
mainCTL	ControlHandle	Development:THINK C: Mac #includes:ControlMgr.h
resCtl	ControlHandle	Development:THINK C: Mac #includes:ControlMgr.h
m_helpCTL	ControlHandle	Development:THINK C: Mac #includes:ControlMgr.h

Table 2.19-49 Refresh.c Variable Information.

2.19.3.6.1 PrintWindow

PrintWindow sets up the initial window containing the mission table. The mission table text headings are filled in and the buttons are set up and displayed. The parameter *window* specifies the window containing the mission table. The function call is PrintWindow(window). Table 2.19-50 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
window	pointer to ComWindow	Development:SIMNET:CEW: ComWindows.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
r	Rect	Development:THINK C: Mac #includes:MacTypes.h
Calls		
Function	**Where Described**	
Swindow::IsVisible	See Section 2.19.5.13.6.	
TextFont	Standard Quickdraw function for Macintosh.	
TextSize	Standard Quickdraw function for Macintosh.	
MoveTo	Standard Quickdraw function for Macintosh.	
DrawString	Standard Quickdraw function for Macintosh.	
SetRect	Standard Quickdraw function for Macintosh.	
NewControl	Standard Control Manager function for Macintosh.	
Swindow::GetWindowPtr	See Section 2.19.5.13.10.	
ComWindow::AddHotSpot	See Section 2.19.5.3.2.	
DrawControls	Standard Control Manager function for Macintosh.	
OutLineRect	See Section 2.19.3.10.5.	
FrontWindow	Standard Window Manager function for Macintosh.	
ReDrawClock	See Section 2.22.1.6.3.	

Table 2.19-50 PrintWindow Information.

2.19.3.7 ResItem.c
Development:SIMNET:MCC:CEW:ResItem.c

This file implements the lists of resources (or assets) which can be used on missions.

Variables		
Variable	**Type**	**Where Typedef Declared**
breach_availList	pointer to CheckList	Development:SIMNET:CEW: CheckList.h
move_availList	pointer to CheckList	Development:SIMNET:CEW: CheckList.h
emplace_availList	pointer to CheckList	Development:SIMNET:CEW: CheckList.h
stat_emplace_availList	pointer to CheckList	Development:SIMNET:CEW: CheckList.h
stat_breach_availList	pointer to CheckList	Development:SIMNET:CEW: CheckList.h
stat_move_availList	pointer to CheckList	Development:SIMNET:CEW: CheckList.h
main_window	extern pointer to ComWindow	Development:SIMNET:CEW: ComWindows.h
emplace_dlog	extern pointer to DialogWindow	Development:SIMNET:CEW: Dialog_Windows.h
breach_dlog	extern pointer to DialogWindow	Development:SIMNET:CEW: Dialog_Windows.h
move_dlog	extern pointer to DialogWindow	Development:SIMNET:CEW: Dialog_Windows.h
stat_emplace	extern pointer to DialogWindow	Development:SIMNET:CEW: Dialog_Windows.h
stat_breach	extern pointer to DialogWindow	Development:SIMNET:CEW: Dialog_Windows.h
stat_move	extern pointer to DialogWindow	Development:SIMNET:CEW: Dialog_Windows.h
cewResources	extern pointer to array of ResourceObj	Development:SIMNET:CEW: ResourceObj.h
totalResources	extern short	Standard C type.

Table 2.19-51 resItem.c Variable Information.

2.19.3.7.1 ResourceItems

ResourceItems updates the list of resources used on the specified mission. Resources may only be allocated to one mission. The parameter *wp* specifies which type of mission dialog (emplace, breach, move, etc.), and *i* is the index to a specific mission within the mission list. The function call is ResourceItems(wp, i). Table 2.19-52 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
wp	WindowPtr	Development:THINK C: Mac #includes:WindowMgr.h
i	short	Standard C type.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
r	Rect	Development:THINK C: Mac #includes:MacTypes.h
tempStr	Str255	Development:THINK C: Mac #includes:MacTypes.h
x	short	Standard C type.
Calls		
Function	**Where Described**	
DialogWindow::equal	See Section 2.19.5.5.15.	
DialogWindow::GetItemRect	See Section 2.19.5.5.14.	
CheckList::init	See Section 2.19.5.1.1.	
ResourceObj::GetName	See Section 2.19.5.9.10	
CheckList::AddRow	See Section 2.19.5.1.2.	
TextList::ListUpdate	See Section 2.19.5.15.2.	
CheckList::GetState	See Section 2.19.5.1.6.	

Table 2.19-52 ResourceItems Information.

2.19.3.7.2 CheckList_Handler

CheckList_Handler handles events in the list of Resources (or assets). When the mouse is clicked on a resource, the resource state is toggled to either selected or not selected. The parameter *availList* points to the available resources list. *theEvent* points to the resource row in the list. *state* is the state of that resource. The function call is CheckList_Handler(availList, theEvent, state). Table 2.19-53 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
availList	pointer to CheckList	Development:SIMNET:CEW: CheckList.h
theEvent	pointer to EventRecord	Development:THINK C: Mac #includes:EventMgr.h
state	short	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
eventpt	Point	Development:THINK C: Mac #includes:MacTypes.h
where	short	Standard C type.
Calls		
Function	**Where Described**	
GlobalToLocal	Standard Quickdraw function for Macintosh.	
TextList::List1Click	See Section 2.19.5.15.12.	
CheckList::ToggleState	See Section 2.19.5.1.7.	

Called By	
Function	**Where Described**
CheckEmplacementDialog	See Section 2.19.3.2.5.
CheckBreachDialog	See Section 2.19.3.2.7.
CheckMoveDialog	See Section 2.19.3.2.8.
CheckStatEmplacementDialog	See Section 2.19.3.2.10.
CheckStatBreachDialog	See Section 2.19.3.2.11.
CheckStatMoveDialog	See Section 2.19.3.2.12.

Table 2.19-53 CheckList_Handler Information.

2.19.3.7.3 DeleteResourceLists

DeleteResourceLists deletes all available resource lists. The function call is DeleteResourceLists(). Table 2.19-54 describes the functions called using this function.

Calls	
Function	**Where Described**
TextList::ListDispose	See Section 2.19.5.15.3.
DialogWindow::quit	See Section 2.19.5.5.4.

Table 2.19-54 DeleteResourceLists Information.

2.19.3.8 ResStatusWindow.c

Development:SIMNET:MCC:CEW:ResStatusWindow.c

ResStatusWindow.c implements the available resource status list dialog windows.

Variables		
Variable	**Type**	**Where Typedef Declared**
resource_window	pointer to ComWindow	Development:SIMNET:CEW: ComWindows.h
resList	pointer to TextList	Development:SIMNET:CEW: TextList.h
exitCTL	ControlHandle	Development:THINK C: Mac #includes:ControlMgr.h
helpCTL	ControlHandle	Development:THINK C: Mac #includes:ControlMgr.h
main_window	extern pointer to ComWindow	Development:SIMNET:CEW: ComWindows.h
cewResources	extern pointer to array of ResourceObj	Development:SIMNET:CEW: ResourceObj.h
totalResources	extern short	Standard C type.

Table 2.19-55 ResStatusWindow.c Variable Information.

2.19.3.8.1 DoResourceStatus

DoResourceStatus implements the Resource Status window. When the user clicks the mouse on the Asset Status button the Rsource Status window appears with a list of Resources (or Assets) with their current locations and states. The function call is DoResourceStatus(). Table 2.19-56 describes the functions called using this function.

Calls	
Function	**Where Described**
SetupResWindow	See Section 2.19.3.8.2.
OtherEventLoop	See Section 2.19.3.8.7.
ResCleanUp	See Section 2.19.3.8.8.
Called By	
Function	**Where Described**
MainEventLoop	See Section 2.19.2.1.2.
StatusButton	See Section 2.19.3.5.4.

Table 2.19-56 DoResourceStatus Information.

2.19.3.8.2 SetupResWindow

SetupResWindow sets up the window of lists of Resources (or Assets). The function call is SetupResWindow(). Table 2.19-57 describes the internal variables used and functions called using this function.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
r	Rect	Development:THINK C: Mac #includes:MacTypes.h
tempStr	Str255	Development:THINK C: Mac #includes:MacTypes.h
i	short	Standard C type.
Calls		
Function	**Where Described**	
ComWindow::initial	See Section 2.19.5.3.3.	
ComWindow::show	See Section 2.19.5.3.1.	
SetRect	Standard Quickdraw function for Macintosh.	
NewControl	Standard Control Manager function for Macintosh.	
Swindow::GetWindowPtr	See Section 2.19.5.13.10.	
ComWindow::AddHotSpot	See Section 2.19.5.3.2.	
OutLineRect	See Section 2.19.3.10.5.	
TextList::init	See Section 2.19.5.15.1.	
HeadingSetUp	See Section 2.19.3.8.4.	
ResourceObj::show	See Section 2.19.5.9.13.	
ResourceObj::GetStatus	See Section 2.19.5.9.12.	
ResourceObj::GetMission	See Section 2.19.5.9.5.	
TextList::AddRow	See Section 2.19.5.15.6.	
Called By		
Function	**Where Described**	
DoResourceStatus	See Section 2.19.3.8.1.	

Table 2.19-57 SetupResWindow Information.

2.19.3.8.3 RefreshWindow

RefreshWindow refreshes the screen specified by the parameter *window*. The function call is RefreshWindow(window). Table 2.19-58 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
window	pointer to ComWindow	Development:SIMNET:CEW: ComWindows.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
r	Rect	Development:THINK C: Mac #includes:MacTypes.h
Calls		
Function	**Where Described**	
Swindow::IsVisible	See Section 2.19.5.13.6.	
HeadingSetUp	See Section 2.19.3.8.4.	
TextList::ListUpdate	See Section 2.19.5.15.2.	
DrawControls	Standard Control Manager function for Macintosh.	
Swindow::GetWindowPtr	See Section 2.19.5.13.10.	
OutLineRect	See Section 2.19.3.10.5.	

Table 2.19-58 RefreshWindow Information.

2.19.3.8.4 HeadingSetUp

HeadingSetUp lists the heading titles in the window. The function call is HeadingSetUp(). Table 2.19-59 describes the functions called using this function.

Calls	
Function	**Where Described**
TextFont	Standard Quickdraw function for Macintosh.
TextSize	Standard Quickdraw function for Macintosh.
MoveTo	Standard Quickdraw function for Macintosh.
DrawString	Standard Quickdraw function for Macintosh.
Called By	
Function	**Where Described**
SetupResWindow	See Section 2.19.3.8.2.
RefreshWindow	See Section 2.19.3.8.3.

Table 2.19-59 HeadingSetUp Information.

2.19.3.8.5 DoneStatus

DoneStatus puts up the mission window when the user is done with the Resource Status window. *cw* points to the Resource Status Window; *xpoint* is the location of the mouse click; *r* is outline of the Done button. The function call is DoneStatus(cw, r, xpoint). Table 2.19-60 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
cw	pointer to ComWindow	Development:SIMNET:CEW: ComWindows.h
xpoint	Point	Development:THINK C: Mac #includes:MacTypes.h
r	Rect	Development:THINK C: Mac #includes:MacTypes.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
eventpt	Point	Development:THINK C: Mac #includes:MacTypes.h
x	short	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
TRUE	short	The user clicked on the Done button; the Resource Status window is put away and the Mission window is brought back.
FALSE	short	The user did not click on the Done button.
Calls		
Function	**Where Described**	
PtInRect	Standard Quickdraw function for Macintosh.	
TrackControl	Standard Control Manager function for Macintosh.	
TextList::ListDispose	See Section 2.19.5.15.3.	
KillControls	Standard Control Manager function for Macintosh.	
Swindow::GetWindowPtr	See Section 2.19.5.13.10.	
Swindow::quit	See Section 2.19.5.13.3.	

Table 2.19-60 DoneStatus Information.

2.19.3.8.6 HelpStatus

HelpStatus puts up the Help window when the user clicks on the Help button. Note that the Help window is not implemented. *cw* points to the current window; *xpoint* is the location of the mouse click; *r* is the rectangle outlining the Help button. The function call is HelpStatus(cw, r, xpoint). Table 2.19-61 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
cw	pointer to ComWindow	Development:SIMNET:CEW: ComWindows.h
xpoint	Point	Development:THINK C: Mac #includes:MacTypes.h
r	Rect	Development:THINK C: Mac #includes:MacTypes.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
eventpt	Point	Development:THINK C: Mac #includes:MacTypes.h
x	short	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
TRUE	short	The user clicked on the Help button.
FALSE	short	The user did not click on the Help button.
Calls		
Function	**Where Described**	
PtInRect	Standard Quickdraw function for Macintosh.	
TrackControl	Standard Control Manager function for Macintosh.	
NotImpDialog	See Section 2.19.3.4.3.	

Table 2.19-61 HelpStatus Information.

2.19.3.8.7 OtherEventLoop

OtherEventLoop handles events when the Resource Status window is frontmost. The function call is OtherEventLoop(). Table 2.19-62 describes the internal variables used and functions called using this function.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
myEvent	EventRecord	Development:THINK C: Mac #includes:EventMgr.h
eventpt	Point	Development:THINK C: Mac #includes:MacTypes.h
astr	Str255	Development:THINK C: Mac #includes:MacTypes.h
c	char	Standard C type.
i	short	Standard C type.
status	short	Standard C type.
wwindow	WindowPtr	Development:THINK C: Mac #includes:WindowMgr.h
windowcode	short	Standard C type.

Calls	
Function	**Where Described**
SystemTask	Standard Desk Manager function for Macintosh.
GetNextEvent	Standard Toolbox Event Manager function for Macintosh.
TextList::ListDispose	See Section 2.19.5.15.3.
KillControls	Standard Control Manager function for Macintosh.
Swindow::GetWindowPtr	See Section 2.19.5.13.10.
Swindow::quit	See Section 2.19.5.13.3.
Swindow::update	See Section 2.19.5.13.4.
FindWindow	Standard Window Manager function for Macintosh.
Swindow::equal	See Section 2.19.5.13.8.
HiliteWindow	Standard Window Manager function for Macintosh.
SelectWindow	Standard Window Manager function for Macintosh.
SetPort	Standard Quickdraw function for Macintosh.
GlobalToLocal	Standard Quickdraw function for Macintosh.
ComWindow::CheckHotSpot	See Section 2.19.5.3.4.
FrontWindow	Standard Window Manager function for Macintosh.
TextList::List1Click	See Section 2.19.5.15.12.
NetworkEventHandler	See Section 2.22.1.3.5.
Called By	
Function	**Where Described**
DoResourceStatus	See Section 2.19.3.8.1.

Table 2.19-62 OtherEventLoop Information.

2.19.3.8.8 ResCleanUp

ResCleanUp cleans up any allocated memory when a resource list is deleted. The function call is ResCleanUp(). Table 2.19-63 describes the functions which call this function.

Called By	
Function	**Where Described**
DoResourceStatus	See Section 2.19.3.8.1.

Table 2.19-63 ResCleanUp Information.

2.19.3.9 StatusSupport.c

Development:SIMNET:MCC:CEW:StatusSupport.c

This file contains routines used to support the Resource and Mission Status Windows.

Variables		
Variable	**Type**	**Where Typedef Declared**
mission_fields	array of long	Standard C type.
resource_fields	array of long	Standard C type.

Table 2.19-64 StatusSupport.c Variable Information.

2.19.3.9.1 AssembleMissionString

AssembleMissionString sets up fields in the Mission Table rows so that the columns line up properly. *num* is the mission number field; *type* is the mission type field; *where* is the mission location; *time1* is the mission start field; *time2* is the mission finish field; *status* is the mission status field; *order_type* is the mission order field; *finalstr* is the mission end field. The function call is AssembleMissionString(num, type, where, time1, time2, status, order_type, finalstr). Table 2.19-65 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
num	short	Standard C type.
type	Ptr	Development:THINK C: Mac #includes:MacTypes.h
where	Ptr	Development:THINK C: Mac #includes:MacTypes.h
time1	Ptr	Development:THINK C: Mac #includes:MacTypes.h
time2	Ptr	Development:THINK C: Mac #includes:MacTypes.h
status	Ptr	Development:THINK C: Mac #includes:MacTypes.h
order_type	Ptr	Development:THINK C: Mac #includes:MacTypes.h
finalstr	Str255	Development:THINK C: Mac #includes:MacTypes.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
tempStr	Str255	Development:THINK C: Mac #includes:MacTypes.h
i	short	Standard C type.
Calls		
Function	**Where Described**	
NumToString	Standard Binary to Decimal Conversion Package function for Macintosh.	
stccpy	See Section 2.19.3.9.6.	
Called By		
Function	**Where Described**	
MissionObj::edit	See Section 2.19.5.7.2.	
MissionObj::NewMission	See Section 2.19.5.7.3.	
MissionObj::show	See Section 2.19.5.7.4.	
MissionObj::UpdateMissionString	See Section 2.19.5.7.5.	

Table 2.19-65 AssembleMissionString Information.

2.19.3.9.2 AssembleResourceString

AssembleResourceString sets up fields in the Resource Table rows so that the columns line up properly. *a* is the resource vehicle number field; *b* is the resource location; *c* is the resource speed field; *d* is the resource status field; *finalstr* is the resource end field. The function call is AssembleResourceString(a, b, c, d, finalstr). Table 2.19-66 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
a	Ptr	Development:THINK C: Mac #includes:MacTypes.h
b	Ptr	Development:THINK C: Mac #includes:MacTypes.h
c	Ptr	Development:THINK C: Mac #includes:MacTypes.h
d	Ptr	Development:THINK C: Mac #includes:MacTypes.h
finalstr	Str255	Development:THINK C: Mac #includes:MacTypes.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
tempStr	Str255	Development:THINK C: Mac #includes:MacTypes.h
i	short	Standard C type.
Calls		
Function	**Where Described**	
stccpy	See Section 2.19.3.9.6.	
Called By		
Function	**Where Described**	
ResourceObj::show	See Section 2.19.5.9.13.	

Table 2.19-66 AssembleResourceString Information.

2.19.3.9.3 ReplaceMissionField

ReplaceMissionField replaces the field specified by *num* in the Mission Table with the text specified in *text*. The function call is ReplaceMissionField(num, text, finalstr). Table 2.19-67 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
num	short	Standard C type.
text	Ptr	Development:THINK C: Mac #includes:MacTypes.h
finalstr	Str255	Development:THINK C: Mac #includes:MacTypes.h

Calls	
Function	**Where Described**
ClearField	See Section 2.19.3.9.5.
stccpy	See Section 2.19.3.9.6.
Called By	
Function	**Where Described**
MissionObj::SetStatus	See Section 2.19.5.7.20.
MissionObj::SendEmplacement	See Section 2.19.5.7.27.
MissionObj::SendBreach	See Section 2.19.5.7.28.

Table 2.19-67 ReplaceMissionField Information.

2.19.3.9.4 MarkField

MarkField places a period at the end of the field specified by *num*. The function call is MarkField(num, finalstr). Table 2.19-68 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
num	short	Standard C type.
finalstr	Str255	Development:THINK C: Mac #includes:MacTypes.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
tempStr	Str255	Development:THINK C: Mac #includes:MacTypes.h
Calls		
Function	**Where Described**	
stccpy	See Section 2.19.3.9.6.	

Table 2.19-68 MarkField Information.

2.19.3.9.5 ClearField

ClearField is used to clear a field. The field specified by *num* is filled with blanks. The function call is ClearField(num, finalstr). Table 2.19-69 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
num	short	Standard C type.
finalstr	Str255	Development:THINK C: Mac #includes:MacTypes.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	short	Standard C type.

Called By	
Function	**Where Described**
ReplaceMissionField	See Section 2.19.3.9.3.

Table 2.19-69 ClearField Information.

2.19.3.9.6 stccpy

stccpy copies the C string pointed to by *p* into another C string pointed to by *c*. *num* is the number of characters in the string. The The function call is stccpy(s, p, num). Table 2.19-70 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
s	pointer to char	Standard C type.
p	pointer to char	Standard C type.
num	short	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	short	Standard C type.
Called By		
Function	**Where Described**	
AssembleMissionString	See Section 2.19.3.9.1.	
AssembleResourceString	See Section 2.19.3.9.2.	
ReplaceMissionField	See Section 2.19.3.9.3.	
MarkField	See Section 2.19.3.9.4.	

Table 2.19-70 stccpy Information.

2.19.3.10 Utils.c

Development:SIMNET:MCC:CEW:Utils.c

This file contains utility routines used by the CEW Workstation.

2.19.3.10.1 LocalRectToGlobal

LocalRectToGlobal changes the coordinates of the rectangle *arect* from local to global coordinates. The function call is LocalRectToGlobal(aRect). Table 2.19-71 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
arect	pointer to Rect	Development:THINK C: Mac #includes:MacTypes.h
Calls		
Function	**Where Described**	
LocalToGlobal	Standard Quickdraw function for Macintosh.	

Called By	
Function	**Where Described**
MissionButton	See Section 2.19.3.5.2.

Table 2.19-71 LocalRectToGlobal Information.

2.19.3.10.2 stxcpy

stxcpy copies the C string pointed to by *p* into another C string pointed to by *c*. *num* is the number of characters in the string. The function call is stxcpy(s, p, num). Table 2.19-72 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
s	pointer to char	Standard C type.
p	pointer to char	Standard C type.
num	short	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	short	Standard C type.
Called By		
Function	**Where Described**	
CheckList::AddRow	See Section 2.19.5.1.2.	
CheckList::SetRow	See Section 2.19.5.1.3.	
CheckList::GetRow	See Section 2.19.5.1.4.	

Table 2.19-72 stxcpy Information.

2.19.3.10.3 stpcpy

stpcpy converts the Pascal string pointed to by *p* into a C string pointed to by *c*. The function call is stpcpy(). Table 2.19-73 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
s	pointer to char	Standard C type.
p	pointer to char	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	short	Standard C type.
Called By		
Function	**Where Described**	
CheckUTMString	See Section 2.19.3.3.1.	
ResourceObj::init	See Section 2.19.5.9.1.	
ResourceObj::GetName	See Section 2.19.5.9.10.	

Table 2.19-73 stpcpy Information.

2.19.3.10.4 HiliteRect

HiliteRect causes a button represented by the rectangle *r* to change its appearance when clicked on by the user. The function call is HiliteRect(r). Table 2.19-74 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
r	pointer to Rect	Development:THINK C: Mac #includes:MacTypes.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
hitPattern	Pattern	Development:THINK C: Mac #includes:Quickdraw.h
Calls		
Function	**Where Described**	
PenMode	Standard Quickdraw function for Macintosh.	
PenPat	Standard Quickdraw function for Macintosh.	
PaintRect	Standard Quickdraw function for Macintosh.	
PenNormal	Standard Quickdraw function for Macintosh.	

Table 2.19-74 HiliteRect Information.

2.19.3.10.5 OutLineRect

OutLineRect causes a button represented by the rectangle *r* to become outlined. Generally, this routine is used to outline the OK button. The function call is OutLineRect(r). Table 2.19-75 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
r	pointer to Rect	Development:THINK C: Mac #includes:MacTypes.h
Calls		
Function	**Where Described**	
InsetRect	Standard Quickdraw function for Macintosh.	
PenNormal	Standard Quickdraw function for Macintosh.	
PenSize	Standard Quickdraw function for Macintosh.	
FrameRoundRect	Standard Quickdraw function for Macintosh.	
Called By		
Function	**Where Described**	
PrintWindow	See Section 2.19.3.6.1.	
SetupResWindow	See Section 2.19.3.8.2.	
RefreshWindow	See Section 2.19.3.8.3.	

Table 2.19-75 OutLineRect Information.

2.19.3.10.6 GetCurrentMonth

GetCurrentMonth fills in the parameter *astr* with the current month in a text string format. The function call is GetCurrentMonth(astr). Table 2.19-76 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
astr	register pointer to char	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
now	unsigned long	Standard C type.
dtg	DateTimeGroup	Development:SIMNET:libmac:dtg.h
str	array of 20 char	Standard C type.
Calls		
Function	**Where Described**	
GetDateTime	Standard Operating System Utility function for Macintosh.	
DTGElapsed	See Section 2.22.1.15.3.	
DTGToString	See Section 2.22.1.15.1.	
Called By		
Function	**Where Described**	
CEStringToDTG	See Section 2.19.3.10.7.	

Table 2.19-76 GetCurrentMonth Information.

2.19.3.10.7 CEStringToDTG

CEStringToDTG converts a Combat Engineering Workstation date/time string to a DTG format. *cp* points to the CEW date/time string; *dtg* is filled in with the CEW date in Date Time Group format. The function call is CEStringToDTG(cp, dtg). Table 2.19-77 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
cp	register pointer to char	Standard C type.
dtg	pointer to DateTimeGroup	Development:SIMNET:libmac:dtg.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
tempStr	array of 100 char	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
StringToDTG(tempStr, dtg)	short	The CEW date converted to DTG

Calls	
Function	**Where Described**
GetCurrentMonth	See Section 2.19.3.10.6.
StringToDTG	See Section 2.22.1.15.2.

Table 2.19-77 CEDStringToDTG Information.

2.19.3.10.8 CEDTGToString

CEDTGToString converts a Combat Engineering Workstation date/time in DTG format to a string. *dtg* points to the CEW date in Date Time Group format; *cp* is filled in with the CEW date/time string. The function call is CEDTGToString(dtg, str). Table 2.19-78 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
dtg	register pointer to DateTimeGroup	Development:SIMNET:libmac:dtg.h
str	register pointer to char	Standard C type.
Called By		
Function	**Where Described**	
MissionObj::edit	See Section 2.19.5.7.2.	
MissionObj::NewMission	See Section 2.19.5.7.3.	
MissionObj::show	See Section 2.19.5.7.4.	
MissionObj::UpdateMissionString	See Section 2.19.5.7.5.	

Table 2.19-78 CEDTGToString Information.

2.19.4 Minefield Geometry

2.19.4.1 poly_area.c

Development:SIMNET:MCC:CEW:poly_area.c

This file contains routines used to calculate the area of a user created minefield, given the coordinates of the polygon's vertices.

Variables		
Variable	**Type**	**Where Typedef Declared**
errString	Str255	Development:THINK C: Mac #includes:MacTypes.h

Table 2.19-79 poly_area.c Variable Information.

2.19.4.1.1 poly_area

poly_area computes the area of a polygon. This routine sets up a data structure called *new_verts*, then calls the routine poly_area_loc() to perform the area calculation. *num_verts* is the number of vertices; *verts* is the array of vertices (x,y) coordinates; *areap* is filled in with the area by poly_area_loc(). It is assumed that the distinct vertices are listed

without repeating the first one at the end (*verts*[0] to *verts*[*num_verts* - 1], with the last vertex not equal to the first). It is also assumed that the polygon has no self-shortersections. Note that this routine does not work well if the polygon has vertices in a row. The function call is poly_area(verts, num_verts, areap). Table 2.19-80 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
verts	pointer to LongPt	Development:SIMNET:libmac:longpt.h
num_verts	short	Standard C type.
areap	pointer to double	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
code	short	Standard C type.
i	register short	Standard C type.
new_verts	pointer to LongPt	Development:SIMNET:libmac:longpt.h
Return Values		
Return Value	**Type**	**Meaning**
1	short	unsuccessful
code	short	If equal to 0, then successful; If equal to 1, then unsuccessful.
Calls		
Function	**Where Described**	
NewPtr	Standard Memory Manager function for Macintosh.	
Notify	See Section 2.19.3.4.4.	
poly_area_loc	See Section 2.19.4.1.2.	
Called By		
Function	**Where Described**	
DetermineMines	See Section 2.19.3.2.14.	

Table 2.19-80 poly_area Information.

2.19.4.1.2 poly_area_loc

poly_area_loc calculates the area of the polygon, where *verts* is the array of vertices (x,y) coordinates and *num_verts* is the number of vertices. *areap* is filled in with the area of the polygon. The routine first checks that the parameters define an actual polygon. If the polygon is a triangle (*num_verts* = 3), its area is calculated and filled into *areap*.

If the polygon is made up of more than three vertices, the routine calculates its area as follows: A triangle is created from the first three vertices: *verts*[*vert* -1], *verts*[*vert*], and *verts*[*vert* + 1]. If no other vertex of the polygon is within this triangle and the triangle is located on the inside of the polygon, the triangle is removed from the polygon, and the area of the triangle is added to *areap*. The vertex *verts*[*vert*] is removed from the original polygon, leaving a new, smaller polygon. The area of this polygon is reduced by recursively calling poly_area_loc(). If the reduction is successful, the routine returns 0 and

areap contains the area of the original polygon. If the reduction is unsuccessful, 1 is returned.

The function call is poly_area_loc(verts, num_verts, areap). Table 2.19-81 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
verts	pointer to LongPt	Development:SIMNET:libmac:longpt.h
num_verts	short	Standard C type.
areap	pointer to double	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
vert	register short	Standard C type.
i	register short	Standard C type.
vert_inside	short	Standard C type.
x_max	long	Standard C type.
y_max	long	Standard C type.
inpt	LongPt	Development:SIMNET:libmac:longpt.h
outpt	LongPt	Development:SIMNET:libmac:longpt.h
triangle	array of LongPt	Development:SIMNET:libmac:longpt.h
area_incr	double	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
1	short	Unsuccessful
0	short	Successful
poly_area_loc(verts, num_verts -1, areap)	short	If equal to 0, the reduction was successful; If equal to 1, the reduction was unsuccessful.
Calls		
Function	**Where Described**	
AREA_TRIANGLE	Macro defined in Development:SIMNET:CEW:poly_area.h.	
Notify	See Section 2.19.3.4.4.	
inside_poly	See Section 2.19.4.1.3.	
poly_area_loc	See Section 2.19.4.1.2.	
Called By		
Function	**Where Described**	
poly_area_loc	See Section 2.19.4.1.2.	
poly_area	See Section 2.19.4.1.1.	

Table 2.19-81 poly_area_loc Information.

2.19.4.1.3 inside_poly

inside_poly determines whether a point lies inside or outside a polygon. *inptp* is the point being tested; *outpt* is a point known to be outside the polygon. The polygon is formed by the array of vertices coordinates *verts*, with the number of vertices equal to *num_verts*. The function call is inside_poly(inptp, outptp, verts, num_verts). Table 2.19-82 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
inptp	pointer to LongPt	Development:SIMNET:libmac:longpt.h
outptp	pointer to LongPt	Development:SIMNET:libmac:longpt.h
verts	pointer to LongPt	Development:SIMNET:libmac:longpt.h
num_verts	short	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	register short	Standard C type.
params	segintparams	Development:SIMNET:CEW:seg_intrsct.h
code_cnt	array of NUM_CODES short	Standard C type.
x_max	long	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
params.code_num_arr[MID_CROSS] & 1	short	The number of MID_CROSS intersections found: If greater than 0, the point lies inside the polygon. If equals 0, the point lies outside the polygon.
Calls		
Function	**Where Described**	
seg_intrsct	See Section 2.19.4.3.1.	
Notify	See Section 2.19.3.4.4.	
Called By		
Function	**Where Described**	
poly_area_loc	See Section 2.19.4.1.2.	

Table 2.19-82 inside_poly Information.

2.19.4.2 poly_area.h

Development:SIMNET:MCC:CEW:poly_area.h

This file contains definitions used within the file poly_area.c. AREA_TRIANGLE is defined to compute the area of a triangle. MAX_NUM_VERTS is defined as the maximum number of vertices in a polygon whose area is calculated by poly_area().

2.19.4.3 seg_intrsct.c

Development:SIMNET:MCC:CEW:seg_intrsct.c

This file contains a routine used to determine whether a line segment crosses or inappropriately touches (i.e. in the middle) any of an array of previous line segments. The routine works with points whose coordinates are long.

2.19.4.3.1 seg_intrsct

seg_intrsct computes information on the intersection of a line segment with an array of connected line segments. The parameter *paramsp* is a segintparams structure formed of the members described in Section 2.19.4.4. *paramsp->pt0p* to *paramsp->pt1p* with the connected line segments array *paramsp->pt_arr[i]* to *paramsp->pt_arr[i + 1]*, where i is equal to 0 to *paramsp->num_pt_arr* - 2. (i.e. the connected points from *paramsp->pt_arr[0]* to *paramsp->pt_arr[num_pt_arr* - 1]. The number of points, *paramsp->num_pt_arr*, must be greater than or equal to 2. For computing points of intersection, *paramsp->size_out_ptsp* should be at least 2, in case of segment overlap. The function call is seg_intrsct(paramsp). Table 2.19-83 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
paramsp	pointer to segintparams	Development:SIMNET:CEW: seg_intrsct.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
use_x	register short	Standard C type.
x_diff	register short	Standard C type.
y_diff	register short	Standard C type.
i	register short	Standard C type.
slope	register double	Standard C type.
code	register short	Standard C type.
abort_code	register short	Standard C type.
num_code	register short	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
DUMMY	short	Unsuccessful
code	short	An intersection code. See return values of check_x_ints and check_y_ints for descriptions.
Calls		
Function	**Where Described**	
check_x_ints	See Section 2.19.4.3.2.	
print_int_code	See Section 2.19.4.3.5.	
check_y_ints	See Section 2.19.4.3.3.	

Called By	
Function	**Where Described**
isLegalMinefield	See Section 2.19.3.2.2.
inside_poly	See Section 2.19.4.1.3.

Table 2.19-83 seg_intrsct Information.

2.19.4.3.2 check_x_ints

check_x_ints checks for an intersection of finite line segments. The two segments are defined by the input coordinates *pt0p* to *pt1p* and *pt2p* to *pt3p*. The slope of the *pt0p* to *pt1p* segment is *slope* (dy/dx), and is assumed to be between -1 and 1. A description code of the type of intersection between the segments is returned. *outcode* contains the list of the intersection codes for which all intersection points are to be reported; *outp* and *outp2* are arrays of intersection points, coded by the type of intersection of the given point; *numoutp* is the number of coded_lpoints that are valid intersection points. The function call is check_x_ints(pt0p, pt1p, slope, pt2p, pt3p, outcode, outp, outp2, numoutp). Table 2.19-84 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
pt0p	pointer to Long_Point	Development:SIMNET:CEW: seg_intrsct.h
pt1p	pointer to Long_Point	Development:SIMNET:CEW: seg_intrsct.h
slope	double	Standard C type.
pt2p	pointer to Long_Point	Development:SIMNET:CEW: seg_intrsct.h
pt3p	pointer to Long_Point	Development:SIMNET:CEW: seg_intrsct.h
outcode	short	Standard C type.
outp	pointer to coded_lpoint	Development:SIMNET:CEW: seg_intrsct.h
outp2	pointer to coded_lpoint	Development:SIMNET:CEW: seg_intrsct.h
numoutp	pointer to short	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
p_leftp	pointer to Long_Point	Development:SIMNET:CEW: seg_intrsct.h
p_rightp	pointer to Long_Point	Development:SIMNET:CEW: seg_intrsct.h
p1_leftp	pointer to Long_Point	Development:SIMNET:CEW: seg_intrsct.h
p1_rightp	pointer to Long_Point	Development:SIMNET:CEW: seg_intrsct.h
dist1	double	Standard C type.
dist2	double	Standard C type.
delta_x	double	Standard C type.
slope2	double	Standard C type.

Return Values		
Return Value	**Type**	**Meaning**
ON_ONE_SIDE	short	The segments miss; separated by x or y.
BOTH_ON_LINE_ABUTTING	short	The given segment is colinear to another and the vertices touch.
BOTH_ON_LINE_OVERLAPPING	short	The segments are colinear and the vertices do not touch
ON_LINE_TO_SIDE	short	A vertex is on an extended line, but not on the segment; there is no intersection.
ON_LINE_ABUTTING	short	One pair of vertices touch
ON_LINE_MID_TOUCH	short	One vertex touches the middle of the given semgnet
MID_CROSS	short	The line segments cross in the middle.
MID_ABUTTING	short	The given segment is abutted by the other.
Calls		
Function	**Where Described**	
POINT_TO_LINE	Macro defined Development:SIMNET:CEW:seg_intrsct.h.	
Called By		
Function	**Where Described**	
seg_intrsct	See Section 2.19.4.3.1.	

Table 2.19-84 check_x_ints Information.

2.19.4.3.3 check_y_ints

check_y_ints checks for an intersection of finite line segments. The two segments are defined by the input parameters *pt0p* to *pt1p* and *pt2p* to *pt3p*. The slope of the *pt0p* to *pt1p* segment is *slope* (dx/dy), and is assumed to be between -1 and 1. A description code of the type of intersection between the segments is returned. *outcode* contains the list of the intersection codes for which all intersection points are to be reported; *outp* and *outp2* are arrays of intersection points, coded by the type of intersection of the given point; *numoutp* is the number of coded_lpoints that are valid intersection points. The function call is check_y_ints(pt0p, pt1p, slope, pt2p, pt3p, outcode, outp, outp2, numoutp). Table 2.19-85 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
pt0p	pointer to Long_Point	Development:SIMNET:CEW: seg_intrsct.h
pt1p	pointer to Long_Point	Development:SIMNET:CEW: seg_intrsct.h
slope	double	Standard C type.
pt2p	pointer to Long_Point	Development:SIMNET:CEW: seg_intrsct.h
pt3p	pointer to Long_Point	Development:SIMNET:CEW: seg_intrsct.h
outcode	short	Standard C type.
outp	pointer to coded_lpoint	Development:SIMNET:CEW: seg_intrsct.h
outp2	pointer to coded_lpoint	Development:SIMNET:CEW: seg_intrsct.h
numoutp	pointer to short	Standard C type.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
p_leftp	pointer to Long_Point	Development:SIMNET:CEW: seg_intrsct.h
p_rightp	pointer to Long_Point	Development:SIMNET:CEW: seg_intrsct.h
p1_leftp	pointer to Long_Point	Development:SIMNET:CEW: seg_intrsct.h
p1_rightp	pointer to Long_Point	Development:SIMNET:CEW: seg_intrsct.h
dist1	double	Standard C type.
dist2	double	Standard C type.
delta_y	double	Standard C type.
slope2	double	Standard C type.

Return Values		
Return Value	**Type**	**Meaning**
ON_ONE_SIDE	short	The segments miss; separated by x or y.
BOTH_ON_LINE_ABUTTING	short	The given segment is colinear to another and the vertices touch.
BOTH_ON_LINE_OVERLAPPING	short	The segments are colinear and the vertices do not touch
ON_LINE_TO_SIDE	short	A vertex is on an extended line, but not on the segment; there is no intersection.
ON_LINE_ABUTTING	short	One pair of vertices touch
ON_LINE_MID_TOUCH	short	One vertex touches the middle of the given semgnet
MID_CROSS	short	The line segments cross in the middle.
MID_ABUTTING	short	The given segment is abutted by the other.

Calls	
Function	**Where Described**
POINT_TO_LINE	Macro defined Development:SIMNET:CEW:seg_intrsct.h.
Called By	
Function	**Where Described**
seg_intrsct	See Section 2.19.4.3.1.

Table 2.19-85 check_y_ints Information.

2.19.4.3.4 reverse_param

reverse_param reverses the x and y coordinates for all member points in the *paramp* parameter. Member points include *pt0p*, *pt1p*, the points in the *pt_arr* array, and the points in the *outptsp* array. The function call is reverse_param(paramp). Table 2.19-86 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
paramp	pointer to segintparams	Development:SIMNET:CEW: seg_intrsct.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	register short	Standard C type.
Calls		
Function	**Where Described**	
REVERSE_COORDS	Macro defined Development:SIMNET:CEW:seg_intrsct.h.	

Table 2.19-86 reverse_param Information.

2.19.4.3.5 print_int_code

print_int_code prints the intersection code description specified by the parameter *n*. This function is only compiled if DEBUG is 1. The function call is print_int_code(n). Table 2.19-87 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
n	short	Standard C type.
Called By		
Function	**Where Described**	
seg_intrsct	See Section 2.19.4.3.1.	

Table 2.19-87 print_int_code Information.

2.19.4.4 seg_intrsct.h

Development:SIMNET:MCC:CEW:seg_intrsct.h

This file contains macros used in the file seg_intrsct.c. It also contains the definitions of the segment intersect codes. Structures are defined for the *segparams* type, the *point* type, the *Long_Point* type, and the *coded_1point* type.

The *segparams* type structure is composed of the following members:

pt0p	-- One point of the segment to be tested.
pt1p	-- The second point of the segment to be tested.
pt_arr	-- The connected segments to be tested against.
num_pt_arr	-- The number of points in the array *pt_arr*.
abort_code	-- The intersection codes on which seg_intrsct() should abort.
num_code	-- The intersection codes which seg_intrsct() should report.
code_num_arr	-- An array allocated by the caller and intialized to 0. On return it contains the number of intersections found with each code. For example, *code_num_arr[MID_CROSS]* contains the number of MIS_CROSS intersections found.
out_pt_code	-- The intersection codes for which all intersection points should be reported.
size_out_ptsp	-- The number of points in the *coded_1point* array, allocated by the caller.
out_ptsp	-- The array of intersection points, coded by the type of intersection of the given point.
num_out_pts	-- The number of *coded_1point*s that are valid intersection points on return of seg_intrsct().

2.19.5 CEW Objects

2.19.5.1 CheckList.c

Development:SIMNET:MCC:CEW:CheckList.c

This file contains routines for setting up the CheckLists. CheckLists are windows containing lists of items with check boxes. An item in the list is selected by clicking in the check box.

2.19.5.1.1 CheckList::init

CheckList::init allocates and initalizes an instance of a CheckList object. *window* is the window in which the list appears; *r* is the rectangle containing the list's viewable content. The function call is CheckList::init(WindowPtr window, Rect r). Table 2.19-88 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
window	WindowPtr	Development:THINK C: Mac #includes:WindowMgr.h
r	Rect	Development:THINK C: Mac #includes:MacTypes.h

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
cell_size	Cell	Development:THINK C: Mac #includes:ListMgr.h
datab	Rect	Development:THINK C: Mac #includes:MacTypes.h
listRect	Rect	Development:THINK C: Mac #includes:MacTypes.h
Return Values		
Return Value	**Type**	**Meaning**
(Handle) this	Handle	A handle to the CheckList object created.
Calls		
Function	**Where Described**	
SetRect	Standard Quickdraw function for Macintosh.	
SetPt	Standard Quickdraw function for Macintosh.	
LNew	Standard List Manager function for Macintosh	
LSetSelect	Standard List Manager function for Macintosh	
LDoDraw	Standard List Manager function for Macintosh	
LUpdate	Standard List Manager function for Macintosh	
FrameRect	Standard Quickdraw function for Macintosh.	
Called By		
Function	**Where Described**	
ResourceItems	See Section 2.19.3.7.1.	

Table 2.19-88 CheckList::init Information.

2.19.5.1.2 CheckList::AddRow

CheckList::AddRow adds a row to the list. The row is filled in with information from the *data* parameter. The check box is set to *state*. The function call is CheckList::AddRow(state, data). Table 2.19-89 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
state	short	Standard C type.
data	Ptr	Development:THINK C: Mac #includes:MacTypes.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
cell_size	Cell	Development:THINK C: Mac #includes:ListMgr.h
dataPtr	Ptr	Development:THINK C: Mac #includes:MacTypes.h

Calls	
Function	**Where Described**
NewPtr	Standard Memory Manager function for Macintosh.
stxcpy	See Section 2.19.3.10.2.
SetPt	Standard Quickdraw function for Macintosh.
LAddRow	Standard List Manager function for Macintosh
LSetCell	Standard List Manager function for Macintosh
LDraw	Standard List Manager function for Macintosh
DisposPtr	Standard Memory Manager function for Macintosh.
Called By	
Function	**Where Described**
ResourceItems	See Section 2.19.3.7.1.

Table 2.19-89 CheckList::AddRow Information.

2.19.5.1.3 CheckList::SetRow

CheckList::SetRow fills in the specified *row* with the input *state* and *data* information. The function call is CheckList::SetRow(row, state, data). Table 2.19-90 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
row	short	Standard C type.
data	pointer to char	Standard C type.
state	short	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
cell_size	Cell	Development:THINK C: Mac #includes:ListMgr.h
dataPtr	Ptr	Development:THINK C: Mac #includes:MacTypes.h
Calls		
Function	**Where Described**	
NewPtr	Standard Memory Manager function for Macintosh.	
stxcpy	See Section 2.19.3.10.2.	
SetPt	Standard Quickdraw function for Macintosh.	
LSetCell	Standard List Manager function for Macintosh	
LDraw	Standard List Manager function for Macintosh	
DisposPtr	Standard Memory Manager function for Macintosh.	
Called By		
Function	**Where Described**	
SetState	See Section 2.19.5.1.5.	

Table 2.19-90 CheckList::SetRow Information.

2.19.5.1.4 CheckList::GetRow

CheckList::GetRow fills in the *data* and *state* pointers with the data and state information from the specified *row*. The function call is CheckList::GetRow(row, state, data). Table 2.19-91 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
row	short	Standard C type.
data	pointer to char	Standard C type.
state	pointer to short	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
dataPtr	Str255	Development:THINK C: Mac #includes:MacTypes.h
dataLen	short	Standard C type.
cell_size	Cell	Development:THINK C: Mac #includes:ListMgr.h
Calls		
Function	**Where Described**	
SetPt	Standard Quickdraw function for Macintosh.	
LGetCell	Standard List Manager function for Macintosh	
stxcpy	See Section 2.19.3.10.2.	
Called By		
Function	**Where Described**	
SetState	See Section 2.19.5.1.5.	
GetState	See Section 2.19.5.1.6.	

Table 2.19-91 CheckList::GetRow Information.

2.19.5.1.5 CheckList::SetState

CheckList::SetState sets the state of the specified *row* to *state*. Each row in the CheckList contains a data item and a check box. The item is selected by clicking in the check box; *state* tells whether the item is selected. The function call is CheckList::SetState(row, state). Table 2.19-92 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
state	short	Standard C type.
row	short	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
data	Str255	Development:THINK C: Mac #includes:MacTypes.h
oldstate	short	Standard C type.

Calls	
Function	**Where Described**
GetRow	See Section 2.19.5.1.4.
SetRow	See Section 2.19.5.1.3.
Called By	
Function	**Where Described**
ToggleState	See Section 2.19.5.1.7.
MissionObj::edit	See Section 2.19.5.7.2.
MissionObj::NewMission	See Section 2.19.5.7.3.
MissionObj::show	See Section 2.19.5.7.4.

Table 2.19-92 CheckList::SetState Information.

2.19.5.1.6 CheckList::GetState

CheckList::GetState returns the state of the specified *row* in the CheckList. Each row in the CheckList contains a data item and a check box. The item is selected by clicking in the check box; *state* tells whether the item is selected. The function call is CheckList::GetState(row). Table 2.19-93 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
row	short	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
data	Str255	Development:THINK C: Mac #includes:MacTypes.h
state	short	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
state	short	The state of the row (either selected or not)
Calls		
Function	**Where Described**	
GetRow	See Section 2.19.5.1.4.	

Called By	
Function	**Where Described**
setEmplacementTime	See Section 2.19.3.2.4.
CheckEmplacementDialog	See Section 2.19.3.2.5.
setBreachTime	See Section 2.19.3.2.6.
CheckBreachDialog	See Section 2.19.3.2.7.
CheckMoveDialog	See Section 2.19.3.2.8.
ResourceItems	See Section 2.19.3.7.1.
ToggleState	See Section 2.19.5.1.7.
MissionObj::edit	See Section 2.19.5.7.2.

Table 2.19-93 CheckList::GetState Information.

2.19.5.1.7 CheckList::ToggleState

CheckList::ToggleState changes the state of the specified *row*. If the item was selected, it is changed to not selected. If the item was not selected, it is changed to selected. The function call is CheckList::ToggleState(row). Table 2.19-94 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
row	short	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
x	short	Standard C type.
Calls		
Function	**Where Described**	
GetState	See Section 2.19.5.1.6.	
SetState	See Section 2.19.5.1.5.	
Called By		
Function	**Where Described**	
CheckList_Handler	See Section 2.19.3.7.2.	

Table 2.19-94 CheckList::ToggleState Information.

2.19.5.2 CheckList.h

Development:SIMNET:MCC:CEW:CheckList.h

This file contains the declaration of the CheckList class as a child (or derivation) of the TextList class.

2.19.5.3 ComWindows.c

Development:SIMNET:MCC:CEW:ComWindows.c

2.19.5.3.1 ComWindow::show

ComWindow::show makes the ComWindow visible by calling the Swindow class declaration member function show(). The Asset (or Resource) window is an example of a ComWindow. The function call is ComWindow::show(). Table 2.19-95 describes the functions called using this function.

Calls	
Function	**Where Described**
Swindow::show	See Section 2.19.5.13.2.
Called By	
Function	**Where Described**
main	See Section 2.19.2.1.1.
SetupResWindow	See Section 2.19.3.8.2.

Table 2.19-95 ComWindow::show Information.

2.19.5.3.2 ComWindow::AddHotSpot

ComWindow::AddHotSpot adds a hot spot to the ComWindow. A hot spot is an area in the window which performs a function when the user clicks in it (e.g. the OK button). *r* is the outline of the hot spot area; *func* contains the functionality of the hot spot. The function call is ComWindow::AddHotSpot(Rect r, ProcPtr func). Table 2.19-96 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
r	Rect	Development:THINK C: Mac #includes:MacTypes.h
func	ProcPtr	Development:THINK C: Mac #includes:MacTypes.h
Calls		
Function	**Where Described**	
SetRect	Standard Quickdraw function for Macintosh.	
Called By		
Function	**Where Described**	
PrintWindow	See Section 2.19.3.6.1.	
SetupResWindow	See Section 2.19.3.8.2.	

Table 2.19-96 ComWindow::AddHotSpot Information.

2.19.5.3.3 ComWindow::initial

ComWindow::initial allocates and initializes an instance of a ComWindow object. *ID* specifies the ID number of the window; *func* specifies the window's update function. The function call is ComWindow::initial(short ID, ProcPtr func). Table 2.19-97 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
ID	short	Standard C type.
func	ProcPtr	Development:THINK C: Mac #includes:MacTypes.h
Return Values		
Return Value	**Type**	**Meaning**
this	pointer to ComWindow	A handle to this instance of the ComWindow object.
Calls		
Function	**Where Described**	
Swindow::init	See Section 2.19.5.13.1.	
Called By		
Function	**Where Described**	
main	See Section 2.19.2.1.1.	
SetupResWindow	See Section 2.19.3.8.2.	

Table 2.19-97 ComWindow::initial Information.

2.19.5.3.4 ComWindow::CheckHotSpot

ComWindow::CheckHotSpot checks to see if the mouse point is in a hot spot. *pt* is the mouse point. The function call is ComWindow::CheckHotSpot(Point pt). Table 2.19-98 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
pt	Point	Development:THINK C: Mac #includes:MacTypes.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	short	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
-1	short	The mouse point is not in a hot spot.
i	short	The resource number of a hot spot.
Calls		
Function	**Where Described**	
PtInRect	Standard Quickdraw function for Macintosh.	

Called By	
Function	**Where Described**
MainEventLoop	See Section 2.19.2.1.2.
OtherEventLoop	See Section 2.19.3.8.7.

Table 2.19-98 ComWindow::CheckHotSpot Information.

2.19.5.4 ComWindows.h

Development:SIMNET:MCC:CEW:ComWindows.h

This file contains the definition of the HotSpot type structure and a declaration of the ComWindow class as a child of the Swindow class.

2.19.5.5 Dialog_Windows.c

Development:SIMNET:MCC:CEW:Dialog_Windows.c

This file contains implementation routines for Dialog Windows.

2.19.5.5.1 DialogWindow::init

DialogWindow::init ComWindow::initial allocates and initializes an instance of a DialogWindow object. *ID* specifies the ID number of the window; *func* specifies the window's check function; *hand* specifies the window's special handler function. The function call is DialogWindow::init(short ID, ProcPtr func, ProcPtr hand). Table 2.19-99 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
ID	short	Standard C type.
func	ProcPtr	Development:THINK C: Mac #includes:MacTypes.h
hand	ProcPtr	Development:THINK C: Mac #includes:MacTypes.h
Return Values		
Return Value	**Type**	**Meaning**
this	pointer to DialogWindow	A handle to this instance of the DialogWindow object.
Calls		
Function	**Where Described**	
NewPtr	Standard Memory Manager function for Macintosh.	
GetNewDialog	Standard Dialog Manager function for Macintosh.	
Called By		
Function	**Where Described**	
MacInits	See Section 2.19.2.1.3.	

Table 2.19-99 DialogWindow::init Information.

2.19.5.5.2 DialogWindow::handler

DialogWindow::handler handles events in DialogWindow objects. The function call is DialogWindow::handler(). Table 2.19-100 describes the internal variables used and functions called using this function.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
itemhit	short	Standard C type.
done	short	Standard C type.
lastitem	short	Standard C type.
aevent	EventRecord	Development:THINK C: Mac #includes:EventMgr.h
c	char	Standard C type.
type	short	Standard C type.
ltype	short	Standard C type.
h_item	Handle	Development:THINK C: Mac #includes:MacTypes.h
lh_item	Handle	Development:THINK C: Mac #includes:MacTypes.h
box	Rect	Development:THINK C: Mac #includes:MacTypes.h
lbox	Rect	Development:THINK C: Mac #includes:MacTypes.h
reset	Boolean	Development:THINK C: Mac #includes:MacTypes.h
special	short	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
special	short	A special handler function exists for this DialogWindow
itemhit	short	The resource id of the item which was hit.
Calls		
Function	**Where Described**	
SysBeep	Standard Operating System Utility function for Macintosh.	
SystemTask	Standard Desk Manager function for Macintosh.	
GetNextEvent	Standard Toolbox Event Manager function for Macintosh	
SpecialHandlerFunc		
CheckFunc		
IsDialogEvent	Standard Dialog Manager function for Macintosh.	
DialogSelect	Standard Dialog Manager function for Macintosh.	
GetDItem	Standard Dialog Manager function for Macintosh.	
Called By		
Function	**Where Described**	
activate	See Section 2.19.5.5.3.	

Table 2.19-100 DialogWindow::handler Information.

2.19.5.5.3 DialogWindow::activate

DialogWindow::activate brings the DialogWindow to the front, and makes it visible and usable. The function call is DialogWindow::activate(). Table 2.19-101 describes the internal variables used and functions called using this function.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
lastButton	short	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
lastButton	short	???
Calls		
Function	**Where Described**	
ShowWindow	Standard Window Manager function for Macintosh.	
HiliteWindow	Standard Window Manager function for Macintosh.	
SelectWindow	Standard Window Manager function for Macintosh.	
SetPort	Standard Quickdraw function for Macintosh.	
DialogOutlineItem	See Section 2.19.5.5.19.	
handler	See Section 2.19.5.5.2.	
close	See Section 2.19.5.5.6.	
Called By		
Function	**Where Described**	
ReviewSelection	See Section 2.19.3.2.15.	
MissionSelection	See Section 2.19.3.5.3.	
MissionObj::edit	See Section 2.19.5.7.2.	
MissionObj::show	See Section 2.19.5.7.4.	

Table 2.19-101 DialogWindow::activate Information.

2.19.5.5.4 DialogWindow::quit

DialogWindow::quit closes the dialog, cleaning up its memory allocations. The function call is DialogWindow::quit(). Table 2.19-102 describes the functions called using this function.

Calls	
Function	**Where Described**
CloseDialog	Standard Dialog Manager function for Macintosh.
DisposHandle	Standard Memory Manager function for Macintosh.
DisposPtr	Standard Memory Manager function for Macintosh.
Called By	
Function	**Where Described**
DeleteResourceLists	See Section 2.19.3.7.3.

Table 2.19-102 DialogWindow::quit Information.

2.19.5.5.6 DialogWindow::close

DialogWindow::close makes the DialogWindow not visible. The function call is DialogWindow::close(). Table 2.19-104 describes the functions called using this function.

Calls	
Function	**Where Described**
HideWindow	Standard Window Manager function for Macintosh.
Called By	
Function	**Where Described**
activate	See Section 2.19.5.5.3.

Table 2.19-104 DialogWindow::close Information.

2.19.5.5.7 DialogWindow::GetText

DialogWindow::GetText fills in the text string pointer, *text*, with the text value of *item*. The function call is DialogWindow::GetText(short item, char text). Table 2.19-105 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
item	short	Standard C type.
text	pointer to char	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
type	short	Standard C type.
h_item	Handle	Development:THINK C: Mac #includes:MacTypes.h
box	Rect	Development:THINK C: Mac #includes:MacTypes.h
Calls		
Function	**Where Described**	
GetDItem	Standard Dialog Manager function for Macintosh.	
GetIText	Standard Dialog Manager function for Macintosh.	
Called By		
Function	**Where Described**	
isLegalMinefield	See Section 2.19.3.2.2.	
setEmplacementTime	See Section 2.19.3.2.4.	
CheckEmplacementDialog	See Section 2.19.3.2.5.	
setBreachTime	See Section 2.19.3.2.6.	
CheckBreachDialog	See Section 2.19.3.2.7.	
CheckMoveDialog	See Section 2.19.3.2.8.	
DetermineMines	See Section 2.19.3.2.14.	
MissionObj::edit	See Section 2.19.5.7.2.	

Table 2.19-105 DialogWindow::GetText Information.

2.19.5.5.8 DialogWindow::SetText

DialogWindow::SetText sets the text value of *item* to the text string passed in *text*. The function call is DialogWindow::SetText(short item, char text). Table 2.19-106 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
item	short	Standard C type.
text	pointer to char	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
type	short	Standard C type.
h_item	Handle	Development:THINK C: Mac #includes:MacTypes.h
box	Rect	Development:THINK C: Mac #includes:MacTypes.h
Calls		
Function	**Where Described**	
GetDItem	Standard Dialog Manager function for Macintosh.	
SetIText	Standard Dialog Manager function for Macintosh.	
Called By		
Function	**Where Described**	
setEmplacementTime	See Section 2.19.3.2.4.	
CheckEmplacementDialog	See Section 2.19.3.2.5.	
setBreachTime	See Section 2.19.3.2.6.	
CheckBreachDialog	See Section 2.19.3.2.7.	
CheckMoveDialog	See Section 2.19.3.2.8.	
DetermineMines	See Section 2.19.3.2.14.	
MissionObj::edit	See Section 2.19.5.7.2.	
MissionObj::show	See Section 2.19.5.7.4.	

Table 2.19-106 DialogWindow::SetText Information.

2.19.5.5.9 DialogWindow::SelectText

DialogWindow::SelectText highlights the text in the given *item*. The function call is DialogWindow::SelectText(short item). Table 2.19-107 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
item	short	Standard C type.
Calls		
Function	**Where Described**	
SelIText	Standard Dialog Manager function for Macintosh.	

Called By	
Function	**Where Described**
CheckEmplacementDialog	See Section 2.19.3.2.5.
MissionObj::edit	See Section 2.19.5.7.2.

Table 2.19-107 DialogWindow::SelectText Information.

2.19.5.5.10 DialogWindow::GetValue

DialogWindow::GetValue returns the control value of the given *item*. The function call is DialogWindow::GetValue(short item). Table 2.19-108 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
item	short	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
type	short	Standard C type.
h_item	Handle	Development:THINK C: Mac #includes:MacTypes.h
box	Rect	Development:THINK C: Mac #includes:MacTypes.h
Return Values		
Return Value	**Type**	**Meaning**
GetCtlValue((ControlHandle) h_item)	short	The control value of the specified item.
Calls		
Function	**Where Described**	
GetDItem	Standard Dialog Manager function for Macintosh.	
GetCtlValue	Standard Control Manager function for Macintosh.	
Called By		
Function	**Where Described**	
CheckEmplacementDialog	See Section 2.19.3.2.5.	
DetermineMines	See Section 2.19.3.2.14.	
ReviewSelection	See Section 2.19.3.2.15.	
MissionObj::edit	See Section 2.19.5.7.2.	

Table 2.19-108 DialogWindow::GetValue Information.

2.19.5.5.11 DialogWindow::SetValue

DialogWindow::SetValue sets the control value of the specified item. *item* is the DialogWindow item; *value* is the control value to set. The function call is DialogWindow::SetValue(short item, short value). Table 2.19-109 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
item	short	Standard C type.
value	short	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
type	short	Standard C type.
h_item	Handle	Development:THINK C: Mac #includes:MacTypes.h
box	Rect	Development:THINK C: Mac #includes:MacTypes.h
Calls		
Function	**Where Described**	
GetDItem	Standard Dialog Manager function for Macintosh.	
SetCtlValue	Standard Control Manager function for Macintosh.	
Called By		
Function	**Where Described**	
CheckEmplacementDialog	See Section 2.19.3.2.5.	
CheckBreachDialog	See Section 2.19.3.2.7.	
CheckReviewDialog	See Section 2.19.3.2.13.	
ReviewSelection	See Section 2.19.3.2.15.	
MissionObj::edit	See Section 2.19.5.7.2.	
MissionObj::show	See Section 2.19.5.7.4.	

Table 2.19-109 DialogWindow::SetValue Information.

2.19.5.5.12 DialogWindow::HiliteItem

DialogWindow::HiliteItem highlights the value of the item. *item* is the DialogWindow item; *value* is the control value of *item*. The function call is DialogWindow::HiliteItem(short item, short value). Table 2.19-110 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
item	short	Standard C type.
value	short	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
type	short	Standard C type.
val	short	Standard C type.
h_item	Handle	Development:THINK C: Mac #includes:MacTypes.h
box	Rect	Development:THINK C: Mac #includes:MacTypes.h

Calls	
Function	**Where Described**
GetDItem	Standard Dialog Manager function for Macintosh.
HiliteControl	Standard Control Manager function for Macintosh.
Called By	
Function	**Where Described**
ReviewSelection	See Section 2.19.3.2.15.
MissionObj::show	See Section 2.19.5.7.4.

Table 2.19-110 DialogWindow::HiliteItem Information.

2.19.5.5.13 DialogWindow::SetUserItem

DialogWindow::SetUserItem sets the DialogWindow's user item to the specified handler routine. This user specified handler routine will be used to handle events while the dialog is active. *item* is the user item; *routine* is the handler routine. The function call is DialogWindow::SetUserItem(short item, ProcPtr routine). Table 2.19-111 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
item	short	Standard C type.
routine	ProcPtr	Development:THINK C: Mac #includes:MacTypes.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
type	short	Standard C type.
h_item	Handle	Development:THINK C: Mac #includes:MacTypes.h
box	Rect	Development:THINK C: Mac #includes:MacTypes.h
Calls		
Function	**Where Described**	
GetDItem	Standard Dialog Manager function for Macintosh.	
SetDItem	Standard Dialog Manager function for Macintosh.	
Called By		
Function	**Where Described**	
MacInits	See Section 2.19.2.1.3.	

Table 2.19-111 DialogWindow::SetUserItem Information.

2.19.5.5.14 DialogWindow::GetItemRect

DialogWindow::GetItemRect returns the box that describes the specified item. The function call is DialogWindow::GetItemRect(short item). Table 2.19-112 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
item	short	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
type	short	Standard C type.
h_item	Handle	Development:THINK C: Mac #includes:MacTypes.h
box	Rect	Development:THINK C: Mac #includes:MacTypes.h
Return Values		
Return Value	**Type**	**Meaning**
box	Rect	The box that describes the item.
Calls		
Function	**Where Described**	
GetDItem	Standard Dialog Manager function for Macintosh.	
Called By		
Function	**Where Described**	
ResourceItems	See Section 2.19.3.7.1.	

Table 2.19-112 DialogWindow::GetItemRect Information.

2.19.5.5.15 DialogWindow::equal

DialogWindow::equal determines if *other* is the same dialog as this object instance. The function call is DialogWindow::equal(WindowPtr other). Table 2.19-113 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
other	WindowPtr	Development:THINK C: Mac #includes:WindowMgr.h
Return Values		
Return Value	**Type**	**Meaning**
other == dialog	Boolean	TRUE if *other* is the same dialog; FALSE if *other* is not the same dialog.
Called By		
Function	**Where Described**	
ResourceItems	See Section 2.19.3.7.1.	

Table 2.19-113 DialogWindow::equal Information.

2.19.5.5.16 DialogWindow::GetDialogPtr

DialogWindow::GetDialogPtr returns the dialog pointer of this DialogWindow object. The function call is DialogWindow::GetDialogPtr(). Table 2.19-114 describes the values returned by this function.

Return Values		
Return Value	**Type**	**Meaning**
dialog	DialogPtr	The dialog pointer of this DialogWindow object.
Called By		
Function	**Where Described**	
CheckEmplacementDialog	See Section 2.19.3.2.5.	
CheckBreachDialog	See Section 2.19.3.2.7.	
CheckMoveDialog	See Section 2.19.3.2.8.	

Table 2.19-114 DialogWindow::GetDialogPtr Information.

2.19.5.5.17 DialogWindow::BringToFront

DialogWindow::BringToFront brings this DialogWindow to the front of the screen. The function call is DialogWindow::BringToFront(). Table 2.19-115 describes the functions called using this function.

Calls	
Function	**Where Described**
HiliteWindow	Standard Window Manager function for Macintosh.
SelectWindow	Standard Window Manager function for Macintosh.
SetPort	Standard Quickdraw function for Macintosh.

Table 2.19-115 DialogWindow::BringToFront Information.

2.19.5.5.18 DialogOutlineItem

DialogOutlineItem highlights the specified item in the specified dialog. *dialog* is the specified DialogWindow; *itemNo* is the item to highlight. The function call is DialogOutlineItem(dialog, itemNo). Table 2.19-116 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
dialog	register DialogPtr	Development:THINK C: Mac #includes:DialogMgr.h
itemNo	short	Standard C type.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	short	Standard C type.
theItem	Handle	Development:THINK C: Mac #includes:MacTypes.h
r	Rect	Development:THINK C: Mac #includes:MacTypes.h
cp	register pointer to char	Standard C type.
Calls		
Function	**Where Described**	
GetDItem	Standard Dialog Manager function for Macintosh.	
InsetRect	Standard Quickdraw function for Macintosh.	
GetHandleSize	Standard Memory Manager function for Macintosh.	
SetHandleSize	Standard Memory Manager function for Macintosh.	
MemError	Standard Memory Manager function for Macintosh.	
Called By		
Function	**Where Described**	
DialogWindow::activate	See Section 2.19.5.5.3.	

Table 2.19-116 DialogOutlineItem Information.

2.19.5.6 Dialog_Windows.h

Development:SIMNET:MCC:CEW:Dialog_Windows.h

This file contains the declaration of the DialogWindows class.

2.19.5.7 MissionObj.c

Development:SIMNET:MCC:CEW:MissionObj.c

This file implements the MissionObj objects.

Variables		
Variable	**Type**	**Where Typedef Declared**
emplace_dlog	extern pointer to DialogWindow	Development:SIMNET:CEW: Dialog_Windows.h
breach_dlog	extern pointer to DialogWindow	Development:SIMNET:CEW: Dialog_Windows.h
move_dlog	extern pointer to DialogWindow	Development:SIMNET:CEW: Dialog_Windows.h
stat_emplace	extern pointer to DialogWindow	Development:SIMNET:CEW: Dialog_Windows.h
stat_breach	extern pointer to DialogWindow	Development:SIMNET:CEW: Dialog_Windows.h
stat_move	extern pointer to DialogWindow	Development:SIMNET:CEW: Dialog_Windows.h
missionList	extern pointer to TextList	Development:SIMNET:CEW: TextList.h
breach_availList	pointer to CheckList	Development:SIMNET:CEW: CheckList.h
move_availList	pointer to CheckList	Development:SIMNET:CEW: CheckList.h
emplace_availList	pointer to CheckList	Development:SIMNET:CEW: CheckList.h
stat_emplace_availList	pointer to CheckList	Development:SIMNET:CEW: CheckList.h
stat_move_availList	pointer to CheckList	Development:SIMNET:CEW: CheckList.h
stat_breach_availList	pointer to CheckList	Development:SIMNET:CEW: CheckList.h
totalResources	extern short	Standard C type.
serviceq	extern pointer to ServiceObj	Development:SIMNET:CEW: ServiceObj.h
cewResources	extern pointer to array of ResourceObj	Development:SIMNET:CEW: ResourceObj.h
str_mission	pointer to array of char	Standard C type.

Table 2.19-117 MissionObj.c Variable Information.

2.19.5.7.1 MissionObj::init

MissionObj::init initializes and allocates an instance of the MissionObj Object. *ID* is an index into the Mission List. The function call is MissionObj::init(short ID). Table 2.19-118 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
ID	short	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
this	pointer to MissionObj	A handle to this instance of the MissionObj Object.

Called By	
Function	**Where Described**
MissionSelection	See Section 2.19.3.5.3.

Table 2.19-118 MissionObj::init Information.

2.19.5.7.2 MissionObj::edit

MissionObj::edit fills in the correct fields when the user edits a mission. The mission number, the start and finish times, the depth of the minefield, and the number of mines required are filled in. *edit_reason* is the reason for the edit (either because a new mission is being created, the user requested an edit through the Review dialog, or an asset was destroyed). The function call is MissionObj::edit(edit_reason). Table 2.19-119 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
edit_reason	short	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
tempStr	Str255	Development:THINK C: Mac #includes:MacTypes.h
aStr	Str255	Development:THINK C: Mac #includes:MacTypes.h
bStr	Str255	Development:THINK C: Mac #includes:MacTypes.h
tempLong	long	Standard C type.
tempInt	short	Standard C type.
lastButton	short	Standard C type.
i	short	Standard C type.
mf_pts	short	Standard C type.
the_dlog	register pointer to DialogWindow	Development:SIMNET:CEW: Dialog_Windows.h
currentTime	DateTimeGroup	Development:SIMNET:libmac: dtg.h
time_diff	long	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
FALSE	short	The mission was aborted.
TRUE	short	The mission is pending.

Calls	
Function	**Where Described**
NumToString	Standard Binary to Decimal Conversion Package function for Macintosh.
DialogWindow::SetText	See Section 2.19.5.5.8.
DialogWindow::SetValue	See Section 2.19.5.5.11.
DTGToString	See Section 2.22.1.15.1.
CheckList::SetState	See Section 2.19.5.1.5.
DialogWindow::SelectText	See Section 2.19.5.5.9.
DialogWindow::GetText	See Section 2.19.5.5.7.
DialogWindow::GetValue	See Section 2.19.5.5.10.
CheckList::GetState	See Section 2.19.5.1.6.
DialogWindow::activate	See Section 2.19.5.5.3.
StringToDTG	See Section 2.22.1.15.2.
Get_DTG	See Section 2.22.1.15.4.
DTGElapsed	See Section 2.22.1.15.3.
CEDTGToString	See Section 2.19.3.10.8.
AssembleMissionString	See Section 2.19.3.9.1.
TextList::SetRow	See Section 2.19.5.15.9.
SetStatus	See Section 2.19.5.7.20.
Called By	
Function	**Where Described**
ReviewSelection	See Section 2.19.3.2.15.
NewMission	See Section 2.19.5.7.3.
ServiceObj::CheckQs	See Section 2.19.5.11.4.

Table 2.19-119 MissionObj::edit Information.

2.19.5.7.3 MissionObj::NewMission

When the user selects the New Mission button and fills in the proper fields, MissionObj::NewMission sets up a new mission and adds it to the appropriate mission list for the *missionType* (either Emplacement, Breach, or Move). The mission number, the start and finish times, the depth of the minefield, and the number of mines required are filled in. The function call is MissionObj::NewMission(short missionType) Table 2.19-120 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
missionType	short	Standard C type.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
tempStr	Str255	Development:THINK C: Mac #includes:MacTypes.h
aStr	Str255	Development:THINK C: Mac #includes:MacTypes.h
bStr	Str255	Development:THINK C: Mac #includes:MacTypes.h
the_list	pointer to CheckList	Development:SIMNET:CEW: CheckList.h
currentTime	DateTimeGroup	Development:SIMNET:libmac: dtg.h
i	short	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
FALSE	short	The mission was not created.
TRUE	short	The mission was created.
Calls		
Function	**Where Described**	
Get_DTG	See Section 2.22.1.15.4.	
Notify	See Section 2.19.3.4.4.	
CheckList::SetState	See Section 2.19.5.1.5.	
edit	See Section 2.19.5.7.2.	
CEDTGToString	See Section 2.19.3.10.8.	
AssembleMissionString	See Section 2.19.3.9.1.	
TextList::AddRow	See Section 2.19.5.15.6.	
SetStatus	See Section 2.19.5.7.20.	
ServiceObj::AddService	See Section 2.19.5.11.2.	
Called By		
Function	**Where Described**	
MissionSelection	See Section 2.19.3.5.3.	

Table 2.19-120 MissionObj::NewMission Information.

2.19.5.7.4 MissionObj::show

MissionObj::show shows the current mission. *show_reason* is the reason the mission is being shown (either because the user asked to see the mission through the Review dialog, or a warning just came due). The function call is MissionObj::show(show_reason). Table 2.19-121 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
show_reason	short	Standard C type.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
tempStr	Str255	Development:THINK C: Mac #includes:MacTypes.h
aStr	Str255	Development:THINK C: Mac #includes:MacTypes.h
bStr	Str255	Development:THINK C: Mac #includes:MacTypes.h
i	short	Standard C type.
lastButton	short	Standard C type.
time_diff	long	Standard C type.
new_start	long	Standard C type.
new_startA	long	Standard C type.
new_end	long	Standard C type.
currentTime	DateTimeGroup	Development:SIMNET:libmac: dtg.h
the_dlog	register pointer to DialogWindow	Development:SIMNET:CEW: Dialog_Windows.h

Calls	
Function	**Where Described**
DialogWindow::HiliteItem	See Section 2.19.5.5.12.
DialogWindow::SetValue	See Section 2.19.5.5.11.
NumToString	Standard Binary to Decimal Conversion Package function for Macintosh.
DialogWindow::SetText	See Section 2.19.5.5.8.
DTGToString	See Section 2.22.1.15.1.
CheckList::SetState	See Section 2.19.5.1.5.
DialogWindow::activate	See Section 2.19.5.5.3.
Notify	See Section 2.19.3.4.4.
StringToDTG	See Section 2.22.1.15.2.
Get_DTG	See Section 2.22.1.15.4.
DTGElapsed	See Section 2.22.1.15.3.
CEDTGToString	See Section 2.19.3.10.8.
AssembleMissionString	See Section 2.19.3.9.1.
TextList::SetRow	See Section 2.19.5.15.9.
SetStatus	See Section 2.19.5.7.20.

Called By	
Function	**Where Described**
ReviewSelection	See Section 2.19.3.2.15.
ServiceObj::CheckQs	See Section 2.19.5.11.4.
ServiceObj::ShowQ	See Section 2.19.5.11.5.

Table 2.19-121 MissionObj::show Information.

2.19.5.7.5 MissionObj::UpdateMissionString

MissionObj::UpdateMissionString updates the text string describing the mission in the mission table. The function call is MissionObj::UpdateMissionString(). Table 2.19-122 describes the internal variable used and functions called using this function.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
retval	Str255	Development:THINK C: Mac #includes:MacTypes.h
aStr	Str255	Development:THINK C: Mac #includes:MacTypes.h
bStr	Str255	Development:THINK C: Mac #includes:MacTypes.h
tempStr	Str255	Development:THINK C: Mac #includes:MacTypes.h
Calls		
Function	**Where Described**	
CEDTGToString	See Section 2.19.3.10.8.	
AssembleMissionString	See Section 2.19.3.9.1.	
TextList::SetRow	See Section 2.19.5.15.9.	
SetStatus	See Section 2.19.5.7.20.	
GetStatus	See Section 2.19.5.7.21.	
Called By		
Function	**Where Described**	
ReviewSelection	See Section 2.19.3.2.15.	
ResetMissionTime	See Section 2.19.5.7.17.	

Table 2.19-122 MissionObj::UpdateMissionString Information.

2.19.5.7.6 MissionObj::GetStartMoveTime

MissionObj::GetStartMoveTime returns the start time of a move mission. The function call is MissionObj::GetStartMoveTime(). Table 2.19-122 describes the values returned by this function.

Return Values		
Return Value	**Type**	**Meaning**
startMoveTime.dtgElapsed	unsigned long	The time the vehicles are supposed to start moving.
Called By		
Function	**Where Described**	
ReviewSelection	See Section 2.19.3.2.15.	
ServiceObj::CheckQs	See Section 2.19.5.11.4.	

Table 2.19-123 MissionObj::GetStartMoveTime Information.

2.19.5.7.7 MissionObj::GetStartActivityTime

MissionObj::GetStartActivityTime returns the start time of an emplacement or breach mission. The function call is MissionObj::GetStartActivityTime(). Table 2.19-124 describes the values returned by this function.

Return Values		
Return Value	**Type**	**Meaning**
startActivityTime.dtgElapsed	unsigned long	The time an emplacement or breach mission is supposed to start.
Called By		
Function	**Where Described**	
ReviewSelection	See Section 2.19.3.2.15.	
ServiceObj::CheckQs	See Section 2.19.5.11.4.	

Table 2.19-124 MissionObj::GetStartActivityTime Information.

2.19.5.7.8 MissionObj::GetEndTime

MissionObj::GetEndTime returns the time the mission ends. The function call is MissionObj::GetEndTime(). Table 2.19-125 describes the values returned by this function.

Return Values		
Return Value	**Type**	**Meaning**
endTime.dtgElapsed	unsigned long	The end time of the mission.
Called By		
Function	**Where Described**	
ReviewSelection	See Section 2.19.3.2.15.	
ServiceObj::CheckQs	See Section 2.19.5.11.4.	

Table 2.19-125 MissionObj::GetEndTime Information.

2.19.5.7.9 MissionObj::SetStartMoveTime

MissionObj::SetStartMoveTime sets the start time of a move mission. *new_time* is the new start time. The function call is MissionObj::SetStartMoveTime(new_time). Table 2.19-126 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
new_time	unsigned long	Standard C type.
Calls		
Function	**Where Described**	
DTGElapsed	See Section 2.22.1.15.3.	

Called By	
Function	**Where Described**
ReviewSelection	See Section 2.19.3.2.15.

Table 2.19-126 MissionObj::SetStartMoveTime Information.

2.19.5.7.10 MissionObj::SetStartActivityTime

MissionObj::SetStartActivityTime sets the start time of an emplacement or breach mission. *new_time* is the new mission start time. The function call is MissionObj::SetStartActivityTime(new_time). Table 2.19-127 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
new_time	unsigned long	Standard C type.
Calls		
Function	**Where Described**	
DTGElapsed	See Section 2.22.1.15.3.	
Called By		
Function	**Where Described**	
ReviewSelection	See Section 2.19.3.2.15.	
ResetMissionTime	See Section 2.19.5.7.17.	

Table 2.19-127 MissionObj::SetStartActivityTime Information.

2.19.5.7.11 MissionObj::SetEndTime

MissionObj::SetEndTime sets the end time of a mission. *new_time* is the new mission end time. The function call is MissionObj::SetEndTime(new_time). Table 2.19-128 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
new_time	unsigned long	Standard C type.
Calls		
Function	**Where Described**	
DTGElapsed	See Section 2.22.1.15.3.	
Called By		
Function	**Where Described**	
ReviewSelection	See Section 2.19.3.2.15.	
ResetMissionTime	See Section 2.19.5.7.17.	

Table 2.19-128 MissionObj::SetEndTime Information.

2.19.5.7.12 MissionObj::GetNumber

MissionObj::GetNumber returns the index into the mission list for this mission. The function call is MissionObj::GetNumber(). Table 2.19-129 describes the values returned by this function.

Return Values		
Return Value	**Type**	**Meaning**
number	short	The index into the mission list.
Called By		
Function	**Where Described**	
ServiceObj::CheckQs	See Section 2.19.5.11.4.	

Table 2.19-129 MissionObj::GetNumber Information.

2.19.5.7.13 MissionObj::SetResStatus

MissionObj::SetResStatus sets the status of each resource assigned to this mission. *state* is the status to set. The function call is MissionObj::SetResStatus(state). Table 2.19-130 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
state	short	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	short	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
TRUE	short	Successful.
Calls		
Function	**Where Described**	
ResourceObj::SetStatus	See Section 2.19.5.9.11.	
Called By		
Function	**Where Described**	
ServiceObj::CheckQs	See Section 2.19.5.11.4.	

Table 2.19-130 MissionObj::SetResStatus Information.

2.19.5.7.14 MissionObj::UpdateResAssigned

MissionObj::UpdateResAssigned updates each resource assigned to this mission in the mission list. The function call is MissionObj::UpdateResAssigned(). Table 2.19-131 describes the internal variables used and functions called using this function.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	short	Standard C type.
Calls		
Function	**Where Described**	
ResourceObj::SetMission	See Section 2.19.5.9.6.	
Called By		
Function	**Where Described**	
SendStartMovement	See Section 2.19.5.7.24.	

Table 2.19-131 MissionObj::UpdateResAssigned Information.

2.19.5.7.15 MissionObj::CheckResStatus

MissionObj::CheckResStatus checks the status of each resource assigned to this mission. The function call is MissionObj::CheckResStatus(). Table 2.19-132 describes the internal variables used and functions called using this function.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	short	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
FALSE	short	One or more of the resources are not ready.
TRUE	short	All resources are ready.
Calls		
Function	**Where Described**	
ResourceObj::GetStatus	See Section 2.19.5.9.12.	
Called By		
Function	**Where Described**	
ServiceObj::CheckQs	See Section 2.19.5.11.4.	

Table 2.19-132 MissionObj::CheckResStatus Information.

2.19.5.7.16 MissionObj::UpdateResPosition

MissionObj::UpdateResPosition updates the position of each resource assigned to this mission. The function call is MissionObj::UpdateResPosition(). Table 2.19-133 describes the internal variables used and functions called using this function.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	short	Standard C type.
mp	MapCoordinates	Development:SIMNET:libmac:map.h
Calls		
Function	**Where Described**	
StringToMapCoordinates	See Section 2.22.1.26.1.	
ResourceObj::SetPosition	See Section 2.19.5.9.4.	
Called By		
Function	**Where Described**	
ServiceObj::CheckQs	See Section 2.19.5.11.4.	

Table 2.19-133 MissionObj::UpdateResPosition Information.

2.19.5.7.17 MissionObj::ResetMissionTime

MissionObj::ResetMissionTime recomputes the timing of a mission. The function call is MissionObj::ResetMissionTime(). Table 2.19-134 describes the internal variables used and functions called using this function.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
ttime	long	Standard C type.
the_list	pointer to CheckList	Development:SIMNET:CEW: CheckList.h
i	short	Standard C type.
x	short	Standard C type.
move_time	float	Standard C type.
activity_time	float	Standard C type.
end_pt	LongPt	Development:SIMNET:libmac: longpt.h
start_pt	LongPt	Development:SIMNET:libmac: longpt.h
string	Str255	Development:THINK C: Mac #includes:MacTypes.h
string2	Str255	Development:THINK C: Mac #includes:MacTypes.h
gemss_number	short	Standard C type.
m57_number	short	Standard C type.
cewPlatoon_number	short	Standard C type.
line_charge_number	short	Standard C type.
aStr	Str255	Development:THINK C: Mac #includes:MacTypes.h
bStr	Str255	Development:THINK C: Mac #includes:MacTypes.h
tempStr	Str255	Development:THINK C: Mac #includes:MacTypes.h
per_hr	float	Standard C type.
charges	float	Standard C type.
length	long	Standard C type.
oldftime	long	Standard C type.
newftime	long	Standard C type.
time_diff	long	Standard C type.
atime_elapsed	long	Standard C type.
percent_done	long	Standard C type.
dtg	DateTimeGroup	Development:SIMNET:libmac: dtg.h
current_time	long	Standard C type.

Calls	
Function	**Where Described**
ResourceObj::DistanceToSecs	See Section 2.19.5.9.9.
ResourceObj::GetStatus	See Section 2.19.5.9.12.
ResourceObj::GetKind	See Section 2.19.5.9.2.
UTMToLongPt	See Section 2.19.3.3.5.
DistBetween2Pts	See Section 2.22.1.23.2.
GetStatus	See Section 2.19.5.7.21.
SetStartActivityTime	See Section 2.19.5.7.10.
SetEndTime	See Section 2.19.5.7.11.
Notify	See Section 2.19.3.4.4.
Get_DTG	See Section 2.22.1.15.4.
UpdateMissionString	See Section 2.19.5.7.5.

Called By	
Function	**Where Described**
ProcessRequest	See Section 2.19.2.1.5.
ServiceObj::CheckQs	See Section 2.19.5.11.4.

Table 2.19-134 MissionObj::ResetMissionTime Information.

2.19.5.7.18 MissionObj::DispatchResources

MissionObj::DispatchResources sends a message to each resource assigned to this mission to start moving. Resources are dispatched and then arrive at their destinations; their travel is not simulated. The function call is MissionObj::DispatchResources(). Table 2.19-135 describes the internal variables used and functions called using this function.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	short	Standard C type.
Calls		
Function	**Where Described**	
SendDispatch	See Section 2.19.5.7.29.	
Called By		
Function	**Where Described**	
SendStartMovement	See Section 2.19.5.7.24.	

Table 2.19-135 MissionObj::DispatchResources Information.

2.19.5.7.19 MissionObj::ArrivalResources

MissionObj::ArrivalResources sends a message to each resource assigned to this mission to arrive at its destination. Resources are dispatched and then arrive at their destinations; their travel is not simulated. The function call is MissionObj::ArrivalResources(). Table 2.19-136 describes the internal variables used and functions called using this function.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	short	Standard C type.
Calls		
Function	**Where Described**	
SendArrival	See Section 2.19.5.7.30.	
Called By		
Function	**Where Described**	
SendStartActivity	See Section 2.19.5.7.25.	

Table 2.19-136 MissionObj::ArrivalResources Information.

2.19.5.7.20 MissionObj::SetStatus

MissionObj::SetStatus sets the status of this mission. *num* is the status code to set for the mission. A mission status can be either Aborted, Pending, Moving, In Progress, or Done. The function call is MissionObj::SetStatus(num). Table 2.19-137 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
num	short	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
data	Str255	Development:THINK C: Mac #includes:MacTypes.h
Calls		
Function	**Where Described**	
TextList::GetRow	See Section 2.19.5.15.10.	
ReplaceMissionField	See Section 2.19.3.9.3.	
TextList::SetRow	See Section 2.19.5.15.9.	
Called By		
Function	**Where Described**	
ReviewSelection	See Section 2.19.3.2.15.	
edit	See Section 2.19.5.7.2.	
NewMission	See Section 2.19.5.7.3.	
show	See Section 2.19.5.7.4.	
UpdateMissionString	See Section 2.19.5.7.5.	
ServiceObj::CheckQs	See Section 2.19.5.11.4.	

Table 2.19-137 MissionObj::SetStatus Information.

2.19.5.7.21 MissionObj::GetStatus

MissionObj::GetStatus returns the status code for this mission. The function call is MissionObj::GetStatus(). Table 2.19-138 describes the values returned by this function.

Return Values		
Return Value	**Type**	**Meaning**
status	short	This mission's status.
Called By		
Function	**Where Described**	
ProcessRequest	See Section 2.19.2.1.5.	
ReviewSelection	See Section 2.19.3.2.15.	
UpdateMissionString	See Section 2.19.5.7.5.	
ResetMissionTime	See Section 2.19.5.7.17.	
ServiceObj::CheckQs	See Section 2.19.5.11.4.	

Table 2.19-138 MissionObj::GetStatus Information.

2.19.5.7.22 MissionObj::GetType

MissionObj::GetType returns the type of this mission. Mission types can be either Emplacement, Breach, or Move. The function call is MissionObj::GetType(). Table 2.19-139 describes the values returned by this function.

Return Values		
Return Value	**Type**	**Meaning**
type	short	This mission's type.

Table 2.19-139 MissionObj::GetType Information.

2.19.5.7.23 MissionObj::GetOrderType

MissionObj::GetOrderType returns the order type of this mission. Missions can either be Warned or Execute. The function call is MissionObj::GetOrderType(). Table 2.19-140 describes the values returned by this function.

Return Values		
Return Value	**Type**	**Meaning**
order_type	short	This mission's order type.
Called By		
Function	**Where Described**	
ServiceObj::CheckQs	See Section 2.19.5.11.4.	

Table 2.19-140 MissionObj::GetOrderType Information.

2.19.5.7.24 MissionObj::SendStartMovement

MissionObj::SendStartMovement dispatches the resources for this move mission. The function call is MissionObj::SendStartMovement(). Table 2.19-141 describes the functions called using this function.

Calls	
Function	**Where Described**
UpdateResAssigned	See Section 2.19.5.7.14.
DispatchResources	See Section 2.19.5.7.18.
Called By	
Function	**Where Described**
ServiceObj::CheckQs	See Section 2.19.5.11.4.

Table 2.19-141 MissionObj::SendStartMovement Information.

2.19.5.7.25 MissionObj::SendStartActivity

MissionObj::SendStartActivity starts the mission activity when the resources have arrived. The function call is MissionObj::SendStartActivity(). Table 2.19-142 describes the functions called using this function.

Calls	
Function	**Where Described**
ArrivalResources	See Section 2.19.5.7.19.
Called By	
Function	**Where Described**
ServiceObj::CheckQs	See Section 2.19.5.11.4.

Table 2.19-142 MissionObj::SendStartActivity Information.

2.19.5.7.26 MissionObj::SendEndMission

MissionObj::SendEndMission ends the mission when the activity is over. The function call is MissionObj::SendEndMission(). Table 2.19-143 describes the functions called using this function.

Calls	
Function	**Where Described**
SendEmplacement	See Section 2.19.5.7.27.
SendBreach	See Section 2.19.5.7.28.
Called By	
Function	**Where Described**
ServiceObj::CheckQs	See Section 2.19.5.11.4.

Table 2.19-143 MissionObj::SendEndMission Information.

2.19.5.7.27 MissionObj::SendEmplacement

MissionObj::SendEmplacement sends information to the host process about the completed minefield emplacement. The function call is MissionObj::SendEmplacement(). Table 2.19-144 describes the internal variables used and functions called using this function.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
req	pointer to CEWEmplaceRequest	Development:SIMNET:CEW: CEW.h
rsp	CEWEmplaceResponse	Development:SIMNET:CEW: CEW.h
err	short	Standard C type.
i	short	Standard C type.
tempStr	Str255	Development:THINK C: Mac #includes:MacTypes.h

Calls	
Function	**Where Described**
NewPtr	Standard Memory Manager function for Macintosh.
UTMToLongPt	See Section 2.19.3.3.5.
ATPPut	See Section 2.22.1.3.3.
ShowCaution	See Section 2.22.1.4.1.
DisposPtr	Standard Memory Manager function for Macintosh.
TextList::GetRow	See Section 2.19.5.15.10.
ReplaceMissionField	See Section 2.19.3.9.3.
TextList::SetRow	See Section 2.19.5.15.9.
Called By	
Function	**Where Described**
SendEndMission	See Section 2.19.5.7.26.

Table 2.19-144 MissionObj::SendEmplacement Information.

2.19.5.7.28 MissionObj::SendBreach

MissionObj::SendBreach sends information to the host process about the completed minefield breach. The function call is MissionObj::SendBreach(). Table 2.19-145 describes the internal variables used and functions called using this function.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
req	pointer to CEWEmplaceRequest	Development:SIMNET:CEW: CEW.h
rsp	CEWEmplaceResponse	Development:SIMNET:CEW: CEW.h
err	short	Standard C type.
i	short	Standard C type.
tempStr	Str255	Development:THINK C: Mac #includes:MacTypes.h
Calls		
Function	**Where Described**	
NewPtr	Standard Memory Manager function for Macintosh	
UTMToLongPt	See Section 2.19.3.3.5.	
ATPPut	See Section 2.22.1.3.3.	
ShowCaution	See Section 2.22.1.4.1.	
DisposPtr	Standard Memory Manager function for Macintosh.	
TextList::GetRow	See Section 2.19.5.15.10.	
ReplaceMissionField	See Section 2.19.3.9.3.	
TextList::SetRow	See Section 2.19.5.15.9.	
Called By		
Function	**Where Described**	
SendEndMission	See Section 2.19.5.7.26.	

Table 2.19-145 MissionObj::SendBreach Information.

2.19.5.7.29 MissionObj::SendDispatch

MissionObj::SendDispatch sends a message to the host process saying that a resource has been dispatched. *num* specifies the resource. The function call is MissionObj::SendDispatch(num). Table 2.19-146 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
num	short	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
req	pointer to CEWEmplaceRequest	Development:SIMNET:CEW: CEW.h
rsp	CEWEmplaceResponse	Development:SIMNET:CEW: CEW.h
err	short	Standard C type.
tempStr	Str255	Development:THINK C: Mac #includes:MacTypes.h
Calls		
Function	**Where Described**	
NewPtr	Standard Memory Manager function for Macintosh.	
ResourceObj::GetCCVNumber	See Section 2.19.5.9.3.	
ATPPut	See Section 2.22.1.3.3.	
ShowCaution	See Section 2.22.1.4.1.	
DisposPtr	Standard Memory Manager function for Macintosh.	
Called By		
Function	**Where Described**	
DispatchResources	See Section 2.19.5.7.18.	

Table 2.19-146 MissionObj::SendDispatch Information.

2.19.5.7.30 MissionObj::SendArrival

MissionObj::SendArrival sends a message to the host process saying that a resource has arrived. *num* specifies the resource. The function call is MissionObj::SendArrival(num). Table 2.19-147 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
num	short	Standard C type.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
req	pointer to CEWEmplaceRequest	Development:SIMNET:CEW: CEW.h
rsp	CEWEmplaceResponse	Development:SIMNET:CEW: CEW.h
err	short	Standard C type.
coord	UTMCoordinates	Development:SIMNET:CEW: SimnetTypes.h
lp	LongPt	Development:SIMNET:libmac: longpt.h

Calls	
Function	**Where Described**
NewPtr	Standard Memory Manager function for Macintosh.
ResourceObj::GetCCVNumber	See Section 2.19.5.9.3.
ResourceObj::GetPosition	See Section 2.19.5.9.7.
ATPPut	See Section 2.22.1.3.3.
ShowCaution	See Section 2.22.1.4.1.
DisposPtr	Standard Memory Manager function for Macintosh.

Called By	
Function	**Where Described**
ArrivalResources	See Section 2.19.5.7.19.

Table 2.19-147 MissionObj::SendArrival Information.

2.19.5.8 MissionObj.h

Development:SIMNET:MCC:CEW:MissionObj.h

This file contains definitions of mission and order types, and edit and show reasons. It also contains the type definition of MinefieldType, and the declaration of the MissionObj class.

2.19.5.9 ResourceObj.c

Development:SIMNET:MCC:CEW:ResourceObj.c

This file implements the ResourceObj objects.

Variables		
Variable	**Type**	**Where Typedef Declared**
str_re]	pointer to array of char	Standard C type.

Table 2.19-148 ResourceObj.c Variable Information.

2.19.5.9.1 ResourceObj::init

ResourceObj::init initializes this resource object. The resource is given its initial attributes as follows: *akind* is the kind of resource; *a_name* is the name of the resource, *speed* is the maximum speed of the resource; *ccv* is the host value stored as the ccv number. The function call is ResourceObj::init(akind, a_name, speed, ccv). Table 2.19-149 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
akind	long	Standard C type.
a_name	Ptr	Development:THINK C: Mac #includes:MacTypes.h
speed	short	Standard C type.
ccv	long	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
temp	array of 255 char	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
this	pointer to ResourceObj	A handle to this instance of the ResourceObj object.
Calls		
Function	**Where Described**	
stpcpy	See Section 2.19.3.10.3.	
Called By		
Function	**Where Described**	
ProcessRequest	See Section 2.19.2.1.5.	
LoadCannedResources	See Section 2.19.3.1.2.	

Table 2.19-149 *ResourceObj::init* Information.

2.19.5.9.2 ResourceObj::GetKind

ResourceObj::GetKind returns the kind of this resource. The function call is ResourceObj::GetKind(). Table 2.19-150 describes the values returned by this function.

Return Values		
Return Value	**Type**	**Meaning**
kind	long	This resource's kind.
Called By		
Function	**Where Described**	
setEmplacementTime	See Section 2.19.3.2.4.	
CheckEmplacementDialog	See Section 2.19.3.2.5.	
setBreachTime	See Section 2.19.3.2.6.	
CheckBreachDialog	See Section 2.19.3.2.7.	
MissionObj::ResetMission Time	See Section 2.19.5.7.17.	

Table 2.19-150 ResourceObj::GetKind Information.

2.19.5.9.3 ResourceObj::GetCCVNumber

ResourceObj::GetCCVNumber returns the ccv number of this resource. The function call is ResourceObj::GetCCVNumber(). Table 2.19-151 describes the values returned by this function.

Return Values		
Return Value	**Type**	**Meaning**
ccv_number	long	This resource's ccv number.
Called By		
Function	**Where Described**	
ProcessRequest	See Section 2.19.2.1.5.	
MissionObj::SendDispatch	See Section 2.19.5.7.29.	
MissionObj::SendArrival	See Section 2.19.5.7.30.	

Table 2.19-151 ResourceObj::GetCCVNumber Information.

2.19.5.9.4 ResourceObj::SetPosition

ResourceObj::SetPosition sets the position of this resource. *location* is the (x,y) coordinate of the position. The routine checks the validity of the passed location. The function call is ResourceObj::SetPosition(location). Table 2.19-152 describes the parameters used, errors returned and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
location	LongPt	Development.SIMNET:libmac: longpt.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
err9	short	Standard C type.
tempstr	Str255	Development:THINK C: Mac #includes:MacTypes.h
Errors		
Error Name	**Reason for Error**	
badMapStringErr	???	
badMapLettersErr	???	
outOfGridZoneErr	???	
noGridZoneErr	???	
Calls		
Function	**Where Described**	
PointToMapCoordinates	See Section 2.22.1.26.2.	
Notify	See Section 2.19.3.4.4.	

Called By	
Function	**Where Described**
ProcessRequest	See Section 2.19.2.1.5.
LoadCannedResources	See Section 2.19.3.1.2.
MissionObj::UpdateRes Position	See Section 2.19.5.7.16.

Table 2.19-152 ResourceObj::SetPosition Information.

2.19.5.9.5 ResourceObj::GetMission

ResourceObj::GetMission returns the mission assignment for this resource. The function call is ResourceObj::GetMission(). Table 2.19-153 describes the values returned by this function.

Return Values		
Return Value	**Type**	**Meaning**
missionAssigned	short	The mission assignment of this resource.

Called By	
Function	**Where Described**
ProcessRequest	See Section 2.19.2.1.5.
SetupResWindow	See Section 2.19.3.8.2.

Table 2.19-153 ResourceObj::GetMission Information.

2.19.5.9.6 ResourceObj::SetMission

ResourceObj::SetMission assigns this resource to the mission specified by *m*. The function call is ResourceObj::SetMission(m). Table 2.19-154 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
m	short	Standard C type.

Called By	
Function	**Where Described**
MissionObj::UpdateRes Assigned	See Section 2.19.5.7.14

Table 2.19-154 ResourceObj::SetMission Information.

2.19.5.9.7 ResourceObj::GetPosition

ResourceObj::GetPosition fills in the parameter *pos* with the (x,y) coordinates of this resource's position. The function call is ResourceObj::GetPosition(pos). Table 2.19-155 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
pos	pointer to LongPt	Development:SIMNET:libmac: longpt.h
Called By		
Function	**Where Described**	
MissionObj::SendArrival	See Section 2.19.5.7.30.	

Table 2.19-155 ResourceObj::GetPosition Information.

2.19.5.9.8 ResourceObj::GetKph

ResourceObj::GetKph returns the travel speed of this resource (in km/hr). The function call is ResourceObj::GetKph(). Table 2.19-156 describes the values returned by this function.

Return Values		
Return Value	**Type**	**Meaning**
km_per_hr	short	The speed of this resource.

Table 2.19-156 ResourceObj::GetKph Information.

2.19.5.9.9 ResourceObj::DistanceToSecs

ResourceObj::DistanceToSecs calculates the length of time it will take this resource to travel to the passed UTM location, *location*. The function call is ResourceObj::DistanceToSecs(location). Table 2.19-157 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
location	UTMCoordinates	Development:SIMNET:CEW: SimnetTypes.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
start_pt	LongPt	Development:SIMNET:libmac: longpt.h
end_pt	LongPt	Development:SIMNET:libmac: longpt.h
meters_per_second	float	Standard C type.
dis	long	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
0L	unsigned long	The resource's current position is *location*.
dis	unsigned long	The time it will take this resource to travel to *location*.

Calls	
Function	**Where Described**
UTMToLongPt	See Section 2.19.3.3.5.
DistBetween2Pts	See Section 2.22.1.23.2.
Called By	
Function	**Where Described**
setBreachTime	See Section 2.19.3.2.6.
CheckMoveDialog	See Section 2.19.3.2.8.
MissionObj::ResetMission Time	See Section 2.19.5.7.17.

Table 2.19-157 ResourceObj::DistanceToSecs Information.

2.19.5.9.10 ResourceObj::GetName

ResourceObj::GetName fills in *a_name* with the vehicle name of this resource. The function call is ResourceObj::GetName(a_name). Table 2.19-158 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
a_name	Ptr	Development:THINK C: Mac #includes:MacTypes.h
Calls		
Function	**Where Described**	
stpcpy	See Section 2.19.3.10.3.	
Called By		
Function	**Where Described**	
ProcessRequest	See Section 2.19.2.1.5.	
ResourceItems	See Section 2.19.3.7.1.	

Table 2.19-158 ResourceObj::GetName Information.

2.19.5.9.11 ResourceObj::SetStatus

ResourceObj::SetStatus sets this resource's status to *stat*. The function call is ResourceObj::SetStatus(stat). Table 2.19-159 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
stat	short	Standard C type.
Called By		
Function	**Where Described**	
ProcessRequest	See Section 2.19.2.1.5.	
LoadCannedResources	See Section 2.19.3.1.2.	
MissionObj::SetResStatus	See Section 2.19.5.7.13.	

Table 2.19-159 ResourceObj::SetStatus Information.

2.19.5.9.12 ResourceObj::GetStatus

ResourceObj::GetStatus returns the status of this resource. The function call is ResourceObj::GetStatus(). Table 2.19-160 describes the values returned by this function.

Return Values		
Return Value	**Type**	**Meaning**
status	short	The status of this resource.
Called By		
Function	**Where Described**	
SetupResWindow	See Section 2.19.3.8.2.	
MissionObj::CheckResStatus	See Section 2.19.5.7.15.	
MissionObj::ResetMission Time	See Section 2.19.5.7.17.	

Table 2.19-160 ResourceObj::GetStatus Information.

2.19.5.9.13 ResourceObj::show

ResourceObj::show fills in the parameter *string* with relevent information about the position, status, and speed of this resource. The function call is ResourceObj::show(string). Table 2.19-161 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
string	pointer to char	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
tempStr	Str255	Development:THINK C: Mac #includes:MacTypes.h
Calls		
Function	**Where Described**	
NumToString	Standard Binary to Decimal Conversion Package function for Macintosh.	
AssembleResourceString	See Section 2.19.3.9.2.	
Called By		
Function	**Where Described**	
SetupResWindow	See Section 2.19.3.8.2.	

Table 2.19-161 ResourceObj::show Information.

2.19.5.10 ResourceObj.h

Development:SIMNET:MCC:CEW:ResourceObj.h

This file declares the ResourceObj object class.

2.19.5.11 ServiceObj.c
Development:SIMNET:MCC:CEW:ServiceObj.c

This file implements the ServiceObj objects.

2.19.5.11.1 ServiceObj::init

ServiceObj::init initializes and allocates an instance of a ServiceObj object. The ServiceObj object contains the queue of events. The function call is ServiceObj::init(). Table 2.19-162 describes the values returned by this function.

Return Values		
Return Value	**Type**	**Meaning**
this	pointer to ServiceObj	A handle to the ServiceObj object created.
Called By		
Function	**Where Described**	
MacInits	See Section 2.19.2.1.3.	

Table 2.19-162 ServiceObj::init Information.

2.19.5.11.2 ServiceObj::AddService

ServiceObj::AddService adds an event, *a_mission*, to the end of the ServiceList queue. The function call is ServiceObj::AddService(a_mission). Table 2.19-163 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
a_mission	pointer to MissionObj	Development:SIMNET:CEW: MissionObj.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
p	ServicePtr	Development:SIMNET:CEW: ServiceObj.h
q	ServicePtr	Development:SIMNET:CEW: ServiceObj.h
Calls		
Function	**Where Described**	
NewPtr	Standard Memory Manager function for Macintosh.	
CheckQs	See Section 2.19.5.11.4.	
Called By		
Function	**Where Described**	
MissionObj::NewMission	See Section 2.19.5.7.3.	

Table 2.19-163 ServiceObj::AddService Information.

2.19.5.11.3 ServiceObj::DeleteService

ServiceObj::DeleteService removes an event, *x*, from the ServiceList queue. The function call is ServiceObj::DeleteService(x). Table 2.19-164 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
x	pointer to MissionObj	Development:SIMNET:CEW: MissionObj.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
p	ServicePtr	Development:SIMNET:CEW: ServiceObj.h
q	ServicePtr	Development:SIMNET:CEW: ServiceObj.h
Calls		
Function	**Where Described**	
DisposPtr	Standard Memory Manager function for Macintosh.	
Called By		
Function	**Where Described**	
CheckQs	See Section 2.19.5.11.4.	

Table 2.19-164 ServiceObj::DeleteService Information.

2.19.5.11.4 ServiceObj::CheckQs

ServiceObj::CheckQs checks all the services on the queue for start times, resource arrival times, and completion.

If a scheduled mission's execution time has occurred, the routine checks the mission's order type. If the order type is Warn, the routine pops up a dialog to the user allowing the user to choose to either execute or abort the mission. If the order type is Execute, the routine executes the mission. If the resources allocated to the mission have had a change in status, the mission is aborted.

Pending missions are checked, and are removed from the ServiceList if they have completed. Move missions are checked, and their activity started if the resources have completed their travelling.

The function call is ServiceObj::CheckQs(). Table 2.19-165 describes the internal variables used and functions called using this function.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
p	ServicePtr	Development:SIMNET:CEW: ServiceObj.h
q	ServicePtr	Development:SIMNET:CEW: ServiceObj.h
time	unsigned long	Standard C type.
tempStr	Str255	Development:THINK C: Mac #includes:MacTypes.h
st	short	Standard C type.
Calls		
Function	**Where Described**	
GetDateTime	Standard Operating System Utility function for Macintosh.	
MissionObj::GetStartMoveTime	See Section 2.19.5.7.6.	
MissionObj::GetStatus	See Section 2.19.5.7.21.	
MissionObj::GetOrderType	See Section 2.19.5.7.23.	
MissionObj::GetNumber	See Section 2.19.5.7.12.	
Notify	See Section 2.19.3.4.4.	
MissionObj::show	See Section 2.19.5.7.4.	
MissionObj::CheckResStatus	See Section 2.19.5.7.15.	
MissionObj::SetStatus	See Section 2.19.5.7.20.	
MissionObj::edit	See Section 2.19.5.7.2.	
MissionObj::ResetMission Time	See Section 2.19.5.7.17.	
MissionObj::SendStartMovement	See Section 2.19.5.7.24.	
MissionObj::SetResStatus	See Section 2.19.5.7.13.	
MySound	See Section 2.19.3.4.5.	
MissionObj::GetEndTime	See Section 2.19.5.7.8.	
MissionObj::SendEndMission	See Section 2.19.5.7.26.	
DeleteService	See Section 2.19.5.11.3.	
MissionObj::GetStartActivityTime	See Section 2.19.5.7.7.	
MissionObj::UpdateRes Position	See Section 2.19.5.7.16.	
MissionObj::SendStartActivity	See Section 2.19.5.7.25.	
Called By		
Function	**Where Described**	
ReviewSelection	See Section 2.19.3.2.15.	
AddService	See Section 2.19.5.11.2.	
UpdateQs	See Section 2.19.5.11.6.	

Table 2.19-165 ServiceObj::CheckQs Information.

2.19.5.11.5 ServiceObj::ShowQ

ServiceObj::ShowQ is not currently called by the application.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
p	ServicePtr	Development:SIMNET:CEW: ServiceObj.h
q	ServicePtr	Development:SIMNET:CEW: ServiceObj.h
Calls		
Function	**Where Described**	
MissionObj::show	See Section 2.19.5.7.4.	

Table 2.19-166 ServiceObj::ShowQ Information.

2.19.5.11.6 ServiceObj::UpdateQs

ServiceObj::UpdateQs checks the service queues for any events occurring at the current time. The function call is ServiceObj::UpdateQs(). Table 2.19-167 describes the internal variables used and functions called using this function.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
now	unsigned long	Standard C type.
Calls		
Function	**Where Described**	
GetDateTime	Standard Operating System Utility function for Macintosh.	
CheckQs	See Section 2.19.5.11.4.	
Called By		
Function	**Where Described**	
MainEventLoop	See Section 2.19.2.1.2.	

Table 2.19-167 ServiceObj::UpdateQs Information.

2.19.5.12 ServiceObj.h

Development:SIMNET:MCC:CEW:ServiceObj.h

This file defines the Service structure type and declares the ServiceObj class.

2.19.5.13 Simple_Windows.c

Development:SIMNET:MCC:CEW:Simple_Window.c

This file implements the Swindow object utilities.

2.19.5.13.1 Swindow::init

Swindow::init initializes and allocates an instance of an Swindow object. *ID* specifies the ID number of the window; *func* specifies the simple window's update function. The function call is Swindow::init(short ID, ProcPtr func). Table 2.19-168 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
ID	short	Standard C type.
func	ProcPtr	Development:THINK C: Mac #includes:MacTypes.h
Return Values		
Return Value	**Type**	**Meaning**
this	pointer to Swindow	A handle to the Swindow object created.
Calls		
Function	**Where Described**	
NewPtr	Standard Memory Manager function for Macintosh.	
GetNewWindow	Standard Window Manager function for Macintosh.	
Called By		
Function	**Where Described**	
ComWindow::initial	See Section 2.19.5.3.3.	

Table 2.19-168 Swindow::init Information.

2.19.5.13.2 Swindow::show

Swindow::show makes the simple window visible. The function call is Swindow::show(). Table 2.19-169 describes the functions called using this function.

Calls	
Function	**Where Described**
ShowWindow	Standard Window Manager function for Macintosh.
SetPort	Standard Quickdraw function for Macintosh.
Called By	
Function	**Where Described**
ComWindow::show	See Section 2.19.5.3.1.
BringToFront	See Section 2.19.5.13.9.

Table 2.19-169 Swindow::show Information.

2.19.5.13.3 Swindow::quit

Swindow::quit closes this simple window, cleaning up its memory allocations. The function call is Swindow::quit(). Table 2.19-170 describes the functions called using this function.

Calls	
Function	**Where Described**
DisposeWindow	Standard Window Manager function for Macintosh.
DisposPtr	Standard Memory Manager function for Macintosh.
Called By	
Function	**Where Described**
MacQuit	See Section 2.19.2.1.4.
DoneStatus	See Section 2.19.3.8.5.
OtherEventLoop	See Section 2.19.3.8.7.

Table 2.19-170 Swindow::quit Information.

2.19.5.13.4 Swindow::update

Swindow::update redraws the simple window, calling the toolbox functions BeginUpdate and EndUpdate. The function call is Swindow::update(). Table 2.19-171 describes the internal variables used and functions called using this function.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
cport	GrafPtr	Development:THINK C: Mac #includes:
Calls		
Function	**Where Described**	
GetPort	Standard Quickdraw function for Macintosh.	
SetPort	Standard Quickdraw function for Macintosh.	
BeginUpdate	Standard Window Manager function for Macintosh.	
EndUpdate	Standard Window Manager function for Macintosh.	
Called By		
Function	**Where Described**	
ReviewSelection	See Section 2.19.3.2.15.	
MissionSelection	See Section 2.19.3.5.3.	
OtherEventLoop	See Section 2.19.3.8.7.	

Table 2.19-171 Swindow::update Information.

2.19.5.13.5 Swindow::hide

Swindow::hide makes the simple window not visible. The function call is Swindow::hide(). Table 2.19-172 describes the functions called using this function.

Calls	
Function	**Where Described**
HideWindow	Standard Window Manager function for Macintosh.

Table 2.19-172 Swindow::hide Information.

2.19.5.13.6 Swindow::IsVisible

Swindow::IsVisible returns a flag which says whether or not this simple window is visible. The function call is Swindow::IsVisible(). Table 2.19-173 describes the values returned by this function.

Return Values		
Return Value	**Type**	**Meaning**
visible	short	If 0, the Swindow is not visible; If 1, the Swindow is visible.
Called By		
Function	**Where Described**	
PrintWindow	See Section 2.19.3.6.1.	
RefreshWindow	See Section 2.19.3.8.3.	

Table 2.19-173 Swindow::IsVisible Information.

2.19.5.13.7 Swindow::close

Swindow::close closes this simple window. The window is taken down and made not visible. The function call is Swindow::close(). Table 2.19-174 describes the functions called using this function.

Calls	
Function	**Where Described**
CloseWindow	Standard Window Manager function for Macintosh.

Table 2.19-174 Swindow::close Information.

2.19.5.13.8 Swindow::equal

Swindow::equal determines if *other* is the same simple window as this object instance. The function call is Swindow::equal(WindowPtr other). Table 2.19-175 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
other	WindowPtr	Development:THINK C: Mac #includes:WindowMgr.h

Return Values		
Return Value	**Type**	**Meaning**
window == other	Boolean	TRUE if *other* is the same simple window; FALSE if *other* is not the same simple window.
Called By		
Function	**Where Described**	
MainEventLoop	See Section 2.19.2.1.2.	
OtherEventLoop	See Section 2.19.3.8.7.	

Table 2.19-175 Swindow::equal Information.

2.19.5.13.9 Swindow::BringToFront

Swindow::BringToFront brings this Swindow to the front of the screen. The function call is Swindow::BringToFront(). Table 2.19-176 describes the functions called using this function.

Calls	
Function	**Where Described**
show	See Section 2.19.5.13.2.
HiliteWindow	Standard Window Manager function for Macintosh.
SelectWindow	Standard Window Manager function for Macintosh.
SetPort	Standard Quickdraw function for Macintosh.

Table 2.19-176 Swindow::BringToFront Information.

2.19.5.13.10 Swindow::GetWindowPtr

Swindow::GetWindowPtr returns the window pointer of this Swindow object. The function call is Swindow::GetWindowPtr(). Table 2.19-177 describes the values returned by this function.

Return Values		
Return Value	**Type**	**Meaning**
window	WindowPtr	The window pointer of this simple window object.
Called By		
Function	**Where Described**	
MainEventLoop	See Section 2.19.2.1.2.	
PrintWindow	See Section 2.19.3.6.1.	
SetupResWindow	See Section 2.19.3.8.2.	
RefreshWindow	See Section 2.19.3.8.3.	
DoneStatus	See Section 2.19.3.8.5.	
OtherEventLoop	See Section 2.19.3.8.7.	

Table 2.19-177 Swindow::GetWindowPtr Information.

2.19.5.14 Simple_Windows.h
Development:SIMNET:MCC:CEW:simple_Windows.h

This file contains the declaration of the Swindow class.

2.19.5.15 TextList.c
Development:SIMNET:MCC:CEW:TextList.c

This file implements the TextList objects.

2.19.5.15.1 TextList::init

TextList::init allocates and initializes an instance of a TextList object. *window* is the window in which the list appears; *r* is the rectangle containing the list's viewable content. The function call is TextList::init(WindowPtr window, Rect r). Table 2.19-178 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
window	WindowPtr	Development:THINK C: Mac #includes:WindowMgr.h
r	Rect	Development:THINK C: Mac #includes:MacTypes.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
cell_size	Cell	Development:THINK C: Mac #includes:ListMgr.h
datab	Rect	Development:THINK C: Mac #includes:MacTypes.h
listrect	Rect	Development:THINK C: Mac #includes:MacTypes.h
Return Values		
Return Value	**Type**	**Meaning**
(Handle) this	Handle	A handle to the TextList object created.
Calls		
Function	**Where Described**	
SetRect	Standard Quickdraw function for Macintosh.	
SetPt	Standard Quickdraw function for Macintosh.	
LNew	Standard List Manager function for Macintosh	
LSetSelect	Standard List Manager function for Macintosh	
LDoDraw	Standard List Manager function for Macintosh	
LUpdate	Standard List Manager function for Macintosh	
FrameRect	Standard Quickdraw function for Macintosh.	

Called By	
Function	**Where Described**
main	See Section 2.19.2.1.1.
SetupResWindow	See Section 2.19.3.8.2.

Table 2.19-178 TextList::init Information.

2.19.5.15.2 TextList::ListUpdate

TextList::ListUpdate redraws the text list. The function call is TextList::ListUpdate(). Table 2.19-179 describes the functions called using this function.

Calls	
Function	**Where Described**
LUpdate	Standard List Manager function for Macintosh
FrameRect	Standard Quickdraw function for Macintosh.
Called By	
Function	**Where Described**
ResourceItems	See Section 2.19.3.7.1.
RefreshWindow	See Section 2.19.3.8.3.

Table 2.19-179 TextList::ListUpdate Information.

2.19.5.15.3 TextList::ListDispose

TextList::ListDispose removes (or disposes) this text list. The function call is TextList::ListDispose(). Table 2.19-180 describes the functions called using this function.

Calls	
Function	**Where Described**
LDispose	Standard List Manager function for Macintosh
Called By	
Function	**Where Described**
MacQuit	See Section 2.19.2.1.4.
DeleteResourceLists	See Section 2.19.3.7.3.
DoneStatus	See Section 2.19.3.8.5.
OtherEventLoop	See Section 2.19.3.8.7.

Table 2.19-180 TextList::ListDispose Information.

2.19.5.15.4 TextList::GetCellRect

TextList::GetCellRect returns the rectangle surrounding a cell (or row) in the TextList. *rowNum* specifies the cell. The function call is TextList::GetCellRect(short rowNum). Table 2.19-181 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
rowNum	short	Standard C type.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
cell_size	Cell	Development:THINK C: Mac #includes:ListMgr.h
r	Rect	Development:THINK C: Mac #includes:MacTypes.h
Return Values		
Return Value	**Type**	**Meaning**
r	Rect	The rectangle surrounding the specified cell.
Calls		
Function	**Where Described**	
SetPt	Standard Quickdraw function for Macintosh.	
LRect	Standard List Manager function for Macintosh	
FrameRect	Standard Quickdraw function for Macintosh.	
Called By		
Function	**Where Described**	
MainEventLoop	See Section 2.19.2.1.2.	

Table 2.19-181 TextList::GetCellRect Information.

2.19.5.15.5 TextList::GetCount

TextList::GetCount returns the number of items in this TextList. The function call is TextList::GetCount(). Table 2.19-182 describes the values returned by this function.

Return Values		
Return Value	**Type**	**Meaning**
count	short	The number of items on this list.
Called By		
Function	**Where Described**	
setEmplacementTime	See Section 2.19.3.2.4.	
CheckEmplacementDialog	See Section 2.19.3.2.5.	
setBreachTime	See Section 2.19.3.2.6.	
CheckBreachDialog	See Section 2.19.3.2.7.	
CheckMoveDialog	See Section 2.19.3.2.8.	

Table 2.19-182 TextList::GetCount Information.

2.19.5.15.6 TextList::AddRow

TextList::AddRow adds a row to the list. The row is filled in with information from the *data* parameter. The function call is TextList::AddRow(data). Table 2.19-183 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
data	pointer to char	Standard C type.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
cell_size	Cell	Development:THINK C: Mac #includes:ListMgr.h
Calls		
Function	**Where Described**	
SetPt	Standard Quickdraw function for Macintosh.	
LAddRow	Standard List Manager function for Macintosh	
LSetCell	Standard List Manager function for Macintosh	
LDraw	Standard List Manager function for Macintosh	
Called By		
Function	**Where Described**	
SetupResWindow	See Section 2.19.3.8.2.	
MissionObj::NewMission	See Section 2.19.5.7.3.	

Table 2.19-183 TextList::AddRow Information.

2.19.5.15.7 TextList::DeleteRow

TextList::DeleteRow removes the specified row from this text list. The function call is TextList::DeleteRow(short rowNum). Table 2.19-184 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
rowNum	short	Standard C type.
Calls		
Function	**Where Described**	
LDelRow	Standard List Manager function for Macintosh	

Table 2.19-184 TextList::DeleteRow Information.

2.19.5.15.8 TextList::MarkRow

TextList::MarkRow highlights the row specified by *rowNum* in this text list. The function call is TextList::MarkRow(rowNum). Table 2.19-185 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
rowNum	short	Standard C type.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
penState	PenState	Development:THINK C: Mac #includes:Quickdraw.h
r	Rect	Development:THINK C: Mac #includes:MacTypes.h
c	Cell	Development:THINK C: Mac #includes:ListMgr.h
Calls		
Function	**Where Described**	
SetPt	Standard Quickdraw function for Macintosh.	
LRect	Standard List Manager function for Macintosh	
GetPenState	Standard Quickdraw function for Macintosh.	
PenMode	Standard Quickdraw function for Macintosh.	
PenPat	Standard Quickdraw function for Macintosh.	
PaintRect	Standard Quickdraw function for Macintosh.	
SetPenState	Standard Quickdraw function for Macintosh.	

Table 2.19-185 TextList::MarkRow Information.

2.19.5.15.9 TextList::SetRow

TextList::SetRow fills in the specified *row* with the input *data* information. The function call is TextList::SetRow(row, data). Table 2.19-186 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
row	short	Standard C type.
data	pointer to char	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
cell_size	Cell	Development:THINK C: Mac #includes:ListMgr.h
Calls		
Function	**Where Described**	
SetPt	Standard Quickdraw function for Macintosh.	
LSetCell	Standard List Manager function for Macintosh	
LDraw	Standard List Manager function for Macintosh	

Called By	
Function	**Where Described**
MissionObj::edit	See Section 2.19.5.7.2.
MissionObj::show	See Section 2.19.5.7.4.
MissionObj::UpdateMissionString	See Section 2.19.5.7.5.
MissionObj::SetStatus	See Section 2.19.5.7.20.
MissionObj::SendEmplacement	See Section 2.19.5.7.27.
MissionObj::SendBreach	See Section 2.19.5.7.28.

Table 2.19-186 TextList::SetRow Information.

2.19.5.15.10 TextList::GetRow

TextList::GetRow fills in a data pointer with the *data* information from the specified *row*. The function call is TextList::GetRow(row, data). Table 2.19-187 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
row	short	Standard C type.
data	Str255	Development:THINK C: Mac #includes:MacTypes.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
dataPtr	Str255	Development:THINK C: Mac #includes:MacTypes.h
datalen	short	Standard C type.
cell_size	Cell	Development:THINK C: Mac #includes:ListMgr.h
Calls		
Function	**Where Described**	
SetPt	Standard Quickdraw function for Macintosh.	
LGetCell	Standard List Manager function for Macintosh	
stxcpy	See Section 2.19.5.15.13.	
Called By		
Function	**Where Described**	
MissionObj::SetStatus	See Section 2.19.5.7.20.	
MissionObj::SendEmplacement	See Section 2.19.5.7.27.	
MissionObj::SendBreach	See Section 2.19.5.7.28.	

Table 2.19-187 TextList::GetRow Information.

2.19.5.15.11 TextList::ListProc

TextList::ListProc is not called by the application.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
thePt	Point	Development:THINK C: Mac #includes:MacTypes.h
theMods	short	Standard C type.
xstring	Str255	Development:THINK C: Mac #includes:MacTypes.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
dataPtr	Str255	Development:THINK C: Mac #includes:MacTypes.h
dc	short	Standard C type.
cellnum	short	Standard C type.
dataLen	short	Standard C type.
i	short	Standard C type.
a_cell	long	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
-1	short	not used.
cellnum	short	not used.
Calls		
Function	**Where Described**	
LLastClick	Standard List Manager function for Macintosh	
HiWord	Standard Toolbox Utility function for Macintosh.	
LClick	Standard List Manager function for Macintosh	
PtInRect	Standard Quickdraw function for Macintosh.	
LGetCell	Standard List Manager function for Macintosh	
stxcpy	See Section 2.19.5.15.13.	

Table 2.19-188 TextList::ListProc Information.

2.19.5.15.12 TextList::List1Click

List1Click processes a mouse click in this text list. *thePt* is the point of the mouse click; *theMods* specifies any modifier keys to the mouse click; *xstring* is the user entered string. The function call is List1Click(thePt, theMods, xstring). Table 2.19-189 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
thePt	Point	Development:THINK C: Mac #includes:MacTypes.h
theMods	short	Standard C type.
xstring	Str255	Development:THINK C: Mac #includes:MacTypes.h

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
dataPtr	Str255	Development:THINK C: Mac #includes:MacTypes.h
dc	short	Standard C type.
cellnum	short	Standard C type.
dataLen	short	Standard C type.
i	short	Standard C type.
a_cell	long	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
-1	short	Unsuccessful.
cellnum	short	Which cell was clicked on.
Calls		
Function	**Where Described**	
LClick	Standard List Manager function for Macintosh	
LLastClick	Standard List Manager function for Macintosh	
HiWord	Standard Toolbox Utility function for Macintosh.	
PtInRect	Standard Quickdraw function for Macintosh.	
LGetCell	Standard List Manager function for Macintosh	
stxcpy	See Section 2.19.5.15.13.	
Called By		
Function	**Where Described**	
MainEventLoop	See Section 2.19.2.1.2.	
CheckList_Handler	See Section 2.19.3.7.2.	
OtherEventLoop	See Section 2.19.3.8.7.	

Table 2.19-189 TextList::List1Click Information.

2.19.5.15.13 stxcpy

stxcpy copies the C string pointed to by *p* into another C string pointed to by *c*. *num* is the number of characters in the string. The function call is stxcpy(s, p, num). Table 2.19-190 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
s	pointer to char	Standard C type.
p	pointer to char	Standard C type.
num	short	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	short	Standard C type.

Called By	
Function	**Where Described**
TextList::GetRow	See Section 2.19.5.15.10.
TextList::ListProc	See Section 2.19.5.15.11.
TextList::List1Click	See Section 2.19.5.15.12.

Table 2.19-190 stxcpy Information.

2.19.5.16 TextList.h

Development:SIMNET:MCC:CEW:TextList.h

This file contains the declaration of the TextList class.

2.20 The Network Communication Libraries

There are a number of shared libraries that various processes of the MCC use. Below is a section on each library with a brief description of the functions it provides. This structure of this CSC is shown in Figure 2.20-1.

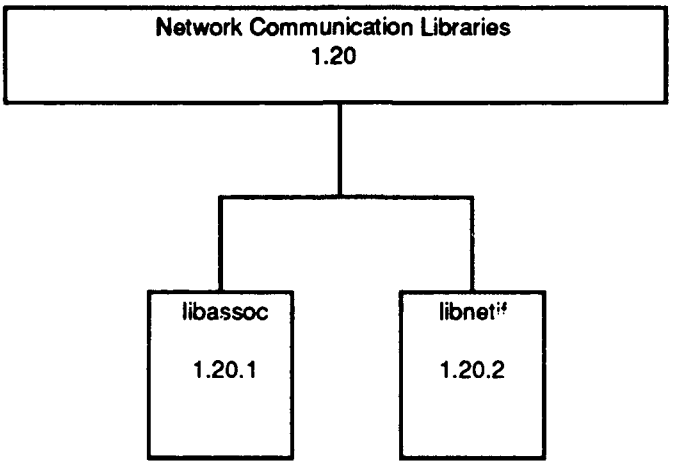


Figure 2.20-1: Network Communication Libraries.

2.20.1 libassoc

The SIMNET Association Layer provides network services to send/receive datagrams and exactly once (xo) transactions. It also provides a subscription service to multicast addresses on the network, allowing an individual application to selectively accept or reject network traffic from different SIMNET exercises, and different protocols within an exercise. More information on the Association Layer can be found in "BBN Report No. 7102, The SIMNET Network and Protocols". Libassoc communicates with the network through libnetif to provide these services.

Libassoc allows an application to open multiple association channels, one per network device to be used. Once a channel is opened, it is referred to by the application through a unique handle, which is returned after successfully opening the channel.

When an error occurs in a libassoc function, it is indicated by an otherwise impossible returned value. This is almost always -1; the individual descriptions specify the details. An error number is made available in the external variable assoc_errno. Table 2.20-1 lists the various libassoc errors.

Error #	Error Name	Description
1	ANETOPEN	Could not open network - The net_open call to libnetif failed. Make sure the network is up and running.
3	ANETACCESS	Could not access network - This is a catch-all for failed calls into libnetif. Once the network has successfully been opened, the most common problem is running out of send buffers. Additional information on failed net calls is available by checking errno (Section 2.20.2 libnetif).
4	ABADHANDLE	Invalid channel handle - The only valid handle is one returned by AssocOpen. Either AssocOpen has not been called, AssocOpen failed and you didn't check the return code (!), or your application has somehow corrupted the handle returned to it by AssocOpen.
6	AFILEOPEN	Could not open file - Currently, the only file that libassoc opens is the "assocDef" file as passed to AssocOpen. Make sure that it has been specified correctly, your application has permission to read it, etc.
7	APARM	Invalid parameter - There is an error in the assocDef file. See the information on AssocDef below which explains how to build an assocDef file.
8	ANOMEMORY	Could not malloc memory - If this error occurs on startup, the INITIAL_DESCRIPTORS parameter is probably too large for the given architecture. If it occurs at some later time, the ADDITIONAL_DESCRIPTORS parameter may be too large, or perhaps descriptors aren't being freed. Make sure that AssocTickAssocLayer is being called.
11	AMAXSUBSCRIBED	Maximum subscriptions exceeded - There is a hard limit to the number of subscriptions allowed by the network interface, as well as a user-definable limit for libassoc. This error occurs if either is exceeded.
12	APKT2BIG	Packet size or threshold too large - Either a request to send a packet larger than the network layer will handle, or in the case of aggregated packets, a threshold size that is too large.

Table 2.20-1 Libassoc Errors.

2.20.1.1 subscribe.c
/simnet/libsrc/libassoc/subscribe.c

subscribe.c contains routines for subscribing to and unsubscribing from multicast groups. Table 2.20-2 describes the variables used by subscribe.c.

Variables		
Variable	**Type**	**Where Typedef Declared**
Errno	extern int	Standard C type.

Table 2.20-2 subscribe.c Variable Information.

2.20.1.1.1 AssocSubscribe

AssocSubscribe subscribes to a multicast group. The routine creates a multicast Ethernet address from the *group* and *protocol* parameters and directs the network interface to start listening to packets with this multicast address. No packets will be received until at least one multicast address has been subscribed to. The *handle* parameter is a low integer referencing the network channel connection. The function call is AssocSubscribe(handle, group, protocol). Table 2.20-3 describes the parameters used, return values and possible errors generated by using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
handle	int	Standard C type.
group	MulticastGroupID	/simnet/common/include/protocol/p_assoc.h
protocol	AssociationUserProtocol	/simnet/common/include/protocol/p_assoc.h
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Unsuccessful.
0	int	Successful.
Errors		
Error Name	**Reason for Error**	
ANETACCESS	Cannot access interface to the Ethernet	
AMAXSUBSCRIBED	Ethernet already has subscribed to a maximum number of multicast addresses.	
ABADHANDLE	Invalid handle (have you called AssocOpen()?)	
Calls		
Function	**Where Described**	
AssocSubscribeWithMask	Section 2.20.1.1.4.	

Table 2.20-3 AssocSubscribe Information.

2.20.1.1.2 AssocUnsubscribe

AssocUnsubscribe unsubscribes from a multicast group. The routine directs the network interface to stop listening to packets with the multicast address formed from the *group* and *protocol* parameters. The *handle* parameter is a low integer referencing the network channel connection. The function call is AssocUnsubscribe(handle, group, protocol). Table 2.20-4 describes the parameters used, return values and possible errors generated by using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
handle	int	Standard C type.
group	MulticastGroupID	/simnet/common/include/protocol/p_assoc.h
protocol	AssociationUserProtocol	/simnet/common/include/protocol/p_assoc.h
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Unsuccessful.
0	int	Successful.
Errors		
Error Name	**Reason for Error**	
ANETACCESS	Cannot access interface to the Ethernet	
ABADHANDLE	Invalid handle (have you called AssocOpen()?)	
Calls		
Function	**Where Described**	
AssocUnsubscribeWithMask	Section 2.21.1.1.5.	

Table 2.20-4 AssocUnsubscribe Information.

2.20.1.1.3 AssocCurrentlySubscribed

AssocCurrentlySubscribed checks to see if the application is currently subscribed to the multicaste group address formed by the *group* and *protocol* parameters. The *handle* parameter is a low integer referencing the network channel connection. The function call is AssocCurrentlySubscribed(handle, group, protocol). Table 2.20-5 describes the parameters used and return values generated by using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
handle	int	Standard C type.
group	MulticastGroupID	/simnet/common/include/protocol/p_assoc.h
protocol	AssociationUserProtocol	/simnet/common/include/protocol/p_assoc.h

Return Values		
Return Value	**Type**	**Meaning**
1	int	Currently subscribed.
0	int	Not currently subscribed.
Calls		
Function	**Where Described**	
AssocCurrentlySubscribedWithMask	Section 2.20.1.1.6.	

Table 2.20-5 AssocCurrentlySubscribed Information.

2.20.1.1.4 AssocSubscribeWithMask

AssocSubscribeWithMask subscribes to a private multicast group (analogous to AssocSubscribe described above). Reference AssocCreatMCAWithMask() in Section 2.20.1.1.8 for a description of the mask. The function call is AssocSubscribeWithMask(handle, group, protocol, mask). Table 2.20-6 describes the parameters used, return values and possible errors generated by using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
handle	int	Standard C type.
group	MulticastGroupID	/simnet/common/include/protocol/p_assoc.h
protocol	AssociationUserProtocol	/simnet/common/include/protocol/p_assoc.h
mask	unsigned long	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
address	NetworkAddress	/simnet/libsrc/libnetif/network.h
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Unsuccessful.
0	int	Successful.
Errors		
Error Name	**Reason for Error**	
ABADHANDLE	Invalid handle (have you called AssocOpen()?).	
Calls		
Function	**Where Described**	
ASSOC_CHECK_HANDLE	Macro defined in /simnet/libsrc/libassoc/assoc_lcl.h.	
AssocCreateMCAWithMask	Section 2.20.1.1.7.	
net_add_mca	Section 2.20.2.11.1.	
AddSubscription	Section 2.20.1.1.9.	

Called By	
Function	**Where Described**
AssocSubscribe	Section 2.20.1.1.1.

Table 2.20-6 AssocSubscribeWithMask Information.

2.20.1.1.5 AssocUnsubscribeWithMask

AssocUnsubscribeWithMask unsubscribes from a private multicast group (analagous to AssocUnsubscribe described above). *group*, *protocol*, and *mask* are used to specify the multicast group. The *handle* parameter is a low integer referencing the network channel connection. The function call is AssocUnsubscribeWithMask(handle, group, protocol, mask). Table 2.20-7 describes the parameters used, return values and possible errors generated by using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
handle	int	Standard C type.
group	MulticastGroupID	/simnet/common/include/protocol/p_assoc.h
protocol	AssociationUserProtocol	/simnet/common/include/protocol/p_assoc.h
mask	unsigned long	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
address	NetworkAddress	/simnet/libsrc/libnetif/network.h
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Unsuccessful.
0	int	Successful.
Errors		
Error Name	**Reason for Error**	
ABADHANDLE	Invalid handle (have you called AssocOpen()?).	
Calls		
Function	**Where Described**	
ASSOC_CHECK_HANDLE	Macro defined in /simnet/libsrc/libassoc/assoc_lcl.h.	
AssocCreateMCAWithMask	Section 2.20.1.1.7.	
net_del_mca	Section 2.20.2.11.2.	
DeleteSubsciption	Section 2.20.1.1.10.	
Called By		
Function	**Where Described**	
AssocUnsubscribe	Section 2.20.1.1.2.	
AssocClose	Section 2.20.1.11.1.	

Table 2.20-7 AssocUnsubscribeWithMask Information.

2.20.1.1.6 AssocCurrentlySubscribedWithMask

AssocCurrentlySubscribedWithMask checks to see if the application is currently subscribed to a given private multicast group (analogous to AssocCurrentlySubscribed described above). *group*, *protocol*, and *mask* are used to specify the multicast group. The function call is AssocCurrentlySubscribedWithMask(handle, group, protocol, mask). Table 2.20-8 describes the parameters used, return values and possible errors generated by using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
handle	int	Standard C type.
group	MulticastGroupID	/simnet/common/include/protocol/p_assoc.h
protocol	AssociationUserProtocol	/simnet/common/include/protocol/p_assoc.h
mask	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	register int	Standard C type.
subCount	register int	Standard C type.
s	register pointer to Subscription	/simnet/libsrc/libassoc/assoc_lcl.h
channel	register pointer to ChannelDescriptor	/simnet/libsrc/libassoc/assoc_lcl.h
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Unsuccessful.
0	int	Successful.
Errors		
Error Name	**Reason for Error**	
ABADHANDLE	Invalid handle (have you called AssocOpen()?).	
Calls		
Function	**Where Described**	
ASSOC_CHECK_HANDLE	Macro defined in /simnet/libsrc/libassoc/assoc_lcl.h.	
Called By		
Function	**Where Described**	
AssocCurrentlySubscribed	Section 2.20.1.1.3.	

Table 2.20-8 AssocCurrentlySubscribedWithMask Information.

2.20.1.1.7 AssocCreateMCA

AssocCreateMCA creates the Multicast Group Address in the buffer pointed to by *address* using the *group* and *protocol* parameters as described in AssocCreateMCAWithMask using a mask of zero. The function call is AssocCreateMCAWithMask(group, protocol, address, mask). Table 2.20-9 describes the parameters used, return values and possible errors generated by using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
group	MulticastGroupID	/simnet/common/include/protocol/p_assoc.h
protocol	AssociationUserProtocol	/simnet/common/include/protocol/p_assoc.h
mask	unsigned long	Standard C type.
address	pointer to NetworkAddress	/simnet/libsrc/libnetif/network.h
Calls		
Function	**Where Described**	
AssocCreateMCAWithMask	Section 2.20.1.1.7.	

Table 2.20-9 AssocCreateMCA Information.

2.20.1.1.8 AssocCreateMCAWithMask

AssocCreateMCAWithMask creates a multicast group address using group, protocol and mask as follows:

Because bits are transmitted from lowest addressed bit to highest addressed bit, a NetworkAddress looks like this:

7 6 5 4 3 2 1 0	15 14 13 12 11 10 9 8	23 22 21 20 19 18 17 16	31 30 29 28 27 26 25 etc.

Bit 0 gets a 1, bit 1 gets a 1, bits 2->7 get 0s, bits 8->31 get bits 0->23 of the mask, bits 32->39 get the group number, and bits 40->47 get the protocol number. This functionality is considered an extension to, rather than a part of, the association layer.

The function call is AssocCreateMCA(group, protocol, mask, address). Table 2.20-10 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
group	MulticastGroupID	/simnet/common/include/protocol/p_assoc.h
protocol	AssociationUserProtocol	/simnet/common/include/protocol/p_assoc.h
address	pointer to NetworkAddress	/simnet/libsrc/libnetif/network.h
mask	unsigned long	Standard C type.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
addr	register pointer to char	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Unsuccessful.
0	int	Successful.
Errors		
Error Name	**Reason for Error**	
ABADHANDLE	Invalid handle (have you called AssocOpen()?).	
Called By		
Function	**Where Described**	
AssocSubscribeWithMask	Section 2.20.1.1.4.	
AssocUnsubscribeWithMask	Section 2.20.1.1.5.	
AssocCreateMCA	Section 2.20.1.1.8.	
AssocSendDatagram	Section 2.20.1.2.1.	
AssocSendAggregate	Section 2.20.1.3.1.	
AssocSendTransact	Section 2.20.1.4.1.	
AssocSendResponse	Section 2.20.1.4.2.	

Table 2.20-10 AssocCreateMCA Information.

2.20.1.1.9 AddSubscription

AddSubscription is called by the routine AssocSubscribe to add a subscription to a multicast group. The function call is AddSubscription(handle, group, protocol, mask). Table 2.20-11 describes the parameters used and errors returned using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
handle	int	Standard C type.
group	MulticastGroupID	/simnet/common/include/protocol/p_assoc.h
protocol	AssociationUserProtocol	/simnet/common/include/protocol/p_assoc.h
mask	unsigned long	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
channel	register pointer to ChannelDescriptor	/simnet/libsrc/libassoc/assoc_lcl.h
subscriptions	register pointer to Subscription	/simnet/libsrc/libassoc/assoc_lcl.h
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Unsuccessful.
0	int	Successful.

Errors	
Error Name	**Reason for Error**
AMAXSUBSCRIBED	Ethernet already has subscribed to a maximum number of multicast addresses.
Called By	
Function	**Where Described**
AssocSubscribeWithMask	Section 2.20.1.1.4.

Table 2.20-11 AddSubscription Information.

2.20.1.1.10 DeleteSubscription

DeleteSubscription is called by the routine AssocUnsubscribe in order to remove a subscription to a multicast group. The function call is DeleteSubscription(handle, group, protocol, mask). Table 2.20-12 describes the parameters used, errors returned and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
handle	int	Standard C type.
group	MulticastGroupID	/simnet/common/include/protocol/p_assoc.h
protocol	AssociationUserProtocol	/simnet/common/include/protocol/p_assoc.h
mask	unsigned long	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	register int	Standard C type.
subCount	register int	Standard C type.
channel	register pointer to ChannelDescriptor	/simnet/libsrc/libassoc/assoc_lcl.h
s	register pointer to Subscription	/simnet/libsrc/libassoc/assoc_lcl.h
Called By		
Function	**Where Described**	
AssocUnsubscribeWithMask	Section 2.20.1.1.5.	

Table 2.20-12 DeleteSubscription Information.

2.20.1.2 send.c

/simnet/libsrc/libassoc/send.c

send.c contains routines for sending datagrams. Table 2.20-13 describes the variables used by send.c.

Variables		
Variable	**Type**	**Where Typedef Declared**
outBuf	AssocNetBuffer	/simnet/libsrc/libassoc/assoc_lcl.h

Table 2.20-13 send.c Variable Information.

2.20.1.2.1 AssocSendDatagram

AssocSendDatagram sends an Association Datagram to the multicast group defined by *group* and *protocol*, and current send mask. *data* points to the data to be contained in the datagram. *length* is the number of bytes pointed to by *data* . The function call is AssocSendDatagram(handle, data, length, group, protocol). Table 2.20-14 describes the parameters used, return values and possible errors generated by using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
handle	int	Standard C type.
data	pointer to char	Standard C type.
length	long int	Standard C type.
group	MulticastGroupID	/simnet/common/include/protocol/p_assoc.h
protocol	AssociationUserProtocol	/simnet/common/include/protocol/p_assoc.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
apdu	register pointer to AssociationPDU	/simnet/common/include/protocol/p_assoc.h
dgSize	int	Standard C type.
addrString[10]	char	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Unsuccessful.
0	int	Successful.
Errors		
Error Name	**Reason for Error**	
ANETACCESS	Cannot access interface to the Ethernet	
Calls		
Function	**Where Described**	
ASSOC_CHECK_HANDLE	Macro defined in /simnet/libsrc/libassoc/assoc_lcl.h.	
CHECK_ASSOC_PKT_LENGTH	Macro defined in /simnet/libsrc/libassoc/assoc_lcl.h.	
AssocCreateMCAWithMask	Section 2.20.1.1.7.	
DATACOPY	Macro defined in /simnet/libsrc/libassoc/assoc_lcl.h.	
PRO_ASSOC_DATAGRAM_SIZE	Macro defined in /simnet/common/include/protocol/p_size.h	
CHECK_NET_PKT_LENGTH	Macro defined in /simnet/libsrc/libassoc/assoc_lcl.h	
AssocPadBuffer	Section 2.20.1.2.2.	
net_snd	Section 2.20.2.19.2.	

Table 2.20-14 AssocSendDatagram Information.

2.20.1.2.2 AssocPadBuffer

AssocPadBuffer expects a pointer to the beginning of a data buffer aPDU and the current length of the buffer. It adds a padding PDU to the end of the PDU in order to bring the total size up to MIN_DATA_SIZE_8023. The function call is AssocPadBuffer(apdu, bufferLength). Table 2.20-15 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
apdu	pointer to AssociationPDU	/simnet/common/include/protocol/p_assoc.h
bufferLength	pointer to int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
padding	int	Standard C type.
Calls		
Function	**Where Described**	
DATACOPY	Macro defined in /simnet/libsrc/libassoc/assoc_lcl.h.	
PRO_ASSOC_PADDING_HDR_SIZE	Macro defined in /simnet/common/include/protocol/p_size.h	
PRO_ASSOC_PADDING_SIZE	Macro defined in /simnet/common/include/protocol/p_size.h	
Called By		
Function	**Where Described**	
AssocSendDatagram	Section 2.20.1.2.1.	
AssocSendAggregate	Section 2.20.1.3.1.	
AssocSendTransact	Section 2.20.1.4.1.	
AssocSendResponse	Section 2.20.1.4.2.	

Table 2.20-15 AssocPadBuffer Information.

2.20.1.3 aggregate.c

/simnet/libsrc/libassoc/aggregate.c

aggregate.c contains routines for sending aggregated datagrams.

2.20.1.3.1 AssocSendAggregate

AssocSendAggregate adds this PDU to a buffer of PDUs of the same *group* and *protocol*. If the current buffer of PDUs has a different group or protocol, or if the current send mask has changed, that buffer is written to the net and a new buffer is filled with the current PDU. *data* is a pointer to the current PDU to be sent and *length* is the number of bytes pointed to by *data*. *timer* is the number of milliseconds the PDU can wait before being sent. *threshold* is the maximum number of bytes allowed in the aggregated packet. To send a PDU out immediately and thus flush the current buffer, set threshold to 0. Aggregate PDUs are sent to the multicast group formed by the group and protocol common to all of the PDUs in the buffer and the send mask in force while it was built. The function

call is AssocSendAggregate(handle, data, length, group, protocol, timer, threshold). Table 2.20-16 describes the parameters used, return values and possible errors generated by using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
handle	int	Standard C type.
data	pointer to char	Standard C type.
length	long int	Standard C type.
group	MulticastGroupID	/simnet/common/include/protocol/p_assoc.h
protocol	AssociationUserProtocol	/simnet/common/include/protocol/p_assoc.h
timer	unsigned long int	Standard C type.
threshold	unsigned long int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
channel	register pointer to ChannelDescriptor	/simnet/libsrc/libassoc/assoc_lcl.h
apdu	register pointer to AssociationPDU	/simnet/common/include/protocol/p_assoc.h
currentTime	long int	Standard C type.
destAddr	NetworkAddress	/simnet/libsrc/libnetif/network.h
Return Values		
Return Value	**Type**	***Meaning***
-1	int	Unsuccessful.
0	int	Successful.
Errors		
Error Name	**Reason for Error**	
ANETACCESS	Cannot access interface to the Ethernet	
Calls		
Function	**Where Described**	
ASSOC_CHECK_HANDLE	Macro defined in /simnet/libsrc/libassoc/assoc_lcl.h.	
CHECK_ASSOC_PKT_LENGTH	Macro defined in /simnet/libsrc/libassoc/assoc_lcl.h.	
CHECK_NET_PKT_LENGTH	Macro defined in /simnet/libsrc/libassoc/assoc_lcl.h.	
net_current_time	Section 2.20.2.8.3.	
AssocCreateMCAWithMask	Section 2.20.1.1.7.	
AssocPadBuffer	Section 2.20.1.2.2.	
net_snd	Section 2.20.2.19.2.	
MIN	Macro defined in /simnet/libsrc/libassoc/assoc_lcl.h.	
DATACOPY	Macro defined in /simnet/libsrc/libassoc/assoc_lcl.h.	
PRO_ASSOC_DATAGRAM_SIZE	Macro defined in /simnet/common/include/protocol/p_size.h	

Called By	
Function	Where Described
AssocTickAssocLayer	Section 2.20.1.8.1.

Table 2.20-16 AssocSendAggregate Information.

2.20.1.4 transact.c
/simnet/libsrc/libassoc/transact.c

transact.c contains routines to interface the host to the transaction protocol capabilities of the association layer library.

2.20.1.4.1 AssocSendTransact

AssocSendTransact sends an Association Transaction PDU directed to respondent and visible to any other observer subscribed to the Ethernet multicast address described by the *group* and *protocol* pair, in conjunction with the current send mask. *respondent* must also be subscribed to this multicast address. *data* points to the data to be transmitted and *length* is the number of bytes in *data*.

If a response to this transaction is received, the *callback* routine will be called with the user-specified *cparam* as one of its parameters. If the transaction times out before a response is received, the *timeout* routine will be called with the used-specified *tparam* as one of its parameters. Note that cparam and tparam must point to space that is statically allocated by the application. When a response is received, the application is notified via the primitive parameter to the AssocReceivePDU routine. No such notification is given in the case of timeout, however.

The callback routine should be defined by the application as:

```
int callback(data, length, respondent, cparam)
char data;
long int length;
SimulationAddress respondent;
long int cparam;
```

where *respondent* is the Simulation Address of the simulator which has responded to the transaction. *length* is the number of bytes in the information pointed to by *data*. *cparam* is a user supplied parameter which may be used to store extra information for the transaction. cparam must point to storage which is allocated by the application.

The timeout routine should be defined by the application as:

```
int timeout(data, length, respondent, tparam)
char data;
long int length;
SimulationAddress respondent;
long int tparam;
```

where *respondent* is the Simulation Address of the simulator who was supposed to respond to the transaction. *length* is the number of bytes in the information pointed to by *data*, which is the original PDU sent by the application. *tparam* is a user supplied parameter which may be used to store extra information for the transaction. Again, tparam must point to storage which is allocated by the application.

The function call is AssocSendTransact(handle, data, length, group, protocol, respondent, callback, cparam, timeout, tparam). Table 2.20-17 describes the parameters used, return values and possible errors generated by using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
handle	int	Standard C type.
data	pointer to char	Standard C type.
length	long int	Standard C type.
group	MulticastGroupID	/simnet/common/include/protocol/p_assoc.h
protocol	AssociationUserProtocol	/simnet/common/include/protocol/p_assoc.h
respondent	SimulationAddress	/simnet/common/include/protocol/address.h
callback	pointer to function returning int	Information below.
cparam	pointer to long int	Standard C type.
timeout	pointer to function returning int	Information below.
tparam	pointer to long int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
td	register pointer to AssocTransDescriptor	/simnet/libsrc/libassoc/assoc_lcl.h
apdu	register pointer to AssociationPDU	/simnet/common/include/protocol/p_assoc.h
reqSize	int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Unsuccessful.
0	int	Successful.
Errors		
Error Name	**Reason for Error**	
ANETACCESS	Cannot access interface to the Ethernet	
ANOMEMORY	Not enough memory for internal Association structures.	

Calls	
Function	**Where Described**
ASSOC_CHECK_HANDLE	Macro defined in /simnet/libsrc/libassoc/assoc_lcl.h.
AssocGetDescriptor	Section 2.20.1.24.3.
AssocCreateMCAWithMask	Section 2.20.1.1.7.
DATACOPY	Macro defined in /simnet/libsrc/libassoc/assoc_lcl.h.
PRO_ASSOC_REQUEST_SIZE	Macro defined in /simnet/common/include/protocol/p_size.h
AssocPadBuffer	Section 2.20.1.2.2.
net_snd	Section 2.20.2.19.2.
AssocFreeDescriptor	Section 2.20.1.24.4.
AssocAddTransaction	Section 2.20.1.23.2.

Table 2.20-17 AssocSendTransact Information.

2.20.1.4.2 AssocSendResponse

AssocSendResponse sends an Association Response PDU directed to the originator of the transaction. Any observers subscribed to the Ethernet multicast address described by the same *group*, *protocol* and send mask as the transaction can also listen to the response. Note that the send mask of the last transaction received is used for the response, not the current send mask of the respondee. This behavior may be modified by using AssocSetRspMask. *cache_response* should be TRUE if the association entity should respond to subsequent retries with the same response. *cache_response* should be FALSE if the application wishes to hear each retry. *data* points to the data in the response and *length* is the number of bytes pointed to by *data* . *transID* must be the same transID as the original request. The function call is AssocSendResponse(handle, data, length, group, protocol,originator, transID, cache_response). Table 2.20-18 describes the parameters used, return values and possible errors generated by using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
handle	int	Standard C type.
data	pointer to char	Standard C type.
length	long int	Standard C type.
group	MulticastGroupID	/simnet/common/include/protocol/p_assoc.h
protocol	AssociationUserProtocol	/simnet/common/include/protocol/p_assoc.h
originator	pointer to SimAddress	
transID	TransactionIdentifier	/simnet/common/include/protocol/address.h
cache_response	long int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
rd	register pointer to AssocTransDescriptor	/simnet/libsrc/libassoc/assoc_lcl.h
apdu	register pointer to AssociationPDU	/simnet/common/include/protocol/p_assoc.h
rspSize	int	Standard C type.

Return Values		
Return Value	**Type**	**Meaning**
-1	int	Unsuccessful.
0	int	Successful.
Errors		
Error Name	**Reason for Error**	
ANETACCESS	Cannot access interface to the Ethernet	
ANOMEMORY	Not enough memory for internal Association structures	
Calls		
Function	**Where Described**	
ASSOC_CHECK_HANDLE	Macro defined in /simnet/libsrc/libassoc/assoc_lcl.h.	
AssocGetDescriptor	Section 2.20.1.24.3.	
AssocCreateMCAWithMask	Section 2.20.1.1.7.	
DATACOPY	Macro defined in /simnet/libsrc/libassoc/assoc_lcl.h.	
PRO_ASSOC_RESPONSE_SIZE	Macro defined in /simnet/common/include/protocol/p_size.h	
AssocPadBuffer	Section 2.20.1.2.2.	
net_snd	Section 2.20.2.19.2.	
AssocFreeDescriptor	Section 2.20.1.24.4.	
AssocCacheResponse	Section 2.20.1.18.2	

Table 2.20-18 AssocSendResponse Information.

2.20.1.5 open.c

/simnet/libsrc/libassoc/open.c

open.c contains routines to initialize the association layer. Table 2.20-19 describes the variables used by open.c.

Variables		
Variable	**Type**	**Where Typedef Declared**
errno	extern int	Standard C type.
channelMap	pointer to array size maxChannels of ChannelDescriptor	/simnet/libsrc/libassoc/assoc_lcl.h
assocPaddingPDU	AssociationPDU	/simnet/common/include/protocol/p_assoc.h

Table 2.20-19 open.c Variable Information.

2.20.1.5.1 AssocOpen

AssocOpen resets the network interface and opens a channel to the specified network. It allocates and initializes all data structures and variables needed by the association network entity. *device* is the name of the device you wish to open (e.g. "/dev/enp0"). *assocDef* is the name of the file to use for association layer definitions/parameters. AssocOpen resets the network interface to insure that the device is not in promiscuous mode, that no multicast groups are currently subscribed to, that the ethernet type code (AssocFamily) is set to the default, and that the input and output buffers are flushed. A number of defaults for the channel are set. Some of these defaults are modifiable by entries in the assocDef file. AssocDef parameters are described in Table 2.20-80.

This routine returns a handle to a network interface that must be used for subsequent association library calls, and may be used directly for network (libnetif) calls. Either AssocOpen or AssocAttach must be called before any other association routine is called. The function call is AssocOpen(device, assocDef). Table 2.20-20 describes the parameters used, return values and possible errors generated by using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
device	pointer to char	Standard C type.
assocDef	pointer to char	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
handle	int	Standard C type.
nib	struct net_info_block	/simnet/libsrc/libnetif/network.h
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Unsuccessful.
association channel (and net) handle	int	Successful.
Errors		
Error Name	**Reason for Error**	
ANETOPEN	Cannot open interface to the Ethernet.	
ANETACCESS	Cannot initialize multicast service or device does not respond.	
ANOMEMORY	Cannot malloc memory for transaction service data structures.	
AFILEOPEN	Cannot open the assocDef file.	
APARM	Missing or invalid site and/or hose parameter is assocDef file.	
Calls		
Function	**Where Described**	
net_open	Section 2.20.2.10.1.	
net_stop	Section 2.20.2.13.2.	
net_norm	Section 2.20.2.7.1.	
net_alive	Section 2.20.2.1.1.	
net_init_mca	Section 2.20.2.11.3.	
SetChannelDefaults	Section 2.20.1.5.3.	
AssocReadParams	Section 2.20.1.22.2.	
net_init_type	Section 2.20.2.18.2.	
net_add_type	Section 2.20.2.18.1.	
net_set_snd_type	Section 2.20.2.19.8.	
net_set_snd_from_addr	Section 2.20.2.19.7.	
AssocCreateFreeList	Section 2.20.1.24.1.	
AssocInitTransactions	Section 2.20.1.23.1.	
AssocInitResponses	Section 2.20.1.18.1.	

Table 2.20-20 AssocOpen Information.

2.20.1.5.2 AssocAttach

AssocAttach opens a channel to a network device without resetting the network interface. It should be called if some other process has already called AssocOpen. Either AssocOpen or AssocAttach must be called before any other association routine is called. The function call is AssocAttach(device, assocDef). Table 2.20-21 describes the parameters used, return values and possible errors generated by using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
device	pointer to char	Standard C type.
assocDef	pointer to char	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
handle	int	Standard C type.
nib	struct net_info_block	/simnet/libsrc/libnetif/network.h
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Unsuccessful.
association channel (and net) handle	int	Successful.
Errors		
Error Name	**Reason for Error**	
ANETOPEN	Cannot open interface to the Ethernet.	
ANETACCESS	Cannot initialize multicast service or device does not respond.	
ANOMEMORY	Cannot malloc memory for transaction service data structures.	
AFILEOPEN	Cannot open the assocDef file.	
APARM	Missing or invalid site and/or hose parameter is assocDef file.	
Calls		
Function	**Where Described**	
net_open	Section 2.20.2.10.1.	
net_alive	Section 2.20.2.1.1.	
SetChannelDefaults	Section 2.20.1.5.3.	
AssocReadParams	Section 2.20.1.22.2.	
net_add_type	Section 2.20.2.18.1.	
net_set_snd_from_addr	Section 2.20.2.19.7.	
AssocCreateFreeList	Section 2.20.1.24.1.	
AssocInitTransactions	Section 2.20.1.23.1.	
AssocInitResponses	Section 2.20.1.18.1.	

Table 2.20-21 AssocAttach Information.

2.20.1.5.3 SetChannelDefaults

SetChannelDefaults sets the channel parameters to the default values. The function call is SetChannelDefaults(handle). Table 2.20-22 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
handle	int	standard
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
channel	register pointer to ChannelDescriptor	/simnet/libsrc/libassoc/assoc_lcl.h
Called By		
Function	**Where Described**	
AssocAttach	Section 2.20.1.5.2.	
AssocOpen	Section 2.20.1.5.1.	

Table 2.20-22 SetChannelDefaults Information.

2.20.1.6 receive.c

/simnet/libsrc/libassoc/receive.c

receive.c contains routines to receive packets for the SIMNET network through the association layer. Table 2.20-23 describes the variables used by receive.c.

Variables		
Variable	**Type**	**Where Typedef Declared**
errno	extern int	Standard C type.

Table 2.20-23 receive.c Variable Information.

2.20.1.6.1 AssocReceivePDU

AssocReceivePDU queries the Association Layer for PDU's from the network. The routine gets a pointer to the next PDU and sets what the data pointer (**data*) to the PDU pointer when appropriate. The data pointer will not be set if the PDU is a response to a transaction the application issued or if the PDU is a retry for which the application has a response cached. If the PDU is a response to a transaction the application issued, the association entity calls the supplied callback routine and returns with *primitive* equal to ASSOC_TRANSACT_CONF. If the PDU is a retry request for which the application has a cached response, the association entity retransmits the response and returns with *primitive* equal to ASSOC_NOTHING_OF_INTEREST. When AssocReceivePDU returns a PDU, protocol is the user protocol to which the PDU belongs, and *primitive* indicates whether the PDU is a datagram, request, etc., as defined in "assoc.h". *length* is the number of bytes in the information pointed to by *data*. The function call is AssocReceivePDU(handle, data, length, group, protocol, primitive, originator, transID, respondent). Table 2.20-24 describes the parameters used, return values and possible errors generated by using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
handle	int	Standard C type.
data	pointer to pointer to char	Standard C type.
length	pointer to long int	Standard C type.
group	pointer to MulticastGroupID	/simnet/common/include/protocol/p_assoc.h
protocol	pointer to AssociationUserProtocol	/simnet/common/include/protocol/p_assoc.h
primitive	pointer to long int	Standard C type.
originator	pointer to SimulationAddress	/simnet/common/include/protocol/address.h
transID	pointer to TransactionIdentifier	/simnet/common/include/protocol/address.h
respondent	pointer to SimulationAddress	/simnet/common/include/protocol/address.h

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
channel	register pointer to ChannelDescriptor	/simnet/libsrc/libassoc/assoc_lcl.h
apdu	register pointer to AssociationPDU	/simnet/common/include/protocol/p_assoc.h
inBuf	pointer to char	Standard C type.
retVal	int	Standard C type.
apduSize	int	Standard C type.

Return Values		
Return Value	**Type**	**Meaning**
-1	int	Read failed.
0	int	Data is returned.
1	int	No data is returned and network buffers are not empty.
2	int	Network buffers are empty.

Errors	
Error Name	**Reason for Error**
ANETACCESS	Cannot access interface to the Ethernet.

Calls	
Function	**Where Described**
ASSOC_CHECK_HANDLE	Macro defined in /simnet/libsrc/libassoc/assoc_lcl.h.
net_release_rcv	Section 2.20.2.17.10.
net_get_rcv	Section 2.20.2.17.9.
REJECT_PACKET	Macro defined in receive.c. Section 2.20.1.6.
PROCESS_DATAGRAM_PDU	Macro defined in /simnet/libsrc/libassoc/assoc_lcl.h.
PRO_ASSOC_DATAGRAM_SIZE	Macro defined in /simnet/common/include/protocol/p_size.h
AssocProcessRequestPDU	Section 2.20.1.20.1.
PRO_ASSOC_REQUEST_SIZE	Macro defined in /simnet/common/include/protocol/p_size.h
AssocProcessResponsePDU	Section 2.20.1.19.1.
PRO_ASSOC_RESPONSE_SIZE	Macro defined in /simnet/common/include/protocol/p_size.h
PRO_ASSOC_PADDING_SIZE	Macro defined in /simnet/common/include/protocol/p_size.h

Table 2.20-24 AssocReceivePDU Information.

Table 2.20-25 lists the values returned in primitive.

Value	Description
ASSOC_NOTHING_OF_INTEREST	Padding PDU or Association Layer retransmitted a cached response.
ASSOC_DATAGRAM_IND	Datagram PDU.
ASSOC_REQUEST_IND	Transaction request to which we are an observer.
ASSOC_RESPONSE_IND	Transaction response to which we are an observer.
ASSOC_TRANSACT_IND	Transaction request destined for my host.
ASSOC_TRANSACT_CONF	Association layer process a callback routine.

Table 2.20-25 Values Returned in Primitive of AssocReceivePDU.

2.20.1.7 block.c

/simnet/libsrc/libassoc/block.c

block.c contains routines to receive packets from the SIMNET network through the association layer. It differs from receive.c (Section 2.20.1.6.) in that calls to AssocWaitForPDU block on the network.

2.20.1.7.1 AssocWaitForPDU

AssocWaitForPDU queries the Association Layer for PDUs from the net, reads the Association PDU, and sets **data* to the address of the PDU. AssocWaitForPDU will block until a packet is received from the network. The packet is then processed in the same manner as in Section 2.20.1.6.1 AssocReceivePDU, except that if there is no data to be returned to the application from that packet, the routine again blocks on the network, until there is a PDU to return. The function call is AssocWaitForPDU(handle, data, length, group, protocol, primitive, originator, transID, respondent). Table 2.20-26 describes the parameters used, return values and possible errors generated by using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
handle	int	Standard C type.
data	pointer to pointer to char	Standard C type.
length	pointer to long int	Standard C type.
group	pointer to MulticastGroupID	/simnet/common/include/protocol/p_assoc.h
protocol	pointer to AssociationUserProtocol	/simnet/common/include/protocol/p_assoc.h
primitive	pointer to long int	Standard C type.
originator	pointer to SimulationAddress	/simnet/common/include/protocol/address.h
transID	pointer to TransactionIdentifier	/simnet/common/include/protocol/address.h
respondent	pointer to SimulationAddress	/simnet/common/include/protocol/address.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
channel	register pointer to ChannelDescriptor	/simnet/libsrc/libassoc/assoc_lcl.h
apdu	register pointer to AssociationPDU	/simnet/common/include/protocol/p_assoc.h
inBuf	pointer to char	Standard C type.
needAPDU	int	Standard C type.
retVal	int	Standard C type.
apduSize	int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Read failed.
0	int	Data is returned.
Errors		
Error Name	**Reason for Error**	
ANETACCESS	Cannot access interface to the Ethernet.	

Calls	
Function	**Where Described**
ASSOC_CHECK_HANDLE	Macro defined in /simnet/libsrc/libassoc/assoc_lcl.h.
net_release_rcv	Section 2.20.2.17.10.
net_get_rcv	Section 2.20.2.17.9.
REJECT_PACKET	Macro defined in block.c. Section 2.20.1.7.
PROCESS_DATAGRAM_PDU	Macro defined in /simnet/libsrc/libassoc/assoc_lcl.h.
PRO_ASSOC_DATAGRAM_SIZE	Macro defined in /simnet/common/include/protocol/p_size.h
AssocProcessRequestPDU	Section 2.20.1.20.1.
PRO_ASSOC_REQUEST_SIZE	Macro defined in /simnet/common/include/protocol/p_size.h
AssocProcessResponsePDU	Section 2.20.1.19.1.
PRO_ASSOC_RESPONSE_SIZE	Macro defined in /simnet/common/include/protocol/p_size.h
PRO_ASSOC_PADDING_SIZE	Macro defined in /simnet/common/include/protocol/p_size.h

Table 2.20-26 AssocWaitForPDU Information.

Table 2.20-27 lists the values returned in primitive.

Value	Description
ASSOC_DATAGRAM_IND	Datagram PDU.
ASSOC_REQUEST_IND	Transaction request to which we are an observer.
ASSOC_RESPONSE_IND	Transaction response to which we are an observer.
ASSOC_TRANSACT_IND	Transaction request destined for my host.

Table 2.20-27 Values Returned in Primitive of AssocWaitForPDU.

2.20.1.8 tick.c

/simnet/libsrc/libassoc/tick.c

tick.c contains routines which perform time related functions of the Association service.

2.20.1.8.1 AssocTickAssocLayer

AssocTickAssocLayer performs time-related functions of the Association service. It will issue retries for those Association Transactions that have not yet received a response and calls a callback routine for those transactions that have timed out to indicate a failed transaction. The callback routine is the one that was specified in the initial call to AssocSendTransact(). If no callback routine was supplied, the transaction is simply cleared. It will clear cached responses that have timed out and send out expired aggregate packets. AssocTickAssocLayer must be called by the application in order to perform these activities and should be called at least once a second to conform to the SIMNET 6.0 protocol. AssocTickAssocLayer replaces AssocUpdateTransactions. The function call is AssocTickAssocLayer(handle). Table 2.20-28 describes the parameters used, return values and possible errors generated by using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
handle	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
channel	register pointer to ChannelDescriptor	/simnet/libsrc/libassoc/assoc_lcl.h
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Unsuccessful.
0	int	Successful.
Errors		
Error Name	**Reason for Error**	
ANETACCESS	Cannot access Ethernet interface.	
Calls		
Function	**Where Described**	
ASSOC_CHECK_HANDLE	Macro defined in /simnet/libsrc/libassoc/assoc_lcl.h.	
AssocTimeOutOldResponse	Section 2.20.1.18.4.	
UpdateTransactions	Section 2.20.1.8.2.	
net_current_time	Section 2.20.2.8.3.	
AssocSendAggregate	Section 2.20.1.3.1.	

Table 2.20-28 AssocTickAssocLayer Information.

2.20.1.8.2 UpdateTransactions

UpdateTransactions is called by AssocTickAssocLayer in order to retry transactions and time out old cached responses. The function call is UpdateTransactions(handle). Table 2.20-29 describes the parameters used, errors returned and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
handle	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
t1	register pointer to AssocTransDescriptor	/simnet/libsrc/libassoc/assoc_lcl.h
channel	register pointer to ChannelDescriptor	/simnet/libsrc/libassoc/assoc_lcl.h
next	pointer to AssocTransDescriptor	/simnet/libsrc/libassoc/assoc_lcl.h
currentTicks	long int	Standard C type

Return Values		
Return Value	**Type**	**Meaning**
-1	int	Unsuccessful.
0	int	Successful.
Errors		
Error Name	**Reason for Error**	
ANETACCESS	Cannot access Ethernet interface.	
Calls		
Function	**Where Described**	
net_current_time	Section 2.20.2.8.3.	
PRO_ASSOC_REQUEST_HDR_SIZE	Macro defined in /simnet/common/include/protocol/p_size.h	
AssocDeleteTransaction	Section 2.20.1.23.3.	
net_snd	Section 2.20.2.19.2.	
AssocRescheduleTransaction	Section 2.20.1.23.5.	
Called By		
Function	**Where Described**	
AssocTickAssocLayer	Section 2.20.1.8.1.	

Table 2.20-29 UpdateTransactions Information.

2.20.1.9 address.c

/simnet/libsrc/libassoc/address.c

address.c contains a routine to get a Simulation Address.

2.20.1.9.1 AssocGetSimAddress

AssocGetSimAddress gets the hosts simulation address. It sets *simAddress* to the site and host numbers of the simulation entity referenced by *handle*. These are taken from the definition file given to AssocOpen(). The function call is AssocGetSimAddress(handle, simAddress). Table 2.20-30 describes the parameters used, return values and possible errors generated by using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
handle	int	Standard C type.
simAddress	pointer to SimulationAddress	/simnet/common/include/protocol/address.h
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Unsuccessful.
0	int	Successful.
Errors		
Error Name	**Reason for Error**	
ABADHANDLE	Invalid network handle.	

Calls	
Function	**Where Described**
ASSOC_CHECK_HANDLE	Macro defined in /simnet/libsrc/libassoc/assoc_lcl.h.

Table 2.20-30 AssocGetSimAddress Information.

2.20.1.10 error.c
/simnet/libsrc/libassoc/error.c

error.c contains all the definitions and routines used for the association layer library's error handling service. Table 2.20-31 describes the variables used by error.c.

Variables		
Variable	**Type**	**Where Typedef Declared**
assoc_errno	int	Standard C type.
errorList	pointer to array MAX_ERROR_STRINGS of char	Standard C type.

Table 2.20-31 error.c Variable Information.

2.20.1.10.1 AssocError

AssocError returns a pointer to a static string that describes the last error encountered during an association library call. The error is obtained by the value in the external variable assoc_errno. AssocError returns a pointer to the string "Error 0" if assoc_errno equals 0. Table 2.20-32 describes the return values generated by this function.

Return Values		
Return Value	**Type**	**Meaning**
errorList	pointer to array assoc_errno of char	String for a valid error number.
errorList	pointer to array MAX_ERROR_STRINGS of char	String for an invalid error number.

2.20.1.11 close.c
/simnet/libsrc/libassoc/close.c

close.c contains a routine to close an association layer channel.

2.20.1.11.1 AssocClose

AssocClose deallocates all memory allocated for a channel, unsubscribes from all subscribed multicast groups and closes the network connection. The function call is AssocClose(handle). Table 2.20-33 describes the parameters used, return values and possible errors generated by using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
handle	int	Standard C type.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
channel	pointer to rgister ChannelDescriptor	/simnet/libsrc/libassoc/assoc_lcl.h
ditch	pointer to pointer to char	Standard C type.
nextDitch	pointer to pointer to char	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Unsuccessful.
association channel (and net) handle	int	Successful.
Errors		
Error Name	**Reason for Error**	
ABADHANDLE	Invalid network handle.	
Calls		
Function	**Where Described**	
ASSOC_CHECK_HANDLE	Macro defined in /simnet/libsrc/libassoc/assoc_lcl.h.	
AssocUnsubscribeWithMask	Section 2.20.1.1.5.	
net_close	Section 2.20.2.3.1.	

Table 2.20-33 AssocClose Information.

2.20.1.12 family.c

/simnet/libsrc/libassoc/family.c

family.c contains a routine to establish a protocol family to be used for a channel.

2.20.1.12.1 AssocSetProtocolFamily

AssocSetProtocolFamily re-initializes the network layer to not accept packets of any type, except the specified family. This is currently done using the ethernet type code. This call is only needed to change types from the default specified in the channel's assocDef file. Family types are defined in assoc.h. They include:

AssocFamilySimnet	listens to SIMNET packets
AssocFamilyData	for network development only
AssocFamilyVoice	listen to voice (radio) packets
AssocFamilyGT	listen to GT sim/cig packets

The function call is AssocSetProtocolFamily(handle, family). Table 2.20-34 describes the parameters used, return values and possible errors generated by using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
handle	int	Standard C type.
family	int	Standard C type.

Return Values		
Return Value	**Type**	**Meaning**
-1	int	Call failed.
0	int	New family is set.
Errors		
Error Name	**Reason for Error**	
ANETACCESS	Cannot access interface to the Ethernet.	
ABADHANDLE	Invalid handle (have you called AssocOpen()?).	
Calls		
Function	**Where Described**	
ASSOC_CHECK_HANDLE	Macro defined in /simnet/libsrc/libassoc/assoc_lcl.h.	
net_init_type	Section 2.20.2.18.2.	
net_add_type	Section 2.20.2.18.1.	
net_set_snd_type	Section 2.20.2.19.8.	

Table 2.20-34 AssocSetProtocolFamily Information.

2.20.1.13 mask.c

/simnet/libsrc/libassoc/mask.c

mask.c contains routines to use address masks. Address masks allow the user to utilize available bits in the ethernet multicast address to create private multicast groups. This is considered and extension to, not part of, the Association Layer.

2.20.1.13.1 AssocSetSendMask

AssocSetSendMask allows the used to specify a mask to be bit-wise *OR*ed into a multicast destination address before it is sent. This allows users to form private multicast groups. Bits 0->23 of the mask are *OR*ed with bits 8->31 of the destination address to form bits 8->31 of the actual multicast address to be used. Normally , these bits are 0. The mask set by AssocSetSendMask is used for sending datagrams and transaction requests. Transaction responses normally use the mask in the address to which the originator sent the request. The function call is AssocSetSendMask(handle, mask). Table 2.20-35 describes the parameters used, return values and possible errors generated by using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
handle	int	Standard C type.
mask	unsigned long	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Unsuccessful.
0	int	Successful.
Errors		
Error Name	**Reason for Error**	
ABADHANDLE	Invalid handle (have you called AssocOpen()?)	

Calls	
Function	**Where Described**
ASSOC_CHECK_HANDLE	Macro defined in /simnet/libsrc/libassoc/assoc_lcl.h.

Table 2.20-35 AssocSetSendMask Information.

2.20.1.13.2 AssocGetRspMask

AssocGetRspMask is used in conjunction with AssocSetRspMask to allow the user to process transactions in different groups out of order. AssocGetRspMask is used to retrieve a response mask when a transaction is first received. The mask is then used by AssocSetRspMask in the response. The function call is AssocGetRspMask(handle, mask). Table 2.20-36 describes the parameters used, return values and possible errors generated by using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
handle	int	Standard C type.
mask	unsigned long	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Unsuccessful.
0	int	Successful.
Errors		
Error Name	**Reason for Error**	
ABADHANDLE	Invalid handle (have you called AssocOpen()?).	
Calls		
Function	**Where Described**	
ASSOC_CHECK_HANDLE	Macro defined in /simnet/libsrc/libassoc/assoc_lcl.h.	

Table 2.20-36 AssocGetRspMask Information.

2.20.1.13.3 AssocSetRspMask

AssocSetRspMask is used with AssocGetRspMask. The mask retrieved by AssocGetRspMask is used to send a response by first calling AssocSetRspMask and then AssocSendResponse. The function call is AssocSetRspMask(handle, mask). Table 2.20-37 describes the parameters used, return values and possible errors generated by using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
handle	int	Standard C type.
mask	unsigned long	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Unsuccessful.
0	int	Successful.

Errors	
Error Name	**Reason for Error**
ABADHANDLE	Invalid handle (have you called AssocOpen()?).
Calls	
Function	**Where Described**
ASSOC_CHECK_HANDLE	Macro defined in /simnet/libsrc/libassoc/assoc_lcl.h.

Table 2.20-37 AssocSetRspMask Information.

2.20.1.14 raw.c

/simnet/libsrc/libassoc/raw.c

raw.c contains a routine to receive raw Association PDUs from the SIMNET network through the association layer. Table 2.20-38 describes the variables used by raw.c.

Variables		
Variable	**Type**	**Where Typedef Declared**
errno	extern int	Standard C type.

Table 2.20-38 raw.c Variable Information.

2.20.1.14.1 AssocReceiveAssocPDU

AssocReceiveAssocPDU gets a pointer to the next available Association PDU (APDU) in the network buffers and sets what*data* points to to this pointer when appropriate. The data pointer will not be set if the APDU is a response to a transaction the application issued or if the APDU is a retry for which the application has a response cached. If the APDU is a response to a transaction the application issued, the association entity calls the supplied callback routine and returns with *primitive* equal to ASSOC_TRANSACT_CONF. If the APDU is a retry request for which the application has a cached response, the association entity retransmits the response and returns with *primitive* equal to ASSOC_NOTHING_OF_INTEREST. When AssocReceiveAssocPDU returns a pointer to an APDU, protocol is the user protocol to which the PDU contained in the APDU belongs, and *primitive* indicates whether the PDU is a datagram, request, etc., as defined in "assoc.h". *length* is the number of bytes in the information pointed to by *data*. The function call is AssocReceiveAssocPDU(handle, data, length, primitive). Table 2.20-39 describes the parameters used, return values and possible errors generated by using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
handle	int	Standard C type.
data	pointer to pointer to char	Standard C type.
length	pointer to long int	Standard C type.
primitive	pointer to long int	Standard C type.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
channel	register pointer to ChannelDescriptor	/simnet/libsrc/libassoc/assoc_lcl.h
inBuf	pointer to char	Standard C type.
apdu	pointer to AssociationPDU	/simnet/common/include/protocol/p_assoc.h
packetLen	int	Standard C type.
retVal	int	Standard C type.
pdu_data	pointer to char	Standard C type.
pdu_length	long int	Standard C type.
group	MulticastGroupID	/simnet/common/include/protocol/p_assoc.h
protocol	AssociationUserProtocol	/simnet/common/include/protocol/p_assoc.h
originator	SimulationAddress	/simnet/common/include/protocol/address.h
transID	TransactionIdentifier	/simnet/common/include/protocol/address.h
respondent	SimulationAddress	/simnet/common/include/protocol/address.h

Return Values		
Return Value	**Type**	**Meaning**
-1	int	Read failed.
0	int	Data is returned.
1	int	No data is returned and network buffers are not empty.
2	int	Network buffers are empty.

Errors	
Error Name	**Reason for Error**
ANETACCESS	Cannot access interface to the Ethernet.

Calls	
Function	**Where Described**
ASSOC_CHECK_HANDLE	Macro defined in /simnet/libsrc/libassoc/assoc_lcl.h.
net_release_rcv	Section 2.20.2.17.10.
net_get_rcv	Section 2.20.2.17.9.
AssocProcessRequestPDU	Section 2.20.1.20.1.
AssocProcessResponsePDU	Section 2.20.1.19.1.

Table 2.20-39 AssocReceiveAssocPDU Information.

Table 2.20-40 lists the values returned in primitive.

Value	Description
ASSOC_NOTHING_OF_INTEREST	Padding PDU or Association Layer retransmitted a cached response.
ASSOC_DATAGRAM_IND	Datagram PDU.
ASSOC_REQUEST_IND	Transaction request to which we are an observer.
ASSOC_RESPONSE_IND	Transaction response to which we are an observer.
ASSOC_TRANSACT_IND	Transaction request destined for my host.
ASSOC_TRANSACT_CONF	Association layer process a callback routine.

Table 2.20-40 Values Returned in Primitive of AssocReceiveAssocPDU.

2.20.1.15 who.c
/simnet/libsrc/libassoc/who.c

who.c contains a routine to determine the ethernet addres of the current packet on each channel.

2.20.1.15.1 AssocGetLastAddress

AssocGetLastAddress sets the buffer pointed to by *who* to the physical ethernet address of the sender of the last packet which was returned to the application (or Null if none has been received). This can be useful for debugging the network. This is considered an extension to, rather than a part of, the association layer. The function call is AssocGetLastAddress(handle, who). Table 2.20-41 describes the parameters used, return values and possible errors generated by using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
handle	int	Standard C type.
who	pointer to NetworkAddress	/simnet/libsrc/libnetif/network.h
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Unsuccessful.
0	int	Successful.
Errors		
Error Name	**Reason for Error**	
ABADHANDLE	Invalid handle (have you called AssocOpen()?).	
Calls		
Function	**Where Described**	
ASSOC_CHECK_HANDLE	Macro defined in /simnet/libsrc/libassoc/assoc_lcl.h.	
net_get_rcv_from_addr	Section 2.20.2.17.4.	

Table 2.20-41 AssocGetLastAddress Information.

2.20.1.16 time_list.c
/simnet/libsrc/libassoc/time_list.c

time_list.c contains routines used to maintain linked lists of pending transactions. These lists are linked in order by time. Each transaction in the list has a pointer to the previous transaction in time (the timeFront pointer) and to the next transaction in time (the timeRear pointer).

2.20.1.16.1 AssocAddToStartOfTimeList

AssocAddToStartOfTimeList adds the transaction*td* to the start of the TimeList. The routine makes sure that *startTimeList* points to the first transaction pointer on the TimeList and *endTimeList* points to the last transaction pointer on the TimeList. The function call is AssocAddToStartOfTimeList(td, startTimeList, endTimeList). Table 2.20-42 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
td	pointer to AssocTransDescriptor	/simnet/libsrc/libassoc/assoc_lcl.h
startTimeList	pointer to pointer to AssocTransDescriptor	/simnet/libsrc/libassoc/assoc_lcl.h
endTimeList	pointer to pointer to AssocTransDescriptor	/simnet/libsrc/libassoc/assoc_lcl.h

Table 2.20-42 AssocAddToStartOfTimeList Information.

2.20.1.16.2 AssocAddToEndOfTimeList

AssocAddToEndOfTimeList adds the transaction*td* to the end of the TimeList. The routine makes sure that *startTimeList* points to the first transaction pointer on the TimeList and *endTimeList* points to the last transaction pointer on the TimeList. The function call is AssocAddToEndOfTimeList(td, startTimeList, endTimeList). Table 2.20-43 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
td	pointer to AssocTransDescriptor	/simnet/libsrc/libassoc/assoc_lcl.h
startTimeList	pointer to pointer to AssocTransDescriptor	/simnet/libsrc/libassoc/assoc_lcl.h
endTimeList	pointer to pointer to AssocTransDescriptor	/simnet/libsrc/libassoc/assoc_lcl.h
Called By		
Function	**Where Described**	
AssocCacheResponse	Section 2.20.1.18.2.	
AssocAddTransaction	Section 2.20.1.23.2.	

Table 2.20-43 AssocAddToEndOfTimeList Information.

2.20.1.16.3 AssocDeleteFromTimeList

AssocDeleteFromTimeList deletes the transaction *td* from the TimeList. The routine makes sure that *startTimeList* points to the first transaction pointer on the TimeList, that *endTimeList* points to the last transaction pointer on the TimeList, and that the TimeList elements remain properly linked by the timerFront and timeRear pointers when *td* is removed from the list. The function call is AssocDeleteFromTimeList(td, startTimeList, endTimeList). Table 2.20-44 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
td	pointer to AssocTransDescriptor	/simnet/libsrc/libassoc/assoc_lcl.h
startTimeList	pointer to pointer to AssocTransDescriptor	/simnet/libsrc/libassoc/assoc_lcl.h
endTimeList	pointer to pointer to AssocTransDescriptor	/simnet/libsrc/libassoc/assoc_lcl.h
Called By		
Function	**Where Described**	
AssocDeleteCachedResponse	Section 2.20.1.18.3.	
AssocDeleteTransaction	Section 2.20.1.23.3.	

Table 2.20-44 AssocDeleteFromTimeList Information.

2.20.1.16.4 AssocMoveToEndOfTimeList

AssocMoveToEndOfTimeList moves the transaction *td* to the end of the TimeList. The routine makes sure that *startTimeList* points to the first transaction pointer on the TimeList and *endTimeList* points to the last transaction pointer on the TimeList. The function call is AssocMoveToEndOfTimeList(td, startTimeList, endTimeList). Table 2.20-45 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
td	pointer to AssocTransDescriptor	/simnet/libsrc/libassoc/assoc_lcl.h
startTimeList	pointer to pointer to AssocTransDescriptor	/simnet/libsrc/libassoc/assoc_lcl.h
endTimeList	pointer to pointer to AssocTransDescriptor	/simnet/libsrc/libassoc/assoc_lcl.h
Called By		
Function	**Where Described**	
AssocRescheduleTransaction	Section 2.20.1.23.5.	

Table 2.20-45 AssocMoveToEndOfTimeList Information.

2.20.1.17 strtok.c
/simnet/libsrc/libassoc/strtok.c

Table 2.20-46 describes the variables used by strtok.c.

Variables		
Variable	**Type**	**Where Typedef Declared**
tok_buf	array 512 of char	Standard C type.
tok_cp	pointer to char	Standard C type.

Table 2.20-46 strtok.c Variable Information.

2.20.1.17.1 strtok

strtok is a utility function used to parse strings. It performs the same functionality as the unix function strtok(). The function call is strtok(operand, set). Table 2.20-47 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
operand	pointer to char	Standard C type.
set	pointer to char	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
ret_val	pointer to char	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
0	pointer to char	the string was empty
ret_val	pointer to char	the parsed string
Called By		
Function	**Where Described**	
AssocReadParams	Section 2.20.1.22.2.	

Table 2.20-47 strtok Information.

2.20.1.18 respondent.c
/simnet/libsrc/libassoc/respondent.c

respondent.c contains routines to implement the portion of the transaction protocol where this host is the respondent. Each element of responseMap is a pointer to a list or "bucket" of response descriptors. The transaction IDs of all transactions in a bucket have low order 7 bits that correspond to the map index, meaning that the low order 7 bits can be used to index the appropriate bucket.

2.20.1.18.1 AssocInitResponse

AssocInitResponse is called by AssocOpen to set up a response bucket map. The function call is AssocInitResponse(handle). Table 2.20-48 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
handle	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	register int	Standard C type.
rd	register pointer to pointer to AssocTransDescriptor	/simnet/libsrc/libassoc/assoc_lcl.h
channel	pointer to ChannelDescriptor	/simnet/libsrc/libassoc/assoc_lcl.h
Called By		
Function	**Where Described**	
AssocOpen	Section 2.20.1.5.1.	
AssocAttach	Section 2.20.1.5.2.	

Table 2.20-48 AssocInitResponse Information.

2.20.1.18.2 AssocCacheResponse

AssocCacheResponse is called by AssocSendResponse (Section 2.20.1.4.2) when a cached response is sent by the host. AssocSendResponse gets a descriptor for the response and fills in the pdubuf and length fields. AssoCacheResponse then links the response into the appropriate bucket for its transaction id and fills in any other descriptor-specific fields. The function call is AssocCacheResponse(handle, rd). Table 2.20-49 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
handle	int	Standard C type.
rd	pointer to AssocTransDescriptor	/simnet/libsrc/libassoc/assoc_lcl.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
apdu	register pointer to AssociationPDU	/simnet/common/include/protocol/p_assoc.h
channel	register pointer to ChannelDescriptor	/simnet/libsrc/libassoc/assoc_lcl.h
Calls		
Function	**Where Described**	
net_current_time	Section 2.20.2.8.3.	
AssocAddToBucket	Section 2.20.1.25.1.	
AssocAddToEndOfTimeList	Section 2.20.1.16.2.	

Called By	
Function	**Where Described**
AssocSendResponse	Section 2.20.1.4.2.

Table 2.20-49 AssocCacheResponse Information.

2.20.1.18.3 AssocDeleteCachedResponse

AssocDeleteCachedResponse is called when a cached response times out. The response is removed from the time list, removed from the bucket list, and added to the free list. The parameter *rd* references the response transaction descriptor. The function call is AssocDeleteCachedResponse(handle, rd). Table 2.20-50 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
handle	int	Standard C type.
rd	pointer to AssocTransDescriptor	/simnet/libsrc/libassoc/assoc_l cl.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
channel	register pointer to ChannelDescriptor	/simnet/libsrc/libassoc/assoc_l cl.h
Calls		
Function	**Where Described**	
AssocDeleteFromTimeList	Section 2.20.1.16.3.	
AssocDeleteFromBucket	Section 2.20.1.25.2.	
AssocFreeDescriptor	Section 2.20.1.24.4.	
Called By		
Function	**Where Described**	
AssocTimeOutOldResponses	Section 2.20.1.18.4.	

Table 2.20-50 AssocDeleteCachedResponse Information.

2.20.1.18.4 AssocTimeOutOldResponses

AssocTimeOutOldResponses goes through the time list and deletes all responses older than CacheTime. It is called either from AssocTickAssocLayer or when the FreeList is empty, in an attempt to free up some space. The function call is AssocTimeOutOldResponses(handle). Table 2.20-51 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
handle	int	Standard C type.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
t1	register pointer to AssocTransDescriptor	/simnet/libsrc/libassoc/assoc_lcl.h
next	pointer to AssocTransDescriptor	/simnet/libsrc/libassoc/assoc_lcl.h
channel	pointer to ChannelDescriptor	/simnet/libsrc/libassoc/assoc_lcl.h
currentTicks	long int	Standard C type.
Calls		
Function	**Where Described**	
net_current_time	Section 2.20.2.8.3.	
AssocDeletedCachedResponse	Section 2.20.1.18.3.	
Called By		
Function	**Where Described**	
AssocTickAssocLayer	Section 2.20.1.8.1.	
AssocGetDescriptor	Section 2.20.1.24.3.	

Table 2.20-51 AssocTimeOutOldResponses Information.

2.20.1.18.5 AssocFindResponse

AssocFindResponse is called when a transaction request arrives, to see if there is a response cached and waiting for it. The function call is AssocFindResponse(handle, originator, tid). Table 2.20-52 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
handle	int	Standard C type.
originator	pointer to Simulation Address	/simnet/common/include/protocol/address.h
tid	TransactionIdentifier	/simnet/common/include/protocol/address.h
Return Values		
Return Value	**Type**	**Meaning**
AssocBucketLookup (originator, tid, channelMap[handle] -> responseMap (RESPONSE_ID_MASK))	pointer to AssocTransDescriptor	The transaction descriptor of a cached response to the transaction request.
Calls		
Function	**Where Described**	
AssocBucketLookup	Section 2.20.1.25.3.	

Called By	
Function	**Where Described**
AssocProcessRequestPDU	Section 2.20.1.20.1.

Table 2.20-52 AssocFindResponse Information.

2.20.1.19 proc_rsp.c
/simnet/libsrc/libassoc/proc_rsp.c

proc_rsp.c contains a routine to process a response APDU.

2.20.1.19.1 AssocProcessResponsePDU

AssocProcessResponsePDU is called by AssocReceivePDU in order to process a response PDU. The routine first determines if the response was intended for the invoking process. If so, it determines which transaction was being responded to, invokes the call back routine, and deletes the transaction from the pending list. If the transaction cannot be found, the response is thrown away. If the response was not intended for the invoking process, it is treated as a datagram. The function call is AssocProcessResponsePDU(handle, apdu, data, length, group, protocol, primitive, originator, transID, respondent, rc). Table 2.20-53 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
handle	int	Standard C type.
apdu	register pointer to AssociationPDU	/simnet/common/include/protocol/p_assoc.h
data	pointer to pointer to char	Standard C type.
length	pointer to long int	Standard C type.
group	pointer to MulticastGroupID	/simnet/common/include/protocol/p_assoc.h
protocol	pointer to AssociationUserProtocol	/simnet/common/include/protocol/p_assoc.h
primitive	pointer to long int	Standard C type.
originator	pointer to SimulationAddress	/simnet/common/include/protocol/address.h
transID	pointer to TransactionIdentifier	/simnet/common/include/protocol/address.h
respondent	pointer to SimulationAddress	/simnet/common/include/protocol/address.h
rc	pointer to int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
td	register pointer to AssocTransDescriptor	/simnet/libsrc/libassoc/assoc_!cl.h
Calls		
Function	**Where Described**	
ASSOC_ADDRESS_EQUAL	Macro defined in /simnet/libsrc/libassoc/assoc.h.	
AssocFindTransaction	Section 2.20.1.23.4.	
AssocDeleteTransaction	Section 2.20.1.23.3.	

Called By	
Function	**Where Described**
AssocReceivePDU	Section 2.20.1.6.1.
AssocWaitForPDU	Section 2.20.1.7.1.
AssocReceiveAssocPDU	Section 2.20.1.14.1.

Table 2.20-54 AssocProcessResponsePDU Information.

2.20.1.20 proc_req.c
/simnet/libsrc/libassoc/proc_req.c

proc_req.c contains a routine to process a request APDU.

2.20.1.20.1 AssocProcessRequestPDU

AssocProcessRequestPDU is called by AssocReceivePDU in order to process a request PDU. The routine first determines if the request was intended for the invoking process. If so, it determines whether a response is cached for the request. If no response is cached, the request is passed to the host for processing. If a response is cached, it is re-transmitted. If the request was not intended for the invoking process, it is treated as a datagram. The function call is AssocProcessRequestPDU(handle, apdu, data, length, group, protocol, primitive, originator, transID, respondent, rc). Table 2.20-55 describes the parameters used, errors returned and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
handle	int	Standard C type.
apdu	register pointer to AssociationPDU	/simnet/common/include/protocol/p_assoc.h
data	pointer to pointer to char	Standard C type.
length	pointer to long int	Standard C type.
group	pointer to MulticastGroupID	/simnet/common/include/protocol/p_assoc.h
protocol	pointer to AssociationUserProtocol	/simnet/common/include/protocol/p_assoc.h
primitive	pointer to long int	Standard C type.
originator	pointer to SimulationAddress	/simnet/common/include/protocol/address.h
transID	pointer to TransactionIdentifier	/simnet/common/include/protocol/address.h
respondent	pointer to SimulationAddress	/simnet/common/include/protocol/address.h
rc	pointer to int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
rd	register pointer to AssocTransDescriptor	/simnet/libsrc/libassoc/assoc_lcl.h
reqAddress	NetworkAddress	/simnet/libsrc/libnetif/network.h

Return Values		
Return Value	**Type**	**Meaning**
-1	int	Unsuccessful.
0	int	Successful.
Errors		
Error Name	**Reason for Error**	
ANETACCESS	Cannot access interface to the Ethernet	
Calls		
Function	**Where Described**	
ASSOC_ADDRESS_EQUAL	Macro defined in /simnet/libsrc/libassoc/assoc.h.	
AssocFindResponse	Section 2.20.1.18.5.	
net_get_rcv_to_addr	Section 2.20.2.17.3.	
net_snd	Section 2.20.2.19.2.	
Called By		
Function	**Where Described**	
AssocReceivePDU	Section 2.20.1.6.1.	
AssocWaitForPDU	Section 2.20.1.7.1.	
AssocReceiveAssocPDU	Section 2.20.1.14.1.	

Table 2.20-55 AssocProcessRequestPDU Information.

2.20.1.21 proc_dgram.c

/simnet/libsrc/libassoc/proc_dgram.c

proc_dgram.c contains a routine for processing datagrams.

2.20.1.21.1 AssocProcessDatagramPDU

AssocProcessDatagramPDU is used to process datagrams, however, is not called with the version 6.6 code; the macro PROCESS_DATAGRAM_PDU is called instead.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
apdu	register pointer to AssociationPDU	/simnet/common/include/protocol/p_assoc.h
data	pointer to pointer to char	Standard C type.
length	pointer to long int	Standard C type.
group	pointer to MulticastGroupID	/simnet/common/include/protocol/p_assoc.h
protocol	pointer to AssociationUserProtocol	/simnet/common/include/protocol/p_assoc.h
primitive	pointer to long int	Standard C type.
originator	pointer to SimulationAddress	/simnet/common/include/protocol/address.h

Table 2.20-56 AssocProcessDatagramPDU Information.

2.20.1.22 params.c
/simnet/libsrc/libassoc/params.c

params.c contains routines to get the association layer parameters for one channel from an "assoc.def" file. Table 2.20-57 describes the variables used by params.c.

Variables		
Variable	**Type**	**Where Typedef Declared**
params	array numParams of PARAM	/simnet/libsrc/libassoc/params.c
families	array MAX_FAMILIES of struct families	/simnet/libsrc/libassoc/params.c

Table 2.20-57 params.c Variable Information.

2.20.1.22.1 upshift

upshift converts tokens from assoc.def to upper case. The function call is upshift(str). Table 2.20-58 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
str	pointer to char	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
cp	register pointer to char	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
str	pointer to char	Upshifted token.
Called By		
Function	**Where Described**	
AssocReadParams	Section 2.20.1.22.2.	
ProcessProtocolFamily	Section 2.20.1.22.8.	

Table 2.20-58 upshift Information.

2.20.1.22.2 AssocReadParams

AssocReadParams is called to read in the ASCII parameter file, *assocDef*. The function call is AssocReadParams(handle, assocDef). Table 2.20-59 describes the parameters used, errors returned and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
handle	int	Standard C type.
assocDef	pointer to char	Standard C type.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
def	pointer to FILE	Standard C type.
buf	array ASSOC_PARM_MAX_LINE_ 1 of char	Standard C type.
tokPtr	pointer to char	Standard C type.
channel	pointer to ChannelDescriptor	/simnet/libsrc/libassoc/assoc_l cl.h
p	register pointer to PARAM	/simnet/libsrc/libassoc/params. c
i	register int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Unsuccessful.
0	int	Successful.
Errors		
Error Name	**Reason for Error**	
AFILEOPEN	Cannot open the assocDef file.	
APARM	Missing or invalid site and/or hose parameter is assocDef file.	
Calls		
Function	**Where Described**	
ASSOC_CHECK_HANDLE	Macro defined in /simnet/libsrc/libassoc/assoc_lcl.h.	
strtok	Section 2.20.1.17.1.	
upshift	Section 2.20.1.22.1.	
Called By		
Function	**Where Described**	
AssocOpen	Section 2.20.1.5.1.	
AssocAttach	Section 2.20.1.5.2.	

Table 2.20-59 AssocReadParams Information.

2.20.1.22.3 ProcessSite

ProcessSite processes the site parameter (read from the assocDef file) for the channel described by *channel*. The function call is ProcessSite(channel, tokPtr). Table 2.20-60 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
channel	pointer to ChannelDescriptor	/simnet/libsrc/libassoc/assoc_l cl.h
tokPtr	pointer to char	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
site	int	Standard C type.

Return Values		
Return Value	**Type**	**Meaning**
-1	int	Unsuccessful.
0	int	Successful.

Table 2.20-60 ProcessSite Information.

2.20.1.22.4 ProcessHost

ProcessHost processes the host parameter (read from the assocDef file) for the channel described by *channel*. The function call is ProcessHost(channel, tokPtr). Table 2.20-61 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
channel	pointer to ChannelDescriptor	/simnet/libsrc/libassoc/assoc_lcl.h
tokPtr	pointer to char	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
host	int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Unsuccessful.
0	int	Successful.

Table 2.20-61 ProcessHost Information.

2.20.1.22.5 ProcessMaxSubscriptions

ProcessMaxSubscriptions processes the maxSubscriptions parameter (read from the assocDef file) for the channel described by *channel*. The function call is ProcessMaxSubscriptions(channel, tokPtr). Table 2.20-62 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
channel	pointer to ChannelDescriptor	/simnet/libsrc/libassoc/assoc_lcl.h
tokPtr	pointer to char	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
subscriptions	int	Standard C type
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Unsuccessful.
0	int	Successful.

Table 2.20-62 ProcessMaxSubscriptions Information

2.20.1.22.6 ProcessInitDescriptors

ProcessInitDescriptors processes the initDescriptors parameter (read from the assocDef file) for the channel described by *channel*. The function call is ProcessInitDescriptors(channel, tokPtr). Table 2.20-63 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
channel	pointer to ChannelDescriptor	/simnet/libsrc/libassoc/assoc_lcl.h
tokPtr	pointer to char	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
initDescriptors	int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Unsuccessful.
0	int	Successful.

Table 2.20-63 ProcessInitDescriptors Information.

2.20.1.22.7 ProcessAddlDescriptors

ProcessAddlDescriptors processes the addlDescriptors parameter (read from the assocDef file) for the channel described by *channel*. The function call is ProcessAddlDescriptors(channel, tokPtr). Table 2.20-64 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
channel	pointer to ChannelDescriptor	/simnet/libsrc/libassoc/assoc_lcl.h
tokPtr	pointer to char	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
addlDescriptors	int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Unsuccessful.
0	int	Successful.

Table 2.20-64 ProcessAddlDescriptors Information.

2.20.1.22.8 ProcessProtocolFamily

ProcessProtocolFamily processes the family parameter (read from the assocDef file) for the channel described by *channel*. The function call is ProcessProtocolFamily(channel, tokPtr). Table 2.20-65 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
channel	pointer to ChannelDescriptor	/simnet/libsrc/libassoc/assoc_l cl.h
tokPtr	pointer to char	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	register int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Unsuccessful.
0	int	Successful.
Calls		
Function	**Where Described**	
upshift	Section 2.20.1.22.1.	

Table 2.20-65 ProcessProtocolFamily Information.

2.20.1.23 origin.c

/simnet/libsrc/libassoc/origin.c

origin.c contains routines to implement that portion of the transaction protocol where this host is the originator.

2.20.1.23.1 AssocInitTransactions

AssocInitTransactions is called by AssocOpen to set up a transaction bucket map. The function call is AssocInitTransactions(handle). Table 2.20-66 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
handle	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	register int	Standard C type.
td	register pointer to pointer to AssocTransDescriptor	/simnet/libsrc/libassoc/assoc_l cl.h
channel	register pointer to ChannelDescriptor	/simnet/libsrc/libassoc/assoc_l cl.h

Called By	
Function	**Where Described**
AssocOpen	Section 2.20.1.5.1.
AssocAttach	Section 2.20.1.5.2.

Table 2.20-66 AssocInitTransactions Information.

2.20.1.23.2 AssocAddTransaction

AssocAddTransaction is called when the host sends a transaction. It is placed at the head of the appropriate transaction bucket, and put at the end of the timeList. AssocAddTransaction assumes that the transaction descriptor, *td*, already has the PDU copied in. The function call is AssocAddTransaction(handle, td, callback, cparam, timeout, tparam). Table 2.20-67 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
handle	int	Standard C type.
td	register pointer to AssocTransDescriptor	/simnet/libsrc/libassoc/assoc_lcl.h
(callback)()	pointer to function returning int	Standard C type.
cparam	pointer to long int	Standard C type.
(timeout)()	pointer to function returning int	Standard C type.
tparam	pointer to long int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
apdu	register pointer to AssociationPDU	/simnet/common/include/protocol/p_assoc.h
channel	pointer to ChannelDescriptor	/simnet/libsrc/libassoc/assoc_lcl.h
Calls		
Function	**Where Described**	
net_current_time	Section 2.20.2.8.3.	
AssocAddToEndOfTimeList	Section 2.20.1.16.2.	
AssocAddToBucket	Section 2.20.1.25.1.	
Called By		
Function	**Where Described**	
AssocSendTransact	Section 2.20.1.4.1.	

Table 2.20-67 AssocAddTransaction Information.

2.20.1.23.3 AssocDeleteTransaction

AssocDeleteTransaction removes a transaction either 1) when a response is received, or 2) when it has been sent retryCountTime. The function call is AssocDeleteTransaction(handle, td). Table 2.20-68 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
handle	int	Standard C type.
td	pointer to AssocTransDescriptor	/simnet/libsrc/libassoc/assoc_lcl.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
channel	pointer to ChannelDescriptor	/simnet/libsrc/libassoc/assoc_lcl.h
Calls		
Function	**Where Described**	
AssocDeleteFromTimeList	Section 2.20.1.16.3.	
AssocDeleteFromBucket	Section 2.20.1.25.2.	
AssocFreeDescriptor	Section 2.20.1.24.4.	
Called By		
Function	**Where Described**	
UpdateTransactions	Section 2.20.1.8.2.	
AssocProcessResponsePDU	Section 2.20.1.19.1.	

Table 2.20-68 AssocDeleteTransaction Information.

2.20.1.23.4 AssocFindTransaction

When a response PDU arrives, AssocFindTransaction checks to see if a transaction with the same transaction id exists. The function call is AssocFindTransaction(handle, tid). Table 2.20-69 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
handle	int	Standard C type.
tid	TransactionIdentifier	/simnet/common/include/protocol/address.h
Return Values		
Return Value	**Type**	**Meaning**
	pointer to AssocTransDescriptor	/simnet/libsrc/libassoc/assoc_lcl.h
Calls		
Function	**Where Described**	
AssocBucketLookup	Section 2.20.1.25.3.	

Called By	
Function	**Where Described**
AssocProcessResponsePDU	Section 2.20.1.19.1.

Table 2.20-69 AssocFindTransaction Information.

2.20.1.23.5 AssocRescheduleTransaction

AssocRescheduleTransaction is called after a transaction has been retried. The transaction descriptor is moved to the end of the time list and a new retry time is calculated. The function call is AssocRescheduleTransaction(handle, td). Table 2.20-70 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
td	pointer to AssocTransDescriptor	/simnet/libsrc/libassoc/assoc_lcl.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
channel	pointer to ChannelDescriptor	/simnet/libsrc/libassoc/assoc_lcl.h
Calls		
Function	**Where Described**	
AssocMoveToEndOfTimeList	Section 2.20.1.16.4.	
Called By		
Function	**Where Described**	
UpdateTransactions	Section 2.20.1.18.2.	

Table 2.20-70 AssocRescheduleTransaction Information.

2.20.1.24 free_list.c
/simnet/libsrc/libassoc/free_list.c

free_list.c contains routines to manage the free list of transaction descriptors. The free list is created as a stack at intialization time, to avoid calling malloc() or free() each time a descriptor is needed.

2.20.1.24.1 AssocCreateFreeList

AssocCreateFreeList creates the free list by allocating enough space for a certain number of descriptors (ASSOC_INITIAL_DESCRIPTORS + an extra char). The function call is AssocCreateFreeList(handle). Table 2.20-71 describes the parameters used and errors returned using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
handle	int	Standard C type.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	register int	Standard C type.
td	register pointer to AssocTransDescriptor	/simnet/libsrc/libassoc/assoc_lcl.h
channel	register pointer to ChannelDescriptor	/simnet/libsrc/libassoc/assoc_lcl.h
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Didn't have enough memory to create list.
0	int	Successful.
Errors		
Error Name	**Reason for Error**	
ANOMEMORY	Not enough memory for internal Association structures.	
Called By		
Function	**Where Described**	
AssocOpen	Section 2.20.1.5.1.	
AssocAttach	Section 2.20.1.5.2.	

Table 2.20-71 AssocCreateFreeList Information.

2.20.1.24.2 AssocGrowFreeList

AssocGrowFreeList mallocs more nodes (descriptor parameters) to the free list when the bottom of the stack is hit. It is assumed that the stack is empty when this routine is called. The function call is AssocGrowFreeList(handle). Table 2.20-72 describes the parameters used, errors returned and functions called using this function.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	register int	Standard C type.
td	register pointer to AssocTransDescriptor	/simnet/libsrc/libassoc/assoc_lcl.h
channel	register pointer to ChannelDescriptor	/simnet/libsrc/libassoc/assoc_lcl.h
ditch	pointer to pointer to char	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Didn't have enough memory to create list.
0	int	Successful.
Errors		
Error Name	**Reason for Error**	
ANOMEMORY	Not enough memory for internal Association structures.	

Called By	
Function	**Where Described**
AssocGetDescriptor	Section 2.20.1.24.3.

Table 2.20-72 AssocGrowFreeList Information.

2.20.1.24.3 AssocGetDescriptor

AssocGetDescriptor attempts to pop a node off the stack. If the stack has some nodes, the descriptor is returned. If you are at the bottom of the free list, the routine first tries to free nodes from the response list by calling AssocTimeOutOldResponses. If AssocTimeOutOldResponses is unsuccessful in freeing up a node, AssocGrowFreeList is called. If AssocGrowFreeList is unsuccessful, Null is returned. The function call is AssocGetDescriptor(handle). Table 2.20-73 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
handle	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
td	register pointer to AssocTransDescriptor	/simnet/libsrc/libassoc/assoc_lcl.h
channel	register pointer to ChannelDescriptor	/simnet/libsrc/libassoc/assoc_lcl.h
Return Values		
Return Value	**Type**	**Meaning**
Null	pointer to AssocTransDescriptor	there are no free nodes
td	pointer to AssocTransDescriptor	address of the free list transaction descriptor
Calls		
Function	**Where Described**	
AssocTimeOutOldResponse	Section 2.20.1.18.4.	
AssocGrowFreeList	Section 2.20.1.24.2.	
Called By		
Function	**Where Described**	
AssocSendTransact	Section 2.20.1.4.1.	
AssocSendResponse	Section 2.20.1.4.2.	

Table 2.20-73 AssocGetDescriptor Information.

2.20.1.24.4 AssocFreeDescriptor

AssocFreeDescriptor pushes a node onto the stack. The function call is AssocFreeDescriptor(handle, td). Table 2.20-74 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
handle	int	Standard C type.
td	register pointer to AssocTransDescriptor	/simnet/libsrc/libassoc/assoc_l cl.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
channel	register pointer to ChannelDescriptor	/simnet/libsrc/libassoc/assoc_l cl.h
Called By		
Function	**Where Described**	
AssocSendTransact	Section 2.20.1.4.1.	
AssocSendResponse	Section 2.20.1.4.2.	
AssocDeleteCachedRespons e	Section 2.20.1.18.3.	
AssocDeleteTransaction	Section 2.20.1.23.3.	

Table 2.20-74 AssocFreeDescriptor Information.

2.20.1.25 bucket.c

/simnet/libsrc/libassoc/bucket.c

bucket.c contains routines to maintain bucket lists of pending transactions. Transactions are sorted into buckets by the low order 7 bits of their transaction ID, resulting in the possibility of each bucket having a linked list of transactions.

2.20.1.25.1 AssocAddToBucket

AssocAddToBucket adds a transaction to a bucket, linking it to any other transactions already in the bucket. The function call is AssocAddToBucket(td, map, mask). Table 2.20-75 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
td	pointer to AssocTransDescriptor	/simnet/libsrc/libassoc/assoc_l cl.h
map	pointer to pointer to AssocTransDescriptor	/simnet/libsrc/libassoc/assoc_l cl.h
mask	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
bucket	register pointer to pointer to AssocTransDescriptor	/simnet/libsrc/libassoc/assoc_l cl.h

Called By	
Function	**Where Described**
AssocCacheResponse	Section 2.20.1.18.2.
AssocAddTransaction	Section 2.20.1.23.2.

Table 2.20-75 AssocAddToBucket Information.

2.20.1.25.2 AssocDeleteFromBucket

AssocDeleteFromBucket removes a transaction from a bucket, maintaining any links within the bucket. The function call is AssocDeleteFromBucket(td, map, mask). Table 2.20-76 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
td	pointer to AssocTransDescriptor	/simnet/libsrc/libassoc/assoc_lcl.h
map	pointer to pointer to AssocTransDescriptor	/simnet/libsrc/libassoc/assoc_lcl.h
mask	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
bucket	register pointer to pointer to AssocTransDescriptor	/simnet/libsrc/libassoc/assoc_lcl.h
Called By		
Function	**Where Described**	
AssocDeleteCachedResponse	Section 2.20.1.18.3.	
AssocDeleteTransaction	Section 2.20.1.23.3.	

Table 2.20-76 AssocDeleteFromBucketInformation.

2.20.1.25.3 AssocBucketLookup

AssocBucketLookup returns the bucket of the transaction defined by *tid* and *who*. The function call is AssocBucketLookup(who, tid, map, mask). Table 2.20-77 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
who	pointer to SimulationAddress	/simnet/common/include/protocol/address.h
tid	TransactionIdentifier	/simnet/common/include/protocol/address.h
map	pointer to pointer to AssocTransDescriptor	/simnet/libsrc/libassoc/assoc_lcl.h
mask	int	Standard C type.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
bucket	register pointer to AssocTransDescriptor	/simnet/libsrc/libassoc/assoc_lcl.h
Return Values		
Return Value	**Type**	**Meaning**
Null	pointer to AssocTransDescriptor	Transaction not in bucket.
bucket->transFront	pointer to AssocTransDescriptor	Desired transaction.
Calls		
Function	**Where Described**	
ASSOC_ADDRESS_EQUAL	Macro defined in /simnet/libsrc/libassoc/assoc.h.	
Called By		
Function	**Where Described**	
AssocFindResponse	Section 2.20.1.18.5.	
AssocFindTransaction	Section 2.20.1.23.4.	

Table 2.20-77 AssocBucketLookup Information.

2.20.1.26 assoc.h
/simnet/libsrc/libassoc/assoc.h

assoc.h contains all the definitions needed by the outside world to use the association layer library. Table 2.20-78 describes the variables used by assoc.h.

Variables		
Variable	**Type**	**Where Typedef Declared**
assoc_errno	extern int	Standard C type.

Table 2.20-78 assoc.h Variable Information.

2.20.1.27 assoc_lcl.h
/simnet/libsrc/libassoc/assoc_lcl.h

assoc_lcl.h contains all the definitions that should be global throughout the associatiaon layer library but hidden from the outside world. Table 2.20-79 describes the variables used by assoc_lcl.h.

Variables		
Variable	**Type**	**Where Typedef Declared**
assocPaddingPDU	extern AssociationPDU	/simnet/common/include/protocol/p_assoc.h
channelMap	extern pointer to array maxChannels of ChannelDescriptor	/simnet/libsrc/libassoc/assoc_lcl.h

Table 2.20-79 assoc_lcl.h Variable Information.

2.20.1.28 defaults.h
/simnet/libsrc/libassoc/defaults.h

defaults.h contains the default values for the assoc.def file.

2.20.1.29 AssocDef

Assoc.def is a generic term for association layer parameter files. Any number of parameter files may exist. The assocDef parameter to the AssocOpen call is a pointer to the name of the parameter file to use for that channel. Thus, channel defaults are set independently by using different parameter files. There are six parameters which can be set for each association channel. They are described in Table 2.20-80.

Parameter	Description
SITE	Site id used for originator site/host pair. Default is siteIrrelevant (see /simnet/common/include/protocol/address.h).
HOST	Host id used for originator site/host pair. Default is hostIrrelevant (see /simnet/common/include/protocol/address.h).
MAX_SUBSCRIPTIONS	Maximum number of multicast addresses to which this channel may subscribe. Each potential subscription costs 8 bytes. Default is 64.
INITIAL_DESCRIPTORS	The number of descriptors allocated for transactions and cached responses at start-up. If this number is exhausted, more are allocated. Each descriptor costs 2628 bytes. Default is 20.
ADDITIONAL_DESCRIPTORS	The number of descriptors allocated additionally each time the current number of descriptors is exhausted. The default is 10.
PROTOCOL_FAMILY	The protocol family to which this channel will listen. The default is ASSOC_FAMILY_SIMNET. This parameter is specified symbolically as one of ASSOC_FAMILY_SIMNET, ASSOC_FAMILY_DATA, ASSOC_FAMILY_VOICE and ASSOC_FAMILY_GT.

Table 2.20-80 Assoc.def Parameters.

Parameters are specified, one per line, as <parameter> <parameter value>. Blank lines, spaces, tabs, etc. are ignored (except that <parameter> and <parameter value> must be separated by one or more blanks or tabs)> Line beginning with '#' are comments. The parameter reading mechanism is case insensitive.

Here is an example:

#This association definition file describes the defaults:

```
SITE                    0
HOST                    0
MAX_SUBSCRIPTIONS       64
INITIAL_DESCRIPTORS     20
ADDITIONAL_DESCRIPTORS  10
PROTOCOL_FAMILY         ASSOC_FAMILY_SIMNET
```

2.20.2 libnetif

Libnetif provides a high performance pathway to an unreliable datagram transport service such as that provided by an IEEE 802.3 network. Libnetif also provides several miscellaneous functions for controlling the network adapter unit. The miscellaneous functions allow applications to take advantage of adapter specific capabilities such as multicast addressing, high granularity timers, and intelligent on-board processors.

2.20.2.1 net_ctl.c

/simnet/libsrc/libnetif/net_ctl.c

net_clt.c contains routines used to perform control functions on the network interface.

2.20.2.1.1 net_alive

net_alive probes the network interface referenced by the descriptor, *h* , to determine whether it is alive. Success means that the code loaded onto the network card is running. Failure means that the code has died and is not running. The function call is net_alive(h). Table 2.20-81 describes the parameters used, return values and possible errors generated by using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
-1	int	network interface doesn't respond, errno set to indicate error
0	int	successful
Errors		
Error Name	**Reason for Error**	
EBADF	Not a valid descriptor.	
EIO	Network interface did not respond.	
Calls		
Function	**Where Described**	
CHECK_HANDLE	Macro defined in /simnet/libsrc/libnetif/libnet.h	
net_access	Section 2.20.2.20.1.	
Called By		
Function	**Where Described**	
open_cmc	Section 2.20.2.10.4.	

Table 2.20-81 net_alive Information.

2.20.2.1.2 net_res

net_res resets the network interface controller specified by descriptor *h*. The interface controller is initialized to its power-up condition. The function call is net_res(h). Table 2.20-82 describes the parameters used, return values and possible errors generated by using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
-1	int	operation failed, errno set to indicate error
0	int	successful
Errors		
Error Name	**Reason for Error**	
EBADF	Not a valid descriptor.	
EIO	The network interface did not respond.	
Calls		
Function	**Where Described**	
CHECK_HANDLE	Macro defined in /simnet/libsrc/libnetif/libnet.h	
net_access	Section 2.20.2.20.1.	

Table 2.20-82 net_res Information.

2.20.2.1.3 net_settimeout

net_settimeout is called to specify a timeout of *value* seconds for net_recvs and net_sends to the network interface specified by *h*. The default condition is no timeout. A timeout value of 0 will cause no timeout. The function call is net_settimeout(h, value). Table 2.20-83 describes the parameters used, return values and possible errors generated by using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.
value	int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
-1	int	operation failed, errno set to indicate error
0	int	successful
Errors		
Error Name	**Reason for Error**	
EBADF	Not a valid descriptor.	

Calls	
Function	**Where Described**
CHECK_HANDLE	Macro defined in /simnet/libsrc/libnetif/libnet.h.

Table 2.20-83 net_settimeout Information.

2.20.2.1.4 net_iocontrol

net_iocontrol provides a mechanism to access control functions provided by underlying network interface. It arranges for *sendsize* bytes to be copied to the network interface from buffer *sendbuf* and then for *recvsize* bytes to be copied to buffer *recvbuf* from the network interface. The contents of the buffers are I/O control function specific. The function call is net_iocontrol(h, sendbuf, sendsize, recvbuf, recvsize). Table 2.20-84 describes the parameters used, errors returned and functions called using this function with a Butterfly. Table 2.20-85 describes the parameters used, errors returned and functions called using this function with any other computer.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.
sendbuf	pointer to char	Standard C type.
sendsize	int	Standard C type.
recvbuf	pointer to char	Standard C type.
recvsize	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
np	register pointer to struct netlib_pb	/simnet/libsrc/libnetif/libnet.h
ret	int	Standard C type.
rbuf	pointer to char	Standard C type.
rbufoid	OID	
sbuf	pointer to char	Standard C type.
sbufoid	OID	
Return Values		
Return Value	**Type**	**Meaning**
ret	int	0 if successful -1 if not successful.
Calls		
Function	**Where Described**	
CHECK_HANDLE	Macro defined in /simnet/libsrc/libnetif/libnet.h.	
net_access	Section 2.20.2.20.1	

Table 2.20-84 Butterfly net_iocontrol Information.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.
sendbuf	pointer to char	Standard C type.
sendsize	int	Standard C type.
recvbuf	pointer to char	Standard C type.
recvsize	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
np	register pointer to struct netlib_pb	/simnet/libsrc/libnetif/libnet.h
ret	int	Standard C type.
arg	struct enpaccess_arg	enparg.h
Return Values		
Return Value	**Type**	**Meaning**
ret	int	0 if successful. -1 if not successful.
Calls		
Function	**Where Described**	
CHECK_HANDLE	Macro defined in /simnet/libsrc/libnetif/libnet.h.	
net_access	Section 2.20.2.20.1.	

Table 2.20-85 Other net_iocontrol Information.

2.20.2.1.5 net_nopened

net_nopened returns the number of times the interface has been opened. This function will only work on a MassComp computer. Other computers will only return -1. The function call is net_nopened(h). Table 2.20-86 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
id	int	Standard C type.
buf	struct shmid_ds	
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Unsuccessful or computer other than Masscomp used.
buf.shm_nattch	int	Number of thimes interface uhas been opened.

Calls	
Function	**Where Described**
CHECK_HANDLE	Macro defined in /simnet/libsrc/libnetif/libnet.h.

Table 2.20-86 net_nopened Information.

2.20.2.1.6 net_loopback

net_loopback turns the loopback of packets on and off. If *flag* is 1, packets are looped. If *flag* is 0, they are not looped. The default condition after initialization is that packets are not looped. *arg* is only used if SIMBFLY is NOT defined. The function call is net_loopback(h, flag). Table 2.20-87 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.
flag	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
np	register pointer to struct netlib_pb	/simnet/libsrc/libnetif/libnet.h
ret	int	Standard C type.
command	int	Standard C type.
arg	struct enpaccess_arg	enparg.h
Return Values		
Return Value	**Type**	**Meaning**
ret	int	0 if successful. -1 if not successful.
Calls		
Function	**Where Described**	
CHECK_HANDLE	Macro defined in /simnet/libsrc/libnetif/libnet.h.	
net_access	Section 2.20.2.20.1.	

Table 2.20-87 net_loopback Information.

2.20.2.2 net_addr.c

/simnet/libsrc/libnetif/net_addr.c

2.20.2.2.1 net_addr_compare

net_addr_compare is an address manipulation routine. It compares the two NetworkAddresses *a1* and *a2* and returns an indication of whether thay are equal or not. The function call is net_addr_compare(a1, a2). Table 2.20-88 describes the parameters used and return values generated by using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
a1	pointer to NetworkAddress	/simnet/libsrc/libnetif/network.h
a2	pointer to NetworkAddress	/simnet/libsrc/libnetif/network.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
FALSE	int	NetworkAddressess are not equal.
TRUE	int	NetworkAddresses are equal.

Table 2.20-88 net_addr_compare Information.

2.20.2.2.2 net_addr_bin_to_str

net_addr_bin_to_str converts the NetworkAddress *p* into a null terminated string which is returned to *s*. *s* must point to a buffer at least 13 bytes long to accomodate the string generated by net_addr_bin_to_str. The routine always succeeds so there is no return value. The function call is net_addr_bin_to_str(p, s). Table 2.20-89 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
p	pointer to NetworkAddress	/simnet/libsrc/libnetif/network.h
s	pointer to char	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	int	Standard C type.
j	int	Standard C type.

Table 2.20-89 net_addr_bin_to_str Information.

2.20.2.2.3 net_addr_str_to_bin

net_addr_str_to_bin converts the string *s* into a NetworkAddress in *p*. The function call is net_addr_bin_to_str(p, s). Table 2.20-90 describes the parameters used and return values generated by using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
p	pointer to NetworkAddress	/simnet/libsrc/libnetif/network.h
s	pointer to char	Standard C type.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	int	Standard C type.
x	int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
FALSE	int	String address did not have exactly 12 hex digits. Address is all 0s.
TRUE	int	String address had exactly 12 hex digits.

Table 2.20-90 net_addr_str_to_bin Information.

2.20.2.2.4 net_getaddr

net_getaddr gets the network address of the interface specified by the descriptor *h* to the NetworkAddress *p*. It returns an indication of success or failure. The function call is net_getaddr(h, p). Table 2.20-91 describes the parameters used, return values and possible errors generated by using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.
p	pointer to NetworkAddress	/simnet/libsrc/libnetif/network.h
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Operation failed, errno set to indicate error.
0	int	Successful.
Errors		
Error Name	**Reason for Error**	
EBADF	Not a valid descriptor.	
EIO	The network interface did not respond.	
Calls		
Function	**Where Described**	
CHECK_HANDLE	Macro defined in /simnet/libsrc/libnetif/libnet.h.	
net_access	Section 2.20.2.20.1.	

Table 2.20-91 net_getaddr Information.

2.20.2.2.5 net_zeroaddr

net_zeroaddr puts a zero address into the NetworkAddress *p*. The routine always succeeds so there is no return value. The function call is net_zeroaddr(p). Table 2.20-92 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
p	pointer to NetworkAddress	/simnet/libsrc/libnetif/network.h

Table 2.20-92 net_zeroaddr Information.

2.20.2.2.6 net_addr_format_convert

net_addr_format_convert converts format "XXXXXXXXXXXX", *p1*, to "YY YY YY YY YY YY", *p2*. The function call is net_addr_format_convert(p1, p2). Table 2.20-93 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
p1	register pointer to char	Standard C type.
p2	register pointer to char	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	*register int*	Standard C type.

Table 2.20-93 net_addr_format_convertInformation.

2.20.2.3 net_clos.c

/simnet/libsrc/libnetif/net_clos.c

2.20.2.3.1 net_close

net_close closes the access to the network interface specified by descriptor *h*. It should be called prior to exiting. The function call is net_close(h). Table 2.20-94 describes the parameters used, return values and possible errors generated by using this function with a Butterfly. Table 2.20-95 describes the parameters used, return values and possible errors generated by using this function with a MassComp. Table 2.20-96 describes the parameters used, return values and possible errors generated by using this function with a Sun or a MIPS.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
np	register pointer to struct netlib_pb	/simnet/libsrc/libnetif/libnet.h
ret	int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
ret	int	0 if Successful. -1 if not successful.
Errors		
Error Name	**Reason for Error**	
EBADF	Not a valid descriptor.	
Calls		
Function	**Where Described**	
CHECK_HANDLE	Macro defined in /simnet/libsrc/libnetif/libnet.h.	

Table 2.20-94 Butterfly net_close Information.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
np	register pointer to struct netlib_pb	/simnet/libsrc/libnetif/libnet.h
ret	int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
ret	int	0 if successful. -1 if not successful.
Errors		
Error Name	**Reason for Error**	
EBADF	Not a valid descriptor.	
Calls		
Function	**Where Described**	
CHECK_HANDLE	Macro defined in /simnet/libsrc/libnetif/libnet.h.	
XXX_CLOSE	Macro defined in /simnet/libsrc/libnetif/libnet.h.	
detachshm	Section 2.6.5.2.1 in the Vehicles CSCI.	

Table 2.20-95 MassComp net_close Information.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
np	register pointer to struct netlib_pb	/simnet/libsrc/libnetif/libnet.h
ret	int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
ret	int	0 if successful. -1 if not successful.
Errors		
Error Name	**Reason for Error**	
EBADF	Not a valid descriptor.	
Calls		
Function	**Where Described**	
CHECK_HANDLE	Macro defined in /simnet/libsrc/libnetif/libnet.h.	
XXX_CLOSE	Macro defined in /simnet/libsrc/libnetif/libnet.h.	

Table 2.20-96 Sun and MIPS net_close Information.

2.20.2.4 net_flus.c

/simnet/libsrc/libnetif/net_flus.c

2.20.2.4.1 net_flush

net_flush flushes all packets from the receive and/or transmit queues for the network interface specified by descriptor *h*. The card must be running (reference net_norm in Section 2.20.2.7.1) before this call can be made. *flags* is a bit mask that selects whether the receive queue (NETFLUSHRCV) or the transmit queue (NETFLUSHXMT) or both (NETFLUSHRCV | NETFLUSHXMT) are flushed. An indication of success or failure is returned. The calls to Wait_DualQ() and Unlock_DualQ() are only made if SIMBFLY is defined. The function call is net_flush(h, flags). Table 2.20-97 describes the parameters used, return values and possible errors generated by using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.
flags	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
np	register pointer to struct netlib_pb	/simnet/libsrc/libnetif/libnet.h

Return Values		
Return Value	**Type**	**Meaning**
-1	int	Flush unsuccessful, errno set to indicate error.
0	int	Successful.
Errors		
Error Name	**Reason for Error**	
EBADF	Not a valid descriptor.	
EIO	Network interface did not respond.	
Calls		
Function	**Where Described**	
CHECK_HANDLE	Macro defined in /simnet/libsrc/libnetif/libnet.h.	
Wait_DualQ		
net_access	Section 2.20.2.20.1.	
SETLONG	Macro defined in /simnet/libsrc/libnetif/libnet.h.	
Unlock_DualQ		

Table 2.20-97 net_flush Information.

2.20.2.5 net_stat.c

/simnet/libsrc/libnetif/net_stat.c

2.20.2.5.1 net_get_statistics

net_get_statistics gets the accumlated statistics from the network interface specified by descriptor *h* to the array *stats*. The following statistics are returned:

successful transmissions
multiple retries reported
single retries reported
retry failure reported (check cable)
deferrals reported
buffer errors
silo underruns
late collisions (check cable)
carrier losses (check cable)
babbling transmitter errors
collisions errors
memory errors
packets received
missed packet reported
CRC errors reported (check cable)
framing errors (check cable)
silo overruns
stp bit missing
enp bit missing
lance errors
non-SIMNET packets

The "check cable" annotation indicates a potential cable fault if the associated statistic is increasing. The function call is net_get_statistics(h, stats). The variable *arg* is only used if MIPS is defined. Table 2.20-98 describes the parameters used, return values and possible errors generated by using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.
stats	pointer to long	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
arg	struct enpaccess_arg	enparg.h
Return Values		
Return Value	**Type**	**Meaning**
-1	int	operation failed, errno set to indicate error
0	int	successful
Errors		
Error Name	**Reason for Error**	
EBADF	Not a valid descriptor.	
EIO	The network interface did not respond.	
Calls		
Function	**Where Described**	
CHECK_HANDLE	Macro defined in /simnet/libsrc/libnetif/libnet.h.	
net_access	Section 2.20.2.20.1.	
Called By		
Function	**Where Described**	
net_print_statistics	Section 2.20.2.5.4.	

Table 2.20-98 net_get_statistics Information.

2.20.2.5.2 net_zero_statistics

net_zero_statistics zeros the statistics for the network interface specifed by descriptor *h*. The function call is net_zero_statistics(h). Table 2.20-99 describes the parameters used, return values and possible errors generated by using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
-1	int	operation failed, errno set to indicate error
0	int	successful

Errors	
Error Name	**Reason for Error**
EBADF	Not a valid descriptor.
EIO	The network interface did not respond.
Calls	
Function	**Where Described**
CHECK_HANDLE	Macro defined in /simnet/libsrc/libnetif/libnet.h.
net_access	Section 2.20.2.20.1.

Table 2.20-99 net_zero_statistics Information.

2.20.2.5.3 net_stat_string

net_stat_string returns a null terminated string of length 40 characters (padded with spaces) to *buf* for statistic number *stat*. If no string is available for the statistic *stat*, a zero length string is returned. Section 2.20.2.5.1, net_get_statistics, for a table of available statistics. The function call is net_stat_string(stat, buf). Table 2.20-100 describes the parameters used, return values and possible errors generated by using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
stat	int	Standard C type.
buf	pointer to char	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
-1	int	operation failed, errno set to indicate error
0	int	successful
Errors		
Error Name	**Reason for Error**	
EBADF	Not a valid descriptor.	
EIO	The network interface did not respond.	
Called By		
Function	**Where Described**	
net_print_statistics	Section 2.20.2.5.4.	

Table 2.20-100 net_stat_string Information.

2.20.2.5.4 net_print_statistics

net_print_statistics prints the statistics in the format of the table in the description of net_get_statistics above. The function call is net_print_statistics(h). Table 2.20-101 describes the parameters used, return values and possible errors generated by using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
stats	array N_STATS of long	Standard C type.
str	array 60 of char	Standard C type.
i	register int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
-1	int	operation failed, errno set to indicate error
0	int	successful
Errors		
Error Name	**Reason for Error**	
EBADF	Not a valid descriptor.	
EIO	The network interface did not respond.	
Calls		
Function	**Where Described**	
CHECK_HANDLE	Macro defined in /simnet/libsrc/libnetif/libnet.h.	
net_get_statistics	Section 2.20.2.5.1.	
net_stat_string	Section 2.20.2.5.3.	

Table 2.20-101 net_print_statistics Information.

2.20.2.6 net_info.c

/simnet/libsrc/libnetif/net_info.c

2.20.2.6.1 net_hostbuf_info

net_hostbuf_info obtains a pointer to and the size in bytes of the hostbuf. The pointer is returned in *pbufptr* and the size is returned in *pbufsize*. This memory block is guaranteed to be aligned on a short boundary. The hostbuf is a segment of memory resident on the network card that is accessible to an application running on the host as well as to code running on the network card. It may be used to transfer parameters between an application and filtering code, for example, running on the network card. The host should avoid heavy access to this region as it will consume bus bandwidth on the network card. The function call is net_hostbuf_info(h, pbufptr, pbufsize). Table 2.20-102 describes the parameters used, return values and possible errors generated by using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.
pbufptr	pointer to pointer to char	Standard C type.
pbufsize	pointer to int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
bufs[2]	int	Standard C type.

Return Values		
Return Value	**Type**	**Meaning**
-1	int	Operation failed, errno set to indicate error.
0	int	Successful.
Errors		
Error Name	**Reason for Error**	
EBADF	Not a valid descriptor.	
Calls		
Function	**Where Described**	
CHECK_HANDLE	Macro defined in /simnet/libsrc/libnetif/libnet.h.	
net_access	Section 2.20.2.20.1.	

Table 2.20-102 net_hostbuf_info Information.

2.20.2.6.2 net_sharebuf_info

net_sharebuf_info is called to obtain a pointer to and the size in bytes of the shared buffer, which is a segment of memory resident in the host that the network card and an application running on the host can access. The pointer is returned into *pbufptr* and the size in *pbufsize*. This memory segment is guaranteed to be aligned on a long boundary. The function call is net_sharebuf_info(h, pbufptr, pbufsize). Table 2.20-103 describes the parameters used, return values and possible errors generated by using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.
pbufptr	pointer to pointer to char	Standard C type.
pbufsize	pointer to int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Operation failed, errno set to indicate error.
0	int	Successful.
Errors		
Error Name	**Reason for Error**	
EBADF	Not a valid descriptor.	
Calls		
Function	**Where Described**	
CHECK_HANDLE	Macro defined in /simnet/libsrc/libnetif/libnet.h.	

Table 2.20-103 net_sharebuf_info Information.

2.20.2.6.3 net_bufs

net_bufs returns the number of transmit and receive buffers available in the underlying network interface. *pxmtbus* and *prcvbufs* are pointers to where to return the number of transmit and receive buffers. The function call is net_bufs(h, pxmtbufs, prcvbufs). Table 2.20-104 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.
pxmtbufs	pointer to int	Standard C type.
prcvbufs	pointer to int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
buf[2]	int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Unsuccessful.
0	int	Successful.
Calls		
Function	**Where Described**	
CHECK_HANDLE	Macro defined in /simnet/libsrc/libnetif/libnet.h.	
net_access	Section 2.20.2.20.1.	

Table 2.20-104 net_bufs Information.

2.20.2.6.4 net_interface_type

net_interface_type returns the interface type for the handle, *h*. The current types are NIF_TYPE_CMC and NIF_TYPE_147. The function call is net_interface_type(h). Table 2.20-105 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
nbp[h].type	int	Interface type for handle.
Calls		
Function	**Where Described**	
CHECK_HANDLE	Macro defined in /simnet/libsrc/libnetif/libnet.h.	

Table 2.20-105 net_interface_type Information.

2.20.2.6.5 net_syserror_info

net_syserror_info gets a pointer to the character array of system errors, *perr*, and also returns the number of the system errors, *p_num*. The function call is net_syserror_info(h,perr,p_num). Table 2.20-106 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.
perr	pointer to pointer to char	Standard C type.
p_num	pointer to int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
0	int	Always returned.
Calls		
Function	**Where Described**	
CHECK_HANDLE	Macro defined in /simnet/libsrc/libnetif/libnet.h.	

Table 2.20-106 net_syserror_info Information.

2.20.2.6.6 net_version

net_version returns a null terminated string that contains the version of the underlying network interface software. *version* is a pointer to the buffer into which the version string is returned. *version_size* is the size of the buffer pointed to by *version*. The function call is net_version(h, version, version_size). Table 2.20-107 describes the parameters used, and functions called using this function with a Butterfly. Table 2.20-108 describes the parameters used and functions called using this function with all other computers.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.
version	pointer to char	Standard C type.
version_size	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
np	pointer to struct netlib_pb	/simnet/libsrc/libnetif/libnet.h
ret	int	Standard C type.
buf	pointer to char	Standard C type.
bufoid	OID	
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Unsuccessful.
0	int	Successful.

Calls	
Function	**Where Described**
CHECK_HANDLE	Macro defined in /simnet/libsrc/libnetif/libnet.h.
net_access	Section 2.20.2.20.1.

Table 2.20-107 Butterfly net_version Information.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.
version	pointer to char	Standard C type.
version_size	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
np	pointer to struct netlib_pb	/simnet/libsrc/libnetif/libnet.h
ret	int	Standard C type.
arg	struct enpaccess_arg	enparg.h
buf[VERSION_SIZE]	char	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Unsuccessful.
0	int	Successful.
Calls		
Function	**Where Described**	
CHECK_HANDLE	Macro defined in /simnet/libsrc/libnetif/libnet.h.	
net_access	Section 2.20.2.20.1.	

Table 2.20-108 Other net_version Information.

2.20.2.7 net_mode.c

/simnet/libsrc/libnetif/net_mode.c

2.20.2.7.1 net_norm

net_norm starts the network interface specified by descriptor *h* in the normal mode of operation. The ethernet address of the interface is set to *na* upon the first invocation of the call. The argument *na* must be a valid pointer for subsequent calls by the data will be ignored. The function call is net_norm(h, na). Table 2.20-109 describes the parameters used, return values and possible errors generated by using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.
na	pointer to NetworkAddress	/simnet/libsrc/libnetif/network.h

Return Values		
Return Value	**Type**	**Meaning**
-1	int	Operation failed, errno set to indicate error.
0	int	Successful.
Errors		
Error Name	**Reason for Error**	
EBADF	Not a valid descriptor.	
EIO	The network interface did not respond.	
Calls		
Function	**Where Described**	
do_mode_cmd	Section 2.20.2.7.5.	

Table 2.20-109 net_norm Information.

2.20.2.7.2 net_intloop

net_intloop starts the network interface specified by descriptor *h* in the internal loopback mode of operation. The ethernet address of the interface is set to *na* upon the first invocation of the call. The argument *na* must be a valid pointer for subsequent calls by the data will be ignored. The function call is net_intloop(h, na). Table 2.20-110 describes the parameters used, return values and possible errors generated by using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.
na	pointer to NetworkAddress	/simnet/libsrc/libnetif/network.h
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Operation failed, errno set to indicate error.
0	int	Successful.
Errors		
Error Name	**Reason for Error**	
EBADF	Not a valid descriptor.	
EIO	The network interface did not respond.	
Calls		
Function	**Where Described**	
do_mode_cmd	Section 2.20.2.7.5.	

Table 2.20-110 net_intloop Information.

2.20.2.7.3 net_extloop

net_extloop starts the network interface specified by descriptor *h* in the external loopback mode of operation. The ethernet address of the interface is set to *na* upon the first invocation of the call. The argument *na* must be a valid pointer for subsequent calls by the data will be ignored. The function call is net_extloop(h, na). Table 2.20-111 describes the parameters used, return values and possible errors generated by using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.
na	pointer to NetworkAddress	/simnet/libsrc/libnetif/network.h
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Operation failed, errno set to indicate error.
0	int	Successful.
Errors		
Error Name	**Reason for Error**	
EBADF	Not a valid descriptor.	
EIO	The network interface did not respond.	
Calls		
Function	**Where Described**	
do_mode_cmd	Section 2.20.2.7.5.	

Table 2.20-111 net_extloop Information.

2.20.2.7.4 net_prom

net_prom starts the network interface specified by descriptor *h* in the promicuous mode of operation. The ethernet address of the interface is set to *na* upon the first invocation of the call. The argument *na* must be a valid pointer for subsequent calls by the data will be ignored. The function call is net_prom(h, na). Table 2.20-112 describes the parameters used, return values and possible errors generated by using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.
na	pointer to NetworkAddress	/simnet/libsrc/libnetif/network.h
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Operation failed, errno set to indicate error.
0	int	Successful.

Errors	
Error Name	**Reason for Error**
EBADF	Not a valid descriptor.
EIO	The network interface did not respond.
Calls	
Function	**Where Described**
do_mode_cmd	Section 2.20.2.7.5.

Table 2.20-112 net_prom Information.

2.20.2.7.5 do_mode_cmd

do_mode_cmd is an internal library function for setting the mode of the underlying network interface. *mode* is a mode word (one of NORMMODE, EXTLMODE, INTLMODE, PROMMODE - normal, external loopback, internal loopback and promiscuous modes, respectively), *h* is a descriptor, *na* is a pointer to the network address to which the Ethernet address of the network interface is set. The function call is do_mode_cmd(mode, h, na). Table 2.20-113 describes the parameters used, errors returned and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
mode	int	Standard C type.
h	int	Standard C type.
na	pointer to NetworkAddress	/simnet/libsrc/libnetif/network.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
np	pointer to struct netlib_pb	/simnet/libsrc/libnetif/libnet.h
ret	int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Unsuccessful.
ret	int	Successful.
Calls		
Function	**Where Described**	
CHECK_HANDLE	Macro defined in /simnet/libsrc/libnetif/libnet.h.	
do_mode_cmd_cmc	Section 2.20.2.7.6.	
do_mode_cmd_147	Section 2.20.2.7.7.	
Called By		
Function	**Where Described**	
net_norm	Section 2.20.2.7.1.	
net_intloop	Section 2.20.2.7.2.	
net_extloop	Section 2.20.2.7.3.	
net_prom	Section 2.20.2.7.4.	

Table 2.20-113 do_mode_cmd Information.

2.20.2.7.6 do_mode_cmd_cmc

do_mode_cmd_cmc is the cmc card specific handler for setting the mode. do_mode_cmd_147 is the MVME147 card specific handler for setting the mode. *args* is only used if SIMBFLY is NOT defined.The function call is do_mode_cmd_cmc(mode, h, na, np). Table 2.20-114 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
mode	int	Standard C type.
h	int	Standard C type.
na	pointer to NetworkAddress	/simnet/libsrc/libnetif/network.h
np	pointer to struct netlib_pb	/simnet/libsrc/libnetif/libnet.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
args	struct enprun_arg	
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Unsuccessful.
0	int	Successful.
Calls		
Function	**Where Described**	
SETLONG	Macro defined in /simnet/libsrc/libnetif/libnet.h.	
Ringsize_in_chars	Macro defined in /simnet/libsrc/libnetif/network.h.	
net_access	Section 2.20.2.20.1.	
Called By		
Function	**Where Described**	
do_mode_cmd	Section 2.20.2.7.5.	

Table 2.20-114 do_mode_cmd_cmc Information.

2.20.2.7.7 do_mode_cmd_147

do_mode_cmd_147 is not currently supported on any computer used in SIMNET. The function call is do_mode_cmd_147(mode, h, na, np). Table 2.20-115 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
mode	int	Standard C type.
h	int	Standard C type.
na	pointer to NetworkAddress	/simnet/libsrc/libnetif/network.h
np	pointer to struct netlib_pb	/simnet/libsrc/libnetif/libnet.h

Return Values		
Return Value	**Type**	**Meaning**
ret	int	Always -1.
Called By		
Function	**Where Described**	
do_mode_cmd	Section 2.20.2.7.5.	

Table 2.20-115 do_mode_cmd_147 Information.

2.20.2.8 net_time.c

/simnet/libsrc/libnetif/net_time.c

net_time.c contains routines to access the CMC card timer.

2.20.2.8.1 net_gettime

net_gettime obtains the current value of the 31 bit timer maintained by the network interface specified by descriptor *h*. The timer is incremented every 998.26 microseconds. The function call is net_gettime(h). Table 2.20-116 describes the parameters used, return values and possible errors generated by using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
ret	int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Operation failed, errno set to indicate error.
ret	int	Successful.
Errors		
Error Name	**Reason for Error**	
EBADF	Not a valid descriptor.	
EIO	The network interface did not respond.	
Calls		
Function	**Where Described**	
CHECK_HANDLE	Macro defined in /simnet/libsrc/libnetif/libnet.h.	
net_access	Section 2.20.2.20.1.	

Table 2.20-116 net_gettime Information.

2.20.2.8.2 net_settime

net_settime sets the current value of the 31 bit timer maintained by the network interface specified by descriptor *h*. The timer is incremented every 998.26 microseconds. The function call is net_settime(h, time). Table 2.20-117 describes the parameters used, return values and possible errors generated by using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.
time	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
ret	int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Operation failed, errno set to indicate error.
0	int	Successful.
Errors		
Error Name	**Reason for Error**	
EBADF	Not a valid descriptor.	
EIO	The network interface did not respond.	
Calls		
Function	**Where Described**	
CHECK_HANDLE	Macro defined in /simnet/libsrc/libnetif/libnet.h.	
net_access	Section 2.20.2.20.1.	

Table 2.20-117 net_settime Information.

2.20.2.8.3 net_current_time

net_current_time returns the number of ticks since the last call to net_init_time (Section 2.20.2.8.4). net_init_time must be called once before net_current_time will return a meaningful value. If it is not called for a bit over 60 seconds, it loses track of time, as the 16-bit timer cycles. While this is not a fatal error, the time returned by net_current_time will not be correct. The function call is net_current_time(h). Table 2.20-118 describes the parameters used, return values and possible errors generated by using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
dif	register long	Standard C type.
cur_time	register unsigned long	Standard C type.
np	register pointer to struct netlib_pb	/simnet/libsrc/libnetif/libnet.h
ret	int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Network interface cannot be closed, ermo set to indicate error.
ret	int	Successful.
Errors		
Error Name	**Reason for Error**	
EBADF	Not a valid descriptor.	
EINVAL	The kernel is not set up correctly to allow the timer to be mapped in.	
Calls		
Function	**Where Described**	
CHECK_HANDLE	Macro defined in /simnet/libsrc/libnetif/libnet.h.	

Table 2.20-118 net_current_time Information.

2.20.2.8.4 net_init_time

net_init_time initializes the 16-bit timer on the CMC card. This timer is incremented every 998.26 microseconds. net_init_time must be called once before net_current_time is used. The function call is net_init_time(h). Table 2.20-119 describes the parameters used, return values and possible errors generated by using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
np	register pointer to struct netlib_pb	/simnet/libsrc/libnetif/libnet.h
ret	int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Network interface cannot be closed, ermo set to indicate error.
0	int	Successful.

Errors	
Error Name	**Reason for Error**
EBADF	Not a valid descriptor.
EINVAL	The kernel is not set up correctly to allow the timer to be mapped in.
Calls	
Function	**Where Described**
CHECK_HANDLE	Macro defined in /simnet/libsrc/libnetif/libnet.h.

Table 2.20-119 net_init_time Information.

2.20.2.8.5 net_stomp_time

net_stomp_time directs the network interface to set the 32 bit timer to the timestamp value of the next packet (if any) in the transmit queue. This call has no effect if timestamping is not enabled. It is useful as a "fast forward" function in applications that need to replay timestamped packets. The function call is net_stomp_time(h). Table 2.20-120 describes the parameters used, return values and possible errors generated by using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
-1	int	*Operation failed, errno set to* indicate error.
0	int	Successful.
Errors		
Error Name	**Reason for Error**	
EBADF	Not a valid descriptor.	
Calls		
Function	**Where Described**	
CHECK_HANDLE	Macro defined in /simnet/libsrc/libnetif/libnet.h.	
net_access	Section 2.20.2.20.1.	

Table 2.20-120 net_stomp_time Information.

2.20.2.8.6 net_heartbeat

net_heartbeat returns the heartbeat value. The function call is net_heartbeat(h). Table 2.20-121 describes the parameters used, errors returned and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
np	register pointer to struct netlib_pb	/simnet/libsrc/libnetif/libnet.h
ret	unsigned short	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
-1	unsigned int	Unsuccessful. Device not mapped-in. Error code returned in errno.
0	unsigned int	Unsuccessful.
ret	unsigned int	Heartbeat value.
Errors		
Error Name	**Reason for Error**	
EINVAL	Heartbeat not mapped in.	
Calls		
Function	**Where Described**	
CHECK_HANDLE	Macro defined in /simnet/libsrc/libnetif/libnet.h.	

Table 2.20-121 net_heartbeat Information.

2.20.2.8.7 net_device_base

net_device_base returns the base of the mapped-in device. The function call is net_device_base(h). Table 2.20-122 describes the parameters used, errors returned and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
np	register pointer to struct netlib_pb	/simnet/libsrc/libnetif/libnet.h
ret	int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Unsuccessful. Device not mapped-in. Error code returned in errno.
NULL	int	Network interface unknown or unsupported.
ret	int	Base of the mapped-in device.
Errors		
Error Name	**Reason for Error**	
EINVAL	Device not mapped in.	

Calls	
Function	**Where Described**
CHECK_HANDLE	Macro defined in /simnet/libsrc/libnetif/libnet.h.

Table 2.20-122 net_device_base Information.

2.20.2.9 net_load.c
/simnet/libsrc/libnetif/net_load.c

2.20.2.9.1 net_load

net_load copies *count* bytes starting from *buffer* to *offset* bytes from the start of memory on the network interface. The function call is net_load(h, buffer, count, offset). Table 2.20-123 describes the parameters used, return values and possible errors generated by using this function with a Butterfly. Table 2.20-124 describes the parameters used, return values and possible errors generated by using this function with a MassComp. Table 2.20-125 describes the parameters used, return values and possible errors generated by using this function with an MSC. Table 2.20-126 describes the parameters used, return values and possible errors generated by using this function with a Sun. Table 2.20-127 describes the parameters used, return values and possible errors generated by using this function with a MIPS.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.
buffer	pointer to char	Standard C type.
count	int	Standard C type.
offset	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
np	register pointer to struct netlib_pb	/simnet/libsrc/libnetif/libnet.h
localbuf	pointer to char	Standard C type.
localbufoid	OID	
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Operation failed, errno set to indicate error.
0	int	Successful.
Errors		
Error Name	**Reason for Error**	
EBADF	Not a valid descriptor.	
EIO	The network interface did not respond.	

Calls	
Function	**Where Described**
CHECK_HANDLE	Macro defined in /simnet/libsrc/libnetif/libnet.h.
Do_bt	
net_access	Section 2.20.2.20.1.

Table 2.20-123 Butterfly net_load Information.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.
buffer	pointer to char	Standard C type.
count	int	Standard C type.
offset	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
np	register pointer to struct netlib_pb	/simnet/libsrc/libnetif/libnet.h
nloaded	int	Standard C type.
werr	int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Operation failed, errno set to indicate error.
0	int	Successful.
Errors		
Error Name	**Reason for Error**	
EBADF	Not a valid descriptor.	
EIO	The network interface did not respond.	
Calls		
Function	**Where Described**	
CHECK_HANDLE	Macro defined in /simnet/libsrc/libnetif/libnet.h.	
XXX_LSEEK	Macro defined in /simnet/libsrc/libnetif/libnet.h.	
XXX_WRITE	Macro defined in /simnet/libsrc/libnetif/libnet.h.	

Table 2.20-124 MassComp net_load Information.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.
buffer	pointer to char	Standard C type.
count	int	Standard C type.
offset	int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
0	int	Always returned.
Errors		
Error Name	**Reason for Error**	
EBADF	Not a valid descriptor.	
EIO	The network interface did not respond.	
Calls		
Function	**Where Described**	
CHECK_HANDLE	Macro defined in /simnet/libsrc/libnetif/libnet.h.	

Table 2.20-125 MSC net_load Information.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	*int*	Standard C type.
buffer	pointer to char	Standard C type.
count	int	Standard C type.
offset	int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Operation failed, errno set to indicate error.
0	int	Successful.
Errors		
Error Name	**Reason for Error**	
EBADF	Not a valid descriptor.	
EIO	The network interface did not respond.	
Calls		
Function	**Where Described**	
CHECK_HANDLE	Macro defined in /simnet/libsrc/libnetif/libnet.h.	
movebytes	Function defined in libmove. Section 2.21.5.	

Table 2.20-126 Sun net_load Information.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.
buffer	pointer to char	Standard C type.
count	int	Standard C type.
offset	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
np	register pointer to struct netlib_pb	/simnet/libsrc/libnetif/libnet.h
cc	int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Operation failed, errno set to indicate error.
0	int	Successful.
Errors		
Error Name	**Reason for Error**	
EBADF	Not a valid descriptor.	
EIO	The network interface did not respond.	
Calls		
Function	**Where Described**	
CHECK_HANDLE	Macro defined in /simnet/libsrc/libnetif/libnet.h.	
XXX_LSEEK	Macro defined in /simnet/libsrc/libnetif/libnet.h.	
XXX_WRITE	Macro defined in /simnet/libsrc/libnetif/libnet.h.	

Table 2.20-127 MIPS net_load Information.

2.20.2.9.2 net_unload

net_unload copies *count* bytes from *offset* bytes from the start of the memory on the network interface to *buffer*. The function call is net_unload(h, buffer, count , offset). Table 2.20-128 describes the parameters used, return values and possible errors generated by using this function with a Butterfly. Table 2.20-129 describes the parameters used, return values and possible errors generated by using this function with a MassComp. Table 2.20-130 describes the parameters used, return values and possible errors generated by using this function with a Sun. Table 2.20-131 describes the parameters used, return values and possible errors generated by using this function with an MSC. Table 2.20-132 describes the parameters used, return values and possible errors generated by using this function with a MIPS.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.
buffer	pointer to char	Standard C type.
count	int	Standard C type.
offset	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
np	register pointer to struct netlib_pb	/simnet/libsrc/libnetif/libnet.h
localbuf	pointer to char	Standard C type.
localbufoid	OID	
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Operation failed, errno set to indicate error.
0	int	Successful.
Errors		
Error Name	**Reason for Error**	
EBADF	Not a valid descriptor.	
EIO	The network interface did not respond.	
Calls		
Function	**Where Described**	
CHECK_HANDLE	Macro defined in /simnet/libsrc/libnetif/libnet.h.	
net_access	Section 2.20.2.20.1.	
Do_bt		

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.
buffer	pointer to char	Standard C type.
count	int	Standard C type.
offset	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
np	register pointer to struct netlib_pb	/simnet/libsrc/libnetif/libnet.h
nunloaded	int	Standard C type.
rerr	int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Operation failed, errno set to indicate error.
0	int	Successful.

Errors	
Error Name	**Reason for Error**
EBADF	Not a valid descriptor.
EIO	The network interface did not respond.
Calls	
Function	**Where Described**
CHECK_HANDLE	Macro defined in /simnet/libsrc/libnetif/libnet.h.
XXX_LSEEK	Macro defined in /simnet/libsrc/libnetif/libnet.h.
XXX_READ	Macro defined in /simnet/libsrc/libnetif/libnet.h.

Table 2.20-129 MassComp net_unload Information.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.
buffer	pointer to char	Standard C type.
count	int	Standard C type.
offset	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
np	register pointer to struct netlib_pb	/simnet/libsrc/libnetif/libnet.h
nunloaded	int	Standard C type.
rerr	int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Operation failed, errno set to indicate error.
0	int	Successful.
Errors		
Error Name	**Reason for Error**	
EBADF	Not a valid descriptor.	
EIO	The network interface did not respond.	
Calls		
Function	**Where Described**	
CHECK_HANDLE	Macro defined in /simnet/libsrc/libnetif/libnet.h.	
XXX_LSEEK	Macro defined in /simnet/libsrc/libnetif/libnet.h.	
XXX_READ	Macro defined in /simnet/libsrc/libnetif/libnet.h.	

Table 2.20-130 Sun net_unload Information.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.
buffer	pointer to char	Standard C type.
count	int	Standard C type.
offset	int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
0	int	Always returned.
Errors		
Error Name	**Reason for Error**	
EBADF	Not a valid descriptor.	
EIO	The network interface did not respond.	
Calls		
Function	**Where Described**	
CHECK_HANDLE	Macro defined in /simnet/libsrc/libnetif/libnet.h.	

Table 2.20-131 MSC net_unload Information.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.
buffer	pointer to char	Standard C type.
count	int	Standard C type.
offset	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
np	register pointer to struct netlib_pb	/simnet/libsrc/libnetif/libnet.h
cc	int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Operation failed, errno set to indicate error.
0	int	Successful.
Errors		
Error Name	**Reason for Error**	
EBADF	Not a valid descriptor.	
EIO	The network interface did not respond.	

Calls	
Function	**Where Described**
CHECK_HANDLE	Macro defined in /simnet/libsrc/libnetif/libnet.h.
XXX_LSEEK	Macro defined in /simnet/libsrc/libnetif/libnet.h.
XXX_READ	Macro defined in /simnet/libsrc/libnetif/libnet.h.

Table 2.20-132 MIPS net_unload Information.

2.20.2.10 net_open.c
/simnet/libsrc/libnetif/net_open.c

2.20.2.10.1 net_open

net_open opens an access path to the network interface. *path* specifies the network interface to open. *flags* specifies options to be active for subsequent accesses to the network interface. Currently, the only flag defined is NETINIT, which should be used only by ringstart. Otherwise, *flags* should be 0. The third parameter is only necessary if NETINIT is set. The net_info_block contains default information for the rings and is used only during system intialization by the network daemon. This routine will test to determine whether the network interface card is alive and will fail if it does not respond. The function call is net_open(path, flags, nib). Table 2.20-133 describes the parameters used, return values and possible errors generated by using this function with a Butterfly. Table 2.20-134 describes the parameters used, return values and possible errors generated by using this function with any other computer.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
svr	pointer to char	Standard C type.
flags	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
mb_mr	MBMAPREGION	Standard C type.
h	int	Standard C type.
s	array 120 of char	Standard C type.
p	pointer to char	Standard C type.
np	register pointer to struct netlib_pb	/simnet/libsrc/libnetif/libnet.h
i	int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Operation failed, errno set to indicate error.
h	int	Successful. Valid network interface descriptor for subsequent calls

Errors	
Error Name	**Reason for Error**
ENODEV	The network interface does not exist.
EMFILE	Too many network interfaces open.
EIO	The network interface did not respond.
Calls	
Function	**Where Described**
get_device_number	Section 2.20.2.10.7.
get_type	Section 2.20.2.10.6.
net_access	Section 2.20.2.20.1.

Table 2.20-133 Butterfly net_open Information.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
path	pointer to char	Standard C type.
flags	int	Standard C type.
nib	pointer to struct net_info_block	/simnet/libsrc/libnetif/network.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
h	int	Standard C type.
np	pointer to struct netlib_pb	/simnet/libsrc/libnetif/libnet.h
ret	int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Operation failed, errno set to indicate error.
ret	int	Successful.
Errors		
Error Name	**Reason for Error**	
ENODEV	The network interface does not exist.	
EMFILE	Too many network interfaces open.	
EIO	The network interface did not respond.	
Calls		
Function	**Where Described**	
get_type	Section 2.20.2.10.6.	
open_cmc	Section 2.20.2.10.4.	
open_147	Section 2.20.2.10.5.	

2.20.2.10.2 net_get_parameters

net_get_parameters gets the current network parameters. These include such things as the number of ring elements and size of ring elements, and various memory keys. If the flag is FALSE, the parameters are taken from the process's information block. If the flag is TRUE, the parameters are obtained from the driver directly. Normally, these should be identical, but they may not be if net_open (Section 2.20.2.10.1) was called with the NETINIT flag. The function call is net_get_parameterss(h, nib, flag). Table 2.20-135 describes the parameters used, return values and possible errors generated by using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.
nib	pointer to struct net_info_block	/simnet/libsrc/libnetif/network.h.
flag	int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Operation failed, errno set to indicate error.
0	int	Successful.
Errors		
Error Name	**Reason for Error**	
EBADF	Not a valid descriptor.	
Calls		
Function	**Where Described**	
CHECK_HANDLE	Macro defined in /simnet/libsrc/libnetif/libnet.h.	
net_access	Section 2.20.2.20.1.	
Called By		
Function	**Where Described**	
open_cmc	Section 2.20.2.10.4.	

Table 2.20-135 net_get_parameters Information.

2.20.2.10.3 net_set_parameterss

net_set_parameters copies the passed parameter block to the driver (flag is TRUE) or the per-process parameter block (flag is FALSE). net_set_parameters will rarely be used except in ringstart. The function call is net_set_parameters(h, nib, flag). Table 2.20-136 describes the parameters used, return values and possible errors generated by using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.
nib	pointer to struct net_info_block	/simnet/libsrc/libnetif/network.h
flag	int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Operation failed, errno set to indicate error.
0	int	Successful.
Errors		
Error Name	**Reason for Error**	
EBADF	Not a valid descriptor.	
Calls		
Function	**Where Described**	
CHECK_HANDLE	Macro defined in /simnet/libsrc/libnetif/libnet.h.	
net_access	Section 2.20.2.20.1.	

Table 2.20-136 net_set_parameters Information.

2.20.2.10.4 open_cmc

open_cmc is the open handler for the cmc card. It is an internal library function. *device* and *flags* are the same as for net_open. *nib* is a pointer to a network interface block; *h* is a descriptor; *np* is a pointer to a data structure that is internal to the library and controls the access. The function call is open_cmc(device, flags, nib, h, np). Table 2.20-137 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
device	pointer to char	Standard C type.
flags	int	Standard C type.
nib	pointer to struct net_info_block	/simnet/libsrc/libnetif/network.h
h	int	Standard C type.
np	pointer to struct netlib_pb	/simnet/libsrc/libnetif/libnet.h
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Unsuccessful.
h	int	Successful.

Calls	
Function	**Where Described**
net_access	Section 2.20.2.20.1.
get_device_number	Section 2.20.2.10.7.
net_alive	Section 2.20.2.1.1.
net_get_parameters	Section 2.20.2.10.2.
get_locks	Section 2.20.2.21.1.
map_buffers	Section 2.20.2.21.2.
map_enp	Section 2.20.2.21.3.
unmap_buffers	Section 2.20.2.21.5.
unget_locks	Section 2.20.2.21.6.
XXX_CLOSE	Macro defined in /simnet/libsrc/libnetif/libnet.h.
Called By	
Function	**Where Described**
net_open	Section 2.20.2.10.1.

Table 2.20-137 open_cmc Information.

2.20.2.10.5 open_147

open_147 is not currently supported by any computer used by SIMNET. The function call is open_147(device, flags, nib, h, np). Table 2.20-138 describes the parameters used by using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
device	pointer to char	Standard C type.
flags	int	Standard C type.
nib	pointer to struct net_info_block	/simnet/libsrc/libnetif/network.h
h	int	Standard C type.
np	pointer to struct netlib_pb	/simnet/libsrc/libnetif/libnet.h
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Always returned.
Called By		
Function	**Where Described**	
net_open	Section 2.20.2.10.1.	

Table 2.20-138 open_147 Information.

2.20.2.10.6 get_type

get_type is an internal library function that gets the type of the network interface, whether cmc (NIF_TYPE_CMC) or MVME147 (NIF_TYPE_147). *np* is a pointer to the control structure for the access; *dev* is a pointer to the device name; *flags* are the flags passed to net_open. net_access() is only called if SIMBFLY is defined. The function call is get_type(np,dev,flags). Table 2.20-139 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
np	register pointer to struct netlib_pb	/simnet/libsrc/libnetif/libnet.h
dev	pointer to char	Standard C type.
flags	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
h	int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Unsuccessful.
0	int	Successful.
Calls		
Function	**Where Described**	
net_access	Section 2.20.2.20.1.	
XXX_OPEN	Macro defined in /simnet/libsrc/libnetif/libnet.h.	
XXX_IOCTL	Macro defined in /simnet/libsrc/libnetif/libnet.h.	
XXX_CLOSE	Macro defined in /simnet/libsrc/libnetif/libnet.h.	
Called By		
Function	**Where Described**	
net_open	Section 2.20.2.10.1.	

Table 2.20-139 get_type Information.

2.20.2.10.7 get_device_number

get_device_number always returns 0. The function call is get_device_number(device). Table 2.20-140 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
device	pointer to char	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
0	int	Always returned.
Called By		
Function	**Where Described**	
net_open	Section 2.20.2.10.1.	
open_cmc	Section 2.20.2.10.4.	

Table 2.20-140 get_device_number Information.

2.20.2.11 net_mca.c
/simnet/libsrc/libnetif/net_mca.c

net_mca.c contains routines to control multicast addressing.

2.20.2.11.1 net_add_mca

net_add_mca adds a multicast address to the list of multicast addresses that the network interface is filtering on. The *pna* argument is a pointer to the multicast address. Up to the number of addresses specified in the network.def file (see network.def) may be passes to the network interface. The function call is net_add_mca(h, pna). Table 2.20-141 describes the parameters used, return values and possible errors generated by using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.
pna	pointer to NetworkAddress	/simnet/libsrc/libnetif/network.h
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Operation failed, errno set to indicate error.
0	int	Successful.
Errors		
Error Name	**Reason for Error**	
EBADF	Not a valid descriptor.	
EIO	The network interface did not respond.	
Calls		
Function	**Where Described**	
CHECK_HANDLE	Macro defined in /simnet/libsrc/libnetif/libnet.h.	
net_access	Section 2.20.2.20.1.	

Table 2.20-141 net_add_mca Information.

2.20.2.11.2 net_del_mca

net_del_mca deletes a multicast address from the list of multicast addresses on which the network interface is filtering. The *pna* argument is a pointer to the multicast address. Up to the number of addresses specified in the network.def file (see network.def) may be passes to the network interface. The function call is net_del_mca(h, pna). Table 2.20-142 describes the parameters used, return values and possible errors generated by using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.
pna	pointer to NetworkAddress	/simnet/libsrc/libnetif/network.h

Return Values		
Return Value	**Type**	**Meaning**
-1	int	Operation failed, errno set to indicate error.
0	int	Successful.
Errors		
Error Name	**Reason for Error**	
EBADF	Not a valid descriptor.	
EIO	The network interface did not respond.	
Calls		
Function	**Where Described**	
CHECK_HANDLE	Macro defined in /simnet/libsrc/libnetif/libnet.h.	
net_access	Section 2.20.2.20.1.	

Table 2.20-142 net_del_mca Information.

2.20.2.11.3 net_init_mca

net_init_mca initializes the list of multicast addresses (deletes all addresses). Up to the number of addresses specified in the network.def file (see network.def) may be passed to the network interface. The function call is net_init_mca(h). Table 2.20-143 describes the parameters used, return values and possible errors generated by using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Operation failed, errno set to indicate error.
0	int	Successful.
Errors		
Error Name	**Reason for Error**	
EBADF	Not a valid descriptor	
EIO	The network interface did not respond	
Calls		
Function	**Where Described**	
CHECK_HANDLE	Macro defined in /simnet/libsrc/libnetif/libnet.h.	
net_access	Section 2.20.2.20.1.	

Table 2.20-143 net_init_mca Information.

2.20.2.12 net_stam.c

/simnet/libsrc/libnetif/net_stam.c

2.20.2.12.1 net_stamp_enable

net_stamp_enable enables packet timestamping for the network interface specified by descriptor *h*. The default condition upon power up is timestamping disabled. The function call is net_stamp_enable(h). Table 2.20-144 describes the parameters used, return values and possible errors generated by using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Operation failed, errno set to indicate error.
0	int	Successful.
Errors		
Error Name	**Reason for Error**	
EBADF	Not a valid descriptor.	
EIO	The network interface did not respond.	
Calls		
Function	**Where Described**	
CHECK_HANDLE	Macro defined in /simnet/libsrc/libnetif/libnet.h.	
net_access	Section 2.20.2.20.1.	

Table 2.20-144 net_stamp_enable Information.

2.20.2.12.2 net_stamp_disable

net_stamp_disable disables packet timestamping for the network interface specified by descriptor *h*. The default condition upon power up is timestamping disabled. The function call is net_stamp_disable(h). Table 2.20-145 describes the parameters used, return values and possible errors generated by using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Operation failed, errno set to indicate error.
0	int	Successful.
Errors		
Error Name	**Reason for Error**	
EBADF	Not a valid descriptor.	
EIO	The network interface did not respond.	

Calls	
Function	**Where Described**
CHECK_HANDLE	Macro defined in /simnet/libsrc/libnetif/libnet.h.
net_access	Section 2.20.2.20.1.

Table 2.20-145 net_stamp_disable Information.

2.20.2.12.3 net_get_timestamp

net_get_timestamp gets the timestamp for the last packet received from the network interface specified by descriptor *h*. The value returned has meaning only if timestamping is enabled. Each received packet is timestamped (when enabled) with the number of ticks (where a tick is 998.26 microseconds) that have elapsed since the network interface was booted or the timer on the network interface was set until the packet was received. The function call is net_get_timestamp(h). Table 2.20-146 describes the parameters used, return values and possible errors generated by using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Operation failed, errno set to indicate error.
timestamp of most recent packet	int	Successful.
Errors		
Error Name	**Reason for Error**	
EBADF	Not a valid descriptor.	
Calls		
Function	**Where Described**	
CHECK_HANDLE	Macro defined in /simnet/libsrc/libnetif/libnet.h.	

Table 2.20-146 net_get_timestamp Information.

2.20.2.12.4 net_put_timestamp

net_put_timestamp sets the timestamp value to *time* for the next packet to be transmitted on the network interface specified by descriptor *h*. The value is significant only if timestamping is enabled. The next packet will not actually be transmitted until the timer value of the timer on the network interface equals or is greater than the timestamp value set with net_set_timestamp. The function call is net_put_timestamp(h, time). Table 2.20-147 describes the parameters used, return values and possible errors generated by using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.
time	int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Operation failed, errno set to indicate error.
0	int	Successful.
Errors		
Error Name	**Reason for Error**	
EBADF	Not a valid descriptor.	
Calls		
Function	**Where Described**	
CHECK_HANDLE	Macro defined in /simnet/libsrc/libnetif/libnet.h.	

Table 2.20-147 net_put_timestamp Information.

2.20.2.13 net_run.c
/simnet/libsrc/libnetif/net_run.c

2.20.2.13.1 net_run

net_run causes the network interface to begin executing downloaded code. Initially, the processor is executing out of a rom-based kernel. The net_run call causes control to be transferred to the downloaded code from the rommed kernel on the network card. The processor then executes in an idle loop, waiting for a command to cause it to start executing either in normal mode (Section 2.20.2.7.1, net_norm), internal loopback mode (Section 2.20.2.7.2, net_intloop), or external loopback mode (Section 2.20.2.7.3, net_extloop). Also Section 2.20.2.13.2, net_stop. The function call is net_run(h). Table 2.20-148 describes the parameters used, return values and possible errors generated by using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Operation failed, errno set to indicate error.
0	int	Successful.
Errors		
Error Name	**Reason for Error**	
EBADF	Not a valid descriptor.	
EIO	The network interface did not respond.	

Calls	
Function	**Where Described**
CHECK_HANDLE	Macro defined in /simnet/libsrc/libnetif/libnet.h.
net_access	Section 2.20.2.20.1.

Table 2.20-148 net_run Information.

2.20.2.13.2 net_stop

net_stop causes control to return to the idle loop. It is used in conjunction with net_run. The function call is net_stop(h). Table 2.20-149 describes the parameters used, return values and possible errors generated by using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Operation failed, errno set tc indicate error.
0	int	Successful.
Errors		
Error Name	**Reason for Error**	
EBADF	Not a valid descriptor.	
EIO	The network interface did not respond.	
Calls		
Function	**Where Described**	
CHECK_HANDLE	Macro defined in /simnet/libsrc/libnetif/libnet.h.	
net_access	Section 2.20.2.20.1.	

Table 2.20-149 net_stop Information.

2.20.2.14 net_orecv.c
/simnet/libsrc/libnetif/net_orecv.c

net_orecv.c contains routines to perform data access functions.

2.20.2.14.1 net_recv

net_recv reads the next available packet from the network interface into buffer *buf* if *hdr* is NULL or copies the Ethernet header into *hdr* if *hdr* is non-NULL and places the data portion of the packet into buffer *buf*. In either case, the number of bytes placed in *plen* is obtained from the *bufsize*. Three flag bits may be specified in *flags* which affect how the write access will be performed. If NETLOCK is set, a semaphore is used to enforce exclusive access to the queue of packets. This allows multiple processes to read the packet queues without corrupting the data structures that control access by the network interface and host processes to the packet queues. If NETLOCK is not set, the semaphores are not examined or set and only one process is assumed to ever access the queues. If NETWAIT is set, the net_recv call will poll (spin) until a packet can be obtained from the queue. If

NETWAIT is not set, net_recv will return a TRUE indication if a packet was successfully received and a FALSE indication if not. Note that if both NETWAIT and NETLOCK are set, the process will sleep on the semaphore associated with the queue and the poll the queue until a packet is received. If NETBLOCK is set, the net_recv call will perform a blocking access of the appropriate packet queue. This is true blocking access, i.e., the calling process will sleep. The access will be exclusive, i.e., a semaphore will be used to control access. NETWAIT and NETLOCK may not be specified in conjunction with NETBLOCK. A timeout for blocking accesses may be specified via the net_settimeout call. The function call is net_recv(h, hdr, buf, bufsize, plen, flags). Table 2.20-150 describes the parameters used, return values and possible errors generated by using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.
hdr	pointer to NetworkHeader	/simnet/libsrc/libnetif/network.h
buf	pointer to char	Standard C type.
bufsize	int	Standard C type.
plen	pointer to int	Standard C type.
flags	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
np	register pointer to struct netlib_pb	/simnet/libsrc/libnetif/libnet.h
ret	int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Operation failed, errno set to indicate error.
0	int	Successful.
Errors		
Error Name	**Reason for Error**	
E2BIG	Bad register number.	
EBADF	Not a valid descriptor.	
EINVAL	Memory cannot be mapped.	
EIO	Timeout occured.	
Calls		
Function	**Where Described**	
CHECK_HANDLE	Macro defined in /simnet/libsrc/libnetif/libnet.h.	
recv_cmc	Section 2.20.2.14.2.	
recv_147	Section 2.20.2.14.3.	

Table 2.20-150 net_recv Information.

2.20.2.14.2 recv_cmc

recv_cmc is the receive handler for the cmc card. It is an internal library function. Arguments are the same as net_rcv, except for the addition of *np* which is a pointer to the access control structure. *blocking_args* and the call to net_access() are only used if SIMBFLY is NOT defined. The function call is recv_cmc(h, hdr, buf, bufsize, plen, flags, np). Table 2.20-151 describes the parameters used, errors returned and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.
hdr	pointer to NetworkHeader	/simnet/libsrc/libnetif/network.h
buf	pointer to char	Standard C type.
bufsize	int	Standard C type.
plen	pointer to int	Standard C type.
flags	int	Standard C type.
np	register pointer to struct netlib_pb	/simnet/libsrc/libnetif/libnet.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
rp	pointer to VOLATILE RingElement	/simnet/libsrc/libnetif/network.h
tmp	int	Standard C type.
ret	int	Standard C type.
blocking_args	struct enprwsupport_arg	
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Unsuccessful. Error code returned in erno.
ret	int	0 if successful. -1 if unsuccessful.
Errors		
Error Name	**Reason for Error**	
EWOULDBLOCK		
EINVAL	Memory cannot be mapped.	
E2BIG		
Calls		
Function	**Where Described**	
GETLONG	Macro defined in /simnewt/libsrc/libnetif/libnet.h.	
wait_for_full_ring_element	Section 2.20.2.17.8.	
POLL_LOCK	Macro defined in /simnewt/libsrc/libnetif/netlock.h.	
REMOVE_LOCK	Macro defined in /simnewt/libsrc/libnetif/netlock.h.	
WAIT_FOR_LOCK	Macro defined in /simnewt/libsrc/libnetif/netlock.h.	
net_access	Section 2.20.2.20.1.	
L_REFLECT	compat.h	
DATACOPY	Macro defined in /simnewt/libsrc/libnetif/libnet.h.	
SETLONG	Macro defined in /simnewt/libsrc/libnetif/libnet.h.	

Called By	
Function	**Where Described**
net_recv	Section 2.20.2.14.1.

Table 2.20-151 recv_cmc Information.

2.20.2.14.3 recv_147

recv_147 is not supported by any computer currently used in SIMNET. The function call is recv_147(h, hdr, buf, bufsize, plen, flags, np). Table 2.20-152 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.
hdr	pointer to NetworkHeader	/simnet/libsrc/libnetif/network.h
buf	pointer to char	Standard C type.
bufsize	int	Standard C type.
plen	pointer to int	Standard C type.
flags	int	Standard C type.
np	register pointer to struct netlib_pb	/simnet/libsrc/libnetif/libnet.h
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Always returned.
Called By		
Function	**Where Described**	
net_recv	Section 2.20.2.14.1.	

Table 2.20-152 recv_147 Information.

2.20.2.14.4 net_get_next_packet

net_get_next_packet is not used in the version 6.6 release. The function call is net_get_next_packet(h, hdr, buf, plen, flags). Table 2.20-153 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.
hdr	pointer to pointer to NetworkHeader	/simnet/libsrc/libnetif/network.h
buf	pointer to pointer to char	Standard C type.
plen	pointer to int	Standard C type.
flags	int	Standard C type.

Return Values		
Return Value	**Type**	**Meaning**
net_get_recv(h, hdr, b uf, plen, flags)	int	Not applicable
Calls		
Function	**Where Described**	
net_get_recv	Section 2.20.2.14.6.	

Table 2.20-153 net_get_next_packet Information.

2.20.2.14.5 net_release_next_packet

net_release_next_packet is not used in the version 6.6 release. The function call is net_release_next_packet(h, flags). Table 2.20-154 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.
flags	int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
net_release_recv(h, flags)	int	Not applicable
Calls		
Function	**Where Described**	
net_release_recv	Section 2.20.2.14.7.	

Table 2.20-154 net_release_next_packet Information.

2.20.2.14.6 net_get_recv

net_get_recv gets a pointer to the header, *hdr*, and the data part, *buf*, of the next packet in the queue and also returns the number of bytes in the buffer, *plen*. It will not try to use NULL pointers passed as arguments. The function call is net_get_recv(h, hdr, buf, plen, flags). Table 2.20-155 describes the parameters used, errors returned and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.
hdr	pointer to pointer to NetworkHeader	/simnet/libsrc/libnetif/network.h
buf	pointer to pointer to char	Standard C type.
plen	pointer to int	Standard C type.
flags	int	Standard C type.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
np	register pointer to struct netlib_pb	/simnet/libsrc/libnetif/libnet.h
rp	register pointer to RingElement	/simnet/libsrc/libnetif/network.h
ret	int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
ret	int	0 if successful and packet waiting. -1 if not successful.
Errors		
Error Name	**Reason for Error**	
EWOULDBLOCK		
Calls		
Function	**Where Described**	
CHECK_HANDLE	Macro defined in /simnet/libsrc/libnetif/libnet.h.	
GETLONG	Macro defined in /simnet/libsrc/libnetif/libnet.h.	
Called By		
Function	**Where Described**	
net_get_next_packet	Section 2.20.2.14.4.	

Table 2.20-155 net_get_recv Information.

2.20.2.14.7 net_release_recv

net_release_recv releases the next queue element and returns. The function call is net_release_recv(h, flags). Table 2.20-156 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.
flags	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
np	register pointer to struct netlib_pb	/simnet/libsrc/libnetif/libnet.h
rp	register pointer to RingElement	/simnet/libsrc/libnetif/network.h
tmp	int	Standard C type.
ret	int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
ret	int	0 if successful. -1 if not successful.

Calls	
Function	**Where Described**
CHECK_HANDLE	Macro defined in /simnet/libsrc/libnetif/libnet.h.
GETLONG	Macro defined in /simnet/libsrc/libnetif/libnet.h.
SETLONG	Macro defined in /simnet/libsrc/libnetif/libnet.h.
Called By	
Function	**Where Described**
net_release_next_packet	Section 2.20.2.14.5.

Table 2.20-156 net_release_recv Information.

2.20.2.15 net_osend.c
/simnet/libsrc/libnetif/net_osend.c

2.20.2.15.1 net_send

net_send writes the packet in buffer *buf* to the network interface if *hdr* is NULL or obtains the Ethernet header from *hdr* if *hdr* is non-NULL and the data portion of the packet from buffer *buf*. In either case, the number of bytes read from *buf* is obtained from the *len* argument. Three flag bits may be specified in *flags* which affect how the write access will be performed. If NETLOCK is set, a semaphore is used to enforce exclusive access to the queue of packets. This allows multiple processes to write the packet queues without corrupting the data structures that control access by the network interface and host processes to the packet queues. If NETLOCK is not set, the semaphores are not examined or set and only one process is assumed to ever access the queues. If NETWAIT is set, the net_send call will poll (spin) until a packet can be obtained from the queue. If NETWAIT is not set, net_send will return a TRUE indication if a packet was successfully transmitted and a FALSE indication if not. Note that if both NETWAIT and NETLOCK are set, the process will sleep on the semaphore associated with the queue and poll the queue until a packet is received. If NETBLOCK is set, the net_send call will perform a blocking access of the appropriate packet queue. This is true blocking access, i.e., the calling process will sleep. The access will be exclusive, i.e., a semaphore will be used to control access. NETWAIT and NETLOCK may not be specified in conjunction with NETBLOCK. A timeout for blocking accesses may be specified via the net_settimeout call. The function call is net_send(h, hdr, buf, len, flags). Table 2.20-157 describes the parameters used, return values and possible errors generated by using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.
hdr	pointer to NetworkHeader	/simnet/libsrc/libnetif/network.h
buf	pointer to char	Standard C type.
len	int	Standard C type.
flags	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
np	register pointer to struct netlib_pb	/simnet/libsrc/libnetif/libnet.h
ret	int	Standard C type.

Return Values		
Return Value	**Type**	**Meaning**
-1	int	Operation failed, errno set to indicate error.
0	int	Successful.
Errors		
Error Name	**Reason for Error**	
E2BIG	Bad register number.	
EBADF	Not a valid descriptor.	
EINVAL	Memory cannot be mapped.	
EIO	Timeout occured.	
Calls		
Function	**Where Described**	
CHECK_HANDLE	Macro defined in /simnet/libsrc/libnetif/libnet.h.	
send_cmc	Section 2.20.2.15.2.	
send_147	Section 2.20.2.15.3.	

Table 2.20-157 net_send Information.

2.20.2.15.2 send_cmc

send_cmc is the send handler for the cmc card. It takes the same arguments as net_send (Section 2.20.2.15.1) except for the addition of *np* which is a pointer to the access control structure. *blocking_args* and the call to net_access() are only used if SIMBFLY is NOT defined. The function call is send_cmc(h,hdr,buf,len,flags,np). Table 2.20-158 describes the parameters used, errors returned and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.
hdr	pointer to NetworkHeader	/simnet/libsrc/libnetif/network.h
buf	pointer to char	Standard C type.
len	int	Standard C type.
flags	int	Standard C type.
np	register pointer to struct netlib_pb	/simnet/libsrc/libnetif/libnet.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
rp	pointer to VOLATILE RingElement	/simnet/libsrc/libnetif/network.h
tmp	int	Standard C type.
blocking_args	struct enprwsupport_arg	enparg.h
Return Values		
Return Value	**Type**	**Meaning**
0	int	Successful.
-1	int	Unsuccessful.

Errors	
Error Name	**Reason for Error**
EWOULDBLOCK	
EINVAL	Memory cannot be mapped.
Calls	
Function	**Where Described**
GETLONG	Macro defined in /simnet/libsrc/libnetif/libnet.h.
wait_for_empty_ring_element	Section 2.20.2.19.1.
POLL_LOCK	Macro defined in /simnet/libsrc/libnetif/netlock.h.
REMOVE_LOCK	Macro defined in /simnet/libsrc/libnetif/netlock.h.
WAIT_FOR_LOCK	Macro defined in /simnet/libsrc/libnetif/netlock.h.
net_access	Section 2.20.2.20.1.
DATACOPY	Macro defined in /simnet/libsrc/libnetif/libnet.h.
SETLON G	Macro defined in /simnet/libsrc/libnetif/libnet.h.
L_REFLECT	compat.h
Called By	
Function	**Where Described**
net_send	Section 2.20.2.15.1.

Table 2.20-158 send_cmc Information.

2.20.2.15.3 send_147

send_147 is not currently supported by any computer used in SIMNET. The function call is send_147(h, hsr, buf, len, flags, np). *Table 2.20-159 describes the parameters used by* this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.
hdr	pointer to NetworkHeader	/simnet/libsrc/libnetif/network.h
buf	pointer to char	Standard C type.
len	int	Standard C type.
flags	int	Standard C type.
np	register pointer to struct netlib_pb	/simnet/libsrc/libnetif/libnet.h
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Always returned.
Called By		
Function	**Where Described**	
net_send	Section 2.20.2.15.1.	

Table 2.20-159 send_147 Information.

2.20.2.15.4 net_get_send

net_get_send is not used in the version 6.6 release. The function call is net_get_send(h, hdr, buf, plen, flags). Table 2.20-160 describes the parameters used, errors returned and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.
hdr	pointer to pointer to NetworkHeader	/simnet/libsrc/libnetif/network.h
buf	pointer to pointer to char	Standard C type.
plen	pointer to int	Standard C type.
flags	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
np	register pointer to struct netlib_pb	/simnet/libsrc/libnetif/libnet.h
rp	register pointer to RingElement	/simnet/libsrc/libnetif/network.h
ret	int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
ret	int	0 if successful and buffer available waiting. -1 if not successful.
Errors		
Error Name	**Reason for Error**	
EWOULDBLOCK		
Calls		
Function	**Where Described**	
CHECK_HANDLE	Macro defined in /simnet/libsrc/libnetif/libnet.h.	
GETLONG	Macro defined in /simnet/libsrc/libnetif/libnet.h.	

Table 2.20-160 net_get_send Information.

2.20.2.15.5 net_release_send

net_release_send releases the next queue element and returns. The function call is net_release_send(h, flags). Table 2.20-161 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.
flags	int	Standard C type.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
np	register pointer to struct netlib_pb	/simnet/libsrc/libnetif/libnet.h
rp	pointer to RingElement	/simnet/libsrc/libnetif/network.h
tmp	int	Standard C type.
ret	int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
ret	int	0 if successful. -1 if not successful.
Calls		
Function	**Where Described**	
CHECK_HANDLE	Macro defined in /simnet/libsrc/libnetif/libnet.h.	
GETLONG	Macro defined in /simnet/libsrc/libnetif/libnet.h.	
SETLONG	Macro defined in /simnet/libsrc/libnetif/libnet.h.	

Table 2.20-161 net_release_send Information.

2.20.2.16 net_reg.c
/simnet/libsrc/libnetif/net_reg.c

net_reg.c contains routines to access the network interface registers.

2.20.2.16.1 net_reg_read

net_reg_read reads the value of the network interface register specified by *regnum* and returns the value into *pval*. The network interface registers are a block of NUMBER_OF_REGISTERS long words directly mapped into a process's segment for rapid access. The registers are available for communication with on-board packet filtering routines (see net_filter). The ENP30 implementation (Multibus) reads and writes a short (16 bits) at a time, so accesses are not necessarily atomic. The function call is net_reg_read(h, regnum, pval). Table 2.20-162 describes the parameters used, return values and possible errors generated by using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.
regnum	int	Standard C type.
pval	pointer to long	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Operation failed, errno set to indicate error.
0	int	Successful.

Errors	
Error Name	**Reason for Error**
EBADF	Not a valid descriptor.
EINVAL	Memory cannot be mapped.
E2BIG	Bad register number.
Calls	
Function	**Where Described**
CHECK_HANDLE	Macro defined in /simnet/libsrc/libnetif/libnet.h.
GETLONG	Macro defined in /simnet/libsrc/libnetif/libnet.h.

Table 2.20-162 net_reg_read Information.

2.20.2.16.2 net_reg_write

net_reg_write writes the value *val* into the network interface register specified by *regnum*. The network interface registers are a block of NUMBER_OF_REGISTERS long words directly mapped into a process's segment for rapid access. The registers are available for communication with on-board packet filtering routines (see net_filter). The ENP30 implementation (Multibus) reads and writes a short (16 bits) at a time, so accesses are not necessarily automatic. The function call is net_reg_write(h, regnum, val). Table 2.20-163 describes the parameters used, return values and possible errors generated by using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.
regnum	int	Standard C type.
val	long	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Operation failed, errno set to indicate error.
0	int	Successful.
Errors		
Error Name	**Reason for Error**	
EBADF	Not a valid descriptor.	
EINVAL	Memory cannot be mapped.	
E2BIG	Bad register number.	
Calls		
Function	**Where Described**	
CHECK_HANDLE	Macro defined in /simnet/libsrc/libnetif/libnet.h.	
SETLONG	Macro defined in /simnet/libsrc/libnetif/libnet.h.	

Table 2.20-163 net_reg_write Information.

2.20.2.17 net_recv.c
/simnet/libsrc/libnetif/net_recv.c

2.20.2.17.1 net_reset_lock

net_reset_lock initializes the semaphore governing the ring buffers on the MASSCOMP. This is necessary only for ringstart and in case any routine dies while it has the rings locked. The function call is net_reset_lock(h). Table 2.20-164 describes the parameters used, return values and possible errors generated by using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
su	union su	/simnet/libsrc/libnetif/net_recv.c
np	register pointer to struct *netlib_pb*	/simnet/libsrc/libnetif/libnet.h
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Operation failed, errno set to indicate error.
0	int	Successful.
Errors		
Error Name	**Reason for Error**	
EBADF	Not a valid descriptor.	
Calls		
Function	**Where Described**	
CHECK_HANDLE	Macro defined in /simnet/libsrc/libnetif/libnet.h.	

Table 2.20-164 net_reset_lock Information.

2.20.2.17.2 net_rcv

net_rcv receives a packet. The data portion of the packet is returned to *buf*. *bufsize* is the size of *buf* in bytes. The number of bytes actually copied into *buf* is returned in *len*. The destination and source addresses and the packet type may be obtained from calls to Section 2.20.2.17.3 net_get_rcv_to_addr, Section 2.20.2.17.4 net_get_rcv_from_addr and Section 2.20.2.17.5 net_get_rcvtype. The function call is net_rcv(h, buf, bufsize, len, flags). Table 2.20-165 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.
buf	pointer to char	Standard C type.
bufsize	int	Standard C type.
len	pointer to int	Standard C type.
flags	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
np	register pointer to struct netlib_pb	/simnet/libsrc/libnetif/libnet.h
ret	int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
ret	int	0 if successful. -1 if not successful.
Calls		
Function	**Where Described**	
CHECK_HANDLE	Macro defined in /simnet/libsrc/libnetif/libnet.h.	
recv_cmc_8023	Section 2.20.2.17.6.	
recv_147_8023	Section 2.20.2.17.7.	

Table 2.20-165 net_rcv Information.

2.20.2.17.3 net_get_rcv_to_addr

net_get_rcv_to_addr gets the destination address of the last packet received via the calls to Section 2.20.2.17.2 net_rcv or Section 2.20.2.17.9 net_get_rcv. The function call is net_get_rcv_to_addr(h, to). Table 2.20-166 describes the parameters used, errors returned and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.
to	pointer to NetworkAddress	/simnet/libsrc/libnetif/network.h
Return Values		
Return Value	**Type**	**Meaning**
0	int	Successful.
-1	int	Unsuccessful. Error code returned in errno
Errors		
Error Name	**Reason for Error**	
EBADF	Bad handle.	

Calls	
Function	**Where Described**
CHECK_HANDLE	Macro defined in /simnet/libsrc/libnetif/libnet.h.

Table 2.20-166 net_get_rcv_to_addr Information.

2.20.2.17.4 net_get_rcv_from_addr

net_get_rcv_from_addr gets the source address of the last packet received via the calls to Section 2.20.2.17.2 net_rcv or Section 2.20.2.17.9 net_get_rcv. The function call is net_get_rcv_from_addr(h, from). Table 2.20-167 describes the parameters used, errors returned and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.
from	pointer to NetworkAddress	/simnet/libsrc/libnetif/network.h
Return Values		
Return Value	**Type**	**Meaning**
0	int	Successful.
-1	int	Unsuccessful. Error code returned in errno.
Errors		
Error Name	**Reason for Error**	
EBADF	Bad handle.	
Calls		
Function	**Where Described**	
CHECK_HANDLE	Macro defined in /simnet/libsrc/libnetif/libnet.h.	

Table 2.20-167 net_get_rcv_from_addr Information.

2.20.2.17.5 net_get_rcv_type

net_get_rcv_type gets the type of the last packet received via the calls to Section 2.20.2.17.2 net_rcv or Section 2.20.2.17.9 net_get_rcv. The function call is net_get_rcv_type(h). Table 2.20-168 describes the parameters used, errors returned and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
npb[h].rcvType	int	Type of last packet received.

Errors	
Error Name	**Reason for Error**
EBADF	Bad handle.
Calls	
Function	**Where Described**
CHECK_HANDLE	Macro defined in /simnet/libsrc/libnetif/libnet.h.

Table 2.20-168 net_get_rcv_type Information.

2.20.2.17.6 recv_cmc_8023

recv_cmc_8023 is the receive handler for the cmc card for 802.3 packets. The parameters are the same as for net_rcv except for the addition of *np* which is a pointer to the access control structure for the access. *blocking_args* and the call to net_access will only be used if SIMBFLY is NOT defined. The function call is recv_cmc_8023(h, buf, bufsize, plen, flags, np). Table 2.20-169 describes the parameters used, errors returned and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.
buf	pointer to char	Standard C type.
bufsize	int	Standard C type.
plen	pointer to int	Standard C type.
flags	int	Standard C type.
np	register pointer to struct netlib_pb	/simnet/libsrc/libnetif/libnet.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
datap	pointer to VOLATILE char	Standard C type.
size	int	Standard C type.
rp	register pointer to QueueElement	/simnet/libsrc/libnetif/network.h
tmp	int	Standard C type.
ret	int	Standard C type.
blocking_args	struct enprwsupport_arg	
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Unsuccessful. Error code returned in errno.
ret	int	Successful. ret will be 0.
Errors		
Error Name	**Reason for Error**	
EWOULDBLOCK	Input queue empty, no packet available.	
EINVAL	Unknown flag.	
E2BIG	Packet too big for supplied buffer.	

Calls	
Function	**Where Described**
GETLONG	Macro defined in /simnet/libsrc/libnetif/libnet.h.
wait_for_full_ring_element	Section 2.20.2.17.8.
POLL_LOCK	Macro defined in /simnet/libsrc/libnetif/netlock.h.
REMOVE_LOCK	Macro defined in /simnet/libsrc/libnetif/netlock.h.
WAIT_FOR_LOCK	Macro defined in /simnet/libsrc/libnetif/netlock.h.
net_access	Section 2.20.2.20.1.
L_REFLECT	compat.h
movewords	*Function defined in libmove. Section 2.21.5.*
GET_ETHER_TYPE	Macro defined in /simnet/libsrc/libnetif/network.h.
GET_DATA_PTR	Macro defined in /simnet/libsrc/libnetif/network.h.
DATACOPY	Macro defined in /simnet/libsrc/libnetif/libnet.h.
SETLONG	Macro defined in /simnet/libsrc/libnetif/libnet.h.
Called By	
Function	**Where Described**
net_rcv	Section 2.20.2.17.2.

Table 2.20-169 recv_cmc_8023 Information.

2.20.2.17.7 recv_147_8023

recv_147_8023 is not supported for any computer currently used by SIMNET. The function call is recv_147_8023(h, buf, bufsize, plen, size, np). Table 2.20-170 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.
buf	pointer to char	Standard C type.
bufsize	int	Standard C type.
plen	pointer to int	Standard C type.
flags	int	Standard C type.
np	register pointer to struct netlib_pb	/simnet/libsrc/libnetif/libnet.h
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Always returned.
Called By		
Function	**Where Described**	
net_rcv	Section 2.20.2.17.2.	

Table 2.20-170 recv_147_8023 Information.

2.20.2.17.8 wait_for_full_ring_element

wait_for_full_ring_element is a waiting function necessary for the MIPS computer. The function call is wait_for_full_ring_element(rp). Table 2.20-171 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
rp	pointer to VOLATILE RingElement	/simnet/libsrc/libnetif/network.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
dummy	int	Standard C type.
i	register int	Standard C type.
Called By		
Function	**Where Described**	
recv_cmc	Section 2.20.2.14.2.	
recv_cmc_8023	Section 2.20.2.17.6.	

Table 2.20-171 wait_for_full_ring_element Information.

2.20.2.17.9 net_get_rcv

net_get_rcv gets a pointer, *buf*, to the data part of the next packet in the queue and also returns the number of bytes in the buffer, *plen*. The function call is net_get_rcv(h, buf, plen, flags). Table 2.20-172 describes the parameters used, errors returned and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.
buf	pointer to pointer to char	Standard C type.
plen	pointer to int	Standard C type.
flags	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
np	register pointer to struct netlib_pb	/simnet/libsrc/libnetif/libnet.h
rp	register pointer to QueueElement	/simnet/libsrc/libnetif/network.h
ret	int	Standard C type.
size	int	Standard C type.
datap	pointer to char	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
ret	int	0 if successful -1 if not successful

Errors	
Error Name	**Reason for Error**
EWOULDBLOCK	Input queue empty, no packet available.
Calls	
Function	**Where Described**
CHECK_HANDLE	Macro defined in /simnet/libsrc/libnetif/libnet.h.
GETLONG	Macro defined in /simnet/libsrc/libnetif/libnet.h.
L_REFLECT	compat.h
movewords	Function defined in libmove. Section 2.21.5.
GET_ETHER_TYPE	Macro defined in /simnet/libsrc/libnetif/network.h.
GET_DATA_PTR	Macro defined in /simnet/libsrc/libnetif/network.h.
DATACOPY	Macro defined in /simnet/libsrc/libnetif/libnet.h.

Table 2.20-172 net_get_rcv Information.

2.20.2.17.10 net_release_rcv

net_release_rcv releases the next queue element and returns. The function call is net_release_rcv(h, flags). Table 2.20-173 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.
flags	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
np	register pointer to struct netlib_pb	/simnet/libsrc/libnetif/libnet.h
rp	register pointer to QueueElement	/simnet/libsrc/libnetif/network.h
tmp	int	Standard C type.
ret	int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
ret	int	0 if successful. -1 if not successful.
Calls		
Function	**Where Described**	
CHECK_HANDLE	Macro defined in /simnet/libsrc/libnetif/libnet.h.	
GETLONG	Macro defined in /simnet/libsrc/libnetif/libnet.h.	
SETLONG	Macro defined in /simnet/libsrc/libnetif/libnet.h.	

Table 2.20-173 net_release_rcv Information.

2.20.2.18 net_type.c
/simnet/libsrc/libnetif/net_type.c

net_type.c contains routines for ethernet type code checking.

2.20.2.18.1 net_add_type

net_add_type adds a type code to the list of types that the network interface is checking for. The function call is net_add_type(h, type). Table 2.20-174 describes the parameters used and functions called using this function while using a Butterfly. Table 2.20-175 describes the parameters used and functions called using this function with any other computer.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.
type	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
np	register pointer to struct netlib_pb	/simnet/libsrc/libnetif/libnet.h
ret	int	Standard C type.
buf	pointer to char	Standard C type.
bufoid	OID	
Return Values		
Return Value	**Type**	**Meaning**
ret	int	0 if successful. -1 if not successful.
Calls		
Function	**Where Described**	
CHECK_HANDLE	Macro defined in /simnet/libsrc/libnetif/libnet.h.	
net_access	Section 2.20.2.20.1.	

Table 2.20-174 Butterfly net_add_type Information.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.
type	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
np	register pointer to struct netlib_pb	/simnet/libsrc/libnetif/libnet.h
ret	int	Standard C type.
arg	struct enpaccess_arg	enparg.h

Return Values		
Return Value	**Type**	**Meaning**
ret	int	0 if successful. -1 if not successful.
Calls		
Function	**Where Described**	
CHECK_HANDLE	Macro defined in /simnet/libsrc/libnetif/libnet.h.	
net_access	Section 2.20.2.20.1.	

Table 2.20-175 Other net_add_type Information.

2.20.2.18.2 net_init_type

net_init_type initializes a list of type codes the network interface is checking for. If SIMBFLY is defined the last internal variable, *arg*, is not used. The function call is net_init_type(h). Table 2.20-176 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
np	register pointer to struct netlib_pb	/simnet/libsrc/libnetif/libnet.h
ret	int	Standard C type.
arg	struct enpaccess_arg	enparg.h
Return Values		
Return Value	**Type**	**Meaning**
ret	int	0 if successful. -1 if not successful.
Calls		
Function	**Where Described**	
CHECK_HANDLE	Macro defined in /simnet/libsrc/libnetif/libnet.h.	
net_access	Section 2.20.2.20.1.	

Table 2.20-176 net_init_type Information.

2.20.2.19 net_send.c

/simnet/libsrc/libnetif/net_send.c

2.20.2.19.1 wait_for_empty_ring_element

wait_for_empty_ring_element is used to wait while doing nothing. It is necessary for the MIPS machine. The function call is wait_for_empty_ring_element(rp). Table 2.20-177 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
rp	pointer to VOLATILE RingElement	/simnet/libsrc/libnetif/network.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
dummy	int	Standard C type.
i	register int	Standard C type.
Called By		
Function	**Where Described**	
send_cmc	Section 2.20.2.15.2.	
send_cmc_8023	Section 2.20.2.19.3.	

Table 2.20-177 wait_for_empty_ring_element Information.

2.20.2.19.2 net_snd

net_snd sends a packet. The function call is net_snd(h, to, buf, len, flags). Table 2.20-178 describes the parameters used, errors returned and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.
to	pointer to NetworkAddress	/simnet/libsrc/libnetif/network.h
buf	pointer to char	Standard C type.
len	int	Standard C type.
flags	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
np	register pointer to struct netlib_pb	/simnet/libsrc/libnetif/libnet.h
ret	int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
ret	int	0 if successful, -1 if not successful
Calls		
Function	**Where Described**	
CHECK_HANDLE	Macro defined in /simnet/libsrc/libnetif/libnet.h.	
send_cmc_8023	Section 2.20.2.19.3.	
send_147_8023	Section 2.20.2.19.4.	

Table 2.20-178 net_snd Information.

2.20.2.19.3 send_cmc_8023

send_cmc_8023 is the send handler for the cmc card for 802.3 packets. The parameters are the same as for net_snd except for the addition of *np* which is a pointer to the access control structure for the access. *blocking_args* and the call to net_access() are only used if the computer is not a Butterfly. The function call is send_cmc_8023(h, to, buf, len, flags, np). Table 2.20-179 describes the parameters used, errors returned and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.
to	pointer to NetworkAddress	/simnet/libsrc/libnetif/network.h
buf	pointer to char	Standard C type.
len	int	Standard C type.
flags	int	Standard C type.
np	register pointer to struct netlib_pb	/simnet/libsrc/libnetif/libnet.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
rp	pointer to VOLATILE QueueElement	/simnet/libsrc/libnetif/network.h
tmp	int	Standard C type.
blocking_args	struct enprwsupport_arg	enparg.h
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Unsuccessful. Error code returned in errno.
0	int	Successful.
Errors		
Error Name	**Reason for Error**	
EWOULDBLOCK	No room in output queue.	
EINVAL	Unknown flag.	

Calls	
Function	**Where Described**
GETLONG	Macro defined in /simnet/libsrc/libnetif/libnet.h.
wait_for_empty_ring_element	Section 2.20.2.19.1.
POLL_LOCK	Macro defined in /simnet/libsrc/libnetif/netlock.h.
REMOVE_LOCK	Macro defined in /simnet/libsrc/libnetif/netlock.h.
WAIT_FOR_LOCK	Macro defined in /simnet/libsrc/libnetif/netlock.h.
net_access	Section 2.20.2.20.1.
movewords	Function defined in libmove. Section 2.21.5.
SET_DSAP	Macro defined in /simnet/libsrc/libnetif/network.h.
SET_SSAP	Macro defined in /simnet/libsrc/libnetif/network.h.
SET_CONTROL	Macro defined in /simnet/libsrc/libnetif/network.h.
SET_PROTOID	Macro defined in /simnet/libsrc/libnetif/network.h.
SET_ETHER_TYPE	Macro defined in /simnet/libsrc/libnetif/network.h.
DATACOPY	Macro defined in /simnet/libsrc/libnetif/libnet.h.
GET_DATA_PTR	Macro defined in /simnet/libsrc/libnetif/network.h.
SETLONG	Macro defined in /simnet/libsrc/libnetif/libnet.h.
L_REFLECT	compat.h
Called By	
Function	**Where Described**
net_snd	Section 2.20.2.19.2.

Table 2.20-179 send_cmc_8023 Information.

2.20.2.19.4 send_147_8023

send_147_8023 is not supported by any computer used in SIMNET. This function will always return -1. The function call is send_147_8023(h, to, buf, len, flags, np) Table 2.20-180 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.
to	pointer to NetworkAddress	/simnet/libsrc/libnetif/network.h
buf	pointer to char	Standard C type.
len	int	Standard C type.
flags	int	Standard C type.
np	register pointer to struct netlib_pb	/simnet/libsrc/libnetif/libnet.h
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Always returned.
Called By		
Function	**Where Described**	
net_snd	Section 2.20.2.19.2.	

Table 2.20-180 send_147_8023 Information.

2.20.2.19.5 net_set_snd_from_addr

net_set_snd_from_addr sets the source address for subsequent calls to net_snd(). If this is any multicast address (least significant bit of the first byte equal to 1) then the default address of the interface is used. net_set_snd_from_addr returns 0 if successful and -1 if not. The function call is net_set_snd_from_addr(h, from). Table 2.20-181 describes the parameters used, errors returned and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.
from	pointer to NetworkAddress	/simnet/libsrc/libnetif/network.h
Return Values		
Return Value	**Type**	**Meaning**
0	int	Successful.
-1	int	Unsuccessful.
Errors		
Error Name	**Reason for Error**	
EBADF	Bad handle.	
Calls		
Function	**Where Described**	
CHECK_HANDLE	Macro defined in /simnet/libsrc/libnetif/libnet.h.	

Table 2.20-181 net_set_snd_from_addr Information.

2.20.2.19.6 net_set_snd_type

net_set_snd_type sets the packet type for subsequent calls to net_snd(). It returns 0 if successful and -1 if not. The function call is net_set_snd_type(h, type). Table 2.20-182 describes the parameters used, errors returned and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.
type	int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
0	int	Successful.
-1	int	Unsuccessful.
Errors		
Error Name	**Reason for Error**	
EBADF	Bad handle.	

Calls	
Function	**Where Described**
CHECK_HANDLE	Macro defined in /simnet/libsrc/libnetif/libnet.h.

Table 2.20-182 net_set_snd_type Information.

2.20.2.19.7 net_get_snd

net_get_snd gets a pointer, *buf*, to the data part of the next packet in the queue. *plen* returns the size of *buf* in bytes. net_get_snd returns 0 if there is a buffer available and -1 if not. The function call is net_get_snd(h, buf, plen, flags). Table 2.20-183 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.
buf	pointer to pointer to char	Standard C type.
plen	pointer to int	Standard C type.
flags	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
np	register pointer to struct netlib_pb	/simnet/libsrc/libnetif/libnet.h
rp	register pointer to QueueElement	/simnet/libsrc/libnetif/network.h
ret	int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
ret	int	Successful if 0, unsuccessful if -1.
Calls		
Function	**Where Described**	
CHECK_HANDLE	Macro defined in /simnet/libsrc/libnetif/libnet.h.	
GETLONG	Macro defined in /simnet/libsrc/libnetif/libnet.h.	
GET_DATA_PTR	Macro defined in /simnet/libsrc/libnetif/network.h.	

Table 2.20-183 net_get_snd Information.

2.20.2.19.8 net_release_snd

net_release_snd releases the next queue element and returns. The function call is net_release_snd(h, to, len, flags). Table 2.20-184 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.
to	pointer to NetworkAddress	/simnet/libsrc/libnetif/network.h
len	int	Standard C type.
flags	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
np	register pointer to struct netlib_pb	/simnet/libsrc/libnetif/libnet.h
rp	register pointer to QueueElement	/simnet/libsrc/libnetif/network.h
tmp	int	Standard C type.
ret	int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
ret	int	Successful if 0, unsuccessful if -1.
Calls		
Function	**Where Described**	
CHECK_HANDLE	Macro defined in /simnet/libsrc/libnetif/libnet.h.	
GETLONG	Macro defined in /simnet/libsrc/libnetif/libnet.h.	
movewords	Function defined in libmove. Section 2.21.5.	
SET_DSAP	Macro defined in /simnet/libsrc/libnetif/network.h.	
SET_SSAP	Macro defined in /simnet/libsrc/libnetif/network.h.	
SET_CONTROL	Macro defined in /simnet/libsrc/libnetif/network.h.	
SET_PROTOID	Macro defined in /simnet/libsrc/libnetif/network.h.	
SET_ETHER_TYPE	Macro defined in /simnet/libsrc/libnetif/network.h.	
DATACOPY	Macro defined in /simnet/libsrc/libnetif/libnet.h.	
GET_DATA_PTR	Macro defined in /simnet/libsrc/libnetif/network.h.	
SETLONG	Macro defined in /simnet/libsrc/libnetif/libnet.h.	
L_REFLECT	compat.h	

Table 2.20-184 net_release_snd Information.

2.20.2.20 net_acce.c
/simnet/libsrc/libnetif/net_acce.c

2.20.2.20.1 net_access

net_access is the common access routine to the network server. net_access is an internal library call. It returns the error code provided by the operating system the library is running on. *h* is the access handle and *command* is the command code to execute. The command specific arguments are assumed to be set up by the caller. The handle is assumed to be valid. The function call is net_access(h, command). Table 2.20-185 describes the parameters used and functions called using this function with a Butterfly computer. Table 2.20-186 describes the parameters used and functions called using this function with a Sun, MIPS, MSC or MassComp computer.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.
command	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
np	register pointer to struct netlib_pb	/simnet/libsrc/libnetif/libnet.h
ret	int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
np->net_req->ret_code	int	error code

Called By	
Function	**Where Described**
net_alive	Section 2.20.2.1.1.
net_res	Section 2.20.2.1.2.
net_iocontrol	Section 2.20.2.1.4.
net_loopback	Section 2.20.2.1.6.
net_getaddr	Section 2.20.2.2.4.
net_flush	Section 2.20.2.4.1.
net_get_statistics	Section 2.20.2.5.1.
net_zero_statistics	Section 2.20.2.5.2.
net_hostbuf_info	Section 2.20.2.6.1.
net_bufs	Section 2.20.2.6.3.
net_version	Section 2.20.2.6.6.
do_mode_cmd_cmc	Section 2.20.2.7.6.
net_gettime	Section 2.20.2.8.1.
net_settime	Section 2.20.2.8.2.
net_stomp_time	Section 2.20.2.8.5.
net_load	Section 2.20.2.9.1.
net_unload	Section 2.20.2.9.2.
net_open	Section 2.20.2.10.1.
net_get_parameters	Section 2.20.2.10.2.
net_set_parameters	Section 2.20.2.10.3.
open_cmc	Section 2.20.2.10.4.
get_type	Section 2.20.2.10.6.
net_add_mca	Section 2.20.2.11.1.
net_del_mca	Section 2.20.2.11.2.
net_init_mca	Section 2.20.2.11.3.
net_stamp_enable	Section 2.20.2.12.1.
net_stamp_disable	Section 2.20.2.12.2.
net_run	Section 2.20.2.13.1.
net_stop	Section 2.20.2.13.2.
recv_cmc	Section 2.20.2.14.2.
send_cmc	Section 2.20.2.15.2.
recv_cmc_8023	Section 2.20.2.17.6.
net_add_type	Section 2.20.2.18.1.
net_init_type	Section 2.20.2.18.2.
send_cmc_8023	Section 2.20.2.19.3.

Table 2.20-185 Butterfly net_access Information.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.
command	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
np	register pointer to struct netlib_pb	/simnet/libsrc/libnetif/libnet.h
ret	int	Standard C type.

Return Values		
Return Value	**Type**	**Meaning**
-1	int	Unknown NIF type.
ret	int	error code
Calls		
Function	**Where Described**	
access_cmc	Section 2.20.2.20.2.	
access_147	Section 2.20.2.20.3.	
Called By		
Function	**Where Described**	
net_alive	Section 2.20.2.1.1.	
net_res	Section 2.20.2.1.2.	
net_iocontrol	Section 2.20.2.1.4.	
net_loopback	Section 2.20.2.1.6.	
net_getaddr	Section 2.20.2.2.4.	
net_flush	Section 2.20.2.4.1.	
net_get_statistics	Section 2.20.2.5.1.	
net_zero_statistics	Section 2.20.2.5.2.	
net_hostbuf_info	Section 2.20.2.6.1.	
net_bufs	Section 2.20.2.6.3.	
net_version	Section 2.20.2.6.6.	
do_mode_cmd_cmc	Section 2.20.2.7.6.	
net_gettime	Section 2.20.2.8.1.	
net_settime	Section 2.20.2.8.2.	
net_stomp_time	Section 2.20.2.8.5.	
net_load	Section 2.20.2.9.1.	
net_unload	Section 2.20.2.9.2.	
net_open	Section 2.20.2.10.1.	
net_get_parameters	Section 2.20.2.10.2.	
net_set_parameters	Section 2.20.2.10.3.	
open_cmc	Section 2.20.2.10.4.	
get_type	Section 2.20.2.10.6.	
net_add_mca	Section 2.20.2.11.1.	
net_del_mca	Section 2.20.2.11.2.	
net_init_mca	Section 2.20.2.11.3.	
net_stamp_enable	Section 2.20.2.12.1.	
net_stamp_disable	Section 2.20.2.12.2.	
net_run	Section 2.20.2.13.1.	
net_stop	Section 2.20.2.13.2.	
recv_cmc	Section 2.20.2.14.2.	
send_cmc	Section 2.20.2.15.2.	
recv_cmc_8023	Section 2.20.2.17.6.	
net_add_type	Section 2.20.2.18.1.	
net_init_type	Section 2.20.2.18.2.	
send_cmc_8023	Section 2.20.2.19.3.	

Table 2.20-186 Sun, MIPS, MSC and MassComp net_access Information.

2.20.2.20.2 access_cmc

access_cmc retuns the access code to the cmc card. The function call is access_cmc(np, command). Table 2.20-187 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
command	int	Standard C type.
np	pointer to struct netlib_pb	/simnet/libsrc/libnetif/libnet.h
Return Values		
Return Value	**Type**	**Meaning**
errno	int	Unsuccessful. Error code for error.
0	int	Successful.
Calls		
Function	**Where Described**	
XXX_OPEN	Macro defined in /simnet/libsrc/libnetif/libnet.h.	
XXX_IOCTL	Macro defined in /simnet/libsrc/libnetif/libnet.h.	
Called By		
Function	**Where Described**	
net_access	Section 2.20.2.20.1.	

Table 2.20-187 access_cmc Information.

2.20.2.20.3 access_147

access_147 is not supported by any computers used in SIMNET. This function will always return 0. The function call is access_147(np). Table 2.20-188 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
np	pointer to struct netlib_pb	/simnet/libsrc/libnetif/libnet.h
Return Values		
Return Value	**Type**	**Meaning**
0	int	Always returned.
Called By		
Function	**Where Described**	
net_access	Section 2.20.2.20.1.	

Table 2.20-188 access_147 Information.

2.20.2.21 net_stuf.c
/simnet/libsrc/libnetif/net_stuf.c

None of the following functions are compiled if SIMBFLY is defined. Table 2.20-189 describes the variables used by net_stuf.c.

Variables		
Variable	**Type**	**Where Typedef Declared**
force_some_compilation	int	Standard C type.

Table 2.20-189 net_stuf.c Variable Information.

2.20.2.21.1 get_locks

get_locks gets the locks for the rings. It does nothing and always returns 0 for Sun, MIPS and MSC computers. The function call is get_locks(np). Table 2.20-190 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
np	pointer to struct netlib_pb	/simnet/libsrc/libnetif/libnet.h
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Unsuccessful.
0	int	Successful. Always returned for Sun, MIPS and MSC computers.
Called By		
Function	**Where Described**	
open_cmc	Section 2.20.2.10.4.	

Table 2.20-190 get_locks Information.

2.20.2.21.2 map_buffers

map_buffers maps in the buffers. On the MassComp it attaches the buffers, while on the Sun it just maps them in. The function call is map_buffers(np, flags). Table 2.20-191 describes the parameters used and functions called using this function with a MassComp computer. Table 2.20-192 describes the parameters used and functions called using this function with a MIPS machine. Table 2.20-193 describes the parameters used and functions called using this function with a Sun. Table 2.20-194 describes the parameters used by this function with an MSC.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
np	pointer to struct netlib_pb	/simnet/libsrc/libnetif/libnet.h
flags	int	Standard C type.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
ptr	pointer to char	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Unsuccessful.
0	int	Successful.
Calls		
Function	**Where Described**	
attachshm	Section 2.6.5.1.1 in the Vehicles CSCI.	
Ringsize_in_chars	Macro defined in /simnet/libsrc/libnetif/network.h.	
Called By		
Function	**Where Described**	
open_cmc	Section 2.20.2.10.4.	

Table 2.20-191 MassComp map_buffers Information.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
np	pointer to struct netlib_pb	/simnet/libsrc/libnetif/libnet.h
flags	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
ptr	pointer to char	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Unsuccessful.
0	int	Successful.
Calls		
Function	**Where Described**	
Ringsize_in_chars	Macro defined in /simnet/libsrc/libnetif/network.h.	
Called By		
Function	**Where Described**	
open_cmc	Section 2.20.2.10.4.	

Table 2.20-192 MIPS map_buffers Information.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
np	pointer to struct netlib_pb	/simnet/libsrc/libnetif/libnet.h
flags	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
ptr	pointer to char	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Unsuccessful.
0	int	Successful.
Called By		
Function	**Where Described**	
open_cmc	Section 2.20.2.10.4.	

Table 2.20-193 Sun map_buffers Information.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
np	pointer to struct netlib_pb	/simnet/libsrc/libnetif/libnet.h
flags	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
ptr	pointer to char	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
0	int	Always returned.
Called By		
Function	**Where Described**	
open_cmc	Section 2.20.2.10.4.	

Table 2.20-194 MSC map_buffers Information.

2.20.2.21.3 map_enp

map_enp maps in the CMC card. The function call is map_enp(np). Table 2.20-195 describes the parameters used and functions called using this function with a Masscomp computer. Table 2.20-196 describes the parameters used by this function with a Sun, MIPS or MSC computer.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
np	pointer to struct netlib_pb	/simnet/libsrc/libnetif/libnet.h

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
enp_base	pointer to char	Standard C type.
ptr	pointer to char	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Unsuccessful.
0	int	Successful.
Called By		
Function	**Where Described**	
open_cmc	Section 2.20.2.10.4.	

Table 2.20-195 MassComp map_enp Information.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
np	pointer to struct netlib_pb	/simnet/libsrc/libnetif/libnet.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
enp_base	pointer to char	Standard C type.
ptr	pointer to char	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
0	int	Always returned.
Called By		
Function	**Where Described**	
open_cmc	Section 2.20.2.10.4.	

Table 2.20-196 Sun, MIPS or MSC map_enp Information.

2.20.2.21.4 unmap_enp

unmap_enp frees up the memory allocated to map the timer. If the compuer is not a MassComp then nothing happens but 0 is still returned. The function call is unmap_enp(np). Table 2.20-197 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
np	pointer to struct netlib_pb	/simnet/libsrc/libnetif/libnet.h
Return Values		
Return Value	**Type**	**Meaning**
0	int	Always returned.

Table 2.20-197 unmap_enp Information.

2.20.2.21.5 unmap_buffers

unmap_buffers undoes the action of Section 2.20.2.21.2 map_buffers. The function call is unmap_buffers(np). Table 2.20-198 describes the parameters used, errors returned and functions called using this function when used on a Mascomp computer. Table 2.20-199 describes the parameters used, errors returned and functions called using this function on a Sun or a MIPS.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
np	pointer to struct netlib_pb	/simnet/libsrc/libnetif/libnet.h
Return Values		
Return Value	**Type**	**Meaning**
0	int	Always returned.
Calls		
Function	**Where Described**	
detachshm	Section 2.6.5.2.1 in the Vehicles CSCI.	
Called By		
Function	**Where Described**	
open_cmc	Section 2.20.2.10.4.	

Table 2.20-198 MassComp unmap_buffers Information.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
np	pointer to struct netlib_pb	/simnet/libsrc/libnetif/libnet.h
Return Values		
Return Value	**Type**	**Meaning**
0	int	Successful.
-1	int	Unsuccessful. Call to munmap failed.
Called By		
Function	**Where Described**	
open_cmc	Section 2.20.2.10.4.	

Table 2.20-199 Sun and MIPS unmap_buffers Information.

2.20.2.21.6 unget_locks

unget_locks undoes the action of Section 2.20.2.21.1 get_locks. The function call is unget_locks(np). Table 2.20-200 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
np	pointer to struct netlib_pb	/simnet/libsrc/libnetif/libnet.h
Return Values		
Return Value	**Type**	**Meaning**
0	int	Always returned.
Called By		
Function	**Where Described**	
open_cmc	Section 2.20.2.10.4.	

Table 2.20-200 unget_locks Information.

2.20.2.22 net_data.c

/simnet/libsrc/libnetif/net_data.c

Variables		
Variable	**Type**	**Where Typedef Declared**
statistics_strings[][41]	matrix of char	Standard C type.
max_statistics_string	int	Standard C type.
BroadcastAddress	NetworkAddress	/simnet/libsrc/libnetif/network.h
FillinAddress	NetworkAddress	/simnet/libsrc/libnetif/network.h
npb_space	array NUM_NETHANDLES of struct netlib_pb	/simnet/libsrc/libnetif/libnet.h
npb	pointer to struct netlib_pb	/simnet/libsrc/libnetif/libnet.h
enpmapsize	unsigned	/usr/include/sys/sem.h
sem_unlock	struct sembuf	/usr/include/sys/sem.h
sem_lock_wait	struct sembuf	/usr/include/sys/sem.h
sem_lock_nowait	struct sembuf	/usr/include/sys/sem.h

Table 2.20-201 net_data.c Variable Information.

2.20.2.23 pktq.h

/simnet/libsrc/libnetif/pktq.h declares the structure of the elements of the queue of packets.

2.20.2.24 network.h

/simnet/libsrc/libnetif/network.h contains constants and declarations for using libnetif. This file contains the public interface to the library.

2.20.2.25 netlock.h

/simnet/libsrc/libnetif/netlock.h defines macros for locking and unlocking the incoming queue of packets.

2.20.2.26 libnet.h

/simnet/libsrc/libnetif/libnet.h is the private header file for libnetif. It declares and defines constants, macros, types, and includes for libnetif.

2.20.2.27 network.def

The network.def file specifies the configuration of the network interface. This file is read at boot time and each restart by the network daemon. Parameters are stored in the driver and obtained via library calls as needed. A change in network.def does not take effect until the network daemon is restarted. If the size of any memory segment is changed (such as the inring, outring, or sharebuf) the shared memory should be removed before restarting the network. An example network.def file is shown below:

```
#
# network definition file
#
# This file contains data that defines the configuration of
# the SIMNET network interface.
#

device enp0
address fffffffffff
inringkey 5
inringnum 32
outringkey 6
outringnum 4
sharekey 7
sharesize 4096
xmtsemkey 4
rcvsemkey 3
hostbufsize 0
multicast 64
```

The device entry specifies the network interface device in /dev.

The address entry specifies the ethernet address of the interface. Any multicast address (such as the broadcast address) directs the interface to use its ROMed address.

The inringkey entry specifies the shared memory key to use for the receive (input) ring.

The inringnum entry specifies the number of receive ring buffers.

The outringkey entry specifies the shared memory key to use for the transmit (output) ring.

The outringnum entry specifies the number of transmit ring buffers.

The sharekey entry specifies the shared memory key to use for the shared memory buffer. This buffer resides in the host and is accessible to the application and the code running on the network card.

The sharesize entry is the number of bytes allocated to the shared buffer.

The xmtsemkey entry specifies the semaphore key used for the transmit ring lock.

The rcvsemkey entry specifies the semaphore key used for the receive ring lock.

The hostbufsize entry reserves a segment of on-card memory for host use. This is a count in bytes for the nuymber of bytes to reserve.

The multicast entry specifies the number of multicast addresses that the card can support.

2.21 The MCC Libraries

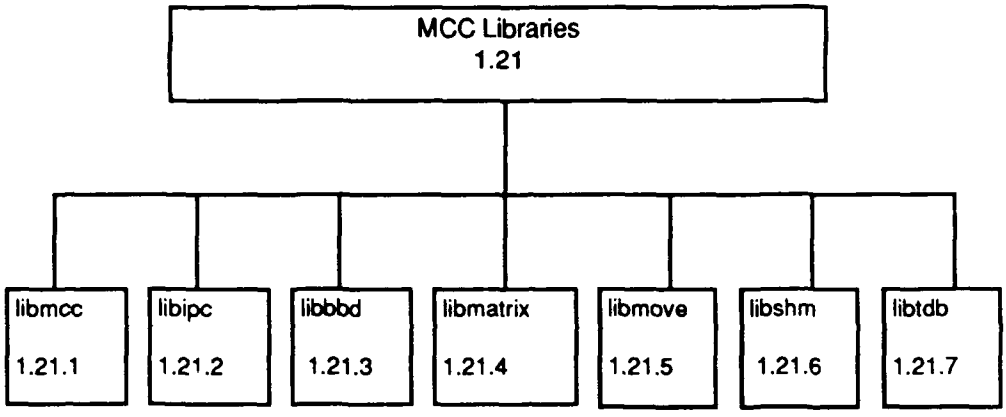


Figure 2.21-1: MCC Libraries.

2.21.1 libmcc

The MCC library (libmcc) provides access to many services shared among and throughout the MCC host processes. Abstract data types simulator tables, ccv status entries and supply truck tables directly support the MCC simulation. More basic ADT's such as fast, hash accessed queues, data type conversion routines and protocol interface routines provide flexible and efficient front ends to internal data structures.

2.21.1.1 atatp.c

/simnet/mcc/libmcc/atatp.c

atatp.c implements an AppleTalk Transaction Protocol. The AppleTalk Transaction Protocol allows MCC host processes to send and receive AppleTalk messages to and from MCC Macintosh consoles. The protocol provides a reliable end-to-end datagram service between sender and receiver. A shared table is used to maintain state information on outstanding transactions. Table 2.21-1 describes the variables used by atatp.c.

Variables		
Variable	**Type**	**Where Typedef Declared**
tid	short	Standard C type.
activeTransactions	pointer to array of Transaction	/simnet/mcc/include/bridge.h

Table 2.21-1 atatp.c Variable Information.

2.21.1.1.1 ATPSendRequest

ATPSendRequest sends an ATP Request packet. The function call is ATPSendRequest(t). The function is void so there are no errors or return values. Table 2.21-2 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
t	register pointer to Transaction	/simnet/mcc/include/bridge.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
msg	MCCMessageBuffer	/simnet/mcc/include/MCC_ipc.h
Calls		
Function	**Where Described**	
SendPacket	See Section 2.21.1.4.3.	
SetAlarm	See Section 2.21.2.4.2.	
ATPTimeout	See Section 2.21.1.1.3.	
Called By		
Function	**Where Described**	
ATPTimeout	See Section 2.21.1.1.3.	
ATPRequest	See Section 2.21.1.1.4.	

Table 2.21-2 ATPSendRequest Information.

2.21.1.1.2 ATPSendResponse

ATPSendResponse sends an ATP Response packet. The function call is ATPSendResponse(t). The function is void so there are no errors or return values. Table 2.21-3 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
t	register pointer to Transaction	/simnet/mcc/include/bridge.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
msg	MCCMessageBuffer	/simnet/mcc/include/MCC_ipc.h
Calls		
Function	**Where Described**	
SendPacket	See Section 2.21.1.4.3.	
Called By		
Function	**Where Described**	
ATPResponse	See Section 2.21.1.1.6.	
ATReceive	See Section 2.21.1.1.7.	

Table 2.21-3 ATPSendResponse Information.

2.21.1.1.3 ATPTimeout

ATPTimeout is called when a timeout expires. The function call is ATPTimeout(param, a). Table 2.21-4 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
param	register int	Standard C type.
a	Alarm	/simnet/rcs/libipc/alarm.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	int	Standard C type.
Calls		
Function	**Where Described**	
ATPSendRequest	See Section 2.21.1.1.1.	
ATFreeSocket	See Section 2.21.1.4.2.	
Called By		
Function	**Where Described**	
ATPSendRequest	See Section 2.21.1.1.1.	
ATPResponse	See Section 2.21.1.1.6.	
ATReceive	See Section 2.21.1.1.7.	

Table 2.21-4 ATPTimeout Information.

2.21.1.1.4 ATPRequest

ATPRequest sends an ATP request. It places the transaction in the active transaction table, allocates a socket, constructs the ATP Request packet, sends the packet and starts a retry timeout alarm. The function call is ATPRequest(t, priority). Table 2.21-5 describes the parameters used, errors returned and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
t	register pointer to Transaction	/simnet/mcc/include/bridge.h
priority	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
bp	register pointer to char	Standard C type.
i	int	Standard C type.
Errors		
Error Name	**Reason for Error**	
MCC_TOO_MANY_TRANS ACTIONS	The active transaction table is full.	

Calls	
Function	**Where Described**
ATAllocSocket	See Section 2.21.1.4.1.
ATPSendRequest	See Section 2.21.1.1.1.
Called By	
Function	**Where Described**
ATPTransact	See Section 2.21.1.7.3.
ATPPut	See Section 2.21.1.7.4.

Table 2.21-5 ATPRequest Information.

2.21.1.1.5 ATPGetRequest

ATPGetRequest prepares to receive a request on a socket. The function call is ATPGetRequest(t). Table 2.21-6 describes the parameters used and errors returned using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
t	register pointer to Transaction	/simnet/mcc/include/bridge.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	int	Standard C type.
Errors		
Error Name	**Reason for Error**	
MCC_TOO_MANY_TRANS ACTIONS	The active transaction table is full.	
Called By		
Function	**Where Described**	
OpenHostSocket	See Section 2.21.1.7.1.	
ConsoleReleaseArrived	See Section 2.21.1.7.7.	

Table 2.21-6 ATPGetRequest Information.

2.21.1.1.6 ATPResponse

ATPResponse sends a response to a transaction request. It constructs and sends an ATP Response packet and restarts a release timer if it is an exactly-once transaction. The function call is ATPResponse(t). Table 2.21-7 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
t	register pointer to Transaction	/simnet/mcc/include/bridge.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
bp	register pointer to char	Standard C type.

Calls	
Function	**Where Described**
ATPSendResponse	See Section 2.21.1.1.2.
CancelAlarm	See Section 2.21.2.4.3.
SetAlarm	See Section 2.21.2.4.2.
ATPTimeout	See Section 2.21.1.1.3.
Called By	
Function	**Where Described**
SendConsoleResponse	See Section 2.21.1.7.6.

Table 2.21-7 ATPResponse Information.

2.21.1.1.7 ATReceive

ATReceive processes a packet received from the AppleTalk network. This routine is called when a YUMM message bearing a DDP packet arrives from the Appletalk Network. The function call is ATReceive(type, kind, length, data, reqID). Table 2.21-8 describes the parameters used, errors returned and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
type	int	Standard C type.
length	int	Standard C type.
kind	long	Standard C type.
reqID	long	Standard C type.
data	register pointer to char	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
bp	register pointer to char	Standard C type.
t	register pointer to Transaction	/simnet/mcc/include/bridge.h
alarmsEnabled	int	Standard C type.
ccField	short	Standard C type.
seqField	short	Standard C type.
tidField	short	Standard C type.
i	short	Standard C type.
data	register pointer to char	Standard C type.
Errors		
Error Name	**Reason for Error**	
MCC_UNEXPECTED_MESSAGE	Unexpected message	
MCC_BAD_ATALK_PACKET	Invalid AppleTalk packet received	
MCC_ORPHAN_REQUEST	ATP Request discarded.	
MCC_ORPHAN_RESPONSE	ATP Response discarded.	
MCC_RESPONSE_SEQ_NUMBER	ATP Response has non-zero sequence number	
MCC_ORPHAN_RELEASE	ATP Release discarded.	

Calls	
Function	**Where Described**
TraceMessage	See Section 2.21.1.19.3.
AlarmsEnabled	See Section 2.21.2.4.4.
ATFreeBuffer	See Section 2.21.1.2.2.
ATPSendResponse	See Section 2.21.1.1.2.
CancelAlarm	See Section 2.21.2.4.3.
SetAlarm	See Section 2.21.2.4.2.
ATPTimeout	See Section 2.21.1.1.3.
ATAllocBuffer	See Section 2.21.1.2.1.
SendPacket	See Section 2.21.1.4.3.
ATFreeSocket	See Section 2.21.1.4.2.
Called By	
Function	**Where Described**
OpenHostSocket	See Section 2.21.1.7.1.

Table 2.21-8 ATReceive Information.

2.21.1.2 atbuf.c

/simnet/mcc/libmcc/atbuf.c

atbuf.c implements buffer management in shared memory for AppleTalk datagrams.

2.21.1.2.1 ATAllocBuffer

ATAllocBuffer obtains an AppleTalk buffer from the shared pool. The function call is ATAllocBuffer(). Table 2.21-8 describes the parameters used, return values and possible errors generated by using this function.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
buf	register int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
-1	int	operation failed, errno set to indicate error
buf	int	successful
Errors		
Error Name	**Reason for Error**	
MCC_NO_BUFFER	No AppleTalk buffer in the shared pool.	
Calls		
Function	**Where Described**	
Lock	See Section 2.21.2.6.3.	
Unlock	See Section 2.21.2.6.4.	

Called By	
Function	**Where Described**
ATReceive	See Section 2.21.1.1.7.
NBPLookup	See Section 2.21.1.3.1.
ATPTransact	See Section 2.21.1.7.3.
ATPPut	See Section 2.21.1.7.4.

Table 2.21-9 ATAllocBuffer Information.

2.21.1.2.2 ATFreeBuffer

ATFreeBuffer returns an AppleTalk buffer, *buf*, to the shared pool. The function call is ATFreeBuffer(buf). Table 2.21-10 describes the parameters used and possible errors generated by using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
buf	int	Standard C type.
Errors		
Error Name	**Reason for Error**	
MCC_BUFFER_FREE	Buffer not marked as in use.	
Called By		
Function	**Where Described**	
ATReceive	See Section 2.21.1.1.7.	
NBPLookup	See Section 2.21.1.3.1.	
NBPReceive	See Section 2.21.1.3.2.	
ATPTransact	See Section 2.21.1.7.3.	
ATPPutComplete	See Section 2.21.1.7.5.	
ConsoleReleaseArrived	See Section 2.21.1.7.7.	

Table 2.21-10 ATFreeBuffer Information.

2.21.1.3 atnbp.c

/simnet/mcc/libmcc/atnbp.c

atnbp.c implements a simple name lookup on the AppleTalk Network. This module is used to lookup the address of the objects on the AppleTalk Network which are associated with MCC Macintosh consoles so that those objects may communicate with the MCC host system. Table 2.21-11 describes the variables used by atnbp.c.

Variables		
Variable	**Type**	**Where Typedef Declared**
a	pointer to ATalkAddress	/simnet/mcc/include/at_addr.h
replyReceived	int	Standard C type.

Table 2.21-11 atnbp.c Variable Information.

2.21.1.3.1 NBPLookup

NBPLookup looks up an entity name and returns its address. This function implements a very limited name lookup. At most, one matching address will be found. Retry interval and count are fixed. The function call is NBPLookup(ojbStr, typeStr, zoneStr, a). Table 2.21-12 describes the parameters used, internal variables used, and functions called by using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
objStr	pointer to char	Standard C type.
typeStr	pointer to char	Standard C type.
zoneStr	pointer to char	Standard C type.
a	pointer to ATalkAddress	Standard Appletalk.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
bp	register pointer to char	Standard C type.
ddpSocket	int	Standard C type.
retry	int	Standard C type.
timeLeft	int	Standard C type.
msg	pointer to MCCMessageBuffer	/simnet/mcc/include/MCC_ipc.h
Return Values		
Return Value	**Type**	**Meaning**
replyReceived	int	Address of entity
Calls		
Function	**Where Described**	
ATAllocSocket	See Section 2.21.1.4.1.	
ATAllocBuffer	See Section 2.21.1.2.1.	
SendPacket	See Section 2.21.1.4.3.	
ATFreeSocket	See Section 2.21.1.4.2.	
ATFreeBuffer	See Section 2.21.1.2.2.	
Called By		
Function	**Where Described**	
ActivateConsole	See Section 2.21.1.7.2.	

Table 2.21-12 NBPLookup Information.

2.21.1.3.2 NBPReceive

NBPReceive handles an incomming LkUp-Reply packet. The function checks to see that the message is from the AppleTalk network and that the packet is an NBP packet. It also checks to see that it is a LkUp-Reply packet and extracts is entity address. The function call is NBPReceive(type, kind, length, data, reqID). Table 2.21-13 describes the parameters used, errors returned and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
type	int	Standard C type.
length	int	Standard C type.
kind	long	Standard C type.
reqID	long	Standard C type.
data	register pointer to char	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
bp	register pointer to char	Standard C type.
Errors		
Error Name	**Reason for Error**	
MCC_UNEXPECTED_MESSAGE	Unexpected message.	
MCC_BAD_ATALK_PACKET	Invalid AppleTalk packet received.	
Calls		
Function	**Where Described**	
TraceMessage	See Section 2.21.1.19.3.	
ATFreeBuffer	See Section 2.21.1.2.2.	

Table 2.21-13 NBPReceive Information.

2.21.1.4 atskt.c

/simnet/mcc/libmcc/atskt.c

atskt.c manages allocation and deallocation of data pathways to the AppleTalk network. Unix sockets are used to communicate with the ATRecv and ATSend processes which encapsulate the actual messages into AppleTalk packets.

2.21.1.4.1 ATAllocSocket

ATAllocSocket obtains an AppleTalk DDP socket. A YUMM socket is opened and associated with the DDP socket. The function call is ATAllocSocket(handler, priority). Table 2.21-14 describes the parameters used, errors returned and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
handler	pointer to a void function	Standard C type.
priority	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
s	register int	Standard C type.
msgSocket	YUMMSocket	/simnet/rcs/libipc/yumm.h

Return Values		
Return Value	**Type**	**Meaning**
s	int	DDP socket number
Errors		
Error Name	**Reason for Error**	
MCC_SOCKET_TABLE_FULL	Socket table full.	
Calls		
Function	**Where Described**	
OpenSocket	See Section 2.21.2.10.3.	
Lock	See Section 2.21.2.6.3.	
Unlock	See Section 2.21.2.6.4.	
Called By		
Function	**Where Described**	
ATPRequest	See Section 2.21.1.1.4.	
NBPLookup	See Section 2.21.1.3.1.	
OpenHostSocket	See Section 2.21.1.7.1.	

Table 2.21-14 ATAllocSocket Information.

2.21.1.4.2 ATFreeSocket

ATFreeSocket releases an AppleTalk DDP socket, *skt*. The function call is ATFreeSocket(skt). Table 2.21-15 describes the parameters used, errors returned and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
skt	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
cp	register pointer to char	Standard C type.
Errors		
Error Name	**Reason for Error**	
MCC_SOCKET_NOT_OPEN	Socket is not in use.	
Calls		
Function	**Where Described**	
CloseSocket	See Section 2.21.2.10.4.	
Called By		
Function	**Where Described**	
ATPTimeout	See Section 2.21.1.1.3.	
ATReceive	See Section 2.21.1.1.7.	
NBPLookup	See Section 2.21.1.3.1.	

Table 2.21-15 ATFreeSocket Information.

2.21.1.4.3 SendPacket

SendPacket sends a packet on the AppleTalk network. The function call is SendPacket(msg). Table 2.21-16 describes the parameter used and function called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
msg	pointer to MCCMessageBuffer	/simnet/mcc/include/MCC_ipc.h
Calls		
Function	**Where Described**	
SendMCCMsg1	Macro defined in /simnet/mcc/include/MCC_ipc.h.	
Called By		
Function	**Where Described**	
ATPSendRequest	See Section 2.21.1.1.1.	
ATPSendResponse	See Section 2.21.1.1.2.	
ATReceive	See Section 2.21.1.1.7.	
NBPLookup	See Section 2.21.1.3.1.	

Table 2.21-16 SendPacket Information.

2.21.1.5 azimuth.c

/simnet/mcc/libmcc/azimuth.c

azimuth.c provides conversion from 32 bit fixed point fraction of a circle to mils and vice versa.

2.21.1.5.1 mil_to_fixedpt

mil_to_fixedpt converts mils to a 32 bit fixed point fraction of a circle. The function call is mil_to_fixedpt(mils). Table 2.21-17 describes the parameters used,internal variables used, and return values generated by using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
mils	unsigned short	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
pts	unsigned long	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
pts	unsigned long	32 bit fixed point fraction of a circle

Table 2.21-17 mil_to_fixedpt Information.

2.21.1.5.2 fixedpt_to_mil

fixedpt_to_mil converts a 32 bit fixed point fraction of a circle to mils. The function call is fixedpt_to_mil(pts). Table 2.21-18 describes the parameters used, internal variables used, and return values generated by using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
pts	unsigned long	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
mils	unsigned short	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
mils	unsigned short	the value of the circle in mils
Called By		
Function	**Where Described**	
ProcessSimQueryRequest	See Section 2.21.1.25.4.	

Table 2.21-18 fixedpt_to_mil Information.

2.21.1.6 ccv.c

/simnet/mcc/libmcc/ccv.c

ccv.c handles requests from Macintoshes to place and remove computer controlled vehicles from the terrain. These routines package up the Macintosh requests and forward them on to the controlling MCC host process.

2.21.1.6.1 DisplayCCV

DisplayCCV makes a computer-controlled vehicle visible on the terrain. A message is sent to the Mother process and if the vehicle is placed, the data in the vehicle table will be changed to reflect its actual location. The function call is DisplayCCV(vehicle, type, alignment, role, company, loc, azimuth). Table 2.21-19 describes the parameters used, errors returned and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
vehicle	unsigned short	Standard C type.
guises	pointer to VehicleGuises	/simnet/common/include/protocolbasic.h
unit	pointer to OrganizationalUnit	/simnet/common/include/protocolbasic.h
marking	pointer to VehicleMarking	/simnet/common/include/protocolbasic.h
capabilities	VehicleCapabilities	/simnet/common/include/protocolbasic.h
loc	pointer to LongPt	/simnet/common/include/global/longpt.h
azimuth	short	Standard C type.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
msg	MCCMessageBuffer	/simne/mcc/include/MCC_ipc.h
respKind	long	Standard C type.
respLen	int	Standard C type.
errCode	int	Standard C type.
Calls		
Function	**Where Described**	
Transact	See Section 2.21.2.10.9.	

Table 2.21-19 DisplayCCV Information.

2.21.1.6.2 HideCCV

HideCCV makes a computer-controlled vehicle, *vehicle*, invisible. The function call is HideCCV(vehicle). Table 2.21-20 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
vehicle	unsigned short	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
msg	MCCMessageBuffer	/simnet/mcc/include/MCC_ipc.h
Calls		
Function	**Where Described**	
SendMCCMsg1	Macro defined in /simnet/mcc/include/MCC_ipc.h.	

Table 2.21-20 HideCCV Information.

2.21.1.7 console.c

/simnet/mcc/libmcc/console.c

This module forms the high level interface mechanism used by MCC host processes to communicate with the Macintosh consoles they control. It provides a transaction based communication mechanism through the other lower level transaction services. Table 2.21-21 describes the variables used by console.c.

Variables		
Variable	**Type**	**Where Typedef Declared**
hostSocket	int	Standard C type.
(requestHandler)()	function that returns int	Standard C type.
(responseHandler)()	function that returns int	Standard C type.
trans	pointer to Transaction	/simnet/mcc/include/bridge.h

Table 2.21-21 console.c Variable Information.

2.21.1.7.1 OpenHostSocket

OpenHostSocket opens a socket that the host console process will use to receive requests. First a socket is allocated through which requests may be received and then asynchronous ATP GetRequests are queued up. The function call is OpenHostSocket(reqHdler, rspHdler). Table 2.21-22 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
reqHdler	pointer to function that returns int	Standard C type.
rspHdler	pointer to function that returns int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	register int	Standard C type.
Calls		
Function	**Where Described**	
ATAllocSocket	See Section 2.21.1.4.1.	
ATReceive	See Section 2.21.1.1.7.	
ATPGetRequest	See Section 2.21.1.1.5.	

Table 2.21-22 OpenHostSocket Information.

2.21.1.7.2 ActivateConsole

ActivateConsole activates the Macintosh console, *console*. The name of the console on the AppleTalk network is looked up and then a ConsoleStartUp request is sent. The function call is ActivateConsole(console). Table 2.21-23 describes the parameters used, errors returned and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
console	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
start	ConsoleStartUpRequest	/simnet/mcc/include/console.h
Errors		
Error Name	**Reason for Error**	
MCC_MAC_NOT_RESPONDING	Mac not responding. It will retry in 30 seconds.	
MCC_STARTUP_FAILED	Start up failed. It will retry in 30 seconds.	

Calls	
Function	**Where Described**
AlarmsEnabled	See Section 2.21.2.4.4.
NBPLookup	See Section 2.21.1.3.1.
MacintoshTime	See Section 2.21.1.27.1.
ATPTransact	See Section 2.21.1.7.3.

Table 2.21-23 ActivateConsole Information.

2.21.1.7.3 ATPTransact

ATPTransact sends an ATP request and gets a response. The function call is ATPTransact(console, code, req, reqLength, rsp, rspLength, xo). Table 2.21-24 describes the parameters used, errors returned and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
console	int	Standard C type.
code	long	Standard C type.
req	pointer to char	Standard C type.
reqLength	int	Standard C type.
rsp	pointer to char	Standard C type.
rspLength	int	Standard C type.
xo	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
t	Transaction	/simnet/mcc/include/bridge.h
alarmsEnabled	int	Standard C type.
result	int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
result	int	If result = 0 , the transction was not successful. Otherwise result equals t.rspLength - atpHdsz
Errors		
Error Name	**Reason for Error**	
MCC_BAD_RESPONSE	Incorrect response from Macintosh.	
Calls		
Function	**Where Described**	
AlarmsEnabled	See Section 2.21.2.4.4	
ATAllocBuffer	See Section 2.21.1.2.1.	
ATPRequest	See Section 2.21.1.1.4.	
ATFreeBuffer	See Section 2.21.1.2.2.	

Called By	
Function	**Where Described**
ActivateConsole	See Section 2.21.1.7.2.

Table 2.21-24 ATPTransact Information.

2.21.1.7.4 ATPPut

ATPPut sends an ATP Request to a console, not waiting for the response. The function call is ATPPut(console, code, req, reqLength, xo). Table 2.21-25 describes the parameters used, errors returned and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
console	int	Standard C type.
code	long	Standard C type.
req	pointer to char	Standard C type.
reqLength	int	Standard C type.
xo	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
t	pointer to Transaction	/simnet/mcc/include/bridge.h
Calls		
Function	**Where Described**	
ATAllocBuffer	See Section 2.21.1.2.1.	
ATPPutComplete	See Section 2.21.1.7.5.	
ATPRequest	See Section 2.21.1.1.4.	

Table 2.21-25 ATPPut Information.

2.21.1.7.5 ATPPutComplete

ATPPutComplete is the completion routine for ATPPut. It is called if a response has been received or the transaction has timed out. The function call is ATPPutComplete(t). Table 2.21-26 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
t	pointer to Transaction	/simnet/mcc/include/bridge.h
Calls		
Function	**Where Described**	
ATFreeBuffer	See Section 2.21.1.2.2.	
Called By		
Function	**Where Described**	
ATPPut	See Section 2.21.1.7.4.	

Table 2.21-26 ATPPutComplete Information.

2.21.1.7.6 SendConsoleResponse

SendConsoleRequest sends an ATP Response to a console. The function call is SendConsoleResponse(t, size). Table 2.21-27 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
t	register pointer to Transaction	/simnet/mcc/include/bridge.h
size	int	Standard C type.
Calls		
Function	**Where Described**	
ATPResponse	See Section 2.21.1.1.6.	
Called By		
Function	**Where Described**	
Process GridInfoRequest	See Section 2.21.1.16.1.	
ProcessSimExistsRequest	See Section 2.21.1.25.1.	
ProcessSimInitRequest	See Section 2.21.1.25.2.	
ProcessSimQueryRequest	See Section 2.21.1.25.4.	

Table 2.21-27 SendConsoleResponse Information.

2.21.1.7.7 ConsoleReleaseArrived

ConsoleReleaseArrived is called on arrival of an ATP Release packet, *t*, signifying completion of a transaction. The function call is ConsoleReleaseArrived(t). Table 2.21-28 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
t	pointer to Transaction	/simnet/mcc/include/bridge.h
Calls		
Function	**Where Described**	
ATFreeBuffer	See Section 2.21.1.2.2.	
ATPGetRequest	See Section 2.21.1.1.5.	

Table 2.21-28 ConsoleReleaseArrived Information.

2.21.1.8 cew.c

/simnet/mcc/libmcc/cew.c

cew.c contains a routine for accessing the CEW asset parameters in shared memory.

2.21.1.8.1 GetCEWParameters

GetCEWParameters finds the CEWParameters object in shared memory corresponding to a particular asset. The function call is GetCEWParameters(kind, assetNumber). Table 2.21-29 describes the parameters used and errors returned using this function

Parameters		
Parameter	**Type**	**Where Typedef Declared**
kind	short	Standard C type.
assetNumber	short	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
pars	register pointer to CEWParameters	/simnet/mcc/include/MCC_pars.h
Return Values		
Return Value	**Type**	**Meaning**
pars + assetNumber	pointer to CEWParameters	CEWParameters object in shared memory that corresponds the the asset specified by *assetNumber*
Errors		
Error Name	**Reason for Error**	
MCC_TRUCK_BADPARM	Bad truck parameter.	

Table 2.21-29 GetCEWParameters Information.

2.21.1.9 data.c

/simnet/mcc/libmcc/data.c

This module defines global data segments shared among all processes. Table 2.21-30 shows the variables defined in data.c.

Variables		
Variable	**Type**	**Where Typedef Declared**
copyright	pointer to char	Standard C type.
authors	pointer to char	Standard C type.
mcc	pointer to MCCParameters	/simnet/mcc/include/MCC_Pars.h
vehicles	pointer to VehicleTable	/simnet/mcc/include/veh_table.h
vList	pointer to FQueue_Head_t	/simnet/mcc/libmcc/fqueue.h
vStatusList	pointer to FQueue_Head_t	/simnet/mcc/libmcc/fqueue.h
aTalk	pointer to ATalkInfo	/simnet/mcc/include/bridge.h
thisProcess	int	Standard C type.
mccSemaphores	int	Standard C type.

Table 2.21-30 data.c Information.

2.21.1.10 decode.c
/simnet/mcc/libmcc/decode.c

decode.c contains routines for dealing with rotation matrices.

2.21.1.10.1 DecodeHullToWorldMatrix

DecodeHullToWorldMatrix derives a rotation about each of three axes from the hull to world transformation matrix. The function call is DecodeHullToWorldMatrix(m, rot). Table 2.21-31 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
m[3][3]	float	Standard C type.
rot[3]	float	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
d	double	Standard C type.
Called By		
Function	**Where Described**	
ProcessSimQueryRequest	See Section 2.21.1.25.4.	

Table 2.21-31 DecodeHullToWorldMatrix Information.

2.21.1.10.2 DecodeRotationVector

DecodeRotationVector decodes a vector, *v*, composed of sine and cosine into a single angle in the range of -PI to +PI radians. The function call is DecodeRotationVector(v). Table 2.21-32 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
v	array of 2 float	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
f	float	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
f	float	Angle of vector.

Table 2.21-32 DecodeRotationVector Information.

2.21.1.11 error.c
/simnet/mcc/libmcc/error.c

The MCC takes advantage of exception handling under RTU V4.0A Unix in order to display and report erroneous or interesting behavior during the course of an exercise. The exception handler traps all exceptions generated by the MCC. Table 2.21-33 shows the external variables used by error.c.

External Variables		
Variable	**Type**	**Where Typedef Declared**
errno	extern int	Standard C.
severity	extern int	Standard C.
sys_nerr	extern int	Standard C.
sys_errlist	extern pointer to array of char	Standard C.

Table 2.21-33 error.c External Variable Information.

2.21.1.11.1 InitErrorHandling

InitErrorHandling initializes a process's handling of errors. The function call is InitErrorHandling(progname). Table 2.21-34 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
progname	pointer to char	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
str	array of 30 char	Standard C type.
Calls		
Function	**Where Described**	
MCC_errinit	See Section 2.21.1.33.1.	
Called By		
Function	**Where Described**	
InitializeProcess	See Section 2.21.1.19.1.	

Table 2.21-34 InitErrorHandling Information.

2.21.1.11.2 SendErrorReport

SendErrorReport sends an Error Report PDU, which is of interest to the NOM system. The function call is SendErrorReport(). Table 2.21-35 describes the internal variables used and functions called using this function.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
buff	array of maxNetworkPDUSize char	Standard C type.
mPDU	pointer to ManagementPDU	/simnet/common/include/protocol/p_mgmt.h
errorVariant	pointer to ErrorReportVariant	/simnet/common/include/protocol/p_mgmt.h
errorText	array of 513 char	Standard C type.
max_err_len	unsigned int	Standard C type.
mSize	unsigned short	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
severity	int	The severity of the error code.
Calls		
Function	**Where Described**	
MGMT_HDR_SIZE	macro defined in /simnet/common/include/protocol/p_mgmt.h	

Table 2.21-35 SendErrorReport Information.

2.21.1.11.3 MCC_SystemError

MCC_SystemError reports an error encountered in a system call. *functionName* is the name of the function you are currently in and *systemCallName* is the name of the system call tht caused the error. The function call is MCC_SystemError(functionName, systemCallName). Table 2.21-36 describes the parameters used and errors returned using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
functionName	pointer to char	Standard C type.
systemCallName	pointer to char	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
s	pointer to char	Standard C type.
str	array of 10 char	Standard C type.
Errors		
Error Name	**Reason for Error**	
MCC_SYSTEM_ERROR	System error.	

Called By	
Function	**Where Described**
OpenNetworkInterface	See Section 2.21.1.22.3.
SpawnProcess	See Section 2.21.1.26.1.

Table 2.21-36 MCC_SystemError Information.

2.21.1.11.4 VehicleIDToTrailer

VehicleIDToTrailer returns an ASCII string identifying a vehicle. This routine returns a string of the form <trailer><element>, given a vehicle number, *vehicleID*. If the simulator is not one of ours, "unknown" is returned. The returned string is in storage internal to this routine. The function call is VehicleIDToTrailer(vehicleID). Table 2.21-37 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
vehicleID	pointer to VehicleID	/simnet/common/include/protocol/basic.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
sim	register pointer to SimulatorStatus	/simnet/mcc/include/veh_table.h
veh	register pointer to VehicleIDStatus	/simnet/mcc/include/veh_table.h
buffer	array of 10 char	Standard C type.
unknown	pointer to char	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
buffer	pointer to char	String identifying vehicle.
unknown	pointer to char	String identifying vehicle.
Calls		
Function	**Where Described**	
FQueue_Retrieve	See Section 2.21.1.13.7.	

Table 2.21-37 VehicleIDToTrailer Information.

2.21.1.12 faad.c

/simnet/mcc/libmcc/faad.c

Forward Area Air Defense vehicles (FAAD) have specific status information relating to parameters such as fuel and armament. This module decodes and encodes FAAD status information for communication between the MCC host process and the SCC Macintosh console.

2.21.1.12.1 EncodeVehicleStatusFAAD

EncodeVehicleStatusFAAD builds an FAAD VehicleSpecificStatus structure from data supplied by the SCC or Place console. The function call is EncodeVehicleStatusFAAD(loc, vss). Table 2.21-38 describes the parameters usedby this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
loc	pointer to SimVehicleStatus	/simnet/mcc/include/sim_xact.h
vss	VehicleSpecificStatus	/simnet/common/include/protocol/status.h

Table 2.21-38 EncodeVehicleStatusFAAD Information.

2.21.1.12.2 DecodeVehicleStatusFAAD

DecodeVehicleStatusFAAD decodes an FAAD VehicleSpecificStatus structure into a form supplied to the SCC Macintosh. The function call is DecodeVehicleStatusFAAD(vss, loc). Table 2.21-39 describes the parameters usedby this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
vss	pointer to VehicleSpecificStatus	/simnet/common/include/protocol/status.h
loc	SimVehicleStatus	/simnet/mcc/include/sim_xact.h

Table 2.21-39 DecodeVehicleStatusFAAD Information.

2.21.1.12.3 TotalFuelFAAD

TotalFuelFAAD adds up the fuel aboard an FAAD. fuel gets the amount of total fuel. The function call is TotalFuelFAAD(vss, fuel). Table 2.21-40 describes the parameters usedby this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
vss	VehicleSpecificStatus	/simnet/common/include/protocol/status.h
fuel	pointer to float	Standard C type.

Table 2.21-40 TotalFuelFAAD Information.

2.21.1.13 fqueue.c

/simnet/mcc/libmcc/fqueue.c

The fqueue module provides the functionality of a fast, hash accessed queue mechanism. The queue is implemented using offsets from a known point in memory to accesses hash and data buckets. An arbitrary number of fqueue objects may exist at any time, and all are independent of each other at the ADT level. Routines are provided to insert, remove, initialize or iterate through a queue. Vehicle Appearance packets, status packets and a list

of activation locations are currently stored using fqueue structures since they must be accessed quickly by a 48 bit key.

2.21.1.13.1 Hash_Get_Entry

Hash_Get_Entry gets an entry from the queue. The function call is Hash_Get_Entry(FQueue_Head, Key, Hash_Bucket_Ptr, Hash_Func). Table 2.21-41 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
FQueue_Head	pointer to FQueue_Head_t	/simnet/mcc/libmcc/fqueue.h
Key	pointer to char	Standard C type.
Hash_Bucket_Ptr	pointer to pointer to Hash_Entry_t	/simnet/mcc/libmcc/fqueue.h
(Hash_Func)()	unsigned int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
Bucket_Index	unsigned int	Standard C type.
Queue_Entry	pointer to FQueue_t	/simnet/mcc/libmcc/fqueue.h
Hash_Bucket	pointer to Hash_Entry_t	/simnet/mcc/libmcc/fqueue.h
Hash_Table	pointer to Hash_Entry_t	/simnet/mcc/libmcc/fqueue.h
i	unsinged int	Standard C type.
Zone	pointer to char	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
Queue_Entry	pointer to FQueue_t	The specified queue entry.
NULL	pointer to FQueue_t	There is no queue entry.
Called By		
Function	**Where Described**	
FQueue_Insert	See Section 2.21.1.13.4.	
FQueue_Remove	See Section 2.21.1.13.5.	
FQueue_Retrieve	See Section 2.21.1.13.7.	

Table 2.21-41 Hash_Get_Entry Information.

2.21.1.13.2 Alloc_Free

Alloc_Free is currently a dummy function which contains no code and always returns 0. The function call is Alloc_Free(FQueue_Head). Table 2.21-42 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
FQueue_Head	pointer to FQueue_Head_t	/simnet/mcc/libmcc/fqueue.h

Return Values		
Return Value	**Type**	**Meaning**
0	int	Always returned.
Called By		
Function	**Where Described**	
FQueue_Insert	See Section 2.21.1.13.4.	

Table 2.21-42 Alloc_Free Information.

2.21.1.13.3 FQueue_Create

FQueue_Create creates an fqueue structure. The function call is FQueue_Create(FQueue_Head_Ptr, Max_Queue_Size, Entry_Size, Alloc_Func, Hash_Func, Key_Size, Table_Size, U_Param, U_Param_Size). Table 2.21-43 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
FQueue_Head_Ptr	pointer to pointer to FQueue_Head_t	/simnet/mcc/libmcc/fqueue.h
Max_Queue_Size	int	Standard C type.
Entry_Size	int	Standard C type.
(Alloc_Func)()	function that returns pointer to int	Standard C type.
(Hash_Func)()	function that returns pointer to unsigned int	Standard C type.
Key_Size	int	Standard C type.
Table_Size	int	Standard C type.
U_Param	pointer to char	Standard C type.
U_Param_Size	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
FQueue_Head	pointer to FQueue_Head_t	/simnet/mcc/libmcc/fqueue.h
New_Freee_Entry	pointer to FQueue_t	/simnet/mcc/libmcc/fqueue.h
Prev_Free_Entry	pointer to FQueue_t	/simnet/mcc/libmcc/fqueue.h
Hash_Table	pointer to Hash_Entry_t	/simnet/mcc/libmcc/fqueue.h
Mem_Block	pointer to char	Standard C type.
Zone	pointer to char	Standard C type.
Head_Mem_Block	pointer to FQueue_Head_t	/simnet/mcc/libmcc/fqueue.h
Start_Of_Zone	unsigned long	Standard C type.
i	int	Standard C type.
Total_Entry_Size	int	Standard C type.
Total_Memory	int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
Total_Memory	int	The total memory used by the fqueue structure.

Table 2.21-43 FQueue_Create Information.

2.21.1.13.4 FQueue_Insert

FQueue_Insert inserts an entry in the fqueue structure. The function call is FQueue_Insert(Fqueue_Head, Key, User_Data, Hash_Func). Table 2.21-44 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
FQueue_Head	pointer to FQueue_Head_t	/simnet/mcc/libmcc/fqueue.h
Key	pointer to char	Standard C type.
User_Data	pointer to char	Standard C type.
(Hash_Func)()	function that returns pointer to unsigned int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
Prev_Free_Entry	pointer to FQueue_t	/simnet/mcc/libmcc/fqueue.h
New_Entry	pointer to FQueue_t	/simnet/mcc/libmcc/fqueue.h
Prev_Entry	pointer to FQueue_t	/simnet/mcc/libmcc/fqueue.h
Queue_Entry	pointer to FQueue_t	/simnet/mcc/libmcc/fqueue.h
Next_Entry	pointer to FQueue_t	/simnet/mcc/libmcc/fqueue.h
Hash_Bucket	pointer to Hash_Entry_t	/simnet/mcc/libmcc/fqueue.h
Zone	pointer to char	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
0	int	Successful
-1	int	This entry is first in the list.
1	int	Unsuccessful
Calls		
Function	**Where Described**	
Hash_Get_Entry	See Section 2.21.1.13.1.	
Alloc_Free	See Section 2.21.1.13.2.	

Table 2.21-44 FQueue_Insert Information.

2.21.1.13.5 FQueue_Remove

FQueue_Remove removes an entry from the fqueue structure. The function call is FQueue_Remove(FQueue_Head, Key, User_Data, Flag, Hash_Func). Table 2.21-45 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
FQueue_Head	pointer to FQueue_Head_t	/simnet/mcc/libmcc/fqueue.h
Key	pointer to char	Standard C type.
User_Data	pointer to char	Standard C type.
Flag	unsigned char	Standard C type.
(Hash_Func)()	function that returns pointer to unsigned int	Standard C type.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
Entry	pointer to FQueue_t	/simnet/mcc/libmcc/fqueue.h
Prev_Entry	pointer to FQueue_t	/simnet/mcc/libmcc/fqueue.h
Next_Entry	pointer to FQueue_t	/simnet/mcc/libmcc/fqueue.h
Hash_Bucket	pointer to Hash_Entry_t	/simnet/mcc/libmcc/fqueue.h
Hash_Get_Entry()	pointer to FQueue_t	/simnet/mcc/libmcc/fqueue.h
Zone	pointer to char	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Unsuccessful
0	int	Successful
Calls		
Function	**Where Described**	
Hash_Get_Entry	See Section 2.21.1.13.1.	

Table 2.21-45 FQueue_Remove Information.

2.21.1.13.6 FQueue_Scan

FQueue_Scan finds the previous entry in the fqueue structure. The function call is FQueue_Scan(FQueue_Head, Prev_Entry_Ptr). Table 2.21-46 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
FQueue_Head	pointer to FQueue_Head_t	/simnet/mcc/libmcc/fqueue.h
Prev_Entry_Ptr	pointer to pointer to FQueue_t	/simnet/mcc/libmcc/fqueue.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
Prev_Entry	pointer to FQueue_t	/simnet/mcc/libmcc/fqueue.h
Zone	pointer to char	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
&Zone[Prev_Entry ->Data_Offset	pointer to char	The previous entry
NULL	pointer to char	There is no previous entry

Table 2.21-46 FQueue_Scan Information.

2.21.1.13.7 FQueue_Retrieve

FQueue_Retrieve an entry from the fqueue structure. The function call is FQueue_Retrieve(FQueue_Head, Key, Hash_Func). Table 2.21-47 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
FQueue_Head	pointer to FQueue_Head_t	/simnet/mcc/libmcc/fqueue.h
Key	pointer to char	Standard C type.
(Hash_Func)()	function that returns pointer to unsigned int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
Entry	pointer to FQueue_t	/simnet/mcc/libmcc/fqueue.h
Hash_Bucket	pointer to Hash_Entry_t	/simnet/mcc/libmcc/fqueue.h
Hash_Get_Entry()	pointer to FQueue_t	/simnet/mcc/libmcc/fqueue.h
Zone	pointer to char	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
NULL	pointer to char	The entry is NULL
&Zone[Entry->Data_Offset]	pointer to char	The specified entry.
Calls		
Function	**Where Described**	
Hash_Get_Entry	See Section 2.21.1.13.1.	
Called By		
Function	**Where Described**	
VehicleIDToTrailer	See Section 2.21.1.11.4.	
ProcessSimQueryRequest	See Section 2.21.1.25.4.	

Table 2.21-47 FQueue_Retrieve Information.

2.21.1.13.8 FQueue_Dump_Bucket

FQueue_Dump_Bucket is currently a dummy function. The function call is FQueue_Dump_Bucket(FQueue_Head, Key). Table 2.21-48 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
FQueue_head	pointer to FQueue_Head_t	/simnet/mcc/libmcc/fqueue.h
Key	pointer to char	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
Entry	pointer to FQueue_t	/simnet/mcc/libmcc/fqueue.h
Hash_Bucket	pointer to Hash_Entry_t	/simnet/mcc/libmcc/fqueue.h
Bucket_Index	int	Standard C type.
i	int	Standard C type.
j	int	Standard C type.

Table 2.21-48 FQueue_Dump_Bucket Information.

2.21.1.13.9 FQueue_Analyze

FQueue_Analyze is currently a dummy function. The function call is FQueue_Analyze(FQueue_Head). Table 2.21-49 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
FQueue_Head	pointer to FQueue_Head_t	/simnet/mcc/libmcc/fqueue.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	int	Standard C type.
Depth	array of 100 int	Standard C type.
Used	int	Standard C type.
Entries	int	Standard C type.
Hash_Bucket	pointer to Hash_Entry_t	/simnet/mcc/libmcc/fqueue.h

Table 2.21-49 FQueue_Analyze Information.

2.21.1.14 fred.c
/simnet/mcc/libmcc/fred.c

Fully Reconfigurably Evaluation Device vehicles (FRED) have specific status information relating to parameters such as fuel and armament. This module decodes and encodes FRED status information for communication between the MCC host process and the SCC Macintosh console.

2.21.1.14.1 EncodeVehicleStatusFRED

EncodeVehicleStatusFRED builds an FRED status structure in a GenericVehicleStatus from data supplied by the SCC or Place console. The function call is EncodeVehicleStatusFRED(loc, vss). Table 2.21-50 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
loc	pointer to SimVehicleStatus	/simnet/mcc/include/sim_xact.h
vss	pointer to VehicleSpecificStatus	/simnet/common/include/protocol/status.h
Called By		
Function	**Where Described**	
ProcessSimInitRequest	See Section 2.21.1.25.2.	

Table 2.21-50 EncodeVehicleStatusFRED Information.

2.21.1.14.2 DecodeVehicleStatusFRED

DecodeVehicleStatusFRED decodes an FRED VehicleSpecificStatus structure into a form supplied to the SCC Macintosh. The function call is DecodeVehicleStatusFRED(vss, loc). Table 2.21-51 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
vss	pointer to VehicleSpecificStatus	/simnet/common/include/protocol/status.h
loc	pointer to SimVehicleStatus	/simnet/mcc/include/sim_xact.h
Called By		
Function	**Where Described**	
ProcessSimQueryRequest	See Section 2.21.1.25.4.	

Table 2.21-51 DecodeVehicleStatusFRED Information.

2.21.1.14.3 fred_encode_failures

fred_encode_failures fills in the error bits of the failures. The function call is fred_encode_failures(vehicle_status). Table 2.21-52 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
vehicle_status	pointer to VehicleStatus	/simnet/common/include/protocol/status.h
Calls		
Function	**Where Described**	
veh_subsys_encode_air_failures	See Section 2.21.1.30.3.	
Called By		
Function	**Where Described**	
ProcessSimInitRequest	See Section 2.21.1.25.2.	

Table 2.21-52 fred_encode_failures Information.

2.21.1.15 generic.c
/simnet/mcc/libmcc/generic.c

generic.c contains routines for handling generic vehicles. Currently, the only information handled is fuel.

2.21.1.15.1 TotalFuelGeneric

TotalFuelGeneric adds up the fuel aboard a Generic vehicle. *fuel* gets the amount of total fuel. The function call is TotalFuelGeneric(vss, fuel). Table 2.21-53 describes the parameters usedby this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
vss	VehicleSpecificStatus	/simnet/common/include/protocol/status.h
fuel	pointer to float	Standard C type.
Called By		
Function	**Where Described**	
TotalVehicleFuel	See Section 2.21.1.28.1.	

Table 2.21-53 TotalFuelGeneric Information.

2.21.1.16 grid.c

/simnet/mcc/libmcc/grid.c

Internally, the MCC uses Cartesian coordinates to coordinate activity of its asset vehicles as they exist on the terrain. The Macintosh user interface consoles display and accept coordinate information in Universal Transverse Mercator format. grid.c provides for loading information from the terrain database onto the Macintosh consoles in the correct format.

2.21.1.16.1 ProcessGridInfoRequest

ProcessGridInfoRequest processes a request for GridInfo from a Mac. The function call is ProcessGridInfoRequest(t). Table 2.21-54 describes the parameters used, internal variables used, and return values generated by using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
t	pointer to Transaction	/simnet/mcc/include/bridge.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
req	pointer to char	Standard C type.
rsp	register pointer to GridZone	/simnet/common/libsrc/libtdb/map.h
Return Values		
Return Value	**Type**	**Meaning**
1	int	successful
0	int	operation failed, errno set to indicate error
Calls		
Function	**Where Described**	
ATPRequestData	Macro defined in /simnet/mcc/include/bridge.h.	
SendConsoleResponse	See Section 2.21.1.7.6.	

Table 2.21-54 ProcessGridInfoRequest Information.

2.21.1.17 guise.c
/simnet/mcc/libmcc/guise.c

Vehicles on the battlefield can have several discrete pairs of guises depending upon the force the simulating entity belongs to. guise.c provides the mapping from a specific vehicle type simulated or activated from an MCC to the pair of guises appropriate to that vehicle. For a more complete discussion of guise semantics, refer to the document, "The SIMNET Network and Protocols," July, 1989. TAble 2.21-55 describes the variables used by guise.c.

Variables		
Variable	**Type**	**Where Typedef Declared**
guiseTable[NUM_OBJECT_GROUPS][NUM_COUNTRIES]	matrix of ObjectType	/simnet/common/include/protocol/basic.h

Table 2.21-55 guise.c Variable Information.

2.21.1.17.1 GetGuises

GetGuises fills in the guises appropriate to the input parameters. The function call is GetGuises(battleScheme, forceID, objType, guises). Table 2.21-56 describes the parameters used, internal variables used, and return values generated by using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
battleScheme	int	Standard C type.
forceID	ForceID	/simnet/common/include/protocol/basic.h
objType	ObjectType	/simnet/common/include/protocol/basic.h
guises	pointer to VehicleGuises	/simnet/common/include/protocol/basic.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
scheme	pointer to BattleSchemes	/simnet/mcc/libmcc/guise.h
i	int	Standard C type.
j	int	Standard C type.
distCntryIndex	int	Standard C type.
otherCountryIndex	int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Invalid forceID or objType.
0	int	Successful: guises filled in.
Called By		
Function	**Where Described**	
ProcessSimInitRequest	*See Section 2.21.1.25.2.*	

Table 2.21-56 GetGuises Information.

2.21.1.18 hash.c
/simnet/mcc/libmcc/hash.c

The hashing functions used by the MCC to hash the 48 bit keys used in various fqueue objects are defined here. The hash functions all return hash table bucket indexes when given a key to hash.

2.21.1.18.1 vList_Hash

vList_Hash returns a hash table bucket index for the vehicle list fqueue. The function call is vList_Hash(key, Param). Table 2.21-57 describes the parameters used, internal variables used, and return values generated by using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
key	register pointer to unsigned short	Standard C type.
Param	register pointer to MCCParameters	/simnet/mcc/include/MCC_pars.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
index	register unsigned long	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
index	register unsigned long	Hash table bucket index

Table 2.21-57 vList_Hash Information.

2.21.1.18.2 vStatusList_Hash

vStatusList_Hash returns a hash table bucket index for the vehicle status list fqueue. The function call is vStatusList_Hash(key, Param). Table 2.21-58 describes the parameters used, internal variables used, and return values generated by using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
key	register pointer to unsigned short	Standard C type
Param	register pointer to MCCParameters	/simnet/mcc/include/MCC_pars.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
index	register unsigned long	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
index	register unsigned long	Hash table bucket index

Table 2.21-58 vStatusList_Hash Information.

2.21.1.19 init.c
/simnet/mcc/libmcc/init.c

All MCC processes initialize or attach to sections of shared memory and critical regions when they are first activated. The processes are all synchronized and initial communications are set up. init.c controls the initialization sequence. Table 2.21-59 describes the variables used by init.c.

Variables		
Variable	**Type**	**Where Typedef Declared**
progname	pointer to char	Standard C type.

Table 2.21-59 init.c Variable Information.

2.21.1.19.1 InitializeProcess

InitializeProcess initializes the standard MCC process enviroment. The function call is InitializeProcess(processNumber, processName, handler, topLevel). Table 2.21-60 describes the parameters used, internal variables used, and return values generated by using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
processNumber	int	Standard C type.
processName	pointer to char	Standard C type.
handler	pointer to function returning int	Standard C type.
topLevel	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	register short	Standard C type.
Calls		
Function	**Where Described**	
InitErrorHandling	See Section 2.21.1.11.1.	
OpenNetworkInterface	See Section 2.21.1.22.3.	
AttachLocks	See Section 2.21.2.6.1.	
AttachSharedMem	See Section 2.21.2.7.1.	
YUMMInit	See Section 2.21.2.10.1.	
YUMMProcess	See Section 2.21.2.10.2.	
OpenSocket	See Section 2.21.2.10.3.	
InitAlarms	See Section 2.21.2.4.1.	

Table 2.21-60 InitializeProcess Information.

2.21.1.19.2 TraceHandler

TraceHandler is the front-end to TraceMessage for registering with YUMM. The function call is TraceHandler(type, tid, dstSkt, rspSkt, kind, length, data). Table 2.21-61 describes the parameters used and functions used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
type	char	Standard C type.
tid	char	Standard C type.
dstSkt	YUMMSocket	/simnet/rcs/libipc/yumm.h
rspSkt	YUMMSocket	/simnet/rcs/libipc/yumm.h
kind	long	Standard C type.
length	int	Standard C type.
data	pointer to char	Standard C type.
Calls		
Function	**Where Described**	
TraceMessage	See Section 2.21.1.19.3.	

Table 2.21-61 TraceHandler Information.

2.21.1.19.3 TraceMessage

TraceMessage prints a message received by an MCC process. The function call is TraceMessage(kind, msg). Table 2.21-62 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
kind	long	Standard C type.
msg	register pointer to MCCMessageBuffer	/simnet/mcc/include/MCC_ipc.h
Called By		
Function	**Where Described**	
ATReceive	See Section 2.21.1.1.7.	
NBPReceive	See Section 2.21.1.3.2.	
TraceHandler	See Section 2.21.1.19.2.	
ProcessSimInitRequest	See Section 2.21.1.25.2.	

Table 2.21-62 TraceMessage Information.

2.21.1.19.4 InitUnit

InitUnit initializes information for a unit. The function call is InitUnit(unit forceID, type, army, corps, division, brigade, battalion, company, platoon, section, squad). Table 2.21-63 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
unit	pointer to OrganizationalUnit	/simnet/common/include/protocol/basic.h
forceID	int	Standard C type.
type	unsigned char	Standard C type.
army	int	Standard C type.
corps	int	Standard C type.
division	int	Standard C type.
brigade	int	Standard C type.
battalion	int	Standard C type.
company	int	Standard C type.
platoon	int	Standard C type.
section	int	Standard C type.
squad	int	Standard C type.
Called By		
Function	**Where Described**	
ProcessSimInitRequest	See Section 2.21.1.25.2.	

Table 2.21-63 InitUnit Information.

2.21.1.20 m1.c

/simnet/mcc/libmcc/m1.c

M1 tanks have specific status information relating to parameters such as fuel and armament. m1.c decodes and encodes M1 status information for communication between the MCC host process and the SCC Macintosh console.

2.21.1.20.1 EncodeVehicleStatusM1

EncodeVehicleStatusM1 builds an M1 VehicleSpcificStatus structure from data supplied by the SCC or Place console. The function call is EncodeVehicleStatusM1(loc, vss). Table 2.21-64 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
loc	pointer to SimVehicleStatus	/simnet/mcc/include/sim_xact.h
vss	pointer to VehicleSpecificStatus	/simnet/common/include/protocol/status.h
Called By		
Function	**Where Described**	
ProcessSimInitRequest	See Section 2.21.1.25.2.	

Table 2.21-64 EncodeVehicleStatusM1 Information.

2.21.1.20.2 DecodeVehicleStatusM1

DecodeVehicleStatusM1 decodes an M1 VehicleSpecificStatus structure into a form supplied to the SCC Macintosh. The function call is DecodeVehicleStatusM1(vss, loc). Table 2.21-65 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
vss	pointer to VehicleSpecificStatus	/simnet/common/include/protocol/status.h
loc	pointer to SimVehicleStatus	/simnet/mcc/include/sim_xact.h
Called By		
Function	**Where Described**	
ProcessSimQueryRequest	See Section 2.21.1.25.4.	

Table 2.21-65 DecodeVehicleStatusM1 Information.

2.21.1.20.3 TotalFuelM1

TotalFuelM1 adds up the fuel aboard an M1. *fuel* gets the amount of total fuel. The function call is TotalFuelM1(vss, fuel). Table 2.21-66 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
vss	pointer to VehicleSpecificStatus	/simnet/common/include/protocol/status.h
fuel	pointer to float	Standard C type.
Called By		
Function	**Where Described**	
TotalVehicleFuel	See Section 2.21.1.28.1.	

Table 2.21-66 TotalFuelM1 Information.

2.21.1.20.4 m1_encode_failures

m1_encode_failures fills in the error bits for the failures. The function call is m1_encode_failures(vehicle_status). Table 2.21-67 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
vehicle_status	pointer to VehicleStatus	/simnet/common/include/protocol/status.h
Calls		
Function	**Where Described**	
veh_subsys_encode_ground_failures	See Section 2.21.1.30.2.	

Called By	
Function	**Where Described**
ProcessSimInitRequest	See Section 2.21.1.25.2.

Table 2.21-67 m1_encode_failures Information.

2.21.1.21 m2.c

/simnet/mcc/libmcc/m2.c

M2/M3 APCs have specific status information relating to parameters such as fuel and armament. m2.c decodes and encodes M2/M3 status information for communication between the MCC host process and the SCC Macintosh console.

2.21.1.21.1 EncodeVehicleStatusM2

EncodeVehicleStatusM2 builds an M2 VehicleSpcificStatus structure from data supplied by the SCC or Place console. The function call is EncodeVehicleStatusM2(loc, vss). Table 2.21-68 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
loc	pointer to SimVehicleStatus	/simnet/mcc/include/sim_xact.h
vss	pointer to VehicleSpecificStatus	/simnet/common/include/protocol/status.h

Called By	
Function	**Where Described**
ProcessSimInitRequest	See Section 2.21.1.25.2.

Table 2.21-68 EncodeVehicleStatusFAAD Information.

2.21.1.21.2 DecodeVehicleStatusM2

DecodeVehicleStatusM2 decodes an M2 VehicleSpecificStatus structure into a form supplied to the SCC Macintosh. The function call is DecodeVehicleStatusM2(vss, loc). Table 2.21-69 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
vss	pointer to VehicleSpecificStatus	/simnet/common/include/protocol/status.h
loc	pointer to SimVehicleStatus	/simnet/mcc/include/sim_xact.h

Called By	
Function	**Where Described**
ProcessSimQueryRequest	See Section 2.21.1.25.4.

Table 2.21-69 DecodeVehicleStatusM2 Information.

2.21.1.21.3 TotalFuelM2

TotalFuelM2 adds up the fuel aboard an M2. *fuel* gets the amount of total fuel. The function call is TotalFuelM2(vss, fuel). Table 2.21-70 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
vss	pointer to VehicleSpecificStatus	/simnet/common/include/protocol/status.h
fuel	pointer to float	Standard C type.
Called By		
Function	**Where Described**	
TotalVehicleFuel	See Section 2.21.1.28.1.	

Table 2.21-70 TotalFuelM2 Information.

2.21.1.21.4 m2_encode_failures

m2_encode_failures fills in the error bits for the failures. The function call is m2_encode_failures(vehicle_status). Table 2.21-71 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
vehicle_status	pointer to VehicleStatus	/simnet/common/include/protocol/status.h
Calls		
Function	**Where Described**	
veh_subsys_encode_ground_failures	See Section 2.21.1.30.2.	
Called By		
Function	**Where Described**	
ProcessSimInitRequest	See Section 2.21.1.25.2.	

Table 2.21-71 m2_encode_failures Information.

2.21.1.22 net.c
/simnet/mcc/libmcc/net.c

Each MCC host process must obtain a data pathway to the network adapter unit upon initialization. Upon successful termination, each process relinquishes the resources used by the data path. net.c provides routines to open and close the data path. Table 2.21-72 describes the variables used by net.c.

Variables		
Variable	**Type**	**Where Typedef Declared**
g_ethernet_two_packets	int	Standard C type.
hdr	NetworkHeader	/simnet/libsrc/libnetif/network.h
networkInterface	int	Standard C type.

Table 2.21-72 net.c Variable Information.

2.21.1.22.1 SAF_NET_SND_KLUDGE

SAF_NET_SND_KLUDGE allows the MCC to send out either Ethernet 2.0 or Ethernet 802.3 packets. The function call is SAF_NET_SND_KLUDGE(h, to, buf, len, flags). Table 2.21-73 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h	int	Standard C type.
to	pointer to NetworkAddress	/simnet/libsrc/libnetif/network.h
buf	pointer to char	Standard C type.
len	int	Standard C type.
flags	int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
net_snd(h, to, buf, len, flags)	int	send Ethernet 2.0 packets.
net_send(h, &hdr, buf, len, flags)	int	send Ethernet 802.3 packets.
Calls		
Function	**Where Described**	
net_snd	See Section 2.20.2.19.2.	
net_send	See Section 2.20.2.15.1.	

Table 2.21-73 SAF_NET_SND_KLUDGE Information.

2.21.1.22.2 set_g_ethernet_two_packets

set_g_ethernet_two_packets set *g_ethernet_two_packets* to TRUE. The function call is set_g_ethernet_two_packets(). This function calls only standard functions and is not called by any other function in this library, so no table is included for it.

2.21.1.22.3 OpenNetworkInterface

OpenNetworkInterface obtains access to the simulation Ethernet. The function call is OpenNetworkInterface(topLevel). Table 2.21-74 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
topLevel	int	Standard C type.
Calls		
Function	**Where Described**	
AssocOpen	See Section 2.20.1.5.1.	
MCC_SystemError	See Section 2.21.1.11.3.	
AssocError	See Section 2.20.1.10.1.	
AssocAttach	See Section 2.20.1.5.2.	
Called By		
Function	**Where Described**	
InitializeProcess	See Section 2.21.1.19.1.	

Table 2.21-74 OpenNetworkInterface Information.

2.21.1.22.4 CloseNetworkInterface

CloseNetworkInterface releases access to the simulation Ethernet. The function call is CloseNetworkInterface(). Table 2.21-75 describes the functions called using this function.

Calls	
Function	**Where Described**
net_close	See Section 2.20.2.3.1.

Table 2.21-75 CloseNetworkInterface Information.

2.21.1.23 radio.c

/simnet/mcc/libmcc/radio.c

Radio power control to MCC controlled TOC radios is controlled through this module. Radios are currently not controlled by fielded MCC systems.

2.21.1.23.1 InitRadios

InitRadios initializes all radios to "off". The function call is InitRadios(). Table 2.21-76 describes the internal variables used and functions called using this function.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
bitmask	array of 9 char	Standard C type.
temp	register short	Standard C type.
i	register short	Standard C type.

Calls	
Function	**Where Described**
bbd_init	See Section 2.1.5.1.11.1 in the Vehicles CSCI.

Table 2.21-76 InitRadios Information.

2.21.1.23.2 RemoveRadios

RemoveRadios uninitializes the radio power control. The function call is RemoveRadios(). Table 2.21-77 describes the functions called using this function.

Calls	
Function	**Where Described**
bbd_uninit	See Section 2.1.5.1.16.1 in the Vehicles CSCI.

Table 2.21-77 RemoveRadios Information.

2.21.1.23.3 TurnRadioOn

TurnRadioOn turns on *radio*. The function call is TurnRadioOn(radio). Table 2.21-78 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
radio	short	Standard C type.
Calls		
Function	**Where Described**	
bbd_bit_out	See Section 2.1.5.1.4.1 in the Vehicles CSCI.	

Table 2.21-78 TurnRadioOn Information.

2.21.1.23.4 TurnRadioOff

TurnRadioOff turns off *radio*. The function call is TurnRadioOff(radio). Table 2.21-79 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
radio	short	Standard C type.
Calls		
Function	**Where Described**	
bbd_bit_out	See Section 2.1.5.1.4.1 in the Vehicles CSCI.	

Table 2.21-79 TurnRadioOff Information.

2.21.1.24 segintr.c
/simnet/mcc/libmcc/segintr.c

segintr.c contains functions which determine if a line segment crosses or touches inappropriately (i.e. in the middle) any of an array of previous line segments.

2.21.1.24.1 seg_intrsct

seg_intrsct computes information on the intersection of a line segment from *pt_0p* to*pt_1p* with the (connected) line segments *pt_arr[0]* to*pt_arr[num_pts - 1]*. It requires that *num_pts* is greater than 2. For computing points of intersection, *paramsp->size_out_ptsp* should be at least 2 (or 1 more than you expect), in case of segment overlap. The function call is seg_intrsct(paramsp). Table 2.21-80 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
paramsp	pointer to segintparams	/simnet/mcc/libmcc/segintr.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
use_x	register int	Standard C type.
x_diff	register int	Standard C type.
y_diff	register int	Standard C type.
i	register int	Standard C type.
slope	register double	Standard C type.
code	register int	Standard C type.
abort_code	register int	Standard C type.
num_code	register int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
code	int	The segment intersection code describing the type of intersection.
DUMMY	int	Not a valid segment.
Calls		
Function	**Where Described**	
check_x_ints	See Section 2.21.1.24.2.	
reverse_param	See Section 2.21.1.24.4.	
check_y_ints	See Section 2.21.1.24.3.	

Table 2.21-80 seg_intrsct Information.

2.21.1.24.2 check_x_ints

check_x_ints checks for an intersection of the finite line segments: *ptp* to *pt_0p* (with slope (dy/dx) assumed between -1 and 1) and *pt_1p* to*pt_2p*. The function call is check_x_ints(pt0p, pt1p, slope, pt2p, pt3p, outcode, outp, outp2, numoutp). Table 2.21-81 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
pt0p	pointer to Long_Point	/simnet/mcc/libmcc/segintr.h
pt1p	pointer to Long_Point	/simnet/mcc/libmcc/segintr.h
slope	double	Standard C type.
pt2p	pointer to Long_Point	/simnet/mcc/libmcc/segintr.h
pt3p	pointer to Long_Point	/simnet/mcc/libmcc/segintr.h
outcode	int	Standard C type.
outp	pointer to coded_lpoint	/simnet/mcc/libmcc/segintr.h
out2p	pointer to coded_lpoint	/simnet/mcc/libmcc/segintr.h
numoutp	pointer to int	Standard C type.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
p_leftp	pointer to Long_Poin t	/simnet/mcc/libmcc/segintr.h
p_rightp	pointer to Long_Point	/simnet/mcc/libmcc/segintr.h
p1_leftp	pointer to Long_Point	/simnet/mcc/libmcc/segintr.h
p1_rightp	pointer to Long_Point	/simnet/mcc/libmcc/segintr.h
dist1	double	Standard C type.
dist2	double	Standard C type.
delta_x	double	Standard C type.
slope2	double	Standard C type.

Return Values		
Return Value	**Type**	**Meaning**
ON_ONE_SIDE	int	The segments miss; separated by x or y.
BOTH_ON_LINE_ABUTTING	int	The given segment is colinear to another and the vertices touch.
BOTH_ON_LINE_OVERLAP PING	int	The segments are colinear and the vertices do not touch
ON_LINE_TO_SIDE	int	A vertex is on an extended line, but not on the segment; there is no intersection.
ON_LINE_ABUTTING	int	One pair of vertices touch
ON_LINE_MID_TOUCH	int	One vertex touches the middle of the given semgnet
MID_CROSS	int	The line segments cross in the middle.
ABUTTING	int	The given segment is abutted by the other.

Calls	
Function	**Where Described**
POINT_TO_LINE	Macro defined in /simnet/mcc/libmcc/segintr.h.

Called By	
Function	**Where Described**
seg_intrsct	See Section 2.21.1.24.1.

Table 2.21-81 check_x_ints Information.

2.21.1.24.3 check_y_ints

check_y_ints checks for an intersection of the finite line segments: *ptp* to *pt_0p* (with slope (dy/dx) assumed between -1 and 1) and *pt_1p* to*pt_2p*. The function call is check_y_ints(pt0p, pt1p, slope, pt2p, pt3p, outcode, outp, outp2, numoutp). Table 2.21-82 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
pt0p	pointer to Long_Point	/simnet/mcc/libmcc/segintr.h
pt1p	pointer to Long_Point	/simnet/mcc/libmcc/segintr.h
slope	double	Standard C type.
pt2p	pointer to Long_Point	/simnet/mcc/libmcc/segintr.h
pt3p	pointer to Long_Point	/simnet/mcc/libmcc/segintr.h
outcode	int	Standard C type.
outp	pointer to coded_lpoint	/simnet/mcc/libmcc/segintr.h
out2p	pointer to coded_lpoint	/simnet/mcc/libmcc/segintr.h
numoutp	pointer to int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
p_leftp	pointer to Long_Poin t	/simnet/mcc/libmcc/segintr.h
p_rightp	pointer to Long_Point	/simnet/mcc/libmcc/segintr.h
p1_leftp	pointer to Long_Point	/simnet/mcc/libmcc/segintr.h
p1_rightp	pointer to Long_Point	/simnet/mcc/libmcc/segintr.h
dist1	double	Standard C type.
dist2	double	Standard C type.
delta_x	double	Standard C type.
slope2	double	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
ON_ONE_SIDE	int	The segments miss; separated by x or y.
BOTH_ON_LINE_ABUTTING	int	The given segment is colinear to another and the vertices touch.
BOTH_ON_LINE_OVERLAP PING	int	The segments are colinear and the vertices do not touch
ON_LINE_TO_SIDE	int	A vertex is on an extended line, but not on the segment; there is no intersection.
ON_LINE_ABUTTING	int	One pair of vertices touch
ON_LINE_MID_TOUCH	int	One vertex touches the middle of the given semgnet
MID_CROSS	int	The line segments cross in the middle.
ABUTTING	int	The given segment is abutted by the other.

Calls	
Function	**Where Described**
POINT_TO_LINE	Macro defined in /simnet/mcc/libmcc/segintr.h.
Called By	
Function	**Where Described**
seg_intrsct	See Section 2.21.1.24.1.

Table 2.21-82 check_y_ints Information.

2.21.1.24.4 reverse_param

reverse_param reverses the x and y coordinates for all member points in the *paramp* parameter. Member points include *pt0p*, *pt1p*, the points in the *pt_arr* array, and the points in the *outptsp* array. The function call is reverse_param(paramp). Table 2.21-83 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
paramp	pointer to segintparams	/simnet/mcc/libmcc/segintr.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	register int	Standard C type.
Calls		
Function	**Where Described**	
REVERSE_COORDS	Macro defined in /simnet/mcc/libmcc/segintr.h.	
Called By		
Function	**Where Described**	
seg_intrsct	See Section 2.21.1.24.1.	

Table 2.21-83 reverse_param Information.

2.21.1.25 sim.c

/simnet/mcc/libmcc/sim.c

sim.c contains code used by the SCC and Placement console processes for placing simulators. It provides the following functions:

- information about simulators available for the exercise;
- requests to place or reconstitute a simulator's vehicle;
- handles requests for the status of a vehicle to be reconstituted; and
- notifies consoles when a requested activation has failed.

2.21.1.25.1 .ProcessSimExistsRequest

ProcessSimExistsRequest processes a request for simulator data from a Mac. The function call is ProcessSimExistsRequest(t). Table 2.21-84 describes the parameters used, internal variables used, and functions called by using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
t	pointer to Transaction	/simnet/mcc/include/bridge.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
req	pointer to SimExistsRequest	/simnet/mcc/include/sim_xact.h
rsp	register pointer to SimExistsResponse	/simnet/mcc/include/sim_xact.h
sim	register pointer to SimulatorStatus	/simnet/mcc/include/veh_table.h
Calls		
Function	**Where Described**	
ATPRequestData	Macro defined in /simnet/mcc/include/bridge.h.	
SendConsoleResponse	See Section 2.21.1.7.6.	
GetOldStyleCompany	See Section 2.21.1.25.6.	

Table 2.21-84 ProcessSimExistsRequest Information.

2.21.1.25.2 ProcessSimInitRequest

ProcessSimInitRequest processes a request to place a vehicle from a Mac. *vehiclePtr* is set to the number of the vehicle activated. The function call is ProcessSimInitRequest(t, vehiclePtr). Table 2.2-85 describes the parameters used, internal variables used, and functions called by using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
t	pointer to Transaction	/simnet/mcc/include/bridge.h
vehiclePtr	pointer to int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
req	register pointer to SimInitRequest	/simnet/mcc/include/sim_xact.h
rsp	pointer to SimInitResponse	/simnet/mcc/include/sim_xact.h
vehicle	unsigned short	Standard C type.
msg	MCCMessageBuffer	/simnet/mcc/include/MCC_ipc.h
rspKind	long	Standard C type.
errCode	int	Standard C type.
rspLen	int	Standard C type.
sim	register pointer to SimulatorStatus	/simnet/mcc/include/veh_table.h

Return Values		
Return Value	**Type**	**Meaning**
1	int	Successful
0	int	Unsuccessful
Calls		
Function	**Where Described**	
ATPRequestData	Macro defined in /simnet/mcc/include/bridge.h.	
InitUnit	See Section 2.21.1.19.4.	
GetGuises	See Section 2.21.1.17.1.	
EncodeVehicleStatusM1	See Section 2.21.1.20.1.	
m1_encode_failures	See Section 2.21.1.20.4.	
EncodeVehicleStatusM2	See Section 2.21.1.21.1.	
m2_encode_failures	See Section 2.21.1.21.4.	
EncodeVehicleStatusFRED	See Section 2.21.1.14.1.	
fred_encode_failures	See Section 2.21.1.14.3.	
Transact	See Section 2.21.2.10.9.	
TraceMessage	See Section 2.21.1.19.3.	
SendConsoleResponse	See Section 2.21.1.7.6.	

Table 2.21-85 ProcessSimInitRequest Information.

2.21.1.25.3 SimGetSimType

SimGetSimType translates a vehicle object code into a SIMNET simulator type. The function call is SimGetSimType(vehicle_object_type). Table 2.21-86 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
vehicle_object_type	ObjectType	/simnet/common/include/protocol/basic.h
Return Values		
Return Value	**Type**	**Meaning**
simulator_SIMNET_M1	SimulatorType	The simulator is an M1.
simulator_SIMNET_M2	SimulatorType	The simulator is an M2.
simulator_SIMNET_FRED	SimulatorType	The simulator is a FRED.
0	SimulatorType	Unknown simulator type.
Called By		
Function	**Where Described**	
ProcessSimQueryRequest	See Section 2.21.1.25.4.	

Table 2.21-86 SimGetSimType Information.

2.21.1.25.4 ProcessSimQueryRequest

ProcessSimQueryRequest responds to a request from a Macintosh for the current status of a vehicle simulator. The function call is ProcessSimQueryRequest(t). Table 2.21-87 describes the parameters used, errors returned and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
t	pointer to Transaction	/simnet/mcc/include/bridge.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
req	pointer to SimQueryRequest	/simnet/mcc/include/sim_xact.h
rsp	pointer to SimQueryResponse	/simnet/mcc/include/sim_xact.h
sim	pointer to SimulatorStatus	/simnet/mcc/include/veh_table.h
veh	pointer to VehicleIDStatus	/simnet/mcc/include/veh_table.h
veh_status	pointer to VehicleStatus	/simnet/common/include/protocol/status.h
turretAzimuth	Angle	/simnet/common/include/protocol/basic.h
vehicleType	unsigned short	Standard C type.
rollPitchYaw	array of 3 float	Standard C type.
appear	pointer to VehicleAppearanceVariant	/simnet/common/include/protocol/p_sim.h
Errors		
Error Name	**Reason for Error**	
MCC_UNKNOWN_VEHICLE_TYPE	Unknown vehicle type.	
Calls		
Function	**Where Described**	
ATPRequestData	Macro defined in /simnet/mcc/include/bridge.h.	
SimGetSimType	See Section 2.21.1.25.3.	
SendConsoleResponse	See Section 2.21.1.7.6.	
FQueue_Retrieve	See Section 2.21.1.13.7.	
DecodeHullToWorldMatrix	See Section 2.21.1.10.1.	
rad_to_mil	Macro defined in /simnet/common/include/global/sim_macros.h.	
fixedpt_to_mil	See Section 2.21.1.5.2.	
Lock	See Section 2.21.2.6.3.	
DecodeVehicleStatusM1	See Section 2.21.1.20.2.	
DecodeVehicleStatusM2	See Section 2.21.1.21.2.	
DecodeVehicleStatusFRED	See Section 2.21.1.14.2.	
Unlock	See Section 2.21.2.6.4.	

Table 2.21-87 ProcessSimQueryRequest Information.

2.21.1.25.5 GetCompanyFromOrg

GetCompanyFromOrg returns the company, given the organizational unit. The function call is GetCompanyFromOrg(organization). Table 2.21-88 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
organization	pointer to OrganizationalUnit	/simnet/common/include protocol/basic.h
Return Values		
Return Value	**Type**	**Meaning**
organization ->hierarchy [companyOrgLevel].unitNumber	char	the company unit.
Called By		
Function	**Where Described**	
GetOldStyleCompany	See Section 2.21.1.25.6.	

Table 2.21-88 GetCompanyFromOrg Information.

2.21.1.25.6 GetOldStyleCompany

GetOldStyleCompany returns a company constant in the old protocol. The function call is GetOldStyleCompany(unit). Table 2.21-89 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
unit	pointer to OrganizationalUnit	/simnet/common/include/global/basic.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
company	char	Standard C type.
GetCompanyFromOrg()	extern char	See Section 2.21.1.25.5.
Return Values		
Return Value	**Type**	**Meaning**
company	char	
assignedScoutPlt	char	
assignedBattalion	char	
assignedTACP	char	
assignedNothing	char	
Calls		
Function	**Where Described**	
GetCompanyFromOrg	See Section 2.21.1.25.5.	

Called By	
Function	**Where Described**
ProcessSimExistsRequest	See Section 2.21.1.25.1.

Table 2.21-89 GetOldStyleCompany Information.

2.21.1.25.7 PutCompanyInSim

PutCompanyInSim assigns a company to a simulator. The function call is PutCompanyInSim(sim, company). Table 2.21-90 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
sim	pointer to SimulatorStatus	/simnet/mcc/include/veh_table.h
company	unsigned char	Standard C type.

Table 2.21-90 PutCompanyInSim Information.

2.21.1.26 spawn.c

/simnet/mcc/libmcc/spawn.c

spawn.c provides for mechanisms to control and coordinate activity and initializing communications between parent and child processes in the host MCC and for terminating communications upon exit. Table 2.21-91 describes the variables used by spawn.c.

Variables		
Variable	**Type**	**Where Typedef Declared**
errno	extern int	Standard C type.

Table 2.21-91 spawn.c Variable Information.

2.21.1.26.1 SpawnProcess

SpawnProcess forks an MCC process, *p*. The function call is SpawnProcess(p). Table 2.21-92 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
p	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
pid	int	Standard C type.
mypid	int	Standard C type.
target_dir	array of 30 char	Standard C type.
Calls		
Function	**Where Described**	
MCC_SystemError	See Section 2.21.1.11.3.	

Table 2.21-92 SpawnProcess Information.

2.21.1.26.2 LookupProcessIdentifier

LookupProcessIdentifier gets a process number from a process ID, *pid*. The function call is LookupProcessIdentifier(pid). Table 2.21-93 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
pid	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
p	int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
p	int	Process number.
0	int	Unsuccessful.

Table 2.21-93 LookupProcessIdentifier Information.

2.21.1.26.3 LookupProcessNumber

LookupProcessNumber gets a process ID from a process number, *p*. The function call is LookupProcessNumber(p). Table 2.21-94 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
p	int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
mcc->pInfo[p].pid	int	Process ID

Table 2.21-94 LookupProcessNumber Information.

2.21.1.26.4 SignalChild

SignalChild sends a signal ato a child process, *p*. The function call is SignalChild(p, s). Table 2.21-95 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
p	int	Standard C type.
s	int	Standard C type.

Table 2.21-95 SignalChild Information.

2.21.1.27 time.c

/simnet/mcc/libmcc/time.c

time.c contains a routine for converting between Macintosh and Masscomp Unix style time. Table 2.21-96 describes the variables used by time.c.

Variables		
Variable	**Type**	**Where Typedef Declared**
localtime()	extern pointer to struc tm	Standard C Library.
time()	extern long	Standard C Library.
tzset()	extern void	Standard C Library.
timezone	extern long	Standard C Library.
daylight	extern int	Standard C Library.

Table 2.21-96 time.c Variable Information.

2.21.1.27.1 MacintoshTime

MacintoshTime returns current local time in Macintosh format. UNIX_TO_MAC is a constant which accounts for the difference between Macintosh time zero (Jan. 1, 1904) and Unix time zero (Jan. 1, 1970). The function call is MacintoshTime(). Table 2.21-97 describes the internal variables used and return values generated by using this function.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
ltime	pointer to struct tm	Standard C library
gtime	long	Standard C type.
dst	long	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
gtime + UNIX_TO_MAC - timezone + dst	long	Current local time in Macintosh format.
Calls		
Function	**Where Described**	
tzset	Standard C Library.	
time	Standard C Library.	
localtime	Standard C Library.	
Called By		
Function	**Where Described**	
ActivateConsole	See Section 2.21.1.7.2.	

Table 2.21-97 MacintoshTime Information.

2.21.1.28 total.c
/simnet/mcc/libmcc/total.c

Tallying module responsible for computing the total amount of fuel on board a vehicle given a vehicle status packet. This module dispatches the appropriate tallying routine for simulator specific fuel loads.

2.21.1.28.1 TotalVehicleFuel

TotalVehicleFuel computes the total fuel on board a vehicle, given a VehicleStatus structure from that vehicle. The function call is TotalVehicleFuel(status, fuel). Table 2.21-98 describes the parameters used and fuctions called by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
status	register pointer to VehicleStatus	/simnet/common/include/protocol/status.h
fuel	pointer to float	Standard C type.
Calls		
Function	**Where Described**	
TotalFuelM1	See Section 2.21.1.20.3.	
TotalFuelM2	See Section 2.21.1.21.3.	
TotalFuelGeneric	See Section 2.21.1.15.1.	

Table 2.21-98 TotalVehicleFuel Information.

2.21.1.29 truck.c
/simnet/mcc/libmcc/truck.c

Status information about computer controlled vehicles acting as supply trucks are stored as a table in shared memory. truck.c acts as the interface between MCC host services manipulating truck resources and the table of truck parameters.

2.21.1.29.1 GetTruckParameters

GetTruckParameters finds the TruckParameters object in shared memory corresponding to a particular service truck. The function call is GetTruckParameters(capabilities, organization, truckNumber). Table 2.21-99 describes the parameters used, errors returned and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
capabilities	pointer to VehicleCapabilities	/simnet/common/include/protocol/basic.h
organization	pointer to OrganizationalUnit	/simnet/common/include/protocol/basic.h
truckNumber	short	Standard C type.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
pars	register pointer to TruckParameters	/simnet/mcc/include/MCC_pars.h
Return Values		
Return Value	**Type**	**Meaning**
pars+truckNumber	pointer to TruckParameters	the TruckParameters object in shared memory corresponding to the truck specified by *truckNumber*.
Errors		
Error Name	**Reason for Error**	
MCC_TRUCK_BADPARM	Invalid truck parameter.	

Table 2.21-99 GetTruckParameters Information.

2.21.1.30 vehsubsys.c

/simnet/mcc/libmcc/vehsubsys.c

vehsubsys.c contains functions which fill in the vehicle subsystem failure bits.

2.21.1.30.1 fill_bytes

fill_bytes fills in *data* with *fill_value*. The function call is fill_bytes(data, fill_value, size). Table 2.21-100 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
data	pointer to char	Standard C type.
fill_value	char	Standard C type.
size	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	int	Standard C type.
Called By		
Function	**Where Described**	
veh_subsys_encode_ground_failures	See Section 2.21.1.30.2.	
veh_subsys_encode_air_failures	See Section 2.21.1.30.3.	

Table 2.21-100 fill_bytes Information.

2.21.1.30.2 veh_subsys_encode_ground_failures

veh_subsys_encode_ground_failures fills in the ground vehicle subsystems failure bits with the appropriate codes. The function call is veh_subsys_encode_ground_failures(failures). Table 2.21-101 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
failures	pointer to VehicleSubsystems	/simnet/common/include/protocol/status.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
ground	pointer to GroundVehicleSubsystems	/simnet/common/include/protocol/status.h
Calls		
Function	**Where Described**	
fill_bytes	See 2.21.1.30.1.	
Called By		
Function	**Where Described**	
m1_encode_failures	See Section 2.21.1.20.4.	
m2_encode_failures	See Section 2.21.1.21.4.	

Table 2.21-101 veh_subsys_encode_ground_failures Information.

2.21.1.30.3 veh_subsys_encode_air_failures

veh_subsys_encode_air_failures fills in the air vehicle subsystems failure bits with the appropriate codes. The function call is veh_subsys_encode_air_failures(failures). Table 2.21-102 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
failures	pointer to VehicleSubsystems	/simnet/common/include/protocol/status.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
air	pointer to AirVehicleSubsystems	/simnet/common/include/protocol/status.h
Calls		
Function	**Where Described**	
fill_bytes	See 2.21.1.30.1.	
Called By		
Function	**Where Described**	
fred_encode_failures	See Section 2.21.1.14.3.	

Table 2.21-102 veh_subsys_encode_air_failures Information.

2.21.1.31 widebdry.c
/simnet/mcc/libmcc/widebdry.c

widebdry.c contains a function which allocates an array of boundary vertices from a sequence of points.

2.21.1.31.1 wide_bdry

wide_bdry allocates an array of boundary vertices (*bdry_vertsp*), given a sequence of points *center_verts[0]* to *center_verts[num_cverts - 1]* created by the caller. The caller must later free the array. The number of valid boundary points in the array is *num_bvertsp*. If the *center_verts* form a path that crosses or touches itself, the path from the boundary vertices will likely cross itself. The function call is wide_bdry(center_verts, num_cverts, width, bdry_vertsp, num_bvertsp). Table 2.21-103 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
center_verts	pointer to Long_Point	/simnet/mcc/libmcc/segintr.h
num_cverts	int	Standard C type.
width	int	Standard C type.
bdry_vertsp	pointer to pointer to Long_Point	/simnet/mcc/libmcc/segintr.h
num_bvertsp	pointer to int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
left_normal	Double_Point	
left_side	Long_Point	/simnet/mcc/libmcc/segintr.h
new_vector	Long_Point	/simnet/mcc/libmcc/segintr.h
test_arr	array of 2 Long_Point	/simnet/mcc/libmcc/segintr.h
outpt_arr	array of 2 coded_lpoint	/simnet/mcc/libmcc/segintr.h
seg_size	register double	Standard C type.
dot_prod	register double	Standard C type.
params	segintparams	/simnet/mcc/libmcc/segintr.h
left_vert	int	Standard C type.
num_bdry_verts	int	Standard C type.
right_vert	int	Standard C type.
i	register int	Standard C type.
gap	int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
1	int	
0	int	
Calls		
Function	**Where Described**	
SET_CODE		
ROUND_LONG		

Table 2.21-103 wide_bdry Information.

2.21.1.32 MCC_errs.err
/simnet/mcc/libmcc/MCC_errs.err

This file defines the severity, name, and textual messages associated with abnormal conditions detected by the MCC host system. The exceptions defined herein are dispatched to the standard exception handler which is then free to attempt to deal with the exception in an appropriate manner, if any.

2.21.1.33 MCC_errs.c
/simnet/mcc/libmcc/MCC_errs.c

MCC_errs.c is an error definition file generated by errcom. Table 2.21-104 describes the variables used by MCC_errs.c.

Variables		
Variable	**Type**	**Where Typedef Declared**
errfacs	extern array of MAXFACILITIES ERRFACS	
MCC_errlist	pointer to array of char	Standard C type.
MCC_errinfo	array of ERRINFO	

Table 2.21-104 MCC_errs.c Variable Information

2.21.1.33.1 MCC_errinit

MCC_errinit is the error initialization routine for the MCC facility. The function call is MCC_errinit(). Table 2.21-105 describes the functions which call MCC_errinit.

Called By	
Function	**Where Described**
InitErrorHandling	See Section 2.21.1.11.1.
mccerrinit_	See Section 2.21.1.33.2.

Table 2.21-105 MCC_errinit Information.

2.21.1.33.2 mccerrinit_

mccerrinit_ is the fortran entry point to the error initialization routine for the MCC facility. The function call is mccerrinit-(). Table 2.21-106 describes the functions called by this function.

Calls	
Function	**Where Described**
MCC_errinit	See Section 2.21.1.33.1.

Table 2.21-106 mccerrinit_ Information.

2.21.2 libipc

Interprocess Communication Package

Libipc provides several discrete services related to interprocess communication. The services provided are:

- Queued, time based alarm events (alarm.*);
- Shared memory management (memory.*);
- A semaphore based mutual exclusion mechanism (lock.*); and
- Transaction and datagram interprocess message services (yumm*.*).

Libipc is used to coordinate activity across the distributed process of the MCC system.

2.21.2.1 ipc.h

/simnet/rcs/libipc/ipc.h

ipc.h defines the functions available in the IPC library. Table 2.21-107 describes the variables used by ipc.h.

Variables		
Variable	**Type**	**Where Typedef Declared**
ipc_mem_errno	extern int	Standard C type.

Table 2.21-107 ipc.h Variable Information

2.21.2.2 keys.h

/simnet/rcs/libipc/keys.h

keys.h defines keys that uniquely identify shared memory segments, groups of semaphores and message queues all used for inter-process communication.

2.21.2.3 alarm.h

/simnet/rcs/libipc/alarm.h

alarm.h is the header file associated with alarm.c (see Section 2.21.2.4).

2.21.2.4 alarm.c

/simnet/rcs/libipc/alarm.c

alarm.c contains routines which provide an alarm service similar to that of the alarm(2) system call, but differing in that multiple alarms can be pending. Table 2.21-108 describes the variables used by alarm.c.

Variables		
Variable	**Type**	**Where Typedef Declared**
alarmSchedule	Alarm	/simnet/rcs/libipc/alarm.h
alarmsEnabled	int	Standard C type.

Table 2.21-108 alarm.c Variable Information.

2.21.2.4.1 InitAlarms

InitAlarms initializes the alarm package. The function call is InitAlarms(). Table 2.21-109 describes the functions called using this function.

Calls	
Function	**Where Described**
IPC_errinit	See Section 2.21.2.13.1.

Table 2.21-109 InitAlarms Information.

2.21.2.4.2 SetAlarm

SetAlarm makes a new entry in the schedule of upcoming alarms. *delay* is how long to wait before alarm is due; *fnc* is a pointer to function call when alarm is due; *param* is the parameter to give the function when it is called. The function call is SetAlarm(delay, fnc, param). Table 2.21-110 describes the parameters used and errors returned using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
delay	int	Standard C type.
(fnc)()	pointer to int	Standard C type.
param	long	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
prevAlarm	unsigned	Standard C type.
a	register Alarm	/simnet/rcs/libipc/alarm.h
p1	register Alarm	/simnet/rcs/libipc/alarm.h
p2	register Alarm	/simnet/rcs/libipc/alarm.h
oldPriority	int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
0	Alarm	Unsuccessful.
a	Alarm	New alarm.
Errors		
Error Name	**Reason for Error**	
IPC_ALARM_MEMORY	Not enough memory.	

Table 2.21-110 SetAlarm Information.

2.21.2.4.3 CancelAlarm

CancelAlarm removes an entry, *a*, from the schedule of upcoming alarms. The function call is CancelAlarm(a). Table 2.21-111 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
a	register Alarm	/simnet/rcs/libipc/alarm.h

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
prevAlarm	unsigned	Standard C type.
p1	register Alarm	/simnet/rcs/libipc/alarm.h
p2	register Alarm	/simnet/rcs/libipc/alarm.h
oldPriority	int	Standard C type.

Table 2.21-111 CancelAlarm Information.

2.21.2.4.4 AlarmsEnabled

AlarmsEnabled enables or disables the alarm interrupts. The value returned is non-zero if and only if the alarms were enabled at the time of the call. The function call is AlarmsEnabled(on). Table 2.21-112 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
on	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
oldPriority	int	Standard C type.
oldAlarmsEnabled	int	Standard C type.
now	long	Standard C type.
a	register Alarm	/simnet/rcs/libipc/alarm.h
Return Values		
Return Value	**Type**	**Meaning**
alarmsEnabled	int	No change in state.
oldAlarmsEnabled	int	Changed state of alarms.

Table 2.21-112 AlarmsEnabled Information.

2.21.2.4.5 DispatchAlarm

DispatchAlarm handles a SIGALARM signal by calling the next scheduled alarm client function. The function call is DispatchAlarm(sig). Table 2.21-113 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
sig	int	Standard C type
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
now	long	Standard C type.
a	register Alarm	/simnet/rcs/libipc/alarm.h
oldPriority	int	Standard C type

Table 2.21-113 DispatchAlarm Information.

2.21.2.5 error.c
/simnet/rcs/libipc/error.c

error.c contains a routine which reports an error encountered in a system call. Table 2.21-114 describes the variables used by error.c.

Variables		
Variable	**Type**	**Where Typedef Declared**
errno	extern int	Standard C type.
sys_nerr	extern int	Standard C type.
sys_errlist	extern pointer to array of char	Standard C type.

Table 2.21-114 error.c Variable Information.

2.21.2.5.1 IPC_SystemError

IPC_SystemError reports an error encountered in a system call. *functionName* is the name of the function you were in when the call was made and *systemCallName* is the name of the system call made. The function call is IPC_SystemError(functionName, systemCallName). Table 2.21-115 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
functionName	pointer to char	Standard C type.
systemCallName	pointer to char	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
s	pointer to char	Standard C type.
str	array of 10 char	Standard C type.
Called By		
Function	**Where Described**	
AttachLocks	See Section 2.21.2.6.1.	
ResetLocks	See Section 2.21.2.6.2.	
Lock	See Section 2.21.2.6.3.	
Unlock	See Section 2.21.2.6.4.	
RemoveLocks	See Section 2.21.2.6.5.	
AttachMsgQueue	See Section 2.21.2.8.1.	
RemoveMsgQueue	See Section 2.21.2.8.2.	
YUMMCleanUp	See Section 2.21.2.10.5.	
Transact	See Section 2.21.2.10.9.	
RcvMsg	See Section 2.21.2.12.1.	
SendData	See Section 2.21.1.12.3.	

Table 2.21-115 IPC_SystemError Information.

2.21.2.6 lock.c
/simnet/rcs/libipc/lock.c

lock.c contains routines which implement exclusive-access locks using semaphores. Table 2.21-116 describes the variables used by lock.c.

Variables		
Variable	**Type**	**Where Typedef Declared**
errno	extern int	Standard C type.

Table 2.21-116 lock.c Vriable Information.

2.21.2.6.1 AttachLocks

AttachLocks gets a group of semaphores to use as locks. If the group has not yet been created, it is created at this time. *key* is the key for the semaphores, *n* is the number of semaphores and *first* is returned TRUE if the semaphore group was created. This function is not good for checking if a key is already in use. No error will be returned, just the semaphore id. The function call is AttachLocks(key, n, first). Table 2.21-117 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
key	int	Standard C type.
n	int	Standard C type.
first	pointer to int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
semid	int	Standard C type.
firstFlag	int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
semid	int	Semaphore id.
Calls		
Function	**Where Described**	
IPC_errinit	See Section 2.21.2.13.1.	
IPC_SystemError	See Section 2.21.2.5.1.	
ResetLocks	See Section 2.21.2.6.2.	
Called By		
Function	**Where Described**	
YUMMInit	See Section 2.21.2.10.1.	

Table 2.21-117 AttachLocks Information.

2.21.2.6.2 ResetLocks

ResetLocks initializes a group of semaphores representing locks. *semid* is semaphore id and *n* is the number of semaphores in the group. The function call is ResetLocks(semid, n). Table 2.21-118 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
semid	int	Standard C type.
n	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	int	Standard C type.
Calls		
Function	**Where Described**	
IPC_SystemError	See Section 2.21.2.5.1.	
Called By		
Function	**Where Described**	
AttachLocks	See Section 2.21.2.6.1.	

Table 2.21-118 ResetLocks Information.

2.21.2.6.3 Lock

Lock locks a shared resource. The function call is Lock(semid, n). Table 2.21-119 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
semid	int	Standard C type.
n	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
sop	struct sembuf	/usr/include/sys/sem.h
Calls		
Function	**Where Described**	
IPC_SystemError	See Section 2.21.2.5.1.	
Called By		
Function	**Where Described**	
GetInternProcID	See Section 2.21.2.12.7.	
CloseSocket	See Section 2.21.2.10.4.	
OpenSocket	See Section 2.21.2.10.3.	

Table 2.21-119 Lock Information.

2.21.2.6.4 Unlock

Unlock unlocks a shared resource. The function call is Unlock(semid, n). Table 2.21-120 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
semid	int	Standard C type.
n	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
sop	struct sembuf	
Calls		
Function	**Where Described**	
IPC_SystemError	See Section 2.21.2.5.1.	
Called By		
Function	**Where Described**	
GetInternProcID	See Section 2.21.2.12.7.	
CloseSocket	See Section 2.21.2.10.4.	
OpenSocket	See Section 2.21.2.10.3.	

Table 2.21-120 Unlock Information.

2.21.2.6.5 RemoveLocks

RemoveLocks removes a group of semaphores, described by *key*, from the system. The function call is RemoveLocks(key). Table 2.21-121 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
key	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
semid	int	Standard C type.
Calls		
Function	**Where Described**	
IPC_SystemError	See Section 2.21.2.5.1.	

Table 2.21-121 RemoveLocks Information.

2.21.2.7 memory.c
/simnet/rcs/libipc/memory.c

memory.c contains routines for managing the memory shared among processes. Table 2.21-122 describes the variables used by memory.c.

Variables		
Variable	**Type**	**Where Typedef Declared**
ipc_mem_errno	int	Standard C type.
errno	extern int	Standard C type.

Table 2.21-122 memory.c Variable Information.

2.21.2.7.1 AttachSharedMem

AttachSharedMem attaches to memory shared among processes. When it is getting the memory, it figures out if this process is the first one to get it and returns so that some initialization can take place. *key* is the key for the shared memory and *size* is the size of the memory. *first* returns TRUE if this is the first process to create shared memory. This process cannot be used to determine if *key* is already in use. It can only return the shared memory id. The function call is AttachSharedMem(key, size, first). Table 2.21-123 describes the parameters used, errors returned and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
key	int	Standard C type.
size	int	Standard C type.
first	pointer to int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
shmat	extern array of char	Standard C type.
sbrk	extern array of char	Standard C type.
id	int	Standard C type.
charp	pointer to char	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
0	pointer to char	Unsuccessful. Check ipc_mem_errno for error code.
charp	pointer to char	Pointer to shared memory.
Errors		
Error Name	**Reason for Error**	
IPC_SHMEM_SBRK	Couldn't grow process.	
IPC_SHMEM_BRK	Couldn't grow process.	
IPC_SHMEM_SHMGET	Couldn't get region.	
IPC_SHMEM_SHMAT	Couldn't attach to region.	
Calls		
Function	**Where Described**	
IPC_errinit	See Section 2.21.2.13.1.	

Called By	
Function	**Where Described**
YUMMInit	See Section 2.21.2.10.1.

Table 2.21-123 AttachSharedMem Information.

2.21.2.7.2 RemoveSharedMem

RemoveSharedMem removes a block of shared memory, described by *key* and *address*, from the system. The function call is RemoveSharedMem(key, address). Table 2.21-124 describes the parameters used and errors returned by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
key	int	Standard C type.
address	pointer to char	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
shmid	int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Unsuccessful.
Errors		
Error Name	**Reason for Error**	
IPC_SHMEM_RM_SHMGET		
IPC_SHMEM_SHMDT	Couldn't delete the region.	
IPC_SHMEM_SHMCTL	Couldn't remove the region.	

Table 2.21-124 RemoveSharedMem Information.

2.21.2.8 msgqueue.c

/simnet/rcs/libipc/msgqueue.c

msgqueue.c contains routines to create, attach and remove a message queue. Table 2.21-125 describes the variables used by msgqueue.c.

Variables		
Variable	**Type**	**Where Typedef Declared**
errno	extern int	Standard C type.

Table 2.21-125 msgqueue.c Variable Information.

2.21.2.8.1 AttachMsgQueue

AttachMsgQueue is used to create a message queue and its ID or return the ID of an already existing message queue. *key* is the message queue key and *first* returns 1 if the message queue was created. This routine cannot be used to determine if *key* is already in use. It will just return the ID of the corresponding message queue. The function call is AttachMsgQueue(key, first). Table 2.21-126 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
key	int	Standard C type.
first	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
queueID	int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
queueID	int	ID of message queue.
Calls		
Function	**Where Described**	
IPC_errinit	See Section 2.21.2.13.1.	
IPC_SystemError	See Section 2.21.2.5.1.	
Called By		
Function	**Where Described**	
YUMMInit	See Section 2.21.2.10.1.	

Table 2.21-126 AttachMsgQueue Information.

2.21.2.8.2 RemoveMsgQueue

RemoveMsgQueue removes the message queue associated with *key*. The function call is RemoveMsgQueue(key). Table 2.21-127 describes the parameters used, errors returned and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
key	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
msgQid	int	Standard C type.
Calls		
Function	**Where Described**	
IPC_SystemError	See Section 2.21.2.5.1.	

Table 2.21-127 RemoveMsgQueue Information.

2.21.2.9 yumm.h
/simnet/rcs/libipc/yumm.h

yumm.h is the external include file for YUMM. Programs using YUMM procedures need to include this file. Table 2.21-128 describes the variables used by this file.

Variables		
Variable	**Type**	**Where Typedef Declared**
YUMMSocket	unsigned char	Standard C type.

Table 2.21-128 yumm.h Variable Information.

2.21.2.10 yummextern.c
/simnet/rcs/libipc/yummextern.c

yummextern.c contains all the external routines for YUMM (Yet Unother Message-passing Mechanism). Table 2.21-129 describes the variables used by yummextern.c.

Variables		
Variable	**Type**	**Where Typedef Declared**
errno	extern int	Standard C type.

Table 2.21-129 yummextern.c Variable Information.

2.21.2.10.1 YUMMInit

YUMMInit does the initialization for the system, if necessary, and for this process. It allocates and attaches a chunk of shared memory for the socket table, allocates and attaches the semaphore table and allocates and attaches a system-wide message queue. It also generates the internal process id. It returns nothing if everything is ok, or the system error if there is a problem. Warning: This procedure is NOT foolproof. If a non-YUMM related process already allocated shared memory, a semaphore group or a message queue with the same key as YUMM uses, havoc could result. Check keys.h. The function call is YUMMInit(). Table 2.21-130 describes the internal variable used and functions called using this function.

Internal Variables	
Variable	**Type**	**Where Typedef Declared**
first	int	Standard C type.
Calls	
Function	**Where Described**
AttachSharedMem	See Section 2.21.2.7.1.
InitShMem	See Section 2.21.2.12.5.
AttachLocks	See Section 2.21.2.6.1.
AttachMsgQueue	See Section 2.21.2.8.1.
IPC_errinit	See Section 2.21.2.13.1.

Table 2.21-130 YUMMInit Information.

2.21.2.10.2 YUMMProcess

YUMMProcess initializes process specific information such as procid and the entry in the socket table for this process' synchronous requests. It also allocates the semaphore for this process. *procname* is the name of the process used to register a socket. The function call is YUMMProcess(procname). Table 2.21-131 describes the parameters used, errors returned and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
procname	pointer to char	Standard C type.
Calls		
Function	**Where Described**	
CleanupDeadSockets	See Section 2.21.2.12.6.	
GetInternProcID	See Section 2.21.2.12.7.	
SetUpSynchReq	See Section 2.21.2.12.8.	

Table 2.21-131 YUMMProcess Information.

2.21.2.10.3 OpenSocket

OpenSocket dynamically allocates a socket and fills in the entry using information which is passed in and information which it can figure out. It uses a semaphore to avoid resource conflicts. *usrASTHdlr* is a pointer to the user's AST handler, *sktName* is the name to associate with the socket (it can be no more than 31 characters), *astprio* is the priority the _sendast should use for this AST, *sktNum* is the socket allocated, or NULL if we don't care to get it and *usrTrace* is a pointer to the debugging routine called when AST comes. The function call is OpenSocket(usrASTHdlr, sktName, astprio, sktNum, usrTrace). Table 2.21-132 describes the parameters used, errors returned and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
(usrASTHdlr)()	function that returns pointer to void	Standard C type.
sktName	pointer to char	Standard C type.
astprio	int	Standard C type.
sktNum	pointer to YUMMSocket	/simnet/rcs/libipc/yumm.h
(usrTrace)()	function that returns pointer to void	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	short	Standard C type.
temp	YUMMSocket	/simnet/rcs/libipc/yumm.h
Return Values		
Return Value	**Type**	**Meaning**
0	short	Socket number allocated.
-1	short	Socket number not allocated, error code returned in errno.

Errors	
Error Name	**Reason for Error**
IPC_DUP_SOCKET_NAME	The current socket name is a duplicate of one already in the table.
IPC_SOCKET_TABLE_FULL	The socket table is full.
Calls	
Function	**Where Described**
Lock	See Section 2.21.2.6.3.
LkUpSocket	See Section 2.21.2.10.10.
Unlock	See Section 2.21.2.6.4.

Table 2.21-132 OpenSocket Information.

2.21.2.10.4 CloseSocket

CloseSocket closes the designated socket according to *always*. If *always* is TRUE it closes the socket no matter what, otherwise, it checks to see if there are outstanding requests on that socket before closing and it will not close if there are any. *sktnumber* is the socket name of the entry to close. The function call is CloseSocket(sktnumber, always). Table 2.21-133 describes the parameters used, errors returned and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
sktnumber	int	Standard C type.
always	char	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
0	short	Successful.
-1	short	Requests outstanding, check errno for error code.
Errors		
Error Name	**Reason for Error**	
IPC_MORE_REQS	More requests for this socket are outstanding.	
Calls		
Function	**Where Described**	
Lock	See Section 2.21.2.6.3.	
Unlock	See Section 2.21.2.6.4.	
Called By		
Function	**Where Described**	
CleanupDeadSockets	See Section 2.21.2.12.6.	
CleanupSockets	See Section 2.21.2.12.4.	

Table 2.21-133 CloseSocket Information.

2.21.2.10.5 YUMMCleanUp

YUMMCleanUp is used to bring down the system and its resources. It deletes and detaches the message queue, detaches semaphores and detaches shared memory. The function call is YUMMCleanUp(). Table 2.21-134 describes the internal variables used and functions called using this function.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
shmbuf	struct shmid_ds	/use/include/sys/shm.h
shmid	int	Standard C type.
CleanupSockets	extern void	See Section 2.21.2.12.4.
Calls		
Function	**Where Described**	
CleanupSockets	See Section 2.21.2.12.4.	
IPC_SystemError	See Section 2.21.2.5.1.	

Table 2.21-134 YUMMCleanUp Information.

2.21.2.10.6 SendMsg

SendMsg is used for one-way message passing. It sends a message via an AST according to the information found at *dstSkt*. *dstSkt* is the socket to deliver AST to now, *msgKind* is the kind of request (user-defined), *msgLen* is the length, in bytes, of *msgData* and *msgData* is a pointer to the message buffer. The function call is SendMsg(dstSkt, msgKind, msgLen, msgData). Table 2.21-135 describes the parameters used, errors returned and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
dstSkt	YUMMSocket	/simnet/rcs/libipc/yumm.h
msgKind	long	Standard C type.
msgLen	int	Standard C type.
msgData	pointer to char	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
messtype	messageType	/simnet/rcs/libipc/yummint.h
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Unsuccessful. Check error code returned in errno.
SendData(messtype, msgKing, msgLen, msgData)	int	Successful. Should return 0.
Errors		
Error Name	**Reason for Error**	
IPC_BAD_SOCKET	Socket not opened by process.	

Calls	
Function	**Where Described**
SendData	See Section 2.21.2.12.3.

Table 2.21-135 SendMsg Information.

2.21.2.10.7 SendReq

SendReq is used for two-way message passing. It sends a request via an AST according to the information found at *dstSkt*. It expects a response to be sent to *rspSkt*. It assumes that *dstSkt* and *rspSkt* are valid. A bit of info indentifying the request may be passed in *msgInfo* if the user so desires. *dstSkt* is the socket to deliver AST to now, *msgKind* is the kind of request (user-defined), *msgLen* is the length, in bytes, of *msgData* and *msgData* is a pointer to the message buffer. The function call is SendReq(dstSkt, rspSkt, msgInfo, msgKind, msgLen, msgData). Table 2.21-136 describes the parameters used, errors returned and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
dstSkt	YUMMSocket	/simnet/rcs/libipc/yumm.h
rspSkt	YUMMSocket	/simnet/rcs/libipc/yumm.h
msgInfo	long	Standard C type.
msgKind	long	Standard C type.
msgLen	int	Standard C type.
msgData	pointer to char	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
messtype	messageType	/simnet/rcs/libipc/yummint.h
tid	short	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Unsuccessful. Check error code returned in errno.
SendData(messtype, msgKing, msgLen, msgData)	int	Successful. Should return 0.
Errors		
Error Name	**Reason for Error**	
IPC_BAD_SOCKET	Socket not opened by process.	
IPC_REQ_FULL	Request table full.	
Calls		
Function	**Where Described**	
GrabReqEntry	See Section 2.21.2.12.2.	
SendData	See Section 2.21.2.12.3.	

Table 2.21-136 SendReq Information.

2.21.2.10.8 SendRsp

SendRsp is used for two-way mesage passing. It sends a response via an AST according to the information found through *reqID*. *reqID* is the number identifying the request this response is for, *msgKind* is the kind of request (user-defined), *msgLen* is the length, in bytes, of *msgData* and *msgData* is a pointer to the message buffer. The function call is SendRsp(). Table 2.21-137 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
reqID	messageType	/simnet/rcs/libipc/yummint.h
msgKind	long	Standard C type.
msgLen	int	Standard C type.
msgData	pointer to char	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
SendData(reqID.s, msgKind, msgLen, msgData)	int	0 if successful. -1 if unsuccessful, error code returned in errno.
Calls		
Function	**Where Described**	
SendData	See Section 2.21.2.12.3.	

Table 2.21-137 SendRsp Information.

2.21.2.10.9 Transact

Transact is used for two-way message passing. It sends out a request and does not return until a response is received or *timeout* has expired. It uses astpause() to accomplish this. *dstSkt* is the socket to deliver AST to now, *msgKind* is the kind of request (user-defined), *msgLen* is the length, in bytes, of *msgData* and *msgData* is a pointer to the message buffer. *rcvKind* returns the kind of response received. *rcvLen* is initialized to the max length of *rcvData* and returns the length of the received data. *rcvData* is a pointer to the data in the response. *timeout* is the time, in milliseconds, to wait for a response. The function call is Transact(dstSkt, msgKind, msgLen, msgData, rcvKind, rcvLen, rcvData, timeout). Table 2.21-138 describes the parameters used, errors returned and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
dstSkt	YUMMSocket	/simnet/rcs/libipc/yumm.h
msgKind	long	Standard C type.
msgLen	int	Standard C type.
msgData	pointer to char	Standard C type.
rcvKind	pointer to long	Standard C type.
rcvLen	pointer to int	Standard C type.
rcvData	pointer to char	Standard C type.
timeout	int	Standard C type.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
messType	messageType	/simnet/rcs/libipc/yummint.h
tid	short	Standard C type.
msg	messageBuf	/simnet/rcs/libipc/yummint.h
savepri	int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
0	int	Successful.
-1	int	Unsuccessful. Error code returned in errno.
Errors		
Error Name	**Reason for Error**	
IPC_BAD_SOCKET	Socket not explicitly opened by process.	
IPC_REQ_FULL	Request table full.	
IPC_TIMED_OUT	Timeout exceeded before response arrived.	
IPC_NO_MESSAGE	No response found in message queue.	
IPC_WRITE_OVERRUN	More data in response than can be put in the rcvData buffer.	
Calls		
Function	**Where Described**	
GrabReqEntry	See Section 2.21.2.12.2.	
SendData	See Section 2.21.2.12.3.	
IPC_SystemError	See Section 2.21.2.5.1.	

Table 2.21-138 Transact Information.

2.21.2.10.10 LkUpSocket

LkUpSocket looks for *name* in the socket table and returns the socket number, *sktnum*, associated with it. It does a linear search. If *name* is NULL, the function returns FALSE. The function call is LkUpSocket(name, sktnum). Table 2.21-139 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
name	pointer to char	Standard C type.
sktnum	pointer to YUMMSocket	/simnet/rcs/libipc/yumm.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	short	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
FALSE	short	Name not found in table.
TRUE	short	Name found in table.

Called By	
Function	**Where Described**
OpenSocket	See Section 2.21.2.10.3.

Table 2.21-139 LkUpSocket Information.

2.21.2.10.11 GetSocketName

GetSocketName returns the name associated with a given socket. Note that if the socket is closed, the name is meaningless. The function call is GetSocketName(sktnum, name). Table 2.21-140 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
sktnum	unsigned char	Standard C type.
name	pointer to char	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
FALSE	short	Socket number is invalid, 'name' is meaningless.
TRUE	short	Socket number is valid, socket name is returned in 'name'

Table 2.21-140 GetSocketName Information.

2.21.2.10.12 ReportYUMMStatus

ReportYUMMStatus returns the status of the system and of each process registered in the statTable. The status consists of listing the various messge pasing counts, the ipc mechanisms involved with YUMM and information about the socket table. The function call is ReportYUMMStatus(). Table 2.21-141 describes the internal variables used by this function.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
oldpri	int	Standard C type.
i	int	Standard C type.

Table 2.21-141 ReportYUMMStatus Information.

2.21.2.11 yumminit.h

/simnet/rcs/libipc/yumminit.h

yumminit.h is the internal include file for the YUMM sources. Table 2.21-142 describes the variables used by yumminit.h.

Variables		
Variable	**Type**	**Where Typedef Declared**
YUMMData	extern YUMMGlobals	/simnet/rcs/libipc/yumminit.h

Table 2.21-142 yumminit.h Variable Information.

2.21.2.12 yummintern.c
/simnet/rcs/libipc/yummintern.c

yummintern.c contains all the internal routines to Yet Unother Message-passing Mechanism (YUMM). Table 2.21-143 describes the variables used by yummintern.c

Variables		
Variable	**Type**	**Where Typedef Declared**
errno	extern int	Standard C type.
_asthandler()	extern int	Standard C type.
YUMMData	YUMMGlobals	/simnet/rcs/libipc/yumminit.h

Table 2.21-143 yummintern.c Variable Information.

2.21.2.12.1 RcvMsg

RcvMsg is a generic AST handler. All ASTs come to RcvMsg first before the user's trace procedure and the user's AST handler are called. It grabs parameters off the message queue using param and passes them to the user's AST handler and trace program, if provided. *oldpc* is the old program counter and *oldpr* is the old priority. The function call is RcvMsg(param, oldpc, oldpri). Table 2.21-144 describes the parameters used, errors returned and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
param	long	Standard C type.
oldpc	int	Standard C type.
oldpr	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
msgbuf	messageBuf	/simnet/rcs/libipc/yummint.h
msgTp	messageType	/simnet/rcs/libipc/yummint.h
msgInfo	long	Standard C type.
msgLen	long	Standard C type.
sockEnt	sktEntry	/simnet/rcs/libipc/yummint.h
Errors		
Error Name	**Reason for Error**	
IPC_NO_MESSAGE	Expected message was not found on the mesage queue.	
IPC_UNKNOWN_TYPE	An unknown type of messge was received.	
Calls		
Function	**Where Described**	
IPC_SystemError	See Section 2.21.2.5.1.	

Table 2.21-144 RcvMsg Information.

2.21.2.12.2 GrabReqEntry

GrabReqEntry enters information for this request into the next free entry in rqstTable. This table is used to hold user-defined information for asynchronous requests and to hold a flag which signifies if a response has come in yet for a synchronous requests. It returns the index of the new table entry if successful and -1 otherwise. The function call is GrabReqEntry(msgInfo, srcSkt). Table 2.21-145 describes the parameters used and errors returned using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
msgInfo	long	Standard C type.
srcSkt	YUMMSocket	/simnet/rcs/libipc/yumm.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	short	Standard C type.
oldpri	int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
i	short	Index of new table entry.
-1	short	Unsuccessful.
Errors		
Error Name	**Reason for Error**	
IPC_REQ_FULL	Request table is already full.	
Called By		
Function	**Where Described**	
Transact	See Section 2.21.2.10.9.	
SendReq	See Section 2.21.2.10.7.	

Table 2.21-145 GrabReqEntry Information.

2.21.2.12.3 SendData

SendData does the actual sending of data by putting parameters on the message queue and then sending an AST to the receiving process so it knows to read off of the message queue. It will not send to a closed socket. It truncates messages which are bigger than MAXMSGSIZE. *msgType* describes the message as either one-way, a synchronous or asynchronous request or an asynchronous respones. *msgKind* describes the kind of message, *msgLen* is the length of the data buffer and *msgData* is a pointer to an MCCMessageBuffer. The function call is SendData(msgType, msgKind, msgLen, msgData). Table 2.21-146 describes the parameters used, errors returned and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
msgType	messageType	/simnet/rcs/libipc/yummint.h
msgKind	long	Standard C type.
msgLen	int	Standard C type.
msgData	pointer to char	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
msgBuf	messageBuf	/simnet/rcs/libipc/yummint.h
sockEnt	sktEntry	/simnet/rcs/libipc/yummint.h
Return Values		
Return Value	**Type**	**Meaning**
0	int	Successful.
-1	int	Unsuccessful.
Errors		
Error Name	**Reason for Error**	
IPC_TOO_LONG	The message was too long.	
IPC_PROCESS_MISSING	The process has disappeared.	
IPC_CLOSED_SOCKET	The socket is closed.	
Calls		
Function	**Where Described**	
IPC_SystemError	See Section 2.21.2.5.1.	
Called By		
Function	**Where Described**	
Transact	See Section 2.21.2.10.9.	
SendRsp	See Section 2.21.2.10.8.	
SendReq	See Section 2.21.2.10.7.	
SendMsg	See Section 2.21.2.10.6.	

Table 2.21-146 SendData Information.

2.21.2.12.4 CleanupSockets

CleanupSockets is called from YUMMCleanup to close any outstanding sockets. The function call is CleanupSockets(). Table 2.21-147 describes the internal variables used and functions called using this function.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	int	Standard C type.
mypid	int	Standard C type.
Calls		
Function	**Where Described**	
CloseSocket	See Section 2.21.2.10.4.	

Called By	
Function	**Where Described**
YUMMCleanUp	See Section 2.21.2.10.5.

Table 2.21-147 CleanupSockets Information.

2.21.2.12.5 InitShMem

InitShMem initializes shared memory. This entails initializing all the YUMMHdlr entries in the socket table and initializing the process id used to get an internal process id for each process. The function call is InitShMem(). Table 2.21-148 describes the internal variables used by this function.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	short	Standard C type.
Called By		
Function	**Where Described**	
YUMMInit	See Section 2.21.2.10.1.	

Table 2.21-148 InitShMem Information.

2.21.2.12.6 CleanupDeadSockets

CleanupDeadSockets goes through the socket table and closes those for which the corresponding process is no longer alive. The function call is CleanupDeadSockets(). Table 2.21-149 describes the internal variables used and functions called using this function.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	int	Standard C type.
Calls		
Function	**Where Described**	
CloseSocket	See Section 2.21.2.10.4.	
Called By		
Function	**Where Described**	
YUMMProcess	See Section 2.21.2.10.2.	

Table 2.21-149 CleanupDeadSockets Information.

2.21.2.12.7 GetInternProcID

GetInternProcID assigns a unique one byte number to procid. procid will be between 1 and NUMSYNCH-1, inclusive. It assigns procid by incrementing the byte counter in shared memory. The function call is GetInternProcID(). Table 2.21-150 describes the functions called using this function.

Calls	
Function	**Where Described**
Lock	See Section 2.21.2.6.3.
Unlock	See Section 2.21.2.6.4.
Called By	
Function	**Where Described**
YUMMProcess	See Section 2.21.2.10.2.

Table 2.21-150 GetInternProcID Information.

2.21.2.12.8 SetUpSynchReq

SetUpSynchReq sets up a socket table entry to be used when a synchronous request is made. It sets up the entry indexed by the internal process id. The function call is SetUpSynchReq(procname). Table 2.21-151 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
procname	pointer to char	Standard C type.
Called By		
Function	**Where Described**	
YUMMProcess	See Section 2.21.2.10.2.	

Table 2.21-151 SetUpSynchReq Information.

2.21.2.13 libipcerrs.c

/simnet/rcs/libipc/libipcerrs.c

libipcerrs.c is the error definition file generated by errcom.

2.21.2.13.1 IPC_errinit

IPC_errinit is the error initialization routine for the IPC-Library facility. The function call is IPC_errinit(). Table 2.21-152 describes the function.

Called By	
Function	**Where Described**
InitAlarms	See Section 2.21.2.4.1.
AttachLocks	See Section 2.21.2.6.1.
AttachSharedMem	See Section 2.21.2.7.1.
AttachMsgQueue	See Section 2.21.2.8.1.
YUMMInit	See Section 2.21.2.10.1.
ipcerrinit	See Section 2.21.2.13.2.

Table 2.21-152 IPC_errinit Information.

2.21.2.13.2 ipcerrinit

ipcerrinit is the Fortran entry point for the IPC-Library facility. The function call is ipcerrinit(). Table 2.21-153 describes the functions called using this function.

Calls	
Function	**Where Described**
IPC_errinit	See Section 2.21.2.13.1.

Table 2.21-153 ipcerrinit Information.

2.21.2.14 libipcerrs.err

/simnet/rcs/libipc/libipcerrs.err

libipcerrs.err defines the errors detected and reported by the IPC library.

2.21.3 libbbd

Reference the Vehicles CSCI documentation Section 2.1.5.1.

2.21.4 libmatrix

Reference the Vehicles CSCI documentation Section 2.6.2.

2.21.5 libmove

Reference the Vehicles CSCI documentation Section 2.6.6.

2.21.6 libshm

Reference the Vehicles CSCI documentation Section 2.6.5.

2.21.7 libtdb

Terrain Database Access

libtdb is a set of utilities for accessing the generic application database used by the MCC, SAF and PVD. It provides a mechanism for localized terrain caching. Optionally, direct memory access can be used instead of caching (implemented in "memory.c")It also generates a number of useful data constructs based on the current terrain, such as the elevation at a given (x,y) coordinate, a hull/world cosine matrix for a given (x,y) coordinate and heading or a list of obstructions to ground traversal in a given area.

2.21.7.1 cache.h

/simnet/common/libsrc/libtdb/cache.h

cache.h contains definitions needed for the terrain caching system. Table 2.21-154 describes the variables used by cache.h.

Variables		
Variable	**Type**	**Where Typedef Declared**
tcc	extern TERRAIN_CACHE_CONTROL	/simnet/common/libsrc/libtdb/cache.h
cache_map	extern pointer to CACHE_MAP	/simnet/common/libsrc/libtdb/cache.h

Tble 2.21-154 cache.h Variable Information.

2.21.7.2 map.h

/simnet/common/libsrc/libtdb/map.h

map.h defines types representing map coordinates and functions for manipulating these types.

2.21.7.3 objects.h

/simnet/common/libsrc/libtdb/objects.h

objects.h provides symbolic names for objects in the database.

2.21.7.4 tdb.h

/simnet/common/libsrc/libtdb/tdb.h

tdb.h contains definitions necessary for using the routines in libtdb. Table 2.21-155 describes the variables used by tdb.h.

Variables		
Variable	**Type**	**Where Typedef Declared**
tdb_info	extern TDB_INFO	/simnet/common/libsrc/libtdb/tdb.h
tdb_errno	extern INT_4	/simnet/common/include/global/mass_stdc.h

Table 2.21-155 tdb.h Variable Information.

2.21.7.5 tdb_local.h
/simnet/common/libsrc/libtdb/tdb_local.h

tdb_local.h contains all the declarations which are external to the source files but internal to the module. Table 2.21-156 describes the variables used by tdb_local.h.

Variables		
Variable	**Type**	**Where Typedef Declared**
size_of_patch_guards	extern INT_4	/simnet/common/include/global/mass_stdc.h
patch_indices	extern pointer to INT_4	/simnet/common/include/global/mass_stdc.h
tdb_patch_guards	extern pointer to PATCH_GUARD	/simnet/common/libsrc/libtdb/terrain.h
guards_included	extern char	Standard C type.
tdb_status	extern *TDB_STATUS*	*/simnet/common/libsrc/libtdb/t*db.h
tdb_method_used	extern int	Standard C type.

Table 2.21-156 tdb_local.h Variable Information.

2.21.7.6 terrain.h
/simnet/common/libsrc/libtdb/terrain.h

terrain.h defines the terrain database format.

The database begins with a database header of type DB_HEADER. This structure is made up of two other structures, DB_INFORMATION and DB_MAP. The database information header field contains such information as the database name and version number, physical dimensions of terrain patches represented, miscellaneous features represented, and UTM to Cartesian coordinate offsets for the given database. The database map contains offsets from the beginning of the database to the patch indices, patch guards, and the first patch in the database.

The database header is followed by the patch indices. These indices are 4-byte integers. Each index is an offset from the beginning of the database to its corresponding patch. Patches are numbered from 0 to n-1, from left to right and from bottom to top. A macro, PATCH_INDEX(), defined in "tdb.h" takes a pointer to a TDB_POINT and returns the corresponding index. For n patches there are n+1 indices, so that the number of bytes for a patch can be determined by index[n+1] - index[n] for all cases.

The patch indices are followed by a list of patch guards with type PATCH_GUARD, one per patch. These patch guards contain minima/maxima information for the given patch, allowing intervisibility routines to determine if a patch needs to be examined more closely.

The listing of patch guards is followed by the listing of the actual patches. Each patch is prefaced by a header of type PATCH_HEADER. A patch header contains information about the number of vertices, edges, polygons, objects, trees, treelines, and tree canopies in the patch, as well as offsets from the beginning of the patch to each of these entities.

Following the patch header is the list of vertices, which are type TDB_POINT. The order in which the vertices appear is undefined.

The vertices are followed by the list of edges in the patch. Each edge has the type EDGE_DESCRIPTOR. The two vertices of the edge appear in the same order as they appear in the list of vertices, thereby eliminating the need to check both edge v1->v2 and edge v2->v1 when doing edge comparisons. Edges also have a grid locator word. The patch is divided into a 4 x 4 grid. The grid boxes are numbered from 1 to 16, starting from the lower-left corner and going left to right, then bottom to top. If an edge enters a given grid box, the appropriate bit is set.

The list of edges is followed by the list of polygons in the patch. A polygon is represented by a POLYGON_DESCRIPTOR. The polygon descriptor contains a grid locator word, indices into the vertex list for the polygon vertices, and a polygon info bit field. Consecutive pairs of vertices represent edges of the polygon. Rather than use bit fields, #defines and macros are provided for accessing the poly info bit fields.

A list of objects follows the list of polygons in the patch. An object is defined by an OBJECT_DESCRIPTOR. The object descriptor contains a grid locator word, the four vertices of the object, and an object info bit field. Again, #defines and macros are provided to access these bit fields.

The object list is followed by a list of trees of type TREE_DESCRIPTOR. The tree descriptor contains the type, centroid, radius and height of the tree.

The next group of data in the patch is the list of treelines. Each treeline is represented by a header of type TREELINE_HEADER followed by a list of vertices of type TDB_POINT. The treeline header contains the number of vertices in the treeline and the treeline's type, extents, height, and elevation. It also contains the number of bytes for the treeline, to make access of successive treelines easier.

The last group of data in the patch is a list of tree canopies. Each canopy is represented by a CANOPY_HEADER, followed by a list of the treelines in the canopy, a list of canopy polygon vertices, a list of canopy polygon edges, and finally a list of canopy polygons. The vertices/edges/polygons scheme is analogous to that used for terrain polygons. Currently, no direct support for tree canopies is available from the libtdb routines.

2.21.7.7 cache_alloc.c

/simnet/common/libsrc/libtdb/cache_alloc.c

cache_alloc.c contains routines for allocating memory for and terminating the terrain caching system. Table 2.21-157 describes the variables used by cache_alloc.c. *patch_guards_obj* and *patch_indices_obj* are variables that are only used when SIMBFLY is defined.

Variables		
Variable	**Type**	**Where Typedef Declared**
total_cache_memory_required	INT_4	/simnet/common/include/global/mass_stdc.h
patch_guards_obj	extern OID	
patch_indices_obj	OID	

Table 2.21-157 cache_alloc.c Variable Information.

2.21.7.7.1 cache_init

cache_init initializes and allocates the memory for the terrain caching system. *number_of_patches_in_cache* is the number of terrain patches that will be cached. The function call is cache_init(number_of_patches_in_cache). Table 2.21-158 describes the parameters used, errors returned and functions called using this function with a MassComp computer. Table 2.21-159 describes the parameters used, errors returned and functions called using this function with a Butterfly.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
number_of_patches_in_cache	INT_4	/simnet/common/include/global/mass_stdc.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
size_of_cache_map	INT_4	/simnet/common/include/global/mass_stdc.h
size_of_indices	INT_4	/simnet/common/include/global/mass_stdc.h
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Unsuccessful.
0	int	Successful.
Errors		
Error Name	**Reason for Error**	
TE_CACHE_MALLOC	malloc failed for terrain cache.	
TE_LOCK	Could not lock down pages in memory.	
Calls		
Function	**Where Described**	
init_patch_indices	See Section 2.21.7.10.1.	
init_terrain_cache	See Section 2.21.7.10.2.	
init_cache_map	See Section 2.21.7.10.3.	
Called By		
Function	**Where Described**	
tdb_init_cache	See Section 2.21.7.27.1.	

Table 2.21-158 MassComp cache_init Information.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
number_of_patches_in_cache	INT_4	/simnet/common/include/global/mass_stdc.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
size_of_cache_map	INT_4	/simnet/common/include/global/mass_stdc.h
size_of_indices	INT_4	/simnet/common/include/global/mass_stdc.h
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Unsuccessful.
0	int	Successful.
Errors		
Error Name	**Reason for Error**	
TE_CACHE_MALLOC	malloc failed for terrain cache.	
Calls		
Function	**Where Described**	
init_patch_indices	See Section 2.21.7.10.1.	
init_terrain_cache	See Section 2.21.7.10.2.	
init_cache_map	See Section 2.21.7.10.3.	
Called By		
Function	**Where Described**	
tdb_init_cache	See Section 2.21.7.27.1.	

Table 2.21-159 Butterfly cache_init Information.

2.21.7.7.2 cache_and_file_terminate

cache_and_file_terminate terminates the terrain caching system, freeing the index file space and closing the terrain binary or poly file. The function call is cache_and_file_terminate(). Table 2.21-160 describes the internal variables used, errors returned and functions called using this function with a MassComp computer. Table 2.21-161 describes the internal variables used, errors returned and functions called using this function with a Butterfly.

Return Values		
Return Value	**Type**	**Meaning**
-1	int	Unsuccessful.
0	int	Successful.
Errors		
Error Name	**Reason for Error**	
TE_NOT_INITED	Cache has not been initialized.	

Called By	
Function	**Where Described**
tdb_terminate	See Section 2.21.7.27.2.

Table 2.21-160 MassComp cache_and_file_terminate Information.

Return Values		
Return Value	**Type**	**Meaning**
-1	int	Unsuccessful.
0	int	Successful.

Errors	
Error Name	**Reason for Error**
TE_NOT_INITED	Cache has not been initialized.

Called By	
Function	**Where Described**
tdb_terminate	See Section 2.21.7.27.2.

Table 2.21-161 Butterfly cache_and_file_terminate Information.

2.21.7.8 cache_cntl.c

/simnet/common/libsrc/libtdb/cache_cntl.c

cache_cntl.c contains routines for enabling and disabling the terrain cache.

2.21.7.8.1 tdb_cache_enable

tdb_cache_enable enables the terrain cache. The routine resets the cache structure by calling init_terrain_cache() and init_cache_map(), and then sets the value of *tdb_status.cache_enabled* to TRUE to make the cache available. The function call is tdb_cache_enable(). Table 2.21-162 describes the functions called using this function.

Calls	
Function	**Where Described**
init_terrain_cache	See Section 2.21.7.10.2.
init_cache_map	See Section 2.21.7.10.3.

Table 2.21-162 tdb_cache_enable Information.

2.21.7.8.2 tdb_cache_disable

tdb_cache_disable disables the terrain cache by setting the value of *tdb_status.cache_enabled* to FALSE. While the cache is disabled, the memory which it uses can be re-used to grab large "stripes" of consecutive patches using the function tdb_get_stripe(). When the user is finished with a stripe of patches, the cache must be re-enabled by making a call to tdb_cache_enable(). Note that all patches currently cached are removed when the cache is disabled. The function call is tdb_cache_disable(). Table 2.21-163 describes this function.

Called By	
Function	**Where Described**
tdb_init_memory	Se Section 2.21.7.25.1.

Table 2.21-163 tdb_cache_disable Information.

2.21.7.8.3 tdb_p_cache_enabled

tdb_p_cache_enabled returns the value of *tdb_status.cache_enabled.* If the value is TRUE, the terrain cache is enabled; if the value is FALSE, the terrain cache is disabled. The function call is tdb_p_cache_enabled(). Table 2.21-164 describes the return values generated by this function.

Return Values		
Return Value	**Type**	**Meaning**
TRUE	int	Cache is enabled.
FALSE	int	Cache is disabled.

Table 2.21-164 tdb_p_cache_enabled Information.

2.21.7.9 cache_data.c

/simnet/common/libsrc/libtdb/cache_data.c

cache_data.c contains storage allocation for globals in the terrain caching system. Table 2.21-165 describes the variables used by cache_data.c.

Variables		
Variable	**Type**	**Where Typedef Declared**
patch_indices	pointer to INT_4	/simnet/common/include/global/mass_stdc.h
tcc	TERRAIN_CACHE_CONTROL	/simnet/common/libsrc/libtdb/cache.h
cache_map	pointer to CACHE_MAP	/simnet/common/libsrc/libtdb/cache.h
size_of_patch_gurards	INT_4	/simnet/common/include/global/mass_stdc.h
tdb_patch_guards	pointer to PATCH_GUARD	/simnet/common/libsrc/libtdb/terrain.h
guards_included	char	Standard C type.

Table 2.21-165 cache_data.c Variable Information.

2.21.7.10 cache_init.c

/simnet/common/libsrc/libtdb/cache_init.c

cache_init.c contains routines for initializing the terrain caching system. *patch_guards_obj* is only used if SIMBFLY is defined. Table 2.21-166 describes the variables used by cache_init.c.

Variables		
Variable	**Type**	**Where Typedef Declared**
patch_guards_obj	OID	

Table 2.21-166 cache_init.c Variable Information.

2.21.7.10.1 init_patch_indices

init_patch_indices initializes the patch indices by reading in the index file. *patch_indices* points to the locked memory allocated to hold the entire index file. This index array contains pointers to each patch in the terrain file, with one index per patch. The function call is init_patch_indices(patch_indices). Table 2.21-167 describes the parameters used, errors returned and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
patch_indices	register pointer to INT_4	/simnet/common/include/global/mass_stdc.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
bytes_to_read	register INT_4	/simnet/common/include/global/mass_stdc.h
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Unsuccessful.
0	int	Successful
Errors		
Error Name	**Reason for Error**	
TE_OPEN_READ	Open or read on database failed.	
Called By		
Function	**Where Described**	
memory_init	See Section 2.21.7.25.2.	
cache_init	See Section 2.21.7.7.1.	

Table 2.21-167 init_patch_indices Information.

2.21.7.10.2 init_terrain_cache

init_terrain_cache initializes the local terrain cache. *tccp* is a pointer to the local terrain cache control. *number_of_patches* is the number of patches in the cache. The function call is init_terrain_cache(tccp, number_of_patches). Table 2.21-168 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
tccp	register pointer to TERRAIN_CACHE_CONTROL.	/simnet/common/libsrc/libtdb/cache.h
number_of_patches	INT_4	/simnet/common/include/global/mass_stdc.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	register INT_4	/simnet/common/include/global/mass_stdc.h
start_data	register pointer to char	Standard C type.
Called By		
Function	**Where Described**	
tdb_cache_enable	See Section 2.21.7.8.1.	
cache_init	See Section 2.21.7.7.1.	

Table 2.21-168 init_terrain_cache Information.

2.21.7.10.3 init_cache_map

init_cache_map initializes the cache map. *cache_map* maps the patch indices to cache indices. The function call is init_cache_map(cache_map). Table 2.21-169 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
cache_map	register pointer to CACHE_MAP	/simnet/common/libsrc/libtdb/cache.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	register INT_4	/simnet/common/include/global/mass_stdc.h
Called By		
Function	**Where Described**	
tdb_cache_enable	See Section 2.21.7.8.1	
cache_init	See Section 2.21.7.7.1.	

Table 2.21-169 init_cache_map Information.

2.21.7.10.4 init_patch_guards

init_patch_guards initializes the patch guards. Patch guards are the minimum and maximum elevation limits of the patch, used in intervisibility calculations. The patch guards for the entire terrain data base can be read into locked memory. The function call is init_patch_guards(). Table 2.21-170 describes the internal variables used, errors returned and functions called using this function with a MassComp computer. Table 2.21-171 describes the internal variables used, errors returned and functions called using this function with a Butterfly.

Return Values		
Return Value	**Type**	**Meaning**
NULL	pointer to PATCH_GUARD	Unsuccessful
tdb_patch_guards	pointer to PATCH_GUARD	Pointer to the successfully initialized patch guards.
Errors		
Error Name	**Reason for Error**	
TE_CACHE_MALLOC	malloc failed for terrain cache.	
TE_LOCK	Could not lock pages down in memory.	
TE_OPEN_READ	Open or read on the database failed.	

Table 2.21-170 MassComp init_patch_guards Information.

Return Values		
Return Value	**Type**	**Meaning**
NULL	pointer to PATCH_GUARD	Unsuccessful
tdb_patch_guards	pointer to PATCH_GUARD	Pointer to the successfully initialized patch guards
Errors		
Error Name	**Reason for Error**	
TE_CACHE_MALLOC	malloc failed for terrain cache.	
TE_OPEN_READ	Open or read on the database failed.	

Table 2.21-171 Butterfly init_patch_guards Information.

2.21.7.11 cache_query.c

/simnet/common/libsrc/libtdb/cache_query.c

cache_query.c contains routines for retrieving local terrain and preprocessed terrain patches from the cache.

2.21.7.11.1 terrain_cache_inquire

terrain_cache_inquire retrieves a preprocessed terrain patch from cache. *patch_index* is the number of the patch of interest. The function returns a pointer to the start of the terrain patch if successful, a null pointer otherwise. The function call is terrain_cache_inquire(patch_index). Table 2.21-172 describes the parameters used, errors returned and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
patch_index	INT_4	/simnet/common/include/global/mass_stdc.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
cache_index	register INT_4	/simnet/common/include/global/mass_stdc.h
inquiry_size	INT_4	/simnet/common/include/global/mass_stdc.h
bytes_read	INT_4	/simnet/common/include/global/mass_stdc.h
i	INT_4	/simnet/common/include/global/mass_stdc.h
got_ont	char	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
NULL	pointer to char	Unsuccessful
tcc.cache[cache_index].data	pointer to char	Pointer to the start of the terrain patch.
Errors		
Error Name	**Reason for Error**	
TE_NO_SLOT	Can't find slot in the cache queue.	
TE_PATCH_2BIG	A terrain patch was larger than MAX_LOCAL_PATCH_SIZE.	
TE_OPEN_READ	Open or read on the database failed.	
Calls		
Function	**Where Described**	
rotate_queue	See Section 2.21.7.12.3.	
dequeue_terrain_patch	See Section 2.21.7.12.1.	
enqueue_terrain_patch	See Section 2.21.7.12.2.	
Called By		
Function	**Where Described**	
tdb_get_terrain	See Section 2.21.7.18.1.	

Table 2.21-172 terrain_cache_inquire Information.

2.21.7.11.2 tdb_get_stripe

tdb_get_stripe returns a pointer to a "stripe" of consecutive terrain patches from the patch indiciated by*start* up to and including the patch indicated by *end*. Individual patches can be extracted from the "stripe" using the information in the global array pointed to by *patch_indices*. It reuses the cache memory, so caching must be turned off before you call it, and turned back on when you are done (tdb_disable_cache() and tdb_enable_cache()). If the patches cannot be retrieved, a null pointer is returned and the global terrain error variable *tdb_errno* is set. The function call is tdb_get_stripe(start, end). Table 2.21-173 describes the parameters used and errors returned using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
start	register INT_4	/simnet/common/include/global/mass_stdc.h
end	register INT_4	/simnet/common/include/global/mass_stdc.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
inquiry_size	INT_4	/simnet/common/include/global/mass_stdc.h
bytes_read	INT_4	/simnet/common/include/global/mass_stdc.h
Return Values		
Return Value	**Type**	**Meaning**
NULL	pointer to char	Unsuccessful.
tcc.cache[0].data	pointer to char	Pointer to block of patches.
Errors		
Error Name	**Reason for Error**	
TE_STRIPE_2BIG	A terrain patch was larger than MAX_LOCAL_PATCH_SIZE.	
TE_OPEN_READ	Open or read on the database failed.	

Table 2.21-173 tdb_get_stripe Information.

2.21.7.12 cache_queue.c

/simnet/common/libsrc/libtdb/cache_queue.c

cache_queue.c contains routines for queueing patches in the terrain cache. Table 2.21-174 describes the variables used by cache_queue.c.

Variables		
Variable	**Type**	**Where Typedef Declared**
tcc	extern TERRAIN_CACHE_CONTROL	/simnet/common/libsrc/libtdb/cache.h

Table 2.21-174 cache_queue.c Variable Information.

2.21.7.12.1 dequeue_terrain_patch

dequeue_terrain_patch removes the patch specified by *cache_index* from the terrain queue. The function call is dequeue_terrain_patch(cache_index). Table 2.21-175 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
cache_index	register INT_4	/simnet/common/include/global/mass_stdc.h
Called By		
Function	**Where Described**	
terrain_cache_inquire	See Section 2.21.7.11.1.	
rotate_queue	See Section 2.21.7.12.3.	

Table 2.21-175 dequeue_terrain_patch Information.

2.21.7.12.2 enqueue_terrain_patch

enqueue_terrain_patch places the patch specified by *cache_index* onto the end of the terrain queue. The function call is enqueue_terrain_patch(cache_index). Table 2.21-176 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
cache_index	register INT_4	/simnet/common/include/global/mass_stdc.h
Called By		
Function	**Where Described**	
terrain_cache_inquire	See Section 2.21.7.11.1.	
rotate_queue	See Section 2.21.7.12.3.	

Table 2.21-176 enqueue_terrain_patch Information.

2.21.7.12.3 rotate_queue

rotate_queue is called by terrain_cache_inquire. The patch specified by *index* is first removed from the terrain queue, then placed on the end of the queue. The function call is rotate_queue(). Table 2.21-177 describes the internal variables used and functions called using this function.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
index	INT_4	/simnet/common/include/global/mass_stdc.h
Return Values		
Return Value	**Type**	**Meaning**
index	int	the patch index

Calls	
Function	**Where Described**
dequeue_terrain_patch	See Section 2.21.7.12.1.
enqueue_terrain_patch	See Section 2.21.7.12.2.
Called By	
Function	**Where Described**
terrain_cache_inquire	See Section 2.21.7.11.1.

Table 2.21-177 rotate_queue Information.

2.21.7.13 consistent.c

/simnet/common/libsrc/libtdb/consistent.c

consistent.c contains a routine which checks terrain consistency.

2.21.7.13.1 tdb_consistent

tdb_consistent is used in debugging to check the terrain consistency. The function call is tdb_consistent(). Table 2.21-178 describes the internal variables used and functions called using this function.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	register INT_4	/simnet/common/include/global/mass_stdc.h
j	register INT_4	/simnet/common/include/global/mass_stdc.h
coord	TDB_POINT	/simnet/common/libsrc/libtdb/tdb.h
Calls		
Function	**Where Described**	
tdb_get_z	See Section 2.21.7.16.2.	
tdb_error	See Section 2.21.7.17.1.	

Table 2.21-178 tdb_consistent Information.

2.21.7.14 data.c
/simnet/common/libsrc/libtdb/data.c

data.c contains all the storage allocation for globals in the tdb library. Table 2.21-179 describes the variables used by data.c.

Variables		
Variable	**Type**	**Where Typedef Declared**
tdb_info	TDB_INFO	/simnet/common/libsrc/libtdb/tdb.h
tdb_status	TDB_STATUS	/simnet/common/libsrc/libtdb/tdb.h
tdb_errno	int	Standard C type.
tdb_err_string	array of 120 char	Standard C type.
patch_guards	pointr to PATCH_GUARD	/simnet/common/libsrc/libtdb/terrain.h
tdb_method_used	int	Standard C type.

Table 2.21-179 data.c Variable Information.

2.21.7.15 dump.c
/simnet/common/libsrc/libtdb/dump.c

dump.c contains routines for dumping out information about a terrain patch during debugging. Table 2.21-180 describes the variables used by dump.c.

Variables		
Variable	**Type**	**Where Typedef Declared**
terrain_object_names	pointer to array of MAX_NUM_OBJECT_TYPES char	Standard C type.
terrain_texture_names	pointer to array of MAX_NUM_TEXTURE_TYPES char	Standard C type.
tdb_dumpfile	pointer to FILE	Standard C type.

Table 2.21-180 dump.c Variable Information.

2.21.7.15.1 tdb_set_dumpfile

tdb_set_dumpfile allows you to specify an alternate output *file* for the tdb print and tdb dump routines. Otherwise, by default, output is sent to stderr. The function call is tdb_set_dumpfile(file). Table 2.21-181 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
file	pointer to FILE	Standard C type.
Called By		
Function	**Where Described**	
tdb_init_memory	See Section 2.21.7.25.1.	
tdb_init_cache	See Section 2.21.7.27.1.	

Table 2.21-181 tdb_set_dumpfile Information.

2.21.7.15.2 tdb_get_dumpfile

tdb_get_dumpfile returns a pointer to *tdb_dumpfile*. The function call is tdb_get_dumpfile(). Table 2.21-182 describes the return values generated by this function.

Return Values		
Return Value	**Type**	**Meaning**
tdb_dumpfile	pointer to FILE	Pointer to tdb_dumpfile.
Called By		
Function	**Where Described**	
tdb_print_version	See Section 2.21.7.31.1.	
tdb_print_format_compatible	See Section 2.21.7.31.2.	
tdb_print_db_format	See Section 2.21.7.31.3.	

Table 2.21-182 tdb_get_dumpfile Information.

2.21.7.15.3 tdb_dump_terrain

tdb_dump_terrain prints all information in the patch specified by *coord*. The result is voluminous, and is intended for debugging. By default all output is sent to stderr. tdb_set_dumpfile allows you to specify an alternate output file. The function returns 0 if there is no problem accessing the requested patch. Otherwise, -1 is returned and the global terrain error variable *tdb_errno* is set. The function call is tdb_dump_terrain(coord). Table 2.21-183 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
coord	pointer to TDB_POINT	/simnet/common/libsrc/libtdb/tdb.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
terrain	register pointer to char	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Unsuccessful.
0	int	Successful.
Calls		
Function	**Where Described**	
tdb_get_terrain	See Section 2.21.7.18.1.	
dump_terrain	See Section 2.21.7.15.4.	

Table 2.21-183 tdb_dump_terrain Information.

2.21.7.15.4 dump_terrain

dump_terrain gets information for a particular *patch*. The function call is dump_terrain(patch). Table 2.21-184 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
patch	register pointer to char	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	register INT_4	/simnet/common/include/global/mass_stdc.h
no_polygons	INT_4	/simnet/common/include/global/mass_stdc.h
no_objects	INT_4	/simnet/common/include/global/mass_stdc.h
no_trees	INT_4	/simnet/common/include/global/mass_stdc.h
no_treelines	INT_4	/simnet/common/include/global/mass_stdc.h
no_edges	INT_4	/simnet/common/include/global/mass_stdc.h
no_canopies	INT_4	/simnet/common/include/global/mass_stdc.h
no_canopy_vertices	INT_4	/simnet/common/include/global/mass_stdc.h
no_canopy_edges	INT_4	/simnet/common/include/global/mass_stdc.h
vertices	register pointer to TERRAIN_POINT	/simnet/common/libsrc/libtdb/terrain.h
patch_header	register pointer to PATCH_HEADER	/simnet/common/libsrc/libtdb/terrain.h
polygons	pointer to POLYGON_DESCRIPTOR	/simnet/common/libsrc/libtdb/terrain.h
objects	pointer to OBJECT_DESCRIPTOR	/simnet/common/libsrc/libtdb/terrain.h
treeline	pointer to TREELINE_HEADER	/simnet/common/libsrc/libtdb/terrain.h
trees	pointer to TREE_DESCRIPTOR	/simnet/common/libsrc/libtdb/terrain.h
edges	pointer to EDGE_DESCRIPTOR	/simnet/common/libsrc/libtdb/terrain.h
canopies	pointer to CANOPY_HEADER	/simnet/common/libsrc/libtdb/terrain.h
canopy_vertices	pointer to TERRAIN_POINT	/simnet/common/libsrc/libtdb/terrain.h
canopy_edges	pointer to EDGE_DESCRIPTOR	/simnet/common/libsrc/libtdb/terrain.h

Calls	
Function	**Where Described**
tdb_print_polygon	See Section 2.21.7.15.5.
tdb_print_edge	See Section 2.21.7.15.6.
tdb_print_object	See Section 2.21.7.15.7.
tdb_print_trline	See Section 2.21.7.15.8.
tdb_print_tree	See Section 2.21.7.15.10.
tdb_print_canopy	See Section 2.21.7.15.14.
Called By	
Function	**Where Described**
tdb_dump_terrain	See Section 2.21.7.15.3.

Table 2.21-184 dump_terrain Information.

2.21.7.15.5 tdb_print_polygon

tdb_print_polygon prints information for a specified polygon. *poly* is a pointer to the polygon descriptor. *vertices* is a pointer to the terrain point. By default all output is sent to stderr. tdb_set_dumpfile allows you to specify an alternate output file. The function call is tdb_print_polygon(poly, vertices). Table 2.21-185 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
poly	register pointer to POLYGON_DESCRIPTOR	/simnet/common/libsrc/libtdb/terrain.h
vertices	register pointer to TERRAIN_POINT	/simnet/common/libsrc/libtdb/terrain.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	register INT_4	/simnet/common/include/global/mass_stdc.h
no_vertices	register INT_4	/simnet/common/include/global/mass_stdc.h
Calls		
Function	**Where Described**	
print_grid_locator	See Section 2.21.7.15.11.	
GET_POLY_NUM_VERTS	Macro defined in /simnet/common/libsrc/libtdb/terrain.h.	
GET_POLY_PRIORITY	Macro defined in /simnet/common/libsrc/libtdb/terrain.h.	
GET_POLY_SHADING	Macro defined in /simnet/common/libsrc/libtdb/terrain.h.	
P_POLY_DUPLICATE	Macro defined in /simnet/common/libsrc/libtdb/terrain.h.	
GET_POLY_SOIL	Macro defined in /simnet/common/libsrc/libtdb/terrain.h.	
GET_POLY_MIN_X	Macro defined in /simnet/common/libsrc/libtdb/terrain.h.	
GET_POLY_MIN_Y	Macro defined in /simnet/common/libsrc/libtdb/terrain.h.	
GET_POLY_MIN_Z	Macro defined in /simnet/common/libsrc/libtdb/terrain.h.	
GET_POLY_MAX_X	Macro defined in /simnet/common/libsrc/libtdb/terrain.h.	
GET_POLY_MAX_Y	Macro defined in /simnet/common/libsrc/libtdb/terrain.h.	
GET_POLY_MAX_Z	Macro defined in /simnet/common/libsrc/libtdb/terrain.h.	

Called By	
Function	**Where Described**
dump_terrain	See Section 2.21.7.15.4.

Table 2.21-185 tdb_print_polygon Information.

2.21.7.15.6 tdb_print_edge

tdb_print_edge prints information for a specified edge. *edge* is a pointer to the edge descriptor. *vertices* is a pointer to the terrain point. By default all output is sent to stderr. tdb_set_dumpfile allows you to specify an alternate output file. The function call is tdb_print_edge(edge, vertices). Table 2.21-186 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
edge	register pointer to EDGE_DESCRIPTOR	/simnet/common/libsrc/libtdb/terrain.h
vertices	register pointer to TERRAIN_POINT	/simnet/common/libsrc/libtdb/terrain.h
Calls		
Function	**Where Described**	
print_grid_locator	See Section 2.21.7.15.11.	
Called By		
Function	**Where Described**	
dump_terrain	See Section 2.21.7.15.4.	

Table 2.21-186 tdb_print_edge Information.

2.21.7.15.7 tdb_print_object

tdb_print_object prints information about the specified *object*. By default all output is sent to stderr. tdb_set_dumpfile allows you to specify an alternate output file. The function call is tdb_print_object(object). Table 2.21-187 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
object	register pointer to OBJECT_DESCRIPTOR	/simnet/common/libsrc/libtdb/terrain.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	register INT_4	/simnet/common/include/global/mass_stdc.h
type	unsigned INT_4	/simnet/common/include/global/mass_stdc.h

Calls	
Function	**Where Described**
GET_OBJECT_TYPE	Macro defined in /simnet/common/libsrc/libtdb/terrain.h.
GET_OBJECT_MIN_X	Macro defined in /simnet/common/libsrc/libtdb/terrain.h.
GET_OBJECT_MIN_Y	Macro defined in /simnet/common/libsrc/libtdb/terrain.h.
GET_OBJECT_MIN_Z	Macro defined in /simnet/common/libsrc/libtdb/terrain.h.
GET_OBJECT_MAX_X	Macro defined in /simnet/common/libsrc/libtdb/terrain.h.
GET_OBJECT_MAX_Y	Macro defined in /simnet/common/libsrc/libtdb/terrain.h.
GET_OBJECT_MAX_Z	Macro defined in /simnet/common/libsrc/libtdb/terrain.h.
Called By	
Function	**Where Described**
dump_terrain	See Section 2.21.7.15.4.

Table 2.21-187 tdb_print_object Information.

2.21.7.15.8 tdb_print_trline

tdb_print_trline prints information for a specified *treeline*. It assumes that the treeline header is followed in memory by the list of vertices for that treeline (as in the terrain cache). By default all output is sent to stderr. tdb_set_dumpfile allows you to specify an alternate output file. The function call is tdb_print_trline(treeline). Table 2.21-188 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
treeline	register pointer to TREELINE_HEADER	/simnet/common/libsrc/libtdb/terrain.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	register INT_4	/simnet/common/include/global/mass_stdc.h
vertices	register pointer to TERRAIN_POINT	/simnet/common/libsrc/libtdb/terrain.h
Calls		
Function	**Where Described**	
print_trl_hdr	See Section 2.21.7.15.9.	
Called By		
Function	**Where Described**	
dump_terrain	See Section 2.21.7.15.4.	
tdb_print_canopy	See Section 2.21.7.15.14.	

Table 2.21-188 tdb_print_trline Information.

2.21.7.15.9 print_trl_hdr

print_trl_hdr prints only treeline header information for a specified *treeline*. By default all output is sent to stderr. tdb_set_dumpfile allows you to specify an alternate output file. The function call is print_trl_hdr(treeline). Table 2.21-189 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
treeline	register pointer to TREELINE_HEADER	/simnet/common/libsrc/libtdb/terrain.h
Called By		
Function	**Where Described**	
tdb_print_trline	See Section 2.21.7.15.8.	

Table 2.21-189 print_trl_hdr Information.

2.21.7.15.10 tdb_print_tree

tdb_print_tree prints information for a specified *tree*. By default all output is sent to stderr. tdb_set_dumpfile allows you to specify an alternate output file. The function call is tdb_print_tree(tree). Table 2.21-190 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
tree	register pointer to TREE_DESCRIPTOR	/simnet/common/libsrc/libtdb/terrain.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
type	int	Standard C type.
Called By		
Function	**Where Described**	
dump_terrain	See Section 2.21.7.15.4.	

Table 2.21-190 tdb_print_tree Information.

2.21.7.15.11 print_grid_locator

print_grid_locator prints information for a specified grid locator. By default all output is sent to stderr. tdb_set_dumpfile allows you to specify an alternate output file. The function call is print_grid_locator(grid_loc). Table 2.21-191 describes the parameters used, errors returned and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
grid_loc	HWORD	/simnet/common/include/global/mass_stdc.h

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	INT_4	
j	INT_4	
Called By		
Function	**Where Described**	
tdb_print_polygon	See Section 2.21.7.15.5.	
tdb_print_edge	See Section 2.21.7.15.6.	
print_c_poly	See Section 2.21.7.15.16.	

Table 2.21-191 print_grid_locator Information.

2.21.7.15.12 tdb_print_db_info

tdb_print_db_info prints information for the *db_info* specified. By default all output is sent to stderr. tdb_set_dumpfile allows you to specify an alternate output file. The function call is tdb_print_db_info(db_info). Table 2.21-192 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
db_info	pointer to DB_INFORMATION	/simnet/common/libsrc/libtdb/terrain.h
Calls		
Function	**Where Described**	
print_statisitics	See Section 2.21.7.15.18.	
print_terrain_map	See Section 2.21.7.15.13.	
P_FEATURE_TREE	Macro defined in /simnet/common/libsrc/libtdb/terrain.h.	
P_FEATURE_CANOPY	Macro defined in /simnet/common/libsrc/libtdb/terrain.h.	
P_FEATURE_RELPOLY	Macro defined in /simnet/common/libsrc/libtdb/terrain.h.	
P_FEATURE_MICRO	Macro defined in /simnet/common/libsrc/libtdb/terrain.h.	

Table 2.21-192 tdb_print_db_info Information.

2.21.7.15.13 print_terrain_map

print_terrain_map prints information for a specified *map*. By default all output is sent to stderr. tdb_set_dumpfile allows you to specify an alternate output file. The function call is print_terrain_map(map). Table 2.21-193 describes the parameters used, errors returned and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
map	pointer to TerrainMap	/simnet/common/libsrc/libtdb/map.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	int	Standard C type.

Called By	
Function	**Where Described**
tdb_print_db_info	See Section 2.21.7.15.12.

Table 2.21-193 print_terrain_map Information.

2.21.7.15.14 tdb_print_canopy

tdb_print_canopy prints canopy information. By default all output is sent to stderr. tdb_set_dumpfile allows you to specify an alternate output file. The function call is tdb_print_canopy(header, vertices). Table 2.21-194 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
header	pointer to CANOPY_HEADER	/simnet/common/libsrc/libtdb/terrain.h
vertices	register pointer to TERRAIN_POINT	/simnet/common/libsrc/libtdb/terrain.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
boundary	register pointer to TREELINE_HEADER	/simnet/common/libsrc/libtdb/terrain.h
polygons	register pointer to POLYGON_DESCRIPTOR	/simnet/common/libsrc/libtdb/terrain.h
i	int	Standard C type.
no_boundaries	int	Standard C type.
Calls		
Function	**Where Described**	
print_c_hdr	See Section 2.21.7.15.15.	
tdb_print_trline	See Section 2.21.7.15.8.	
print_c_poly	See Section 2.21.7.15.16.	
Called By		
Function	**Where Described**	
dump_terrain	See Section 2.21.7.15.4.	

Table 2.21-194 tdb_print_canopy Information.

2.21.7.15.15 print_c_hdr

print_c_hdr prints information for a specified canopy *header*. By default all output is sent to stderr. tdb_set_dumpfile allows you to specify an alternate output file. The function call is print_c_hdr(header). Table 2.21-195 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
header	pointer to CANOPY_HEADER	/simnet/common/libsrc/libtdb/terrain.h

Called By	
Function	**Where Described**
tdb_print_canopy	See Section 2.21.7.15.14.

Table 2.21-195 print_c_hdr Information.

2.21.7.15.16 print_c_poly

print_c_poly prints canopy polygon information. By default all output is sent to stderr. tdb_set_dumpfile allows you to specify an alternate output file. The function call is print_c_poly(poly, vertices). Table 2.21-196 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
poly	register pointer to POLYGON_DESCRIPTOR	/simnet/common/libsrc/libtdb/terrain.h
vertices	register pointer to TERRAIN_POINT	/simnet/common/libsrc/libtdb/terrain.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	register INT_4	/simnet/common/include/global/mass_stdc.h
no_vertices	register INT_4	/simnet/common/include/global/mass_stdc.h
Calls		
Function	**Where Described**	
print_grid_locator	See Section 2.21.7.15.11.	
GET_CANOPY_POLY_NUM_VERTS	Macro defined in /simnet/common/libsrc/libtdb/terrain.h.	
GET_CANOPY_POLY_PRIORITY	Macro defined in /simnet/common/libsrc/libtdb/terrain.h.	
GET_CANOPY_POLY_TYPE	Macro defined in /simnet/common/libsrc/libtdb/terrain.h.	
GET_CANOPY_POLY_MIN_X	Macro defined in /simnet/common/libsrc/libtdb/terrain.h.	
GET_CANOPY_POLY_MIN_Y	Macro defined in /simnet/common/libsrc/libtdb/terrain.h.	
GET_CANOPY_POLY_MIN_Z	Macro defined in /simnet/common/libsrc/libtdb/terrain.h.	
GET_CANOPY_POLY_MAX_X	Macro defined in /simnet/common/libsrc/libtdb/terrain.h.	
GET_CANOPY_POLY_MAX_Y	Macro defined in /simnet/common/libsrc/libtdb/terrain.h.	
GET_CANOPY_POLY_MAX_Z	Macro defined in /simnet/common/libsrc/libtdb/terrain.h.	
Called By		
Function	**Where Described**	
tdb_print_canopy	See Section 2.21.7.15.14.	

Table 2.21-196 print_c_poly Information.

2.21.7.15.17 tdb_print_cache_status

tdb_print_cache_status prints current status information for the cache. Output includes whether the cache is initialized, whether the cache is enabled, cache size, cache hits, and cache faults. By default all output is sent to stderr. tdb_set_dumpfile allows you to specify an alternate output file. The function call is tdb_print_cache_status(). Table 2.21-197 describes the internal variables used by this function.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
cache_index	register INT_4	/simnet/common/include/global/mass_stdc.h

Table 2.21-197 tdb_print_cache_status Information.

2.21.7.15.18 print_statistics

print_statistics prints statistical information. By default all output is sent to stderr. tdb_set_dumpfile allows you to specify an alternate output file. The function call is print_statistics(stats). Table 2.21-198 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
stats	pointer to DB_STATS	/simnet/common/libsrc/libtdb/terrain.h
Called By		
Function	**Where Described**	
tdb_print_db_info	See Section 2.21.7.15.12.	

Table 2.21-198 print_statistics Information.

2.21.7.15.19 tdb_get_db_name

tdb_get_db_name returns the name of the terrain database being used. The function call is tdb_get_db_name(name). Table 2.21-199 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
name	register pointer to char	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
name	pointer to char	Name of database.

Table 2.21-199 tdb_get_db_name Information.

2.21.7.15.20 tdb_get_db_version

tdb_get_db_version returns the version number of the terrain database being used. The function call is tdb_get_db_version(). Table 2.21-200 describes the return value generated by this function.

Return Values		
Return Value	**Type**	**Meaning**
tdb_info.db_header.db_info.version	INT_2	Database version.

Table 2.21-200 tdb_get_db_version Information.

2.21.7.15.21 init_object_and_texture_names

init_object_and_texture_names initializes the list of terrain texture names and terrain object names. The function call is init_object_and_texture_names(). Table 2.21-201 describes the internal variables used by this function.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	register int	Standard C type.
Called By		
Function	**Where Described**	
tdb_init_memory	See Section 2.21.7.25.1.	
tdb_init_cache	See Section 2.21.7.27.1.	

Table 2.21-201 init_object_and_texture_names Information.

2.21.7.15.22 get_texture_name_list

get_texture_name_list returns the list of terrain texture names. The function call is get_texture_name_list(). Table 2.21-202 describes the return value generated by this function.

Return Values		
Return Value	**Type**	**Meaning**
terrain_texture_names	pointer to pointer to char	the list of terrain texture names
Called By		
Function	**Where Described**	
tdb_thing_string	See Section 2.21.7.28.2.	

Table 2.21-202 get_texture_name_list Information.

2.21.7.15.23 get_object_name_list

get_object_name_list returns the list of terrain object names. The function call is get_object_name_list(). Table 2.21-203 describes the return value generated by this function.

Return Values		
Return Value	**Type**	**Meaning**
terrain_object_names	pointer to pointer to char	the list of terrain object names
Called By		
Function	**Where Described**	
tdb_thing_string	See Section 2.21.7.28.2.	

Table 2.21-203 get_object_name_list Information.

2.21.7.16 elevation.c

/simnet/common/libsrc/libtdb/elevation.c

elevation.c contains routines for getting an elevation from an (x,y) coordinate.

2.21.7.16.1 tdb_shade_get_z

tdb_shade_get_z fills in the z value with the elevation on the terrain at the (x,y) *coord*inates, returns the soil type of the supporting polygon, and sets the appropriate shade flag. The routine prioritizes polygons in order to search through to find which polygon includes the specified point. If the location is off the terrain database, the z value is set to 0.

check_objects allows the user to specify whether to check the elevation of objects in addition to terrain polygons. If *check_objects* is set to TDB_GROUND_ONLY, only terrain polygons are considered. If *check_objects* is TDB_CHECK_OBJECTS, then objects are considered along with the terrain. In the event that support is an object, SOIL_NA is returned.

The memory location pointed to by *shade_flag* is set to 0 if the support polygon is not shaded for a given sun angle, and 1 if it is. For TDB_SHADE_NO_SUN, the shade value is always 0.

tdb_shade_get_z returns the value -1 and sets the global terrain error variable *tdb_errno* if it can not determine the elevation at the given coordinates. If the location is off the terrain, *tdb_errno* is set to TE_OFF_TERRAIN.

The function call is tdb_shade_get_z(coord, check_objects, sun_angle, shade_flag). Table 2.21-204 describes the parameters used, errors returned and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
coord	register pointer to TDB_POINT	/simnet/common/libsrc/libtdb/tdb.h
check_objects	int	Standard C type.
sun_angle	int	Standard C type.
shade_flag	pointer to int	Standard C type.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
objects	register pointer to OBJECT_DESCRIPTOR	/simnet/common/libsrc/libtdb/terrain.h
no_objects	INT_4	/simnet/common/include/global/mass_stdc.h
i	register INT_4	/simnet/common/include/global/mass_stdc.h
support_poly	pointer to POLYGON_DESCRIPTOR	/simnet/common/libsrc/libtdb/terrain.h
grid_map	pointer to GRID_MAP	/simnet/common/libsrc/libtdb/terrain.h
vertices	pointer to TERRAIN_POINT	/simnet/common/libsrc/libtdb/terrain.h
polys	pointer to POLYGON_DESCRIPTOR	/simnet/common/libsrc/libtdb/terrain.h
patch_header	pointer to PATCH_HEADER	/simnet/common/libsrc/libtdb/terrain.h
support_found	int	Standard C type.
patch	register pointer to char	Standard C type.
prev_height	register REAL_8	/simnet/common/include/global/mass_stdc.h
current_soil	char	Standard C type.

Return Values		
Return Value	**Type**	**Meaning**
-1	int	Unsuccessful.
current_soil	int	Successful with no shade.
GET_POLY_SOIL(support_poly->poly_info)	int	Successful with shade.

Errors	
Error Name	**Reason for Error**
TE_OFF_TERRAIN	Point given is off terrain database.
TE_NO_SUPPORT	No support polygon could be found for an otherwise acceptable point.

Calls	
Function	**Where Described**
tdb_get_terrain	See Section 2.21.7.18.1.
GET_OBJECT_MIN_X	Macro defined in /simnet/common/libsrc/libtdb/terrain.h.
GET_OBJECT_MIN_Y	Macro defined in /simnet/common/libsrc/libtdb/terrain.h.
GET_OBJECT_MAX_X	Macro defined in /simnet/common/libsrc/libtdb/terrain.h.
GET_OBJECT_MAX_Y	Macro defined in /simnet/common/libsrc/libtdb/terrain.h.
object_include	See Section 2.21.7.22.2.
find_support	See Section 2.21.7.16.3.
find_height_on_poly	See Section 2.21.7.16.5.
P_POLY_MORNING_SHADE	Macro defined in /simnet/common/libsrc/libtdb/terrain.h.
P_POLY_NOON_SHADE	Macro defined in /simnet/common/libsrc/libtdb/terrain.h.
P_POLY_EVENING_SHADE	Macro defined in /simnet/common/libsrc/libtdb/terrain.h.
GET_POLY_SOIL	Macro defined in /simnet/common/libsrc/libtdb/terrain.h.
Called By	
Function	**Where Described**
tdb_get_z	See Section 2.21.7.16.2.
tracks_set_support_plane	See Section 2.21.7.20.2.

Table 2.21-204 tdb_shade_get_z Information.

2.21.7.16.2 tdb_get_z

tdb_get_z fills in the z value with the elevation on the terrain at the (x,y) *coord*inates, and returns the soil type of the supporting polygon. If the location is off the terrain database, the z value is set to 0.

check_objects allows the user to specify whether to check the elevation of objects in addition to terrain polygons. If *check_objects* is set to TDB_GROUND_ONLY, only terrain polygons are considered. If *check_objects* is TDB_CHECK_OBJECTS, then objects are considered along with the terrain. In the event that the support is an object, SOIL_NA is returned.

tdb_get_z returns the value -1 and sets the global terrain error variable *tdb_errno* if it can not determine the elevation at the given coordinates. If the location is off the terrain, *tdb_errno* is set to TE_OFF_TERRAIN.

The function call is tdb_get_z(coord, check_objects). Table 2.21-205 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
coord	register pointer to TDB_POINT	/simnet/common/libsrc/libtdb/tdb.h
check_objects	int	Standard C type
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
dummy_shade	int	Standard C type.

Return Values		
Return Value	**Type**	**Meaning**
tdb_shade_get_z(coord, check_objects, TDB_SHADE_NO_SUN, &dummy_shade)	int	Successful.
Calls		
Function	**Where Described**	
tdb_shade_get_z	See Section 2.21.7.16.1.	
Called By		
Function	**Where Described**	
tdb_consistent	See Section 2.21.7.13.1.	
tracks_set_support_plane	See Section 2.21.7.20.2.	

Table 2.21-205 tdb_get_z Information.

2.21.7.16.3 find_support

find_support determines if there is a support polygon at the given *coord*inate. The function call is find_support(support_poly, coord, grid_map, vertices, polys). Table 2.21-206 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
support_poly	*pointer to pointer to* POLYGON_DESCRIPTOR	/simnet/common/libsrc/libtdb/terrain.h
coord	pointer to TDB_POINT	/simnet/common/libsrc/libtdb/tdb.h
grid_map	register pointer to GRID_MAP	/simnet/common/libsrc/libtdb/terrain.h
vertices	pointer to TERRAIN_POINT	/simnet/common/libsrc/libtdb/terrain.h
polys	register pointer to POLYGON_DESCRIPTOR	/simnet/common/libsrc/libtdb/terrain.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
grid_number	register INT_2	/simnet/common/include/global/mass_stdc.h
start_poly_index	register INT_4	/simnet/common/include/global/mass_stdc.h
end_poly_index	register INT_4	/simnet/common/include/global/mass_stdc.h
i	register INT_4	/simnet/common/include/global/mass_stdc.h

Return Values		
Return Value	**Type**	**Meaning**
FALSE	int	A support polygon exists at point *coord*.
TRUE	int	A support polygon does not exist at point *coord*.
Calls		
Function	**Where Described**	
get_grid_number	See Section 2.21.7.16.6.	
p_poly_provides_support	See Section 2.21.7.16.4.	

Table 2.21-206 find_support Information.

2.21.7.16.4 p_poly_provides_support

p_poly_provides_support determines whether the given *coord*inate is included in the specified polygon, *current_poly*. The function call is p_poly_provides_support(current_poly, coord, vertices). Table 2.21-207 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
current_poly	register pointer to POLYGON_DESCRIPTOR	/simnet/common/libsrc/libtdb/terrain.h
coord	register pointer to TDB_POINT	/simnet/common/libsrc/libtdb/tdb.h
vertices	register pointer to TERRAIN_POINT	/simnet/common/libsrc/libtdb/terrain.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
poly_vertices	register pointer to HWORD	
poly_info	POLY_INFO	/simnet/common/libsrc/libtdb/terrain.h
Return Values		
Return Value	**Type**	**Meaning**
FALSE	int	The polygon does not support the given point.
TRUE	int	The polygon supports the given point.
Calls		
Function	**Where Described**	
GET_POLY_MIN_X	Macro defined in /simnet/common/libsrc/libtdb/terrain.h.	
GET_POLY_MIN_Y	Macro defined in /simnet/common/libsrc/libtdb/terrain.h.	
GET_POLY_MAX_X	Macro defined in /simnet/common/libsrc/libtdb/terrain.h.	
GET_POLY_MAX_Y	Macro defined in /simnet/common/libsrc/libtdb/terrain.h.	
polygon_include	See Section 2.21.7.22.1.	
GET_POLY_NUM_VERTS	Macro defined in /simnet/common/libsrc/libtdb/terrain.h.	

Called By	
Function	**Where Described**
find_support	See Section 2.21.7.16.3.

Table 2.21-207 p_poly_provides_support Information.

2.21.7.16.5 find_height_on_poly

find_height_on_poly determines the elevation of the given (x,y) *coord*inate on the specified polygon, *current_poly*. The routine calculates the elevation (*coord.z*) based on the dot product of the vector v0->*coord* and a normal vector to the polygon equaling zero. If the normal vector is 0, -1 is returned, signifying an error. If the elevation is successfully calculated, 0 is returned. The function call is find_height_on_poly(current_poly, coord, vertices). Table 2.21-208 describes the parameters used, errors returned and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
current_poly	pointer to POLYGON_DESCRIPTOR	/simnet/common/libsrc/libtdb/terrain.h
coord	register pointer to TDB_POINT	/simnet/common/libsrc/libtdb/tdb.h
vertices	pointer to TERRAIN_POINT	/simnet/common/libsrc/libtdb/terrain.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
normal_x	register REAL_8	/simnet/common/include/global/mass_stdc.h
normal_y	register REAL_8	/simnet/common/include/global/mass_stdc.h
normal_z	register REAL_8	/simnet/common/include/global/mass_stdc.h
fv0	register pointer to TERRAIN_POINT	/simnet/common/libsrc/libtdb/terrain.h
fv1	register pointer to TERRAIN_POINT	/simnet/common/libsrc/libtdb/terrain.h
fv2	register pointer to TERRAIN_POINT	/simnet/common/libsrc/libtdb/terrain.h
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Unsuccessful.
0	int	Successful.
Errors		
Error Name	**Reason for Error**	
TE_NO_SUPPORT	No support polygon could be found for an otherwise acceptable point.	

Table 2.21-208 find_height_on_poly Information.

2.21.7.16.6 get_grid_number

get_grid_number returns the grid number that the point *coord* lies in. The function call is get_grid_number(coord). Table 2.21-209 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
coord	register pointer to TDB_POINT	/simnet/common/libsrc/libtdb/tdb.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
lower_left	TDB_POINT	/simnet/common/libsrc/libtdb/tdb.h
patch_coord	TDB_POINT	/simnet/common/libsrc/libtdb/tdb.h
lower_left_p	register pointer to TDB_POINT	/simnet/common/libsrc/libtdb/tdb.h
patch_coord_p	register pointer to TDB_POINT	/simnet/common/libsrc/libtdb/tdb.h
x_dir	register INT_4	/simnet/common/include/global/mass_stdc.h
y_dir	register INT_4	/simnet/common/include/global/mass_stdc.h
grid_size	register INT_4	/simnet/common/include/global/mass_stdc.h
patch_size	register INT_4	/simnet/common/include/global/mass_stdc.h
float_patch_size	register double	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
x_dir + GRIDS_PER_PATCH y_dir	INT_4	The grid number that the point *coord* lies in.
Called By		
Function	**Where Described**	
find_support	See Section 2.21.7.16.3.	

Table 2.21-209 get_grid_number Information.

2.21.7.17 error.c

/simnet/common/libsrc/libtdb/error.c

error.c contains routines for handling terrain error condition codes. Table 2.21-210 describes the variables used by error.c

Variables		
Variable	**Type**	**Where Typedef Declared**
tdb_err_string	array of MAX_ERR_STRING char	Standard C type.

Table 2.21-210 error.c Variable Information.

2.21.7.17.1 tdb_error

tdb_error returns a pointer to a short error message (no new-line) describing the last error encountered during a call to one of the terrain library routines. The error number is taken from the external variable *tdb_errno* which is set when errors occur but not cleared when non-erroneous calls are made. The function call is tdb_error(). Table 2.21-211 describes the internal variables used and functions called using this function.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
format	array of MAX_DB_FORMAT_LENGTH char	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
tdb_err_string	pointer to char	the error message
Calls		
Function	**Where Described**	
tdb_get_db_format	See Section 2.21.7.31.5.	
Called By		
Function	**Where Described**	
tdb_consistent	See Section 2.21.7.13.1.	

Table 2.21-211 tdb_error Information.

2.21.7.17.2 tdb_p_on_database

tdb_p_on_database determines whether the given point *p* is on the database. The function call is tdb_p_on_database(p). Table 2.21-212 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
p	pointer to TDB_POINT	/simnet/common/libsrc/libtdb/tdb.h
Return Values		
Return Value	**Type**	**Meaning**
FALSE	int	Point not on database.
TRUE	int	Point on database.

Table 2.21-212 tdb_p_on_database Information.

2.21.7.18 get_patch.c
/simnet/common/libsrc/libtdb/get_patch.c

get_patch.c contains a routine for retrieving terrain patches from the cache memory.

2.21.7.18.1 tdb_get_terrain

tdb_get_terrain returns a pointer to the patch which includes the (x,y) location point *coord*. If the patch cannot be retrieved, a null pointer is returned and the global terrain error variable *tdb_errno* is set. The function call is tdb_get_terrain(coord). Table 2.21-213 describes the parameters used, errors returned and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
coord	pointer to TDB_POINT	/simnet/common/libsrc/libtdb/tdb.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
patch_index	INT_4	/simnet/common/include/global/mass_stdc.h
Return Values		
Return Value	**Type**	**Meaning**
NULL	pointer to char	The patch cannot be retrieved
terrain_cache_inquire(patch_index)	pointer to char	The patch including *coord*
terrain_memory_inquire(patch_index	pointer to char	The patch including *coord*
Errors		
Error Name	**Reason for Error**	
TE_CACHE_DISABLED	Tried to access cache while it was disabled.	
TE_OFF_TERRAIN	Point given is off terrain database.	
Calls		
Function	**Where Described**	
PATCH_INDEX	Macro defined in /simnet/common/libsrc/libtdb/tdb.h.	
terrain_cache_inquire	See Section 2.21.7.11.1.	
terrain_memory_inquire	See Section 2.21.7.25.4.	
Called By		
Function	**Where Described**	
tdb_dump_terrain	See Section 2.21.7.15.3.	
tdb_object_count	See Section 2.21.7.26.2.	
tdb_nth_object	See Section 2.21.7.26.4.	
tdb_close_object	See Section 2.21.7.26.6.	
tdb_obstr_object	See Section 2.21.7.26.8.	
tdb_trline_count	See Section 2.21.7.29.2.	
tdb_nth_trline	See Section 2.21.7.29.4.	
tdb_close_trline	See Section 2.21.7.29.6.	
tdb_tree_count	See Section 2.21.7.30.2.	
tdb_nth_tree	See Section 2.21.7.30.4.	
tdb_close_tree	See Section 2.21.7.30.6.	

Table 2.21-213 tdb_get_terrain Information.

2.21.7.19 gr_loc_num.c
/simnet/common/libsrc/libtdb/gr_loc_num.c

gr_loc_num.c contains a routine for getting a grid number from a patch.

2.21.7.19.1 tdb_get_grid_number

tdb_get_grid_number returns a grid number, given a *point* in patch coordinates and the *patch_size*. The number returned has a range from 0 to NUM_POLY_GRIDS-1, where NUM_POLY_GRIDS is NUM_GRIDS_PER_SIDE squared. The function call is tdb_get_grid_number(point, patch_size). Table 2.21-214 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
point	register pointer to TDB_POINT	/simnet/common/libsrc/libtdb/tdb.h
patch_size	INT_4	/simnet/common/include/global/mass_stdc.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
x_dir	register INT_4	/simnet/common/include/global/mass_stdc.h
y_dir	register INT_4	/simnet/common/include/global/mass_stdc.h
grid_size	register INT_4	/simnet/common/include/global/mass_stdc.h
Return Values		
Return Value	**Type**	**Meaning**
x_dir + (NUM_GRIDS_PER_SIDE y_dir))	INT_2	Grid number of *point*.

Table 2.21-214 tdb_get_grid_number Information.

2.21.7.20 h_to_w.c
/simnet/common/libsrc/libtdb/h_to_w.c

h_to_w.c contains routines for determining an appropriate hull-to-world matrix. Table 2.21-215 describes the variables used by h_to_w.c.

Variables		
Variable	**Type**	**Where Typedef Declared**
rear_wheel	array of 3 REAL_8	/simnet/common/include/global/mass_stdc.h
left_wheel	array of 3 REAL_8	/simnet/common/include/global/mass_stdc.h
right_wheel	array of 3 REAL_8	/simnet/common/include/global/mass_stdc.h
radius_wheel	REAL_8	/simnet/common/include/global/mass_stdc.h
support_rear	array of 3 REAL_8	/simnet/common/include/global/mass_stdc.h
support_right	REAL_8	/simnet/common/include/global/mass_stdc.h
support_left	REAL_8	/simnet/common/include/global/mass_stdc.h
current_soil	int	Standard C type.
global_shade	int	Standard C type.
global_sun_angle	int	Standard C type.

Table 2.21-215 h_to_w.c Variable Information.

2.21.7.20.1 tdb_get_hull_to_world

tdb_get_hull_to_world creates a hull-to-world matrix which will provide the proper transformation from hull coordinates to world coordinates for a vehicle on the terrain at the location *point* and with the given *heading* (in world coordinate radians). *hull_to_world* is the input pointer to the matrix which is filled out by this routine.

In case of failure, -1 is returned and the global terrain error variable *tdb_errno* is set, otherwise 0 is returned. If the failure is due to the given location being off the terrain (*tdb_errno* == TE_OFF_TERRAIN), the hull-to-world matrix passed is set to a valid rotation matrix for level support with the given heading about the z-axis.

The function call is tdb_get_hull_to_world(point, heading, hull_to_world). Table 2.21-216 describes the parameters used, errors returned and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
point	pointer to TDB_POINT	/simnet/common/libsrc/libtdb/tdb.h
heading	REAL_8	/simnet/common/include/global/mass_stdc.h
hull_to_world	matrix of REAL_8	/simnet/common/include/global/mass_stdc.h

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
coord	array of 3 REAL_8	/simnet/common/include/global/mass_stdc.h
w_to_h	matrix of REAL_8	/simnet/common/include/global/mass_stdc.h
o_mat	matrix of REAL_8	/simnet/common/include/global/mass_stdc.h
h_to_o	array of 3 REAL_8	/simnet/common/include/global/mass_stdc.h
u_norm	array of 3 REAL_8	/simnet/common/include/global/mass_stdc.h
temp_norm	array of 3 REAL_8	/simnet/common/include/global/mass_stdc.h
temp0	array of 3 REAL_8	/simnet/common/include/global/mass_stdc.h
temp1	array of 3 REAL_8	/simnet/common/include/global/mass_stdc.h
max_x_limit	REAL_8	/simnet/common/include/global/mass_stdc.h
max_y_limit	REAL_8	/simnet/common/include/global/mass_stdc.h
min_x_limit	REAL_8	/simnet/common/include/global/mass_stdc.h
min_y_limit	REAL_8	/simnet/common/include/global/mass_stdc.h

Return Values		
Return Value	**Type**	**Meaning**
-1	int	Unsuccessful.
0	int	Successful.

Errors	
Error Name	**Reason for Error**
TE_OFF_TERRAIN	Point given is off terrain database.

Calls	
Function	**Where Described**
mat_rot_init	See Section 2.6.2.47.1 in the Vehicles CSCI.
vec_scale	See Section 2.6 .2.64.1 in the Vehicles CSCI.
vec_mat_mul	See Section 2.6.2.56.1 in the Vehicles CSCI.
mat_transpose	See Section 2.6.2.51.1 in the Vehicles CSCI.
tracks_set_support_plane	See Section 2.21.7.20.2.
vec_cross_prod	See Section 2.6.2.66.1. in the Vehicles CSCI.
mat_mat_mul	See Section 2.6.2.32.1 in the Vehicles CSCI.

Called By	
Function	**Where Described**
tdb_shade_place_vehicle	See Section 2.21.7.20.4.

Table 2.21-216 tdb_get_hull_to_world Information.

2.21.7.20.2 tracks_set_support_plane

tracks_set_support_plane computes the unit normal of a support plane in world coordinates. *h_to_w* is the hull-to-world matrix for the vehicle, *h_to_o* is the hull-to-origin vector for the vehicle and *u_norm* receives the resultant unit normal. The function call is tracks_set_support_plane(h_to_w, h_to_o, u_norm). Table 2.21-217 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
h_to_w	register matrix of REAL_8	/simnet/common/include/global/mass_stdc.h
h_to_o	register array of REAL_8	/simnet/common/include/global/mass_stdc.h
u_norm	register array of REAL_8	/simnet/common/include/global/mass_stdc.h
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Unsuccessful.
0	int	Successful.
Calls		
Function	**Where Described**	
vec_sub	See Section 2.6.2.65.1 in the Vehicles CSCI.	
vec_mat_mul	See Section 2.6.2.56.1 in the Vehicles CSCI.	
tdb_get_z	See Section 2.21.7.16.2.	
tdb_shade_get_z	See Section 2.21.7.16.1.	
tracks_calc_unit_normal	See Section 2.21.7.20.3.	
Called By		
Function	**Where Described**	
tdb_get_hull_to_world	See Section 2.21.7.20.1.	

Table 2.21-217 tracks_set_support_plane Information.

2.21.7.20.3 tracks_calc_unit_normal

tracks_calc_unit_normal calculates the unit normal to a support plane defined by three points, *p1*, *p2* and *p3*. *result* receives the unit normal. This unit normal is also passed to the kinematics code. The function call is tracks_calc_unit_normal(p1, p2, p3, result). Table 2.21-218 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
p1	register array of REAL_8	/simnet/common/include/global/mass_stdc.h
p2	array of REAL_8	/simnet/common/include/global/mass_stdc.h
p3	array of REAL_8	/simnet/common/include/global/mass_stdc.h
result	register array of REAL_8	/simnet/common/include/global/mass_stdc.h

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
v1	array of 3 REAL_8	/simnet/common/include/global/mass_stdc.h
v2	array of 3 REAL_8	/simnet/common/include/global/mass_stdc.h
Calls		
Function	**Where Described**	
vec_sub	See Section 2.6.2.65.1 in the Vehicles CSCI.	
vec_cross_prod	See Section 2.6.2.66.1 in the Vehicles CSCI.	
vec_normalize	See Section 2.6.2.63.1 in the Vehicles CSCI.	
Called By		
Function	**Where Described**	
tracks_set_support_plane	See Section 2.21.7.20.2.	

Table 2.21-218 tracks_calc_unit_normal Information.

2.21.7.20.4 tdb_shade_place_vehicle

tdb_shade_place_vehicle creates a hull-to-world matrix which will provide the proper transformation from hull coordinates to world coordinates for a vehicle on the terrain at the location *point* and with the given *heading*. It determines the elevation at the support point, setting the z location accordingly, and returns the soil type of the supporting polygon. It also determines if the support polygon is in shade for the given *sun_angle*. The memory location pointed to by *shade*is set to 1 if the polygon is shaded, 0 otherwise. *hull_to_world* is the input pointer to the matrix which is filled out by this routine. In case of failure, -1 is returned and the global terrain error variable *tdb_errno* is set. The function call is tdb_shade_place_vehicle(point, heading, hull_to_world, sun_angle, shade). Table 2.21-219 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
point	register pointer to TDB_POINT	/simnet/common/libsrc/libtdb/tdb.h
heading	REAL_8	/simnet/common/include/global/mass_stdc.h
hull_to_world	matrix of REAL_8	/simnet/common/include/global/mass_stdc.h
sun_angle	int	Standard C type.
shade	pointer to int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
normal_x	register REAL_8	/simnet/common/include/global/mass_stdc.h
normal_y	register REAL_8	/simnet/common/include/global/mass_stdc.h
normal_z	register REAL_8	/simnet/common/include/global/mass_stdc.h

Return Values		
Return Value	**Type**	**Meaning**
-1	int	Unsuccessful.
current_soil	int	Successful
Calls		
Function	**Where Described**	
tdb_get_hull_to_world	See Section 2.21.7.20.1.	
Called By		
Function	**Where Described**	
tdb_place_vehicle	See Section 2.21.7.20.5.	

Table 2.21-219 tdb_shade_place_vehicle Information.

2.21.7.20.5 tdb_place_vehicle

tdb_place_vehicle creates a hull-to-world matrix which will provide the proper transformation from hull coordinates to world coordinates for a vehicle on the terrain at the location *point* and with the given *heading*. It determines the elevation at the support point, setting the z location accordingly, and returns the soil type of the supporting polygon. *hull_to_world* is the input pointer to the matrix which is filled out by this routine. In case of failure, -1 is returned and the global terrain error variable *tdb_errno* is set. The function call is tdb_place_vehicle(point, heading, hull_to_world). Table 2.21-220 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
point	register pointer to TDB_POINT	/simnet/common/libsrc/libtdb/tdb.h
heading	REAL_8	/simnet/common/include/global/mass_stdc.h
hull_to_world	matrix of REAL_8	/simnet/common/include/global/mass_stdc.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
dummy_shade	int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
tdb_shade_place_vehicle(point, heading, hull_to_world, TDB_SHADE_NO_SUN, &dummy_shade)	int	If equal to the soil type, the routine was successful; If equal to -1, the routine was unsuccessful
Calls		
Function	**Where Described**	
tdb_shade_place_vehicle	See Section 2.21.7.20.4.	

Table 2.21-220 tdb_place_vehicle Information.

2.21.7.21 header.c
/simnet/common/libsrc/libtdb/header.c

header.c contains a routine for reading the db_header for a terrain database.

2.21.7.21.1 tdb_read_header

tdb_read_header reads the terrain data base db_header. The function call is tdb_read_header(info). Table 2.21-221 describes the parameters used and errors returned by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
info	register pointer to TDB_INFO	/simnet/common/libsrc/libtdb/tdb.h
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Unsuccessful.
0	int	Successful.
Errors		
Error Name	**Reason for Error**	
TE_OPEN_READ	Open or read on the database failed.	
Called By		
Function	**Where Described**	
tdb_get_tdb_info	See Section 2.21.7.27.3.	

Table 2.21-221 tdb_read_header Information.

2.21.7.22 include.c
/simnet/common/libsrc/libtdb/include.c

2.21.7.22.1 polygon_include

polygon_include determines whether the point *p1* is inside or outside the given polygon, *poly*. The function call is polygon_include(poly, num_verts, vertices, p1). Table 2.21-222 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
poly	pointer to POLYGON_DESCRIPTOR	/simnet/common/libsrc/libtdb/terrain.h
num_verts	register INT_4	/simnet/common/include/global/mass_stdc.h
vertices	register pointer to TERRAIN_POINT	/simnet/common/libsrc/libtdb/terrain.h
p1	register pointer to TDB_POINT	/simnet/common/libsrc/libtdb/tdb.h

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
temp_point1	register pointer to TERRAIN_POINT	/simnet/common/libsrc/libtdb/terrain.h
temp_point2	register pointer to TERRAIN_POINT	/simnet/common/libsrc/libtdb/terrain.h
i	register INT_4	/simnet/common/include/global/mass_stdc.h
z	register float	Standard C type.
dir	register INT_4	/simnet/common/include/global/mass_stdc.h
Return Values		
Return Value	**Type**	**Meaning**
OUTSIDE	int	Point not included in polygon.
INSIDE	int	Point included in polygon.
Calls		
Function	**Where Described**	
V_CROSS	Macro defined in /simnet/common/libsrc/libtdb/include.c.	
Called By		
Function	**Where Described**	
p_poly_provides_support	See Section 2.21.7.16.4.	

Table 2.21-222 polygon_include Information.

2.21.7.22.2 object_include

object_include determines whether the point *p1* is inside or outside the given *object*. The function call is object_include(object, p1). Table 2.21-223 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
object	pointer to OBJECT_DESCRIPTOR	/simnet/common/libsrc/libtdb/terrain.h
p1	register pointer to TDB_POINT	/simnet/common/libsrc/libtdb/tdb.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
temp_point1	register pointer to TERRAIN_POINT	/simnet/common/libsrc/libtdb/terrain.h
temp_point2	register pointer to TERRAIN_POINT	/simnet/common/libsrc/libtdb/terrain.h
i	register INT_4	/simnet/common/include/global/mass_stdc.h
z	register float	Standard C type.
dir	register INT_4	/simnet/common/include/global/mass_stdc.h
num_verts	INT_4	/simnet/common/include/global/mass_stdc.h

Return Values		
Return Value	**Type**	**Meaning**
OUTSIDE	int	Point not included in oblect.
INSIDE	int	Point included in object
Calls		
Function	**Where Described**	
V_CROSS	Macro defined in /simnet/common/libsrc/libtdb/include.c.	

Table 2.21-223 object_include Information.

2.21.7.23 lock.c
/simnet/common/libsrc/libtdb/lock.c

lock.c contains routines for locking and unlocking patches in the cache so they can't be swapped out. Note that they are removed if the cache is disabled.

2.21.7.23.1 tdb_lock_patch

tdb_lock_patch marks a terrain patch, specified by *coord*, as unpurgeable. Normally, a patch runs the risk of being purged any time a new patch is needed. It may therefore be necessary to lock in patches to which pointers are maintained. Locked patches can still be removed from the cache if the cache is disabled, leaving dangling pointers. If the patch requested is not in the cache, -1 is returned and the global terrain error variable *tdb_errno* is set. Otherwise 0 is returned. The function call is tdb_lock_patch(coord). Table 2.21-224 describes the parameters used, errors returned and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
coord	pointer to TDB_POINT	/simnet/common/libsrc/libtdb/tdb.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
patch_index	register INT_4	/simnet/common/include/global/mass_stdc.h
cache_index	register INT_4	/simnet/common/include/global/mass_stdc.h
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Unsuccessful.
0	int	Successful.
Errors		
Error Name	**Reason for Error**	
TE_NOT_IN_CACHE	Tried to lock a patch not in the cache.	
Calls		
Function	**Where Described**	
PATCH_INDEX	Macro defined in /simnet/common/libsrc/libtdb/tdb.h.	

Table 2.21-224 tdb_lock_patch Information.

2.21.7.23.2 tdb_unlock_patch

tdb_unlock_patch marks a previously locked terrain patch, specified by *coord*, as purgeable. If the patch requested is not in the cache, -1 is returned and the global terrain error variable *tdb_errno* is set. Otherwise 0 is returned. The function call is tdb_unlock_patch(coord). Table 2.21-225 describes the parameters used, errors returned and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
coord	pointer to TDB_POINT	/simnet/common/libsrc/libtdb/tdb.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
patch_index	register INT_4	/simnet/common/include/global/mass_stdc.h
cache_index	register INT_4	/simnet/common/include/global/mass_stdc.h
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Unsuccessful.
0	int	Successful.
Errors		
Error Name	**Reason for Error**	
TE_NOT_IN_CACHE	Tried to unlock a patch not in the cache.	
Calls		
Function	**Where Described**	
PATCH_INDEX	Macro defined in /simnet/common/libsrc/libtdb/tdb.h.	

Table 2.21-225 tdb_unlock_patch Information.

2.21.7.24 map.c

/simnet/common/libsrc/libtdb/map.c

map.c contains routines to convert between military and terrain coordinates. Table 2.21-226 describes the variables used by map.c.

Variables		
Variable	**Type**	**Where Typedef Declared**
scale[]	INT_4	/simnet/common/include/global/mass_stdc.h

Table 2.21-226 map.c Variable Information.

2.21.7.24.1 tdb_giv_utm_get_xy

tdb_giv_utm_get_xy converts the UTM coordinate string pointed to by *utm_str* to Cartesian coordinates, and places the results where *coord* indicates. A correctly formed UTM coordinate string consists of two alphabetic characters, the grid designator, followed by the x-direction digits, followed by the y-direction digits. The direction digits can contain 2, 3, 4, or 5 digits. The grid designator and direction digits are separated from each other by any number of non-digits. For example: ES-85-95; es * 85000 ** 95000. The routine checks the validity of the UTM coordinates, converting the numeric portion to meters from the lower-left corner of the grid zone. The letter pair is matched to the designation of a grid zone, and the UTM coordinate numbers are checked for inclusion within that grid zone. The function returns 0 upon successful completion. If an error occurs, -1 is returned and the global terrain error variable *tdb_errno* is set. The function call is tdb_giv_utm_get_xy(utm_str, coord). Table 2.21-227 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
utm_str	register pointer to char	Standard C type.
coord	pointer to TDB_POINT	/simnet/common/libsrc/libtdb/tdb.h
Return Values		
Return Value	**Type**	**Meaning**
tdb_map_utm_to_xy(&(tdb_info.db_header.db_info.map_info), utm_str, coord)	int	the terrain coordinates in Cartesian coordinates
Calls		
Function	**Where Described**	
tdb_map_utm_to_xy	See Section 2.21.7.24.2.	

Table 2.21-227 tdb_giv_utm_get_xy Information.

2.21.7.24.2 tdb_map_utm_to_xy

tdb_map_utm_to_xy converts from UTM to Cartesian coordinates, the same as tdb_giv_utm_get_xy. The *map_info* parameter allows conversion for any terrain database without initializing terrain caching. The function returns 0 upon successful completion. If an error occurs, -1 is returned and the global terrain error variable *tdb_errno* is set. The function call is tdb_map_utm_to_xy(map_info, utm_str, coord). Table 2.21-228 describes the parameters used and errors returned by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
map_info	register pointer to TerrainMap	/simnet/common/libsrc/libtdb/map.h
utm_str	pointer to char	Standard C type.
coord	pointer to TDB_POINT	/simnet/common/libsrc/libtdb/tdb.h

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	register INT_2	/simnet/common/include/global/mass_stdc.h
ch	register INT_2	/simnet/common/include/global/mass_stdc.h
zone	register pointer to GridZone	/simnet/common/libsrc/libtdb/map.h
prec	INT_2	/simnet/common/include/global/mass_stdc.h
pt	LongPt	/simnet/mcc/include/longpt.h
str[maxMapCoordString]	char	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Unsuccessful.
0	int	Successful.
Errors		
Error Name	**Reason for Error**	
TE_BADF_UTM	Invalid format for UTM string.	
TE_BAD_ALPHA	Not a known grid zone.	
TE_OFF_GRID	The numeric part of a UTM string is not in the grid zone specified.	
Called By		
Function	**Where Described**	
tdb_giv_utm_get_xy	See Section 2.21.7.24.1.	

Table 2.21-228 tdb_map_utm_to_xy Information.

2.21.7.24.3 tdb_giv_xy_get_utm

tdb_giv_xy_get_utm converts the Cartesian coordinates given in *coord* into UTM coordinates, formats a string based on the *prec*ision and *separator* specified, and copies the result into *utm_str*. The *prec*ision indicates the number of digits in each direction; hence an 8-digit UTM coordinate string has a precision of 4. The *separator* is placed between the alphabetic grid designators and the x-direction digits, and again between the x-direction digits and the y-direction digits. A UTM coordinate string with a precision of 3 and a separator of - would have the format AA-000-000. If *separator* is the null character, no separator will be included in the string. The function returns 0 upon successful completion. If an error occurs, -1 is returned and the global terrain error variable *tdb_errno* is set. The function call is tdb_giv_xy_get_utm(coord, utm_str, prec, separator). Table 2.21-229 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
coord	register pointer to TDB_POINT	/simnet/common/libsrc/libtdb/tdb.h
utm_str	register pointer to char	Standard C type
prec	register INT_4	/simnet/common/include/global/mass_stdc.h
separator	char	Standard C type

Return Values		
Return Value	**Type**	**Meaning**
tdb_map_xy_to_utm(&(tdb_info.db_header.db_info.map_info), coord, utm_str, prec, separator)	INT_4	If equal to 0, then operation was successful; if equal to -1, ann error is returned.
Calls		
Function	**Where Described**	
tdb_map_xy_to_utm	See Section 2.21.7.24.4.	

Table 2.21-229 tdb_giv_xy_get_utm Information.

2.21.7.24.4 tdb_map_xy_to_utm

tdb_map_xy_to_utm converts from Cartesian to UTM coordinates, the same as tdb_giv_xy_get_utm. The *map_info* parameter allows conversion for any terrain database without initializing terrain caching. The function returns 0 upon successful completion. If an error occurs, -1 is returned and the global terrain error variable *tdb_errno* is set. The function call is tdb_map_xy_to_utm(map_info, coord, utm_str, prec, separator). Table 2.21-230 describes the parameters used and errors returned by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
map_info	pointer to TerrainMap	/simnet/common/libsrc/libtdb/map.h
coord	pointer to TDB_POINT	/simnet/common/libsrc/libtdb/tdb.h
utm_str	pointer to char	Standard C type.
prec	INT_4	/simnet/common/include/global/mass_stdc.h
separator	char	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	register INT_2	/simnet/common/include/global/mass_stdc.h
n	register INT_4	/simnet/common/include/global/mass_stdc.h
cp	register pointer to char	Standard C type.
pt	LongPt	/simnet/mcc/include/longpt.h
zone	pointer to GridZone	/simnet/common/libsrc/libtdb/map.h
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Unsuccessful.
0	int	Successful.

Errors	
Error Name	**Reason for Error**
TE_NO_GRID_ZONE	No grid zone know for a given point.
Called By	
Function	**Where Described**
tdb_giv_xy_get_utm	See Section 2.21.7.24.3.

Table 2.21-230 tdb_map_xy_to_utm Information.

2.21.7.25 memory.c

/simnet/common/libsrc/libtdb/memory.c

memory.c contains routines for initializing and terminating the tdb terrain direct memory access system.

2.21.7.25.1 tdb_init_memory

tdb_init_memory disables the tdb caching system and enables the terrain data base direct memory access system. The function call is tdb_init_memory(pathname). Table 2.21-231 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
pathname	register pointer to char	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Unsuccessful.
0	int	Successful.
Errors		
Error Name	**Reason for Error**	
TE_BAD_FORMAT	Database has wrong format for this version of libtdb.	
Calls		
Function	**Where Described**	
tdb_get_tdb_info	See Section 2.21.7.27.3.	
tdb_right_format	See Section 2.21.7.31.4.	
init_object_and_texture_names	See Section 2.21.7.15.21.	
tdb_set_dumpfile	See Section 2.21.7.15.1.	
memory_init	See Section 2.21.7.25.2.	
tdb_cache_disable	See Section 2.21.7.8.2.	

Table 2.21-231 tdb_init_memory Information.

2.21.7.25.2 memory_init

memory_init initalizes and allocates the memory for the tdb direct memory access system. The function call is memory_init(). Table 2.21-232 describes the internal variables used, errors returned and functions called using this function.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
size_of_indices	long	Standard C type.
size_of_patches	long	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Unsuccessful. Always returned when SIMBFLY is defined.
0	int	Successful.
Errors		
Error Name	**Reason for Error**	
TE_CACHE_MALLOC	malloc failed for terrain cache.	
TE_OPEN_READ	Open or read on the database failed.	
Calls		
Function	**Where Described**	
init_patch_indices	See Section 2.21.7.10.1.	
Called By		
Function	**Where Described**	
tdb_init_memory	See Section 2.21.7.25.1.	

Table 2.21-232 memory_init Information.

2.21.7.25.3 memory_terminate

memory_terminate terminates the tdb direct memory access system, freeing the index file space and closing the terrain binary or poly file. The function call is memory_terminate(). Table 2.21-233 describes the internal variables used and errors returned by this function.

Return Values		
Return Value	**Type**	**Meaning**
-1	int	Unsuccessful. Always returned when SIMBFLY is defined.
0	int	Successful.
Errors		
Error Name	**Reason for Error**	
TE_NOT_INITED	Memory not previously initialized.	
Called By		
Function	**Where Described**	
tdb_terminate	See Section 2.21.7.27.2.	

Table 2.21-233 memory_terminate Information.

2.21.7.25.4 terrain_memory_inquire

terrain_memory_inquire returns a pointer to the terrain patch referenced by *patch_index*. The function call is terrain_memory_inquire(patch_index). Table 2.21-234 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
patch_index	INT_4	/simnet/common/include/global/mass_stdc.h
Return Values		
Return Value	**Type**	**Meaning**
tdb_info.patch_memory + patch_indices[patch_index] - patch_indices[0]	pointer to char	A pointer to the terrain patch referenced by *patch_index*.
Called By		
Function	**Where Described**	
tdb_get_terrain	See Section 2.21.7.18.1.	

Table 2.21-234 terrain_memory_inquire Information.

2.21.7.26 objects.c

/simnet/common/libsrc/libtdb/objects.c

2.21.7.26.1 count_objects_in_patch

count_objects_in_patch is called to determine how many *objects* are around the *location* bounded by the smaller of *radius* or the path boundary. The routine checks one terrain patch. If the area is not obstructed, 0 is returned. Otherwise, the number of obstructing objects is returned. The function call is count_objects_in_patch(objects, num_objects, location, radius). Table 2.21-235 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
objects	register pointer to OBJECT_DESCRIPTOR	/simnet/common/libsrc/libtdb/terrain.h
num_objects	INT_4	/simnet/common/include/global/mass_stdc.h
location	register pointer to TDB_POINT	/simnet/common/libsrc/libtdb/tdb.h
radius	REAL_8	/simnet/common/include/global/mass_stdc.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	register INT_4	/simnet/common/include/global/mass_stdc.h
num_obstacles	register INT_4	/simnet/common/include/global/mass_stdc.h

Return Values		
Return Value	**Type**	**Meaning**
num_obstacles	int	Number of obstructing obstacles.
Calls		
Function	**Where Described**	
GET_OBJECT_MIN_X	Macro defined in /simnet/common/libsrc/libtdb/terrain.h.	
GET_OBJECT_MAX_X	Macro defined in /simnet/common/libsrc/libtdb/terrain.h.	
GET_OBJECT_MIN_Y	Macro defined in /simnet/common/libsrc/libtdb/terrain.h.	
GET_OBJECT_MAX_Y	Macro defined in /simnet/common/libsrc/libtdb/terrain.h.	
Called By		
Function	**Where Described**	
tdb_object_count	See Section 2.21.7.26.2.	

Table 2.21-235 count_objects_in_patch Information.

2.21.7.26.2 tdb_object_count

tdb_object_count is called to determine if the area at the given *location* with the given *radius* is obstructed by any objects. If the area is not obstructed, 0 is returned. Otherwise, the number of obstructing objects is returned. If an error occurs, the global terrain error variable *tdb_errno* is set. The function call is tdb_object_count(location, radius). Table 2.21-236 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
location	register pointer to TDB_POINT	/simnet/common/libsrc/libtdb/tdb.h
radius	REAL_8	/simnet/common/include/global/mass_stdc.h

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
patch	register pointer to char	Standard C type.
num_obstacles	register INT_4	/simnet/common/include/global/mass_stdc.h
patch_header	pointer to PATCH_HEADER	/simnet/common/libsrc/libtdb/terrain.h
objects	pointer to OBJECT_DESCRIPTOR	/simnet/common/libsrc/libtdb/terrain.h
num_objects_in_patch	INT_4	/simnet/common/include/global/mass_stdc.h
coord	TDB_POINT	/simnet/common/libsrc/libtdb/tdb.h
start_x	REAL_8	/simnet/common/include/global/mass_stdc.h
start_y	REAL_8	/simnet/common/include/global/mass_stdc.h
bound_x	REAL_8	/simnet/common/include/global/mass_stdc.h
bound_y	REAL_8	/simnet/common/include/global/mass_stdc.h
ps	register INT_4	/simnet/common/include/global/mass_stdc.h
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Unsuccessful.
num_obstacles	*int*	Successful. Number of obstacles.
Calls		
Function	**Where Described**	
tdb_get_terrain	See Section 2.21.7.18.1.	
count_objects_in_patch	See Section 2.21.7.26.1.	

Table 2.21-236 tdb_object_count Information.

2.21.7.26.3 get_nth_object_in_patch

get_nth_object_in_patch is called to determine if the patch at the given *location* with the given *radius* contains the nth object, counting from *start*. If it does, *nth_object* will be a pointer to the OBJECT_DESCRIPTOR, and the function returns 0. Otherwise, *nth_object* will be null and the function returns *start* plus the number of objects in the patch. The function call is get_nth_object_in_patch(objects, num_objects, location, radius, start, n, nth_object). Table 2.21-237 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
objects	register pointer to OBJECT_DESCRIPTOR	/simnet/common/libsrc/libtdb/terrain.h
num_objects	register INT_4	/simnet/common/include/global/mass_stdc.h
location	register pointer to TDB_POINT	/simnet/common/libsrc/libtdb/tdb.h
radius	REAL_8	/simnet/common/include/global/mass_stdc.h
start	register INT_4	/simnet/common/include/global/mass_stdc.h
n	register INT_4	/simnet/common/include/global/mass_stdc.h
nth_object	pointer to OBJECT_DESCRIPTOR	/simnet/common/libsrc/libtdb/terrain.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	register INT_4	/simnet/common/include/global/mass_stdc.h
Return Values		
Return Value	**Type**	**Meaning**
start	int	0 or number of objects in patch.
Calls		
Function	**Where Described**	
GET_OBJECT_MIN_X	Macro defined in /simnet/common/libsrc/libtdb/terrain.h.	
GET_OBJECT_MAX_X	Macro defined in /simnet/common/libsrc/libtdb/terrain.h.	
GET_OBJECT_MIN_Y	Macro defined in /simnet/common/libsrc/libtdb/terrain.h.	
GET_OBJECT_MAX_Y	Macro defined in /simnet/common/libsrc/libtdb/terrain.h.	
Called By		
Function	**Where Described**	
tdb_nth_object	See Section 2.21.7.26.4.	

Table 2.21-237 get_nth_object_in_patch Information.

2.21.7.26.4 tdb_nth_object

tdb_nth_object is called to retrieve the nth object in the area at the given *location* with the given *radius*. If *n* is greater than the number of objects in the area, a null pointer is returned and *tdb_errno* is set to TE_NO_OBJECT. If any other error occurs, the function returns -1 and *tdb_errno* is set accordingly. The function call is tdb_nth_object(location, radius, n, nth_object). Table 2.21-238 describes the parameters used, errors returned and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
location	register pointer to TDB_POINT	/simnet/common/libsrc/libtdb/tdb.h
radius	REAL_8	/simnet/common/include/global/mass_stdc.h
n	register INT_4	/simnet/common/include/global/mass_stdc.h
nth_object	pointer to OBJECT_DESCRIPTOR	/simnet/common/libsrc/libtdb/terrain.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
patch	register pointer to char	Standard C type.
start	register INT_4	/simnet/common/include/global/mass_stdc.h
objects	pointer to OBJECT_DESCRIPTOR	/simnet/common/libsrc/libtdb/terrain.h
num_objects_in_patch	INT_4	/simnet/common/include/global/mass_stdc.h
patch_header	pointer to PATCH_HEADER	/simnet/common/libsrc/libtdb/terrain.h
coord	TDB_POINT	/simnet/common/include/global/mass_stdc.h
start_x	REAL_8	/simnet/common/include/global/mass_stdc.h
start_y	REAL_8	/simnet/common/include/global/mass_stdc.h
bound_x	REAL_8	/simnet/common/include/global/mass_stdc.h
bound_y	REAL_8	/simnet/common/include/global/mass_stdc.h
ps	register INT_4	/simnet/common/include/global/mass_stdc.h
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Unsuccessful.
0	int	Successful.
Errors		
Error Name	**Reason for Error**	
TE_NO_OBJECT	A request for an object could not be filled.	
Calls		
Function	**Where Described**	
tdb_get_terrain	See Section 2.21.7.18.1.	
get_nth_object_in_patch	See Section 2.21.7.26.3.	

Table 2.21-238 tdb_nth_object Information.

2.21.7.26.5 get_closest_object_in_patch

get_closest_object_in_patch is called to fetch the closest object in the patch, within the given *radius*, to the given *location*. If found, the routine returns the distance to the object and *objects* is made to point to it. Otherwise, *objects* will be null and the function returns the original radius. The function call is get_closest_object_in_patch(objects, num_objects, location, radius, close_one, found_one). Table 2.21-239 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
objects	register pointer to OBJECT_DESCRIPTOR	/simnet/common/libsrc/libtdb/terrain.h
num_objects	INT_4	/simnet/common/include/global/mass_stdc.h
location	register pointer to TDB_POINT	/simnet/common/libsrc/libtdb/tdb.h
radius	REAL_8	/simnet/common/include/global/mass_stdc.h
close_one	pointer to OBJECT_DESCRIPTOR	/simnet/common/libsrc/libtdb/terrain.h
found_one	pointer to char	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	register INT_4	/simnet/common/include/global/mass_stdc.h
min_x	register INT_4	/simnet/common/include/global/mass_stdc.h
min_y	register INT_4	/simnet/common/include/global/mass_stdc.h
max_x	register INT_4	/simnet/common/include/global/mass_stdc.h
max_y	register INT_4	/simnet/common/include/global/mass_stdc.h
dx	REAL_8	/simnet/common/include/global/mass_stdc.h
dy	REAL_8	/simnet/common/include/global/mass_stdc.h
dx_squared	REAL_8	/simnet/common/include/global/mass_stdc.h
dy_squared	REAL_8	/simnet/common/include/global/mass_stdc.h
old_distance_squared	REAL_8	/simnet/common/include/global/mass_stdc.h
new_distance_squared	REAL_8	/simnet/common/include/global/mass_stdc.h
Return Values		
Return Value	**Type**	**Meaning**
radius	REAL_8	Distance to object or original radius.

Calls	
Function	**Where Described**
GET_OBJECT_MIN_X	Macro defined in /simnet/common/libsrc/libtdb/terrain.h.
GET_OBJECT_MIN_Y	Macro defined in /simnet/common/libsrc/libtdb/terrain.h.
GET_OBJECT_MAX_X	Macro defined in /simnet/common/libsrc/libtdb/terrain.h.
GET_OBJECT_MAX_Y	Macro defined in /simnet/common/libsrc/libtdb/terrain.h.
Called By	
Function	**Where Described**
tdb_close_object	See Section 2.21.7.26.6.

Table 2.21-239 get_closest_object_in_patch Information.

2.21.7.26.6 tdb_close_object

tdb_close_object is called to retrieve the closest object in the area at the given *location* with the given *radius*. If there are no objects in the area, *closest_one* is set to the null pointer, -1 is returned and *tdb_errno* is set to TE_NO_OBJECT. Otherwise, *closest_one* is set to point to the object and the distance to the object is returned. If any other error occurs, the function returns -1 and *tdb_errno* is set accordingly. The function call is tdb_close_object(location, radius, closest_one). Table 2.21-240 describes the parameters used, errors returned and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
location	register pointer to TDB_POINT	/simnet/common/libsrc/libtdb/tdb.h
radius	REAL_8	/simnet/common/include/global/mass_stdc.h
closest_one	pointer to OBJECT_DESCRIPTOR	/simnet/common/libsrc/libtdb/terrain.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
patch	register pointer to char	Standard C type.
objects	register pointer to OBJECT_DESCRIPTOR	/simnet/common/libsrc/libtdb/terrain.h
patch_header	pointer to PATCH_HEADER	/simnet/common/libsrc/libtdb/terrain.h
num_objects	INT_4	/simnet/common/include/global/mass_stdc.h
coord	TDB_POINT	/simnet/common/libsrc/libtdb/tdb.h
start_x	REAL_8	/simnet/common/include/global/mass_stdc.h
start_y	REAL_8	/simnet/common/include/global/mass_stdc.h
bound_x	REAL_8	/simnet/common/include/global/mass_stdc.h
bound_y	REAL_8	/simnet/common/include/global/mass_stdc.h
ps	register INT_4	/simnet/common/include/global/mass_stdc.h
found_one	char	Standard C type.
tmp_found	char	Standard C type.

Return Values		
Return Value	**Type**	**Meaning**
radius	int	Distance to closest object.
-1	int	Unsuccessful.
Errors		
Error Name	**Reason for Error**	
TE_NO_OBJECT	A request for an object could not be filled.	
Calls		
Function	**Where Described**	
tdb_get_terrain	See Section 2.21.7.18.1.	
get_closest_object_in_patch	See Section 2.21.7.26.5.	
Called By		
Function	**Where Described**	
tdb_close_thing	See Section 2.21.7.28.1.	

Table 2.21-240 tdb_close_object Information.

2.21.7.26.7 get_obstr_object_in_patch

get_obstr_object_in_patch is called to fetch the closest object in the patch, with a height of at least *min_height*, which after *expansion* is on the line from *start* to *end*. If found, the routine returns 1 and *result* is made to point to it. Otherwise, *result* will be null and the function returns 0. The function call is get_obstr_object_in_patch(objects, num_objects, start, end, min_height, expansion, result, distance_squared). Table 2.21-241 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
objects	register pointer to OBJECT_DESCRIPTOR	/simnet/common/libsrc/libtdb/terrain.h
num_objects	INT_4	/simnet/common/include/global/mass_stdc.h
start	register pointer to TDB_POINT	/simnet/common/libsrc/libtdb/tdb.h
end	register pointer to TDB_POINT	/simnet/common/libsrc/libtdb/tdb.h
min_height	REAL_8	/simnet/common/include/global/mass_stdc.h
expansion	REAL_8	/simnet/common/include/global/mass_stdc.h
result	pointer to OBJECT_DESCRIPTOR	/simnet/common/libsrc/libtdb/terrain.h
distance_squared	pointer to REAL_4	/simnet/common/include/global/mass_stdc.h

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	register INT_4	/simnet/common/include/global/mass_stdc.h
min_x	register INT_4	/simnet/common/include/global/mass_stdc.h
min_y	register INT_4	/simnet/common/include/global/mass_stdc.h
max_x	register INT_4	/simnet/common/include/global/mass_stdc.h
max_y	register INT_4	/simnet/common/include/global/mass_stdc.h
lower_left	TDB_POINT	/simnet/common/libsrc/libtdb/tdb.h
upper_right	TDB_POINT	/simnet/common/libsrc/libtdb/tdb.h
found_one	int	Standard C type.
tmp_distance_squared	REAL_8	/simnet/common/include/global/mass_stdc.h

Return Values		
Return Value	**Type**	**Meaning**
1	int	Successful.
0	int	Unsuccessful.

Calls	
Function	**Where Described**
GET_OBJECT_MIN_X	Macro defined in /simnet/common/libsrc/libtdb/terrain.h.
GET_OBJECT_MIN_Y	Macro defined in /simnet/common/libsrc/libtdb/terrain.h.
GET_OBJECT_MAX_X	Macro defined in /simnet/common/libsrc/libtdb/terrain.h.
GET_OBJECT_MAX_Y	Macro defined in /simnet/common/libsrc/libtdb/terrain.h.
rectangle_intersected	See Section 2.21.7.26.9.
object_intersected	See Section 2.21.7.26.10.

Called By	
Function	**Where Described**
tdb_obstr_object	See Section 2.21.7.26.8.

Table 2.21-241 get_obstr_object_in_patch Information.

2.21.7.26.8 tdb_obstr_object

tdb_obstr_object is called to retrieve the closest object on the line from *start* to *end* after *expansion* with a height of at least *min_height*. If no such object is found, *object* is set to the null pointer, -1 is returned and *tdb_errno* is set to TE_NO_OBJECT. Otherwise, the object pointed to by *object* is set to the object found and 0 is returned. If an error occurs, the function returns -1 and *tdb_errno* is set accordingly. The function call is tdb_obstr_object(start, end, min_height, expansion, object). Table 2.21-242 describes the parameters used, errors returned and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
start	register pointer to TDB_POINT	/simnet/common/libsrc/libtdb/tdb.h
end	register pointer to TDB_POINT	/simnet/common/libsrc/libtdb/tdb.h
min_height	REAL_8	/simnet/common/include/global/mass_stdc.h
expansion	REAL_8	/simnet/common/include/global/mass_stdc.h
object	pointer to OBJECT_DESCRIPTOR	/simnet/common/libsrc/libtdb/terrain.h

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
patch	register pointer to char	Standard C type.
objects	register pointer to OBJECT_DESCRIPTOR	/simnet/common/libsrc/libtdb/terrain.h
tmp_object	OBJECT_DESCRIPTOR	/simnet/common/libsrc/libtdb/terrain.h
patch_header	pointer to PATCH_HEADER	/simnet/common/libsrc/libtdb/terrain.h
num_objects	INT_4	/simnet/common/include/global/mass_stdc.h
coord	TDB_POINT	/simnet/common/libsrc/libtdb/tdb.h
patch_lower_left	TDB_POINT	/simnet/common/libsrc/libtdb/tdb.h
patch_upper_right	TDB_POINT	/simnet/common/libsrc/libtdb/tdb.h
start_x	REAL_8	/simnet/common/include/global/mass_stdc.h
start_y	REAL_8	/simnet/common/include/global/mass_stdc.h
bound_x	REAL_8	/simnet/common/include/global/mass_stdc.h
bound_y	REAL_8	/simnet/common/include/global/mass_stdc.h
ps	register INT_4	/simnet/common/include/global/mass_stdc.h
found_one	char	Standard C type.
tmp_found	char	Standard C type.
distance_squared	REAL_4	/simnet/common/include/global/mass_stdc.h
tmp_distance_squared	REAL_4	/simnet/common/include/global/mass_stdc.h

Return Values		
Return Value	**Type**	**Meaning**
0	int	Successful.
-1	int	Unsuccessful.

Errors	
Error Name	**Reason for Error**
TE_NO_OBJECT	A request for an object could not be filled.
Calls	
Function	**Where Described**
rectangle_intersected	See Section 2.21.7.26.9.
tdb_get_terrain	See Section 2.21.7.18.1.
get_obstr_object_in_patch	See Section 2.21.7.26.7.

Table 2.21-242 tdb_obstr_object Information.

2.21.7.26.9 rectangle_intersected

rectangle_intersected checks to see if a line crosses through a rectangle using a Cohen-Sutherland clipper. It returns 1 if intersection occurs, 0 otherwise. The *distance_squared* variable is filled in with the square of the distance from the left-center of the box to the closest intersecting point if the *check_distance* flag is set (this is useful for finding the "closest" clipped line segment). The function call is rectangle_intersected(start, end, lower_left, upper_right, check_distance, distance_squared). Table 2.21-243 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
start	register pointer to TDB_POINT	/simnet/common/libsrc/libtdb/tdb.h
end	register pointer to TDB_POINT	/simnet/common/libsrc/libtdb/tdb.h
lower_left	register pointer to TDB_POINT	/simnet/common/libsrc/libtdb/tdb.h
upper_right	register pointer to TDB_POINT	/simnet/common/libsrc/libtdb/tdb.h
check_distance	unsigned int	Standard C type.
distance_squared	pointer to float	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
outcode1	register unsigned int	Standard C type.
outcode2	register unsigned int	Standard C type.
intersected	unsigned int	Standard C type.
dx	float	Standard C type.
dy	float	Standard C type.
x1	float	Standard C type.
y1	float	Standard C type.
x2	float	Standard C type.
y2	float	Standard C type.
xmin	float	Standard C type.
ymin	float	Standard C type.
xmax	float	Standard C type.
ymax	float	Standard C type.
center_y	float	Standard C type.
temp_coord	float	Standard C type.
tmp_distance_squared	float	Standard C type.
temp_outcode	int	Standard C type.

Return Values		
Return Value	**Type**	**Meaning**
1	int	Intersection occurs.
0	int	Intersection does not occur.

Called By	
Function	**Where Described**
get_obstr_object_in_patch	See Section 2.21.7.26.7.
tdb_obstr_object	See Section 2.21.7.26.8.
object_intersected	See Section 2.21.7.28.10.

Table 2.21-243 rectangle_intersected Information.

2.21.7.26.10 object_intersected

object_intersected checks to see if a line, expanded into a rectangle by *expansion*, crosses through an *object*. The rectangle is rotated so that it is parallel to the x-axis and then a modified Sutherland-Cohen is applied. This modified version finds the *distance_ squared* from the original *start* point to the point of intersection. The distance to the closest such point is returned. The routine returns 1 if intersection occurs, 0 otherwise. The function call is object_intersected(start, end, object, expansion, distance_squared). Table 2.21-244 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
start	register pointer to TDB_POINT	/simnet/common/libsrc/libtdb/tdb.h
end	register pointer to TDB_POINT	/simnet/common/libsrc/libtdb/tdb.h
object	pointer to OBJECT_DESCRIPTOR	/simnet/common/libsrc/libtdb/terrain.h
expansion	REAL_8	/simnet/common/include/global/mass_stdc.h
distance_squared	pointer to REAL_8	/simnet/common/include/global/mass_stdc.h

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	register int	Standard C type.
j	register int	Standard C type.
edge_intersects	int	Standard C type.
x_to_origin	float	Standard C type.
y_to_origin	float	Standard C type.
sin_theta	float	Standard C type.
cos_theta	float	Standard C type.
translated_end_x	float	Standard C type.
translated_end_y	float	Standard C type.
hypotenuse	float	Standard C type.
tmp_distance_squared	float	Standard C type.
new_object_vertices[4]	TDB_POINT	/simnet/common/libsrc/libtdb/tdb.h
translated_vertices[4]	TDB_POINT	/simnet/common/libsrc/libtdb/tdb.h
lower_left	TDB_POINT	/simnet/common/libsrc/libtdb/tdb.h
upper_right	TDB_POINT	/simnet/common/libsrc/libtdb/tdb.h
Return Values		
Return Value	**Type**	**Meaning**
1	int	Intersection occurs.
0	int	Intersection does not occur.
Calls		
Function	**Where Described**	
rectangle_intersected	See Section 2.21.7.26.9.	
Called By		
Function	**Where Described**	
get_obstr_object_in_patch	See Section 2.21.7.26.7.	

Table 2.21-244 object_intersected Information.

2.21.7.27 tdb_init.c

/simnet/common/libsrc/libtdb/tdb_init.c

tdb_init.c contains routines for initializing and terminating the tdb terrain caching system.

2.21.7.27.1 tdb_init_cache

tdb_init_cache initializes the terrain caching system. It opens the database file specified, reads the database indices and header, initializes the tdb_info structure, and allocates and locks in enough space to cache up to the number of patches given. The function returns -1 and the global terrain error variable *tdb_errno* is appropriately set if initialization fails for any reason. Otherwise, 0 is returned. The function call is tdb_init_cache(pathname, number_of_patches_in_cache). Table 2.21-245 describes the parameters used, errors returned and functions called using this function

Parameters		
Parameter	**Type**	**Where Typedef Declared**
pathname	register pointer to char	Standard C type.
number_of_patches_in_cache	INT_4	/simnet/common/include/global/mass_stdc.h
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Unsuccessful.
0	int	Successful.
Errors		
Error Name	**Reason for Error**	
TE_BAD_FORMAT	Database has wrong format for this version of libtdb.	
Calls		
Function	**Where Described**	
tdb_get_tdb_info	See Section 2.21.7.27.3.	
tdb_right_format	See Section 2.21.7.31.4.	
init_object_and_texture_names	See Section 2.21.7.15.21.	
tdb_set_dumpfile	See Section 2.21.7.15.1.	
cache_init	See Section 2.21.7.7.1.	

Table 2.21-245 tdb_init_cache Information.

2.21.7.27.2 tdb_terminate

tdb_terminate terminates the data base access system (either the cache system or direct memory access system), frees up the data retrieval space, and closes all relevant files. The function call is tdb_terminate(). Table 2.21-246 describes the functions called using this function.

Return Values		
Return Value	**Type**	**Meaning**
cache_and_file_terminate()	int	Terminate the cache system.
memory_terminate()	int	Terminate the direct memory access system.
Calls		
Function	**Where Described**	
cache_and_file_terminate	See Section 2.21.7.7.2.	
memory_terminate	See Section 2.21.7.25.3.	

Table 2.21-246 tdb_terminate Information.

2.21.7.27.3 tdb_get_tdb_info

tdb_get_tdb_info gets information about the data base from the data base header and file stat size. The function call is tdb_get_tdb_info(pathname, info). Table 2.21-247 describes the parameters used, errors returned and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
pathname	register pointer to char	Standard C type.
info	register pointer to TDB_INFO	/simnet/common/libsrc/libtdb/tdb.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
file_stat	struct stat	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Unsuccessful.
0	int	Successful
Errors		
Error Name	**Reason for Error**	
TE_OPEN_READ	Open or read on the database failed.	
Calls		
Function	**Where Described**	
tdb_read_header	See Section 2.21.7.21.1.	
fstat		
Called By		
Function	**Where Described**	
tdb_init_memory	See Section 2.21.7.25.1.	
tdb_init_cache	See Section 2.21.7.27.1.	

Table 2.21-247 tdb_get_tdb_info Information.

2.21.7.28 things.c

/simnet/common/libsrc/libtdb/things.c

things.c contains routines for retrieving a copy of the closest thing (object, tree, or treeline) in a patch.

2.21.7.28.1 tdb_close_thing

tdb_close_thing determines if the patch at the given location within the given *radius* contains an object, tree, or treeline. If it does, *thing* will become a pointer to the closest thing found and the function returns the distance to it. *Flags* specifies which of the three types of things is to be considered, either singly or combined. The function returns -1 if an error occurs or if nothing is located. If an error occurs, the global terrain error variable

tdb_errno is set. The function call is tdb_close_thing(coord, radius, thing, flags). Table 2.21-248 describes the parameters used, errors returned and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
coord	pointer to TDB_POINT	/simnet/common/libsrc/libtdb/tdb.h
radius	REAL_8	/simnet/common/include/global/mass_stdc.h
thing	pointer to TDB_THING	/simnet/common/libsrc/libtdb/tdb.h
flags	unsigned char	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
tmp_distance	INT_4	/simnet/common/include/global/mass_stdc.h
distance	INT_4	/simnet/common/include/global/mass_stdc.h
object	OBJECT_DESCRIPTOR	/simnet/common/libsrc/libtdb/terrain.h
tree	TREE_DESCRIPTOR	/simnet/common/libsrc/libtdb/terrain.h
treeline	TREELINE_HEADER	/simnet/common/libsrc/libtdb/terrain.h
nothing	int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
-1	INT_4	Unsuccessful.
distance	INT_4	Distance to thing.
Errors		
Error Name	**Reason for Error**	
TE_NO_THING	A request for a thing could not be filled.	
Calls		
Function	**Where Described**	
tdb_close_object	See Section 2.21.7.26.6.	
tdb_close_tree	See Section 2.21.7.30.6.	
tdb_close_trline	See Section 2.21.7.29.6.	

Table 2.21-248 tdb_close_thing Information.

2.21.7.28.2 tdb_thing_string

tdb_thing_string returns the name string of *thing*, which is either an object, tree, treeline, or canopy. The function call is tdb_thing_string(thing, string, string_len). Table 2.21-249 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
thing	pointer to TDB_THING	/simnet/common/libsrc/libtdb/tdb.h
string	pointer to char	Standard C type.
string_len	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
type	int	Standard C type.
objs	pointer to pointer to char	Standard C type.
tex	pointer to pointer to char	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
string	pointer to char	The name string of *thing*.
Calls		
Function	**Where Described**	
get_object_name_list	See Section 2.21.7.15.23.	
GET_OBJECT_TYPE	Macro defined in /simnet/common/libsrc/libtdb/terrain.h.	
get_texture_name_list	See Section 2.21.7.15.22.	

Table 2.21-249 tdb_thing_string Information.

2.21.7.29 treelines.c

/simnet/libsrc/libtdb/treelines.c

2.21.7.29.1 count_treelines_in_patch

count_treelines_in_patch is called to determine if the area at the given *location* with the given *radius* is obstructed by any *treelines* in the given patch. If the area is not obstructed, 0 is returned. Otherwise, the number of obstructing treelines is returned. The function call is count_treelines_in_patch(treelines, num_treelines, location, radius). Table 2.21-250 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
treelines	register pointer to TREELINE_HEADER	/simnet/common/libsrc/libtdb/terrain.h
num_treelines	INT_4	/simnet/common/include/global/mass_stdc.h
location	register pointer to TDB_POINT	/simnet/common/libsrc/libtdb/tdb.h
radius	REAL_8	

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	register INT_4	/simnet/common/include/global/mass_stdc.h
num_obstacles	register INT_4	/simnet/common/include/global/mass_stdc.h
vertices	register pointer to TERRAIN_POINT	/simnet/common/libsrc/libtdb/terrain.h
Return Values		
Return Value	**Type**	**Meaning**
num_obstacles	int	Number of obstructing treelines.
Called By		
Function	**Where Described**	
tdb_trline_count	See Section 2.21.7.29.2.	

Table 2.21-250 count_treelines_in_patch Information.

2.21.7.29.2 tdb_trline_count

tdb_trline_count is called to determine if the area at the given *location* with the given *radius* is obstructed by any treelines. If the area is not obstructed, 0 is returned. Otherwise, the number of obstructing treelines is returned. If an error occurs, -1 is returned and the global terrain error variable *tdb_errno* is set. The function call is tdb_trline_count(location, radius). Table 2.21-251 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
location	register pointer to TDB_POINT	/simnet/common/libsrc/libtdb/tdb.h
radius	REAL_8	/simnet/common/include/global/mass_stdc.h

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
patch	register pointer to char	Standard C type.
num_obstacles	register INT_4	/simnet/common/include/global/mass_stdc.h
patch_header	pointer to PATCH_HEADER	/simnet/common/libsrc/libtdb/terrain.h
treelines	pointer to TREELINE_HEADER	/simnet/common/libsrc/libtdb/terrain.h
num_treelines_in_patch	INT_4	/simnet/common/include/global/mass_stdc.h
coord	TDB_POINT	/simnet/common/libsrc/libtdb/tdb.h
start_x	REAL_8	/simnet/common/include/global/mass_stdc.h
start_y	REAL_8	/simnet/common/include/global/mass_stdc.h
bound_x	REAL_8	/simnet/common/include/global/mass_stdc.h
bound_y	REAL_8	/simnet/common/include/global/mass_stdc.h
ps	register INT_4	/simnet/common/include/global/mass_stdc.h

Return Values		
Return Value	**Type**	**Meaning**
-1	int	Unsuccessful.
num_obstacles	int	Number of obstructing treelines.

Calls	
Function	**Where Described**
tdb_get_terrain	See Section 2.21.7.18.1.
count_treelines_in_patch	See Section 2.21.7.29.1.

Table 2.21-251 tdb_trline_count Information.

2.21.7.29.3 get_nth_treeline_in_patch

get_nth_treeline_in_patch is called to determine if the patch at the given *location* with the given *radius* contains the *n*th treeline, counting from *start*. If it does, *nth_treeline* will be a pointer to the TREELINE_HEADER and the function returns 0. Otherwise, *nth_treeline* will be null and the function returns start plus the number of treelines in the patch. The function call is get_nth_treeline_in_patch(treelines, num_treelines, location, radius, atart, n, nth_treeline). Table 2.21-252 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
treelines	register pointer to TREELINE_HEADER	/simnet/common/libsrc/libtdb/terrain.h
num_treelines	register INT_4	/simnet/common/include/global/mass_stdc.h
location	register pointer to TDB_POINT	/simnet/common/libsrc/libtdb/tdb.h
radius	REAL_8	/simnet/common/include/global/mass_stdc.h
start	register INT_4	/simnet/common/include/global/mass_stdc.h
n	register INT_4	/simnet/common/include/global/mass_stdc.h
nth_treeline	pointer to TREELINE_HEADER	/simnet/common/libsrc/libtdb/terrain.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	register INT_4	/simnet/common/include/global/mass_stdc.h
vertices	register pointer to TERRAIN_POINT	/simnet/common/libsrc/libtdb/terrain.h
Return Values		
Return Value	**Type**	**Meaning**
start	int	Number of treelines in patch.
Called By		
Function	**Where Described**	
tdb_nth_trline	See Section 2.21.7.29.4.	

Table 2.21-252 get_nth_treeline_in_patch Information.

2.21.7.29.4 tdb_nth_trline

tdb_nth_trline is called to retrieve the *n*th treeline in the area at the given *location* with the given *radius*. If *n* is greater than the number of treelines in the area, a null pointer is returned and *tdb_errno* is set to TE_NO_TREELINE. If any other error occurs, the function returns -1 and *tdb_errno* is set accordingly. The function call is tdb_nth_trline(location, radius, n, nth_treeline). Table 2.21-253 describes the parameters used, errors returned and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
location	register pointer to TDB_POINT	/simnet/common/libsrc/libtdb/tdb.h
radius	REAL8	/simnet/common/include/global/mass_stdc.h
n	register INT_4	/simnet/common/include/global/mass_stdc.h
nth_treeline	pointer to TREELINE_HEADER	/simnet/common/libsrc/libtdb/terrain.h

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
patch	register pointer to char	Standard C type.
start	register INT_4	/simnet/common/include/global/mass_stdc.h
treelines	pointer to TREELINE_HEADER	/simnet/common/libsrc/libtdb/terrain.h
num_treelines_in_patch	INT_4	/simnet/common/include/global/mass_stdc.h
patch_header	pointer to PATCH_HEADER	/simnet/common/libsrc/libtdb/terrain.h
coord	TDB_POINT	/simnet/common/libsrc/libtdb/tdb.h
start_x	REAL_8	/simnet/common/include/global/mass_stdc.h
start_y	REAL_8	/simnet/common/include/global/mass_stdc.h
bound_x	REAL_8	/simnet/common/include/global/mass_stdc.h
bound_y	REAL_8	/simnet/common/include/global/mass_stdc.h
ps	register INT_4	/simnet/common/include/global/mass_stdc.h
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Unsuccessful.
0	int	Successful.
Errors		
Error Name	**Reason for Error**	
TE_NO_TREELINE	A request for a treeline could not be filled.	
Calls		
Function	**Where Described**	
tdb_get_terrain	See Section 2.21.7.18.1.	
get_nth_treeline_in_patch	See Section 2.21.7.29.3.	

Table 2.21-253 tdb_nth_trline Information.

2.21.7.29.5 get_closest_treeline_in_patch

get_closest_treeline_in_patch is called to fetch the closest treeline in the patch, within the given *radius*, to the given *location*. If found, the routine returns the distance to the treeline and *treelines* is made to point to it. Otherwise, *treelines* will be null and the function returns the original radius. The function call is get_closest_treeline_in_patch(treelines, num_treelines, location, radius, close_one, found_one). Table 2.21-254 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
treelines	register pointer to TREELINE_HEADER	/simnet/common/libsrc/libtdb/terrain.h
num_treelines	INT_4	/simnet/common/include/global/mass_stdc.h
location	register pointer to TDB_POINT	/simnet/common/libsrc/libtdb/tdb.h
radius	REAL_8	/simnet/common/include/global/mass_stdc.h
close_one	pointer to TREELINE_HEADER	/simnet/common/libsrc/libtdb/terrain.h
found_one	pointer to char	Standard C type.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	register INT_4	/simnet/common/include/global/mass_stdc.h
j	register INT_4	/simnet/common/include/global/mass_stdc.h
min_x	register INT_4	/simnet/common/include/global/mass_stdc.h
min_y	register INT_4	/simnet/common/include/global/mass_stdc.h
max_x	register INT_4	/simnet/common/include/global/mass_stdc.h
max_y	register INT_4	/simnet/common/include/global/mass_stdc.h
dx	REAL_8	/simnet/common/include/global/mass_stdc.h
dy	REAL_8	/simnet/common/include/global/mass_stdc.h
dx_squared	REAL_8	/simnet/common/include/global/mass_stdc.h
dy_squared	REAL_8	/simnet/common/include/global/mass_stdc.h
tmp_distance_squared	REAL_8	/simnet/common/include/global/mass_stdc.h
old_distance_squared	REAL_8	/simnet/common/include/global/mass_stdc.h
new_distance_squared	REAL_8	/simnet/common/include/global/mass_stdc.h
vertices	register pointer to TERRAIN_POINT	/simnet/common/libsrc/libtdb/terrain.h

Return Values		
Return Value	**Type**	**Meaning**
radius	REAL_8	Distance to treeline

Called By	
Function	**Where Described**
tdb_close_trline	See Section 2.21.7.29.6.

Table 2.21-254 get_closest_treeline_in_patch Information.

2.21.7.29.6 tdb_close_trline

tdb_close_trline is called to retrieve the closest treeline in the area at the given *location* with the given *radius*. If there are no treelines in the area, *closest_one* is set to the null pointer, -1 is returned and *tdb_errno* is set to TE_NO_TREELINE. Otherwise, *closest_one* is set to point to the treeline and the distance to the treeline is returned. If any other error occurs, the function returns -1 and *tdb_errno* is set accordingly. The function call is tdb_close_trline(location, radius, closest_one). Table 2.21-255 describes the parameters used, errors returned and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
location	register pointer to TDB_POINT	/simnet/common/libsrc/libtdb/tdb.h
radius	REAL_8	/simnet/common/include/global/mass_stdc.h
closest_one	pointer to TREELINE_HEADER	/simnet/common/libsrc/libtdb/terrain.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
patch	register pointer to char	Standard C type.
treelines	register pointer to TREELINE_HEADER	/simnet/common/libsrc/libtdb/terrain.h
patch_header	pointer to PATCH_HEADER	/simnet/common/libsrc/libtdb/terrain.h
num_treelines	INT_4	/simnet/common/include/global/mass_stdc.h
coord	TDB_POINT	/simnet/common/libsrc/libtdb/tdb.h
start_x	REAL_8	/simnet/common/include/global/mass_stdc.h
start_y	REAL_8	/simnet/common/include/global/mass_stdc.h
bound_x	REAL_8	/simnet/common/include/global/mass_stdc.h
bound_y	REAL_8	/simnet/common/include/global/mass_stdc.h
ps	register INT_4	/simnet/common/include/global/mass_stdc.h
found_one	char	Standard C type.
tmp_found	char	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
radius	int	Distance to treeline.
-1	int	Unsuccessful.
Errors		
Error Name	**Reason for Error**	
TE_NO_TREELINE	A request for a treeline could not be filled	

Calls	
Function	**Where Described**
tdb_get_terrain	See Section 2.21.7.18.1.
get_closest_treeline_in_patch	See Section 2.21.7.29.5.
Called By	
Function	**Where Described**
tdb_close_thing	See Section 2.21.7.28.1.

Table 2.21-255 tdb_close_trline Information.

2.21.7.30 trees.c

/simnet/common/libsrc/libtdb/trees.c

2.21.7.30.1 count_trees_in_patch

count_trees_in_patch is called to determine if the area at the given *location* with the given *radius* is obstructed by any *trees* in the given patch. If the area is not obstructed 0 is returned. Otherwise, the number of obstructing trees is returned. The function call is count_trees_in_patch(trees, num_trees, location, radius). Table 2.21-256 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
trees	register pointer to TREE_DESCRIPTOR	/simnet/common/libsrc/libtdb/terrain.h
num_trees	INT_4	/simnet/common/include/global/mass_stdc.h
location	register pointer to TDB_POINT	/simnet/common/libsrc/libtdb/tdb.h
radius	REAL_8	/simnet/common/include/global/mass_stdc.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	register INT_4	/simnet/common/include/global/mass_stdc.h
num_obstacles	register INT_4	/simnet/common/include/global/mass_stdc.h
Return Values		
Return Value	**Type**	**Meaning**
num_obstacles	int	Number of obstructing trees.
Called By		
Function	**Where Described**	
tdb_tree_count	See Section 2.21.7.30.2.	

Table 2.21-256 count_trees_in_patch Information.

2.21.7.30.2 tdb_tree_count

tdb_tree_count is called to determine if the area at the given *location* with the given *radius* is obstructd by any trees. If the area is not obstructed 0 is returned. Otherwise, the number of obstructing trees is returned. If an error occurs, the function returns -1 and sets the global terrain error variable. The function call is tdb_tree_count(location, radius). Table 2.21-257 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
location	register pointer to TDB_POINT	/simnet/common/libsrc/libtdb/tdb.h
radius	REAL_8	/simnet/common/include/global/mass_stdc.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
patch	register pointer to char	Standard C type.
num_obstacles	register INT_4	/simnet/common/include/global/mass_stdc.h
patch_header	pointer to PATCH_HEADER	/simnet/common/libsrc/libtdb/terrain.h
trees	pointer to TREE_DESCRIPTOR	/simnet/common/libsrc/libtdb/terrain.h
num_trees_in_patch	INT_4	/simnet/common/include/global/mass_stdc.h
coord	TDB_POINT	/simnet/common/libsrc/libtdb/tdb.h
start_x	REAL_8	/simnet/common/include/global/mass_stdc.h
start_y	REAL_8	/simnet/common/include/global/mass_stdc.h
bound_x	REAL_8	/simnet/common/include/global/mass_stdc.h
bound_y	REAL_8	/simnet/common/include/global/mass_stdc.h
ps	register INT_4	/simnet/common/include/global/mass_stdc.h
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Unsuccessful.
num_obstacles	int	Number of obstructing trees.
Calls		
Function	**Where Described**	
tdb_get_terrain	See Section 2.21.7.18.1.	
count_trees_in_patch	See Section 2.21.7.30.1.	

Table 2.21-257 tdb_tree_count Information.

2.21.7.30.3 get_nth_tree_in_patch

get_nth_tree_in_patch is called to determine if the patch at the given *location* with the given *radius* contains the *n*th tree, counting from *start*. If it does, *nth_tree* will be a pointer to the TREE_DESCRIPTOR and the function returns 0. Otherwise, *nth_tree* wil be null and the function returns start plus the number of trees in the patch. The function call is get_nth_tree_in_patch(trees, num_trees, location, radius, start, n, nth_tree). Table 2.21-258 describes the parameters used by using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
trees	register pointer to TREE_DESCRIPTOR	/simnet/common/libsrc/libtdb/terrain.h
num_trees	register INT_4	/simnet/common/include/global/mass_stdc.h
location	register pointer to TDB_POINT	/simnet/common/libsrc/libtdb/tdb.h
radius	REAL_8	/simnet/common/include/global/mass_stdc.h
start	register INT_4	/simnet/common/include/global/mass_stdc.h
n	register INT_4	/simnet/common/include/global/mass_stdc.h
nth_tree	pointer to TREE_DESCRIPTOR	/simnet/common/libsrc/libtdb/terrain.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	register INT_4	/simnet/common/include/global/mass_stdc.h
Return Values		
Return Value	**Type**	**Meaning**
start	int	Number of trees in patch or the nth tree.
Called By		
Function	**Where Described**	
tdb_nth_tree	See Section 2.21.7.30.4.	

Table 2.21-258 get_nth_tree_in_patch Information.

2.21.7.30.4 tdb_nth_tree

tdb_nth_tree is called to retrieve the *n*th tree in the area at the given *location* with the given *radius*. If *n* is greater than the number of trees in the area, a null pointer is returned and *tdb_errno* is set to TE_NO_TREE. If any other error occurs, the function returns -1 and *tdb_errno* is set accordingly. The function call is tdb_nth_tree(location, radius, n, nth_tree). Table 2.21-259 describes the parameters used, errors returned and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
location	register pointer to TDB_POINT	/simnet/common/libsrc/libtdb/tdb.h
radius	REAL_8	/simnet/common/include/global/mass_stdc.h
n	register INT_4	/simnet/common/include/global/mass_stdc.h
nth_tree	pointer to TREE_DESCRIPTOR	/simnet/common/libsrc/libtdb/terrain.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
patch	register pointer to char	Standard C type.
start	register INT_4	/simnet/common/include/global/mass_stdc.h
trees	pointer to TREE_DESCRIPTOR	/simnet/common/libsrc/libtdb/terrain.h
num_trees_in_patch	INT_4	/simnet/common/include/global/mass_stdc.h
patch_header	pointer to PATCH_HEADER	/simnet/common/libsrc/libtdb/terrain.h
coord	TDB_POINT	/simnet/common/libsrc/libtdb/tdb.h
start_x	REAL_8	/simnet/common/include/global/mass_stdc.h
start_y	REAL_8	/simnet/common/include/global/mass_stdc.h
bound_x	REAL_8	/simnet/common/include/global/mass_stdc.h
bound_y	REAL_8	/simnet/common/include/global/mass_stdc.h
ps	register INT_4	/simnet/common/include/global/mass_stdc.h
Return Values		
Return Value	**Type**	**Meaning**
0	int	Successful.
-1	int	Unsuccessful.
Errors		
Error Name	**Reason for Error**	
TE_NO_TREE	A request for a tree could not be filled.	
Calls		
Function	**Where Described**	
tdb_get_terrain	See Section 2.21.7.18.1.	
get_nth_tree_in_patch	See Section 2.21.7.30.3.	

Table 2.21-259 tdb_nth_tree Information.

2.21.7.30.5 get_closest_tree_in_patch

get_closest_tree_in_patch is called to fetch the closest tree in the patch, within the given *radius*, to the given *location*. If found, the routine returns the distance to the tree and *tree* is made to point to it. Otherwise, *tree* will be null and the function returns the original radius. The function call is get_closest_tree_in_patch(tree, num_trees, location, radius, close_one, found_one). Table 2.21-260 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
trees	register pointer to TREE_DESCRIPTOR	/simnet/common/libsrc/libtdb/terrain.h
num_trees	INT_4	/simnet/common/include/global/mass_stdc.h
location	register pointer to TDB_POINT	/simnet/common/libsrc/libtdb/tdb.h
radius	REAL_8	/simnet/common/include/global/mass_stdc.h
close_one	pointer to TREE_DESCRIPTOR	/simnet/common/libsrc/libtdb/terrain.h
found_one	pointer to char	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	register INT_4	/simnet/common/include/global/mass_stdc.h
min_x	register REAL_8	/simnet/common/include/global/mass_stdc.h
min_y	register REAL_8	/simnet/common/include/global/mass_stdc.h
max_x	register REAL_8	/simnet/common/include/global/mass_stdc.h
max_y	register REAL_8	/simnet/common/include/global/mass_stdc.h
dx	REAL_8	/simnet/common/include/global/mass_stdc.h
dy	REAL_8	/simnet/common/include/global/mass_stdc.h
dx_squared	REAL_8	/simnet/common/include/global/mass_stdc.h
dy_squared	REAL_8	/simnet/common/include/global/mass_stdc.h
old_distance_squared	REAL_8	/simnet/common/include/global/mass_stdc.h
new_distance_squared	REAL_8	/simnet/common/include/global/mass_stdc.h
Return Values		
Return Value	**Type**	**Meaning**
radius	REAL_8	Original radius or distance to closest tree.

Called By	
Function	**Where Described**
tdb_close_tree	See Section 2.21.7.30.6.

Table 2.21-260 get_closest_tree_in_patch Information.

2.21.7.30.6 tdb_close_tree

tdb_close_tree is called to retrieve the closest tree in the area at the given *location* with the given *radius*. If there are no trees in the area, *closest_one* is set to the null pointer, -1 is returned and *tdb_errno* is set to TE_NO_TREE. Otherwise, *closest_one* is set to point to the tree and the distance to the tree is returned. If any other error occurs, the function returns -1 and *tdb_errno* is set accordingly. The function call is tdb_close_tree(locaiton, radius, closest_one). Table 2.21-261 describes the parameters used, errors returned and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
location	pointer to TDB_POINT	/simnet/common/libsrc/libtdb/tdb.h
radius	REAL_8	/simnet/common/include/global/mass_stdc.h
closest_one	pointer to TREE_DESCRIPTOR	/simnet/common/libsrc/libtdb/terrain.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
patch	register pointer to char	Standard C type.
trees	register pointer to TREE_DESCRIPTOR	/simnet/common/libsrc/libtdb/terrain.h
patch_header	pointer to PATCH_HEADER	/simnet/common/libsrc/libtdb/terrain.h
num_trees	INT_4	/simnet/common/include/global/mass_stdc.h
coord	TDB_POINT	/simnet/common/libsrc/libtdb/tdb.h
start_x	REAL_8	/simnet/common/include/global/mass_stdc.h
start_y	REAL_8	/simnet/common/include/global/mass_stdc.h
bound_x	REAL_8	/simnet/common/include/global/mass_stdc.h
bound_y	REAL_8	/simnet/common/include/global/mass_stdc.h
ps	register INT_4	/simnet/common/include/global/mass_stdc.h
found_one	char	Standard C type.
tmp_found	char	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
radius	int	Distance to closest tree.
-1	int	Unsuccessful.

Errors	
Error Name	**Reason for Error**
TE_NO_TREE	A request for a tree could not be filled.
Calls	
Function	**Where Described**
tdb_get_terrain	See Section 2.21.7.18.1.
get_closest_tree_in_patch	See Section 2.21.7.30.5.
Called By	
Function	**Where Described**
tdb_close_thing	See Section 2.21.7.28.1.

Table 2.21-261 tdb_close_tree Information.

2.21.7.31 version.c

/simnet/common/libsrc/libtdb/version.c

version.c contains routines for accessing the version numbers of libtdb and any database it accesses (with format >= v3.01).

2.21.7.31.1 tdb_print_version

tdb_print_version prints the terrain database version number. The function call is tdb_print_version(). Table 2.21-262 describes the functions called using this function.

Calls	
Function	**Where Described**
tdb_get_dumpfile	See Section 2.21.7.15.2.

Table 2.21-262 tdb_print_version Information.

2.21.7.31.2 tdb_print_format_compatible

tdb_print_format_compatible prints the database format version number with which this version of libtdb works. The function call is tdb_print_format_compatible(). Table 2.21-263 describes the functions called using this function.

Calls	
Function	**Where Described**
tdb_get_dumpfile	See Section 2.21.7.15.2.

Table 2.21-263 tdb_print_format_compatible Information.

2.21.7.31.3 tdb_print_db_format

tdb_print_db_format prints the database format name. The function call is tdb_print_db_format(). Table 2.21-264 describes the internal variables used and functions called using this function.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
format	array of MAX_DB_FORMAT_LENGTH char	Standard C type.
Calls		
Function	**Where Described**	
tdb_get_dumpfile	See Section 2.21.7.15.2.	
tdb_get_db_format	See Section 2.21.7.31.5.	

Table 2.21-264 tdb_print_db_format Information.

2.21.7.31.4 tdb_right_format

tdb_right_format determines whether the database format is correct. The function call is tdb_right_format(). Table 2.21-265 describes the internal variables used and functions called using this function.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
db_format	array of MAX_DB_FORMAT_LENGTH char	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
FALSE	int	Not correct format.
TRUE	int	Correct format.
Calls		
Function	**Where Described**	
tdb_get_db_format	See Section 2.21.7.31.5.	
Called By		
Function	**Where Described**	
tdb_init_memory	See Section 2.21.7.25.1.	
tdb_init_cache	See Section 2.21.7.27.1.	

Table 2.21-265 tdb_right_format Information.

2.21.7.31.5 tdb_get_db_format

tdb_get_db_format gets the database format. The function call is tdb_get_db_format(buffer). Table 2.21-266 describes the parameters used and errors returned using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
buffer	pointer to char	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
file_desc	int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
NULL	pointer to char	Unsuccessful.
buffer	pointer to char	The database format.
Errors		
Error Name	**Reason for Error**	
TE_OPEN_READ	Open or read on the database failed.	
Called By		
Function	**Where Described**	
tdb_error	See Section 2.21.7.17.1.	
tdb_print_db_format	See Section 2.21.7.31.3.	
tdb_right_format	See Section 2.21.7.31.4.	

Table 2.21-266 tdb_get_db_format Information.

2.22 The Macintosh Libraries and Headers

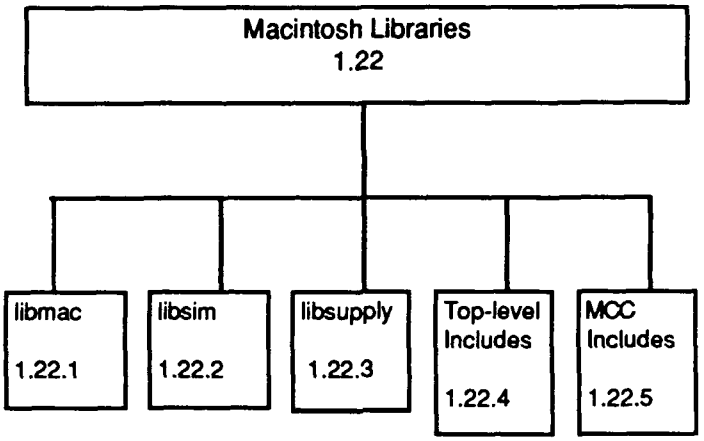


Figure 2.21-1: Macintosh Libraries.

2.22.1 libmac

Folder: "Development:SIMNET:libmac"
This folder contains a library of C functions shared among all MCC Macintosh applications. The library is composed of the following C source code files:

2.22.1.1 libmac.h
Development:SIMNET:libmac:libmac.h

libmac.h defines the interface to many of the functions in the libmac library. Table 2.22-1 describes the variables used by libmac.h.

Variables		
Variable	**Type**	**Where Typedef Declared**
atpSocket	extern int	Standard C type.
hostNode	extern int	Standard C type.
hostSocket	extern int	Standard C type.
application	extern pointer to char	Standard C type.
authors	extern pointer to char	Standard C type.
copyright	extern pointer to char	Standard C type.

Table 2.22-1 libmac.h Variable Information.

2.22.1.2 draw.h
Development:SIMNET:libmac:draw.h

draw.h defines constants that control the proportions of displayed information.

2.22.1.3 atalk.c

Development:SIMNET:libmac:atalk.c

atalk.c contains routines supporting AppleTalk communication with the MCC host. Table 2.22-2 describes the variables used by atalk.c.

Variables		
Variable	**Type**	**Where Typedef Declared**
atpSocket	short	Standard C type.
hostNode	int	Standard C type.
hostSocket	int	Standard C type.
ab	ABRecHandle	Development:THINK C: Mac #includes:Appletalk.h

Table 2.22-2 atalk.c Variable Information.

2.22.1.3.1 SetUpAppleTalk

SetUpAppleTalk initializes the Appletalk communications with the MCC host. The routine returns a handle to the network event that starts up the console. *maxRequestSize* is the maximum number of bytes the request will be. The function call is SetUpAppleTalk(maxRequestSize). Table 2.22-3 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
maxRequestSize	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
addrRcvd	AddrBlock	Development:THINK C: Mac #includes:Appletalk.h
i	int	Standard C type.
errCode	int	Standard C type.
ab	ABRecHandle	Development:THINK C: Mac #includes:Appletalk.h
entity	EntityName	Development:THINK C: Mac #includes:Appletalk.h
hdl	StringHandle	Development:THINK C: Mac #includes:MacTypes.h
dp	DialogPtr	Development:THINK C: Mac #includes:DialogMgr.h
whichDialog	DialogPtr	Development:THINK C: Mac #includes:DialogMgr.h
myEvent	EventRecord	Development:THINK C: Mac #includes:EventMgr.h
Return Values		
Return Value	**Type**	**Meaning**
myEvent.message	Handle	handle to the network event that starts up the console

Calls	
Function	**Where Described**
ATPLoad	Standard Appletalk Manager function for Macintosh.
NBPLoad	Standard Appletalk Manager function for Macintosh.
BigProb	See Section 2.22.1.8.2.
ATPOpenSocket	Standard Appletalk Manager function for Macintosh.
SystemFailure	See Section 2.22.1.8.1.
NewHandle	Standard Memory Manager function for Macintosh.
HLock	Standard Memory Manager function for Macintosh.
NewPtr	Standard Memory Manager function for Macintosh.
HUnlock	Standard Memory Manager function for Macintosh.
ATPGetRequest	Standard Appletalk Manager function for Macintosh.
ATPCloseSocket	Standard Appletalk Manager function for Macintosh.
GetString	Standard Toolbox Utility function for Macintosh.
BlockMove	Standard Memory Manager function for Macintosh.
DisposHandle	Standard Memory Manager function for Macintosh.
NBPRegister	Standard Appletalk Manager function for Macintosh.
GetNewDialog	Standard Dialog Manager function for Macintosh.
LastCaution	See Section 2.22.1.4.3.
SystemTask	Standard Desk Manager function for Macintosh.
GetNextEvent	Standard Toolbox Event Manager function for Macintosh.
ExitToShell	Standard Segment Loader function for Macintosh.
IsDialogEvent	Standard Dialog Manager function for Macintosh.
DialogSelect	Standard Dialog Manager function for Macintosh.
DisposDialog	Standard Dialog Manager function for Macintosh.

Table 2.22-3 SetUpAppleTalk Information.

2.22.1.3.2 SetUpATPRequest

SetUpATPRequest sets up the fields of the ATPRequest record. *ab* is the handle to the Appletalk structure; *userData* is reserved for whatever the user wants; *req* is the pointer to request data; *reqSize* is the size of the request data; *rsp* is the pointer to where response should go; *rspSize* is the maximum size of *rsp*; *xo* indicates whether the transaction should be exactly once or at least once. The function call is SetUpATPRequest(ab, userData, req, reqSize, rsp, rspSize, xo). Table 2.22-4 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
ab	register ABRecHandle	Development:THINK C: Mac #includes:Appletalk.h
userData	long	Standard C type.
req	Ptr	Development:THINK C: Mac #includes:MacTypes.h
reqSize	int	Standard C type.
rsp	Ptr	Development:THINK C: Mac #includes:MacTypes.h
rspSize	int	Standard C type.
xo	int	Standard C type.

Called By	
Function	**Where Described**
ATPPut	See Section 2.22.1.3.3.
DownloadTerrainMap	See Section 2.22.1.3.4.

Table 2.22-4 SetUpATPRequest Information.

2.22.1.3.3 ATPPut

ATPPut sends an ATP request and gets a response. The function call is ATPPut(code, req, reqSize, rsp, RspSize, xo). Table 2.22-5 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
code	long	Standard C type.
req	Ptr	Development:THINK C: Mac #includes:MacTypes.h
reqSize	int	Standard C type.
rsp	Ptr	Development:THINK C: Mac #includes:MacTypes.h
rspSize	int	Standard C type.
xo	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
errCode	int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Unsuccessful
(ab)->atpProto.atpActCount	int	Actual size of the response.
Calls		
Function	**Where Described**	
NewHandle	Standard Memory Manager function for Macintosh.	
SetUpATPRequest	See Section 2.22.1.3.2.	
ATPRequest	Standard Appletalk Manager function for Macintosh.	
ATPError	See Section 2.22.1.3.6.	

Table 2.22-5 ATPPut Information.

2.22.1.3.4 DownloadTerrainMap

DownloadTerrainMap obtains the TerrainMap description of the terrain mapping from the host. The function call is DownloadTerrainMap(userData). Table 2.22-6 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
userData	long	Standard C type.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
ab	register ABRecHandle	Development:THINK C: Mac #includes:Appletalk.h
n	char	Standard C type.
errCode	int	Standard C type.

Calls	
Function	**Where Described**
NewHandle	Standard Memory Manager function for Macintosh.
SetUpATPRequest	See Section 2.22.1.3.2.
ATPRequest	Standard Appletalk Manager function for Macintosh.
ATPError	See Section 2.22.1.3.6.
DisposHandle	Standard Memory Manager function for Macintosh.

Table 2.22-6 DownloadTerrainMap Information.

2.22.1.3.5 NetworkEventHandler

NetworkEventHandler fields a network event signifying completion of an asynchronous AppleTalk transaction. *theEvent* holds information about the network event; *requestHandler* is the pointer to the function to call if network event is a request; *responseHandler* is a pointer to the function to call if this is a response. The function call is NetworkEventHandler(theEvent, requestHandler, responseHandler). Table 2.22-7 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
theEvent	pointer to EventRecord	Development:THINK C: Mac #includes:EventMgr.h
(requestHandler)()	pointer to void function	Standard C type.
(responseHandler)()	pointer to void function	Standard C type.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
ab	register ABRecHandle	Development:THINK C: Mac #includes:Appletalk.h
errCode	int	Standard C type.

Calls	
Function	**Where Described**
ATPError	See Section 2.22.1.3.6.
DisposPtr	Standard Memory Manager function for Macintosh.
DisposHandle	Standard Memory Manager function for Macintosh.
ATPGetRequest	Standard Appletalk Manager function for Macintosh

Table 2.22-7 NetworkEventHandler Information.

2.22.1.3.6 ATPError

ATPError interprets an error that occured with the AppleTalk network. The function call is ATPError(errCode). Table 2.22-8 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
errCode	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
cp	register pointer to char	Standard C type.
buf[100]	char	Standard C type.
Calls		
Function	**Where Described**	
ShowCaution	See Section 2.22.1.4.1.	
Called By		
Function	**Where Described**	
ATPPut	See Section 2.22.1.3.3.	
DownloadTerrainMap	See Section 2.22.1.3.4.	
NetworkEventHandler	See Section 2.22.1.3.5.	

Table 2.22-8 ATPError Information.

2.22.1.4 caution.c

Development:SIMNET:libmac:caution.c

caution.c contains routines for displaying an alert box containing a warning message.

2.22.1.4.1 ShowCaution

ShowCaution puts up an alert box with an arbitrary warning message, *str*. The function call is ShowCaution(str). Table 2.22-9 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
str	pointer to char	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
d	DialogPtr	Development:THINK C: Mac #includes:DialogMgr.h
i	int	Standard C type.
theItem	Handle	Development:THINK C: Mac #includes:MacTypes.h
box	Rect	Development:THINK C: Mac #includes:MacTypes.h

Calls	
Function	**Where Described**
SysBeep	Standard Operating System Utility function for Macintosh.
ShowSimpleDialog	See Section 2.22.1.40.1.
OutlineItem	See Section 2.22.1.27.2.
GetDItem	Standard Dialog Manager function for Macintosh.
SetIText	Standard Dialog Manager function for Macintosh.
SetWRefCon	Standard Window Manager function for Macintosh.
ShowWindow	Standard Window Manager function for Macintosh.
Called By	
Function	**Where Described**
CheckMandatoryFields	See Section 2.22.1.13.1.
showhelp	See Section 2.22.1.19.4.
setuphtopic	See Section 2.22.1.19.16.
DialogSeqPrev	See Section 2.22.1.39.5.
CheckLastField	See Section 2.22.1.11.5.
ATPError	See Section 2.22.1.3.6.

Table 2.22-9 ShowCaution Information.

2.21.1.4.2 SimpleCautionEvent

SimpleCautionEvent is called when a simple dialog gets an event. The function call is SimpleCautionEvent(window, theEvent). Table 2.22-10 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
window	WindowPtr	Development:THINK C: Mac #includes:WindowMgr.h
theEvent	pointer to EventRecord	Development:THINK C: Mac #includes:EventMgr.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
itemHit	int	Standard C type.
Calls		
Function	**Where Described**	
DialogSelect	Standard Dialog Manager function for Macintosh.	
DisposDialog	Standard Dialog Manager function for Macintosh.	
SysBeep	Standard Operating System Utility function for Macintosh.	

Table 2.22-10 SimpleCautionEvent Information.

2.21.1.4.3 LastCaution

LastCaution returns a WindowPtr pointing to the last CautionWindow. If you need to stick your new window behind the last caution box, then call NewWindow with LastCaution. The function call is LastCaution(). Table 2.22-11 describes the internal variables used and functions called using this function.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
wPeek	WindowPeek	Development:THINK C: Mac #includes:WindowMgr.h
fPeek	WindowPeek	Development:THINK C: Mac #includes:WindowMgr.h
Return Values		
Return Value	**Type**	**Meaning**
-1	WindowPtr	No caution boxes up.
fPeek	WindowPtr	Last caution window.
Calls		
Function	**Where Described**	
GetWRefCon	Standard Window Manager function for Macintosh.	
Called By		
Function	**Where Described**	
ShowDialog	See Section 2.22.1.11.1.	
ShowSimpleDialog	See Section 2.22.1.40.1.	
SetUpAppleTalk	See Section 2.22.1.3.1.	

Table 2.22-11 LastCaution Information.

2.22.1.4.4 CountCaution

CountCaution returns the number of caution boxes currently displayed. The function call is CountCaution(). Table 2.22-12 describes the internal variables used and functions called using this function.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
wPeek	WindowPeek	Development:THINK C: Mac #includes:WindowMgr.h
cnt	int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
cnt	int	Number of caution boxes.
Calls		
Function	**Where Described**	
GetWRefCon	Standard Window Manager function for Macintosh.	

Table 2.22-12 CountCaution Information.

2.22.1.5 clock.h
Development:SIMNET:libmac:clock.h

clock.h is the header file associated with the following file, clock.c(Section 2.22.1.6).

2.22.1.6 clock.c
Development:SIMNET:libmac:clock.c

clock.c contains routines for displaying the current date and time in the top right corner of the Macintosh screen. Table 2.22-13 describes the variables used by clock.c.

Variables		
Variable	**Type**	**Where Typedef Declared**
clockPort	GrafPort	Development:THINK C: Mac #includes:Quickdraw.h
lastMinute	unsigned long	Standard C type.

Table 2.22-13 clock.c Variable Information.

2.22.1.6.1 DrawClock

DrawClock writes the current Date Time group, *now*, in the clock port in response to an update. The function call is DrawClock(now). Table 2.22-14 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
now	unsigned long	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
tempPort	GrafPtr	Development:THINK C: Mac #includes:Quickdraw.h
dtg	DateTimeGroup	Development:SIMNET:libmac: dtg.h
str[20]	char	Standard C type.
Calls		
Function	**Where Described**	
GetPort	Standard Quickdraw function for Macintosh.	
SetPort	Standard Quickdraw function for Macintosh.	
EraseRect	Standard Quickdraw function for Macintosh.	
FrameRect	Standard Quickdraw function for Macintosh.	
DTGElapsed	See Section 2.22.1.15.3.	
DTGToString	See Section 2.22.1.15.1.	
MoveTo	Standard Quickdraw function for Macintosh.	
DrawText	Standard Quickdraw function for Macintosh.	

Called By	
Function	**Where Described**
UpdateClock	See Section 2.22.1.6.2.
RedrawClock	See Section 2.22.1.6.3.
InstallClock	See Section 2.22.1.6.4.
SetClock	See Section 2.22.1.6.5.

Table 2.22-14 DrawClock Information.

2.22.1.6.2 UpdateClock

UpdateClock updates the clock every minute. The function call is UpdateClock(). Table 2.22-15 describes the internal variable used and functions called using this function.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
now	unsigned long	Standard C type.
Calls		
Function	**Where Described**	
GetDateTime	Standard Operating System Utility function for Macintosh.	
DrawClock	See Section 2.22.1.6.1.	

Table 2.22-15 UpdateClock Information.

2.22.1.6.3 RedrawClock

RedrawClock redraws the clock with the current time. The function call is RedrawClock(). Table 2.22-16 describes the internal variable used and functions called using this function.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
now	unsigned long	Standard C type.
Calls		
Function	**Where Described**	
GetDateTime	Standard Operating System Utility function for Macintosh.	
DrawClock	See Section 2.22.1.6.1.	

Table 2.22-16 RedrawClock Information.

2.22.1.6.4 InstallClock

InstallClock sets up the Clock GrafPort and displays the current time. The function call is InstallClock(). Table 2.22-17 describes the internal variable used and functions called using this function.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
tempPort	GrafPtr	Development:THINK C: Mac #includes:Quickdraw.h
now	unsigned long	Standard C type.

Calls	
Function	**Where Described**
GetPort	Standard Quickdraw function for Macintosh.
OpenPort	Standard Quickdraw function for Macintosh.
PortSize	Standard Quickdraw function for Macintosh.
MovePortTo	Standard Quickdraw function for Macintosh.
SetPort	Standard Quickdraw function for Macintosh.
GetDateTime	Standard Operating System Utility function for Macintosh.
DrawClock	See Section 2.22.1.6.1.

Table 2.22-17 InstallClock Information.

2.22.1.6.5 SetClock

SetClock sets the clock to the time *now*. The function call is SetClock(now). Table 2.22-18 describes the internal variable used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
now	unsigned long	Standard C type.
Calls		
Function	**Where Described**	
SetDateTime	Standard Operating System Utility function for Macintosh.	
DrawClock	See Section 2.22.1.6.1.	

Table 2.22-18 SetClock Information.

2.22.1.7 control.c

Development:SIMNET:libmac:control.c

control.c contains routines for enabling and disabling controls within dialogs.

2.22.1.7.1 DisableControl

DisableControl disables a control within a dialog. *dialog* is the dialog that control is in; *itemNo* is the itemNo of the control. The function call is DisableControl(dialog, itemNo). Table 2.22-19 describes the parameters used and functions called using this function

Parameters		
Parameter	**Type**	**Where Typedef Declared**
dialog	DialogPtr	Development:THINK C: Mac #includes:DialogMgr.h
itemNo	int	Standard C type
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	int	Standard C type.
theItem	Handle	Development:THINK C: Mac #includes:MacTypes.h
box	Rect	Development:THINK C: Mac #includes:MacTypes.h

Calls	
Function	**Where Described**
GetDItem	Standard Dialog Manager function for Macintosh.
HiliteControl	Standard Control Manager function for Macintosh.
Called By	
Function	**Where Described**
LabelForceButtons	See Section 2.22.1.16.1.

Table 2.22-19 DisableControl Information.

2.22.1.7.2 EnableControl

EnableControl enables a control within a dialog. The function call is EnableControl(dialog, itemNo). Table 2.22-20 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
dialog	DialogPtr	Development:THINK C: Mac #includes:DialogMgr.h
itemNo	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	int	Standard C type.
theItem	*Handle*	Development:THINK C: Mac #includes:MacTypes.h
box	Rect	Development:THINK C: Mac #includes:MacTypes.h
Calls		
Function	**Where Described**	
GetDItem	Standard Dialog Manager function for Macintosh.	
HiliteControl	Standard Control Manager function for Macintosh.	
Called By		
Function	**Where Described**	
LabelForceButtons	See Section 2.22.1.16.1.	

Table 2.22-20 EnableControl Information.

2.22.1.8 bigprob.c

Development:SIMNET:libmac:bigprob.c

bigprob.c contains routines for reporting unrecoverable errors.

2.22.1.8.1 SystemFailure

SystemFailure calls BipProb with a system error code. *errCode* is the error number of type of error; *place* is a string that describes where error occurred. The function call is SystemFailure(errCode, place). Table 2.22-21 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
errCode	int	Standard C type.
place	pointer to char	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
str	array of 256 char	Standard C type.
Calls		
Function	**Where Described**	
BigProb	See Section 2.22.1.8.2.	
Called By		
Function	**Where Described**	
SetUpAppleTalk	See Section 2.22.1.3.1.	

Table 2.22-21 SystemFailure Information.

2.22.1.8.2 BigProb

BigProb puts up an alert, *s*, and halts the program. The function call is BigProb(s). Table 2.22-22 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
s	pointer to char	Standard C type.
Calls		
Function	**Where Described**	
ParamText	Standard Dialog Manager function for Macintosh.	
StopAlert	Standard Dialog Manager function for Macintosh.	
ExitToShell	Standard Segment Loader function for Macintosh.	
Called By		
Function	**Where Described**	
SystemFailure	See Section 2.22.1.8.1.	
SetUpAppleTalk	See Section 2.22.1.3.1.	

Table 2.22-22 BigProb Information.

2.22.1.9 dialog.h
Development:SIMNET:libmac:dialog.h

dialog.h is the header file associated with the following files, which collectively provide support for dialog boxes.

2.22.1.10 dialine.c
Development:SIMNET:libmac:dialine.c

dialine.c contains routines for drawing lines and rectangles within dialogs.

2.22.1.10.1 DrawLine

DrawLine draws a line or a rectangle in a dialog box. The function call is DrawLine(w, itemNo). Table 2.22-23 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
w	register WindowPtr	Development:THINK C: Mac #includes:WindowMgr.h
itemNo	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	int	Standard C type.
field	int	Standard C type.
theItem	Handle	Development:THINK C: Mac #includes:MacTypes.h
r	Rect	Development:THINK C: Mac #includes:MacTypes.h
defn	register pointer to LineFieldDefn	Development:SIMNET:libmac: dialog.h
pnState	PenState	Development:THINK C: Mac #includes:Quickdraw.h
Calls		
Function	**Where Described**	
GetDItem	Standard Dialog Manager function for Macintosh.	
DialogLookupField	See Section 2.22.1.12.1.	
SetPort	Standard Quickdraw function for Macintosh.	
GetPenState	Standard Quickdraw function for Macintosh.	
PenSize	Standard Quickdraw function for Macintosh.	
PenMode	Standard Quickdraw function for Macintosh.	
PenPat	Standard Quickdraw function for Macintosh.	
MoveTo	Standard Quickdraw function for Macintosh.	
LineTo	Standard Quickdraw function for Macintosh.	
FrameRect	Standard Quickdraw function for Macintosh.	
SetPenState	Standard Quickdraw function for Macintosh.	

Table 2.22-23 DrawLine Information.

2.22.1.11 dialog.c

Development:SIMNET:libmac:dialog.c

dialog.c contains routines for initializing, discarding and checking the entries in dialogs, and for handling events associated with dialogs.

2.22.1.11.1 ShowDialog

ShowDialog is called to display a dialog box. The function call is ShowDialog(dStorage, defn). The function returns a pointer to DialogState, a structure defined in dialog.h. *dStorage* is the pointer to DialogState structure; *defn* is the pointer to DialogDefn structure. Table 2.22-24 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
dStorage	register Ptr	Development:THINK C: Mac #includes:MacTypes.h
defn	register pointer to DialogDefn	Development:SIMNET:libmac: dialog.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
fpp	pointer to pointer to FieldDefn	Development:SIMNET:libmac: dialog.h
fnc	pointer to function that returns int	Standard C type.
i	int	Standard C type.
j	int	Standard C type.
theItem	Handle	Development:THINK C: Mac #includes:MacTypes.h
box	Rect	Development:THINK C: Mac #includes:MacTypes.h
Return Values		
Return Value	**Type**	**Meaning**
STATE	DialogState	The specified dialog
Calls		
Function	**Where Described**	
NewPtr	Standard Memory Manager function for Macintosh.	
GetNewDialog	Standard Dialog Manager function for Macintosh.	
LastCaution	See Section 2.22.1.4.3.	
SetWRefCon	Standard Window Manager function for Macintosh.	
OutlineItem	See Section 2.22.1.27.2.	
ShowFixTable	See Section 2.22.1.42.1.	
ShowScrollTable	See Section 2.22.1.30.1.	
GetDItem	Standard Dialog Manager function for Macintosh.	
SetDItem	Standard Dialog Manager function for Macintosh.	
UpdateDialog	See Section 2.22.1.11.2.	

Called By	
Function	**Where Described**
ShowDialogSeqNode	See Section 2.22.1.39.2.

Table 2.22-24 ShowDialog Information.

2.22.1.11.2 UpdateDialog

UpdateDialog loads values into the editable fields of a dialog box specified by *dialog*. The function call is UpdateDialog(dialog). Table 2.22-25 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
dialog	register pointer to DialogState	Development:SIMNET:libmac: dialog.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
fpp	pointer to pointer to FieldDefn	Development:SIMNET:libmac: dialog.h
field	int	Standard C type.
i	int	Standard C type.
theItem	Handle	Development:THINK C: Mac #includes:MacTypes.h
box	Rect	Development:THINK C: Mac #includes:MacTypes.h
str[256]	char	Standard C type.
Calls		
Function	**Where Described**	
GetDItem	Standard Dialog Manager function for Macintosh.	
SetCtlValue	Standard Control Manager function for Macintosh.	
SetIText	Standard Dialog Manager function for Macintosh.	
NumToString	Standard Binary to Decimal Conversion Package function for Macintosh.	
DTGToString	See Section 2.22.1.15.1.	
Called By		
Function	**Where Described**	
ShowDialog	See Section 2.22.1.11.1.	

Table 2.22-25 UpdateDialog Information.

2.22.1.11.3 ThrowDialog

ThrowDialog removes a dialog, *dialog*, from the screen. The function call is ThrowDialog(dialog). Table 2.22-26 describes the parameter used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
dialog	register pointer to DialogState	Development:SIMNET:libmac:dialog.h
Calls		
Function	**Where Described**	
CloseDialog	Standard Dialog Manager function for Macintosh.	
DisposHandle	Standard Memory Manager function for Macintosh.	
Called By		
Function	**Where Described**	
DialogSeqNext	See Section 2.22.1.39.4.	
DialogSeqPrev	See Section 2.22.1.39.5.	

Table 2.22-26 ThrowDialog Information.

2.22.1.11.4 DialogEvent

DialogEvent handles an event, *theEvent*, within a dialog box, specified by *dialog*. *window* is the window in which the event occurred; *theEvent* returns 0 if the event was invalid, otherwise it returns the *itemNo* of the control the event was about. The function call is DialogEvent(window, theEvent). Table 2.22-27 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
window	register WindowPtr	Development:THINK C: Mac #includes:WindowMgr.h
theEvent	pointer to EventRecord	Development:THINK C: Mac #includes:EventMgr.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
defn	register pointer to DialogDefn	Development:SIMNET:libmac:dialog.h
itemNo	int	Standard C type.
field	int	Standard C type.
fpp	pointer to pointer to FieldDefn	Development:SIMNET:libmac:dialog.h
fp	pointer to FieldDefn	Development:SIMNET:libmac:dialog.h
theItem	Handle	Development:THINK C: Mac #includes:MacTypes.h
ctl	ControlHandle	Development:THINK C: Mac #includes:ControlMgr.h
ch	char	Standard C type.

Return Values		
Return Value	**Type**	**Meaning**
0	int	Unsuccessful.
itemNo	int	The resource id of the button pressed.
Calls		
Function	**Where Described**	
SubPt	Standard Quickdraw function for Macintosh.	
DialogSelect	Standard Dialog Manager function for Macintosh.	
ScrollTableActivate	See Section 2.22.1.32.1.	
DialogLookupField	See Section 2.22.1.12.1.	
CheckLastField	See Section 2.22.1.11.5.	
SetRadioButton	See Section 2.22.1.11.6.	
SetCheckBox	See Section 2.22.1.11.7.	
FixTableEvent	See Section 2.22.1.42.3.	
ScrollTableEvent	See Section 2.22.1.32.2.	
Called By		
Function	**Where Described**	
DialogSeqEvent	See Section 2.22.1.39.3.	

Table 2.22-27 DialogEvent Information.

2.22.1.11.5 CheckLastField

CheckLastField checks the syntax of the field just left. The function call is CheckLastField(dialog). Table 2.22-28 describes the parameters used, errors returned and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
dialog	pointer to DialogState	Development:SIMNET:libmac:dialog.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
fp	register pointer to FieldDefn	Development:SIMNET:libmac:dialog.h
i	int	Standard C type.
n	long	Standard C type.
theItem	Handle	Development:THINK C:Mac #includes:MacTypes.h
box	Rect	Development:THINK C:Mac #includes:MacTypes.h
cp	pointer to char	Standard C type.
cStr	pointer to char	Standard C type.
pStr[256]	char	Standard C type.
map	MapCoordinates	Development:SIMNET:libmac:map.h
dtg	DateTimeGroup	Development:SIMNET:libmac:dtg.h

Calls	
Function	**Where Described**
GetDItem	Standard Dialog Manager function for Macintosh.
GetIText	Standard Dialog Manager function for Macintosh.
SelIText	Standard Dialog Manager function for Macintosh.
ShowCaution	See Section 2.22.1.4.1.
StringToNum	Standard Binary to Decimal Conversion Package function for Macintosh.
StringToMapCoordinates	See Section 2.22.1.26.1.
SetIText	Standard Dialog Manager function for Macintosh.
StringToDTG	See Section 2.22.1.15.2.
Called By	
Function	**Where Described**
DialogEvent	See Section 2.22.1.11.4.

Table 2.22-28 CheckLastField Information.

2.22.1.11.6 SetRadioButton

SetRadioButton sets a radio button on and the others in the group off. The function call is SetRadioButton(dialog, itemNo). Table 2.22-29 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
dialog	pointer to DialogState	Development:SIMNET:libmac: dialog.h
itemNo	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
fp	register pointer to RBFieldDefn	Development:SIMNET:libmac: dialog.h
fpp	pointer to pointer to FieldDefn	Development:SIMNET:libmac: dialog.h
i	int	Standard C type.
theItem	Handle	Development:THINK C: Mac #includes:MacTypes.h
box	Rect	Development:THINK C: Mac #includes:MacTypes.h
Calls		
Function	**Where Described**	
DialogLookupField	See Section 2.22.1.12.1.	
GetDItem	Standard Dialog Manager function for Macintosh.	
SetCtlValue	Standard Control Manager function for Macintosh.	
Called By		
Function	**Where Described**	
DialogEvent	SeeSection 2.22.1.11.4.	

Table 2.22-29 SetRadioButton Information.

2.22.1.11.7 SetCheckBox

SetCheckBox sets a check box on or off. *dialog* points to the dialog check box is in; *itemNo* is the itemNo of check box; *value* states whether check box should be turned on or off. The function call is SetCheckBox(dialog, itemNo, value). Table 2.22-30 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
dialog	pointer to DialogState	Development:SIMNET:libmac:dialog.h
itemNo	int	Standard C type.
value	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
fp	pointer to CBFieldDefn	Development:SIMNET:libmac:dialog.h
i	int	Standard C type.
theItem	Handle	Development:THINK C:Mac #includes:MacTypes.h
box	Rect	Development:THINK C:Mac #includes:MacTypes.h
Calls		
Function	**Where Described**	
DialogLookupField	See Section 2.22.1.12.1.	
GetDItem	Standard Dialog Manager function for Macintosh.	
SetCtlValue	Standard Control Manager function for Macintosh.	
Called By		
Function	**Where Described**	
DialogEvent	See Section 2.22.1.11.4.	

Table 2.22-30 SetCheckBox Information.

2.22.1.11.8 SetText

SetText sets a text item to a string. The function call is SetText(dialog, itemNo, cStr). Table 2.22-31 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
dialog	pointer to DialogState	Development:SIMNET:libmac:dialog.h
itemNo	int	Standard C type.
cStr	pointer to char	Standard C type.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	int	Standard C type.
theItem	Handle	Development:THINK C: .Mac #includes:MacTypes.h
box	Rect	Development:THINK C: Mac #includes:MacTypes.h
pStr[256]	char	Standard C type.
Calls		
Function	**Where Described**	
GetDItem	Standard Dialog Manager function for Macintosh.	
SetIText	Standard Dialog Manager function for Macintosh.	

Table 2.22-31 SetText Information.

2.22.1.12 dialookup.c
Development:SIMNET:libmac:dialookup.c

dialookup.c contains a routine for locating a definition of a dialog field given its dialog item number.

2.22.1.12.1 DialogLookupField

DialogLookupField locates a field by item number. The function call is DialogLookupField(state, itemNo). Table 2.22-32 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
state	pointer to DialogState	Development:SIMNET:libmac. dialog.h
itemNo	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
fpp	pointer to pointer to FieldDefn	Development:SIMNET:libmac: dialog.h
i	register int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
-1	int	Not successful.
i	int	Field definition.

Called By	
Function	**Where Described**
DrawLine	See Section 2.22.1.10.1.
DrawFixTable	See Section 2.22.1.42.4.
DialogEvent	See Section 2.22.1.11.4.
SetRadioButton	See Section 2.22.1.11.6.
SetCheckBox	See Section 2.22.1.11.7.

Table 2.22-32 DialogLookupField Information.

2.22.1.13 diamand.c

Development:SIMNET:libmac:diamand.c

diamand.c contains a routine for ensuring that mandatory dialog fields are filled.

2.22.1.13.1 CheckMandatoryFields

CheckMandatoryFields ensures that all mandatory fields are filled in. *state* is the pointer to Dialog to be checked. The function call is CheckMandatoryFields(state). Table 2.22-33 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
state	register pointer to DialogState	Development:SIMNET:libmac:dialog.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
fpp	pointer to pointer to FieldDefn	Development:SIMNET:libmac:dialog.h
i	int	Standard C type.
theItem	Handle	Development:THINK C:Mac #includes:MacTypes.h
box	Rect	Development:THINK C:Mac #includes:MacTypes.h
str[256]	char	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
1	int	All mandatory fields filled in.
0	int	Mandatory field must be filled in.
Calls		
Function	**Where Described**	
GetDItem	Standard Dialog Manager function for Macintosh.	
GetIText	Standard Dialog Manager function for Macintosh.	
SetIText	Standard Dialog Manager function for Macintosh.	
ShowCaution	See Section 2.22.1.4.1.	
CtoPstr		

Table 2.22-33 CheckMandatoryFields Information.

2.22.1.14 dtg.h
Development:SIMNET:libmac:dtg.h

dtg.h is the header file associated with the following file, dtg.c(Section 2.22.1.15).

2.22.1.15 dtg.c
Development:SIMNET:libmac:dtg.c

dtg.c contains routines for input, output, and processing of date time groups. Table 2.22-34 describes the variables used by dtg.c.

Variables		
Variable	**Type**	**Where Typedef Declared**
currentTime	DateTimeGroup	Development:SIMNET:libmac:dtg.h
monthAbbrev[]	char	Standard C type.
monthLength[]	char	Standard C type.

Table 2.22-34 dtg.c Variable Information.

2.22.1.15.1 DTGToString

DTGToString generates a string, *str*, from a DateTimeGroup object, *dtg*. The function call is DTGToString(dtg, str). Table 2.22-35 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
dtg	register pointer to DateTimeGroup	Development:SIMNET:libmac:dtg.h
str	register pointer to char	Standard C type.
Called By		
Function	**Where Described**	
DrawClock	See Section 2.22.1.6.1.	
UpdateDialog	See Section 2.22.1.11.2.	

Table 2.22-35 DTGToString Information.

2.22.1.15.2 StringToDTG

StringToDTG converts a string, *cp*, to a DateTimeGroup object, *dtg*. The string is not expected to include a year, although the year must be specified in the DateTimeGroup object produced. So the appropriate year is deduced from the current time on the Macintosh. The function call is StringToDTG(cp, dtg). Table 2.22-36 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
cp	register pointer to char	Standard C type.
dtg	pointer to DateTimeGroup	Development:SIMNET:libmac:dtg.h

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
day	int	Standard C type.
time	int	Standard C type.
hour	int	Standard C type.
minute	int	Standard C type.
month	int	Standard C type.
monthName[4]	char	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
DTG_SUCCESS	int = 0	DTG string parsed okay.
DTG_ERROR	int = 1	DTG string has lexical error.
DTG_BAD_DAY	int = 2	DTG day out of range.
DTG_BAD_HOUR	int = 3	DTG hour out of range.
DTG_BAD_MINUTE	int = 4	DTG minute out of range.
Calls		
Function	**Where Described**	
GetDateTime	Standard Operating System Utility function for Macintosh.	
Secs2Date	Standard Operating System Utility function for Macintosh.	
Date2Secs	Standard Operating System Utility function for Macintosh.	
Called By		
Function	**Where Described**	
CheckLastField	See Section 2.22.1.11.5.	

Table 2.22-36 StringToDTG Information.

2.22.1.15.3 DTGElapsed

DTGElapsed sets a DateTimeGroup data object, *dtg*, from an elapsed time value, *secs*, (in seconds from Jan. 1, 1904). The function call is DTGElapsed(secs, dtg). Table 2.22-37 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
secs	unsigned long	Standard C type.
dtg	pointer to DateTimeGroup	Development:SIMNET:libmac:dtg.h
Calls		
Function	**Where Described**	
Secs2Date	Standard Operating System Utility function for Macintosh.	
Called By		
Function	**Where Described**	
DrawClock	See Section 2.22.1.6.1.	

Table 2.22-37 DTGElapsed Information.

2.22.1.15.4 Get_DTG

Get_DTG retrieves the current time as a DateTimeGroup, *dtg*. The function call is Get_DTG(dtg). Table 2.22-38 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
dtg	pointer to DateTimeGroup	Development:SIMNET:libmac:dtg.h
Calls		
Function	**Where Described**	
GetDateTime	Standard Operating System Utility function for Macintosh.	
Secs2Date	Standard Operating System Utility function for Macintosh.	

Table 2.22-38 Get_DTG Information.

2.22.1.16 force.c
Development:SIMNET:libmac:force.c

force.c contains routines supporting the concept of forces (e.g., "US" and "Soviet").

2.22.1.16.1 LabelForceButtons

LabelForceButtons labels radio buttons that display a choice of forces. The *itemNo* passed is that of the first of three radio buttons, corresponding to force IDs 1, 2 and 3. The *battleScheme* parameter determines whether the first two buttons are labelled "Offense" and "Defense", or "US" and"Soviet". The *forceID* parameter specifies which button(s) is/are enabled. If it is forceIDIrrelevant or observerForceID, all three buttons are enabled. The function call is LabelForceButtons(dialog, itemNo, battleScheme, forceID). Table 2.22-39 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
dialog	DialogPtr	Development:THINK C: Mac #includes:DialogMgr.h
itemNo	int	Standard C type.
battleScheme	int	Standard C type.
forceID	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	int	Standard C type.
theItem	Handle	Development:THINK C: Mac #includes:MacTypes.h
box	Rect	Development:THINK C: Mac #includes:MacTypes.h
titleA[]	pointer to char	Standard C type.
titleB[]	pointer to char	Standard C type.

Calls	
Function	**Where Described**
GetDItem	Standard Dialog Manager function for Macintosh.
SetCTitle	Standard Control Manager function for Macintosh.
EnableControl	See Section 2.22.1.7.2.
DisableControl	See Section 2.22.1.7.1.

Table 2.22-39 LabelForceButtons Information.

2.22.1.16.2 ForceNamePStr

ForceNamePStr returns the name of a force under a particular battle scheme as a Pascal string. The function call is ForceNamePStr(forceID, battleScheme). The function call returns a pointer to a string. Table 2.22-40 describes the parameters usedby this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
forceID	int	Standard C type.
battleScheme	int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
str	pointer to char	Name of force.

Table 2.22-40 ForceNamePStr Information.

2.22.1.17 help.h

Development:SIMNET:libmac:help.h

help.h is the header file defining the external interface to the following collection of help routines.

2.22.1.18 helplocal.h

Development:SIMNET:libmac:helplocal.h

helplocal.h is the header file containing internal definitions associated with the following collection of help routines.

2.22.1.19 help.c

Development:SIMNET:libmac:help.c

help.c contains routines providing user access to a database of help material. Table 2.22-41 describes the variables used by help.c.

Variables		
Variable	**Type**	**Where Typedef Declared**
curhelp	int	Standard C type.
helpfid	int	Standard C type.
helphandle	Handle	Development:THINK C: Mac #includes:MacTypes.h
helpareaheight	int	Standard C type.
helplineheight	int	Standard C type.
helpmaxorigin	int	Standard C type.
helppageheight	int	Standard C type.
helprect	Rect	Development:THINK C: Mac #includes:MacTypes.h
helpscroll	ControlHandle	Development:THINK C: Mac #includes:ControlMgr.h
helpscrstate	unsigned int	Standard C type.
helptitle[TITLEMAX]	char	Standard C type.
helpupdrgn	RgnHandle	Development:THINK C: Mac #includes:Quickdraw.h
helpvars	Handle	Development:THINK C: Mac #includes:MacTypes.h
holdclip	RgnHandle	Development:THINK C: Mac #includes:Quickdraw.h
lasthelp	int	Standard C type.
nohelp	Boolean	Development:THINK C: Mac #includes:MacTypes.h
numchunks	int	Standard C type.
topiccount	int	Standard C type.
topiclist	ListHandle	Development:THINK C: Mac #includes:ListMgr.h
topichdl	Handle	Development:THINK C: Mac #includes:MacTypes.h
topicnames	Handle	Development:THINK C: Mac #includes:MacTypes.h
topics	Handle	Development:THINK C: Mac #includes:MacTypes.h

Table 2.22-41 help.c Variable Information.

2.22.1.19.1 helpinit

helpinit initializes the help facility. It is called once at program startup. The function call is helpinit(helpfile). Table 2.22-42 describes the parameters used, errors returned and functions called using this function. *helpfile* is the address of a pascal type string containing the name of the help resource file, if NIL is passed, no file will be opened and the help resources will be assumed to be in an already opened resource file.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
helpfile	pointer to char	Standard C type.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
curtopic	int	Standard C type.
helpindex	int	Standard C type.
helpid	int	Standard C type.
initresult	int	Standard C type.
mhlphdl	Handle	Development:THINK C: Mac #includes:MacTypes.h
mhlpptr	pointer to char	Standard C type.
orgcount	int	Standard C type.
textlen	unsigned long	Standard C type.
topiclen	unsigned long	Standard C type.
topicptr	pointer to int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
initresult	int	0 if no errors, otherwise error any error encountered.
Calls		
Function	**Where Described**	
OpenResFile	Standard Resource Manager function for Macintosh.	
ResError	Standard Resource Manager function for Macintosh.	
GetIndResource	Standard Resource Manager function for Macintosh.	
HLock	Standard Memory Manager function for Macintosh.	
NewHandle	Standard Memory Manager function for Macintosh.	
GetResource	Standard Resource Manager function for Macintosh.	
HNoPurge	Standard Memory Manager function for Macintosh.	
SetHandleSize	Standard Memory Manager function for Macintosh.	
BlockMove	Standard Memory Manager function for Macintosh.	
HPurge	Standard Memory Manager function for Macintosh.	
ReleaseResource	Standard Resource Manager function for Macintosh.	
DisposHandle	Standard Memory Manager function for Macintosh.	
HUnlock	Standard Memory Manager function for Macintosh.	

Table 2.22-42 helpinit Information.

2.22.1.19.2 helpshutdown

helpshutdown shut down the help facility. The function call is helpshutdown(). Table 2.22-43 describes the internal variables used and functions called using this function.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
contenthdl	Handle	Development:THINK C: Mac #includes:MacTypes.h
curlen	unsigned int	Standard C type.
curpos	unsigned long	Standard C type.
curvar	pointer to char	Standard C type.
varslen	unsigned long	Standard C type.

Calls	
Function	**Where Described**
DisposHandle	Standard Memory Manager function for Macintosh.
GetHandleSize	Standard Memory Manager function for Macintosh.
CloseResFile	Standard Resource Manager function for Macintosh.

Table 2.22-43 helpshutdown Information.

2.22.1.19.3 sethelpvar

sethelpvar sets the contents of a help variable. The function call is sethelpvar(variable, varcontents). Table 2.22-44 describes the parameters used and functions called using this function. *variable* is a pointer to a string containing the variable name; *varcontents* is a pointer to a string containing the new contents for the variable.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
variable	pointer to char	Standard C type.
varcontents	pointer to char	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
addlen	long	Standard C type.
contenthdl	Handle	Development:THINK C: Mac #includes:MacTypes.h
contlen	int	Standard C type.
count	int	Standard C type.
curlen	long	Standard C type.
temp1ptr	pointer to char	Standard C type.
temp2ptr	pointer to char	Standard C type.
varlen	int	Standard C type.
varnotfound	Boolean	Development:THINK C: Mac #includes:MacTypes.h
Calls		
Function	**Where Described**	
SetHandleSize	Standard Memory Manager function for Macintosh.	
GetHandleSize	Standard Memory Manager function for Macintosh.	
NewHandle	Standard Memory Manager function for Macintosh.	

Table 2.22-44 sethelpvar Information.

2.22.1.19.4 showhelp

showhelp displays the specified help screen. The function call is showhelp(helpid). Table 2.22-45 describes the parameters used and functions called using this function. *helpid* is the id number of the help screen to display. If *helpid* is zero, the topics menu will be displayed and the user can choose any help screen.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
helpid	int	Standard C type.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
done	Boolean	Development:THINK C: Mac #includes:MacTypes.h
helpdloghdl	Handle	Development:THINK C: Mac #includes:MacTypes.h
helpdlogid	int	Standard C type.
helpdlogname[64]	char	Standard C type.
helpdlogtype	ResType	Development:THINK C: Mac #includes:MacTypes.h
helpdptr	DialogPtr	Development:THINK C: Mac #includes:DialogMgr.h

Calls	
Function	**Where Described**
ShowCaution	See Section 2.22.1.4.1.
NewRgn	Standard Quickdraw function for Macintosh.
GetNamedResource	Standard Resource Manager function for Macintosh.
GetResInfo	Standard Resource Manager function for Macintosh.
GetNewDialog	Standard Dialog Manager function for Macintosh.
centerdialog	See Section 2.22.1.19.5.
SetPort	Standard Quickdraw function for Macintosh.
OutlineItem	See Section 2.22.1.27.2.
setuphtopic	See Section 2.22.1.19.16.
setuphpick	See Section 2.22.1.19.8.

Called By	
Function	**Where Described**
DialogSeqEvent	See Section 2.22.1.39.3.

Table 2.22-45 showhelp Information.

2.22.1.19.5 centerdialog

centerdialog centers a dialog box on the current screen. The function call is centerdialog(dptr). Table 2.22-46 describes the parameters used and functions called using this function. *dptr* is a pointer to the dialog to center.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
dptr	DialogPtr	Development:THINK C: Mac #includes:DialogMgr.h

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
dialogheight	int	Standard C type.
dialogleft	int	Standard C type.
dialogtop	int	Standard C type.
dialogwidth	int	Standard C type.
screenheight	int	Standard C type.
screenwidth	int	Standard C type.

Calls	
Function	**Where Described**
MoveWindow	Standard Window Manager function for Macintosh.
Called By	
Function	**Where Described**
showhelp	See Section 2.22.1.19.4.

Table 2.22-46 centerdialog Information.

2.22.1.19.6 hpickevents

hpickevents is the event handler for the hpick dialog box. The function call is hpickevents(wptr, eventptr). Table 2.22-47 describes the parameters used and functions called using this function. *wptr* is a pointer to the dialog box window; *eventptr* is a pointer to the event record.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
wptr	WindowPtr	Development:THINK C: Mac #includes:WindowMgr.h
eventptr	pointer to EventRecord	Development:THINK C: Mac #includes:EventMgr.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
clicked	Point	Development:THINK C: Mac #includes:MacTypes.h
doubleclick	Boolean	Development:THINK C: Mac #includes:MacTypes.h
fontnum	int	Standard C type.
holdport	GrafPtr	Development:THINK C: Mac #includes:Quickdraw.h
itemhandle	Handle	Development:THINK C: Mac #includes:MacTypes.h
itemhit	int	Standard C type.
itemrect	Rect	Development:THINK C: Mac #includes:MacTypes.h
itemtype	int	Standard C type.
orgfont	int	Standard C type.
orgsize	int	Standard C type.
scrollrect	Rect	Development:THINK C: Mac #includes:MacTypes.h
state	int	Standard C type.
testcell	Cell	Development:THINK C: Mac #includes:ListMgr.h
topicptr	pointer to int	Standard C type.
topicsel	Boolean	Development:THINK C: Mac #includes:MacTypes.h

Calls	
Function	**Where Described**
GetPort	Standard Quickdraw function for Macintosh.
SetPort	Standard Quickdraw function for Macintosh.
GetFNum	Standard Font Manager function for Macintosh.
TextFont	Standard Quickdraw function for Macintosh.
TextSize	Standard Quickdraw function for Macintosh.
LActivate	Standard List Manager function for Macintosh
DialogSelect	Standard Dialog Manager function for Macintosh.
LGetSelect	Standard List Manager function for Macintosh.
SetRectRgn	Standard Quickdraw function for Macintosh.
GetClip	Standard Quickdraw function for Macintosh.
SetClip	Standard Quickdraw function for Macintosh.
LDispose	Standard List Manager function for Macintosh.
setuphtopic	See Section 2.22.1.19.16.
DisposDialog	Standard Dialog Manager function for Macintosh.
DisposeRgn	Standard Quickdraw function for Macintosh.
GlobalToLocal	Standard Quickdraw function for Macintosh.
LClick	Standard List Manager function for Macintosh.
PtInRect	Standard Quickdraw function for Macintosh.
GetDItem	Standard Dialog Manager function for Macintosh.
HiliteControl	Standard Control Manager function for Macintosh.

Table 2.22-47 hpickevents Information.

2.22.1.19.7 drawlist

drawlist is called by the dialog manager to update the topic list. The function call is drawlist(dptr, itemno). Table 2.22-48 describes the parameters used and functions called using this function. *dptr* is a pointer to the dialog box to use; *itemno* is the item number of the topic list.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
dptr	DialogPtr	Development:THINK C: Mac #includes:DialogMgr.h
itemno	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
fontnum	int	Standard C type.
orgfont	int	Standard C type.
orgsize	int	Standard C type.
r	Rect	Development:THINK C: Mac #includes:MacTypes.h

Calls	
Function	**Where Described**
PenSize	Standard Quickdraw function for Macintosh.
InsetRect	Standard Quickdraw function for Macintosh.
FrameRect	Standard Quickdraw function for Macintosh.
GetFNum	Standard Font Manager function for Macintosh.
TextFont	Standard Quickdraw function for Macintosh.
TextSize	Standard Quickdraw function for Macintosh.
LUpdate	Standard List Manager function for Macintosh.

Table 2.22-48 drawlist Information.

2.22.1.19.8 setuphpick

setuphpick setups the dialog box to let the user choose a topic. The function call is setuphpick(dptr). Table 2.22-49 describes the parameters used and functions called using this function. *dptr* is a pointer to the dialog box to use.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
dptr	DialogPtr	Development:THINK C: Mac #includes:DialogMgr.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
curcell	Cell	Development:THINK C: Mac #includes:ListMgr.h
curname	pointer to char	Standard C type.
fontnum	int	Standard C type.
hiliteindex	int	Standard C type.
itemhandle	Handle	Development:THINK C: Mac #includes:MacTypes.h
itemrect	Rect	Development:THINK C: Mac #includes:MacTypes.h
itemtype	int	Standard C type.
orgfont	int	Standard C type.
orgsize	int	Standard C type.
state	int	Standard C type.
topicbounds	Rect	Development:THINK C: Mac #includes:MacTypes.h
topicsize	Point	Development:THINK C: Mac #includes:MacTypes.h

Calls	
Function	**Where Described**
helpidindex	See Section 2.22.1.19.18.
GetDItem	Standard Dialog Manager function for Macintosh.
SetCTitle	Standard Control Manager function for Macintosh.
HiliteControl	Standard Control Manager function for Macintosh.
SetIText	Standard Dialog Manager function for Macintosh.
EraseRect	Standard Quickdraw function for Macintosh.
SetRect	Standard Quickdraw function for Macintosh.
GetFNum	Standard Font Manager function for Macintosh.
TextFont	Standard Quickdraw function for Macintosh.
TextSize	Standard Quickdraw function for Macintosh.
LNew	Standard List Manager function for Macintosh.
LDoDraw	Standard List Manager function for Macintosh.
SetDItem	Standard Dialog Manager function for Macintosh.
HLock	Standard Memory Manager function for Macintosh.
LSetCell	Standard List Manager function for Macintosh.
HUnlock	Standard Memory Manager function for Macintosh.
LSetSelect	Standard List Manager function for Macintosh.
LAutoScroll	Standard List Manager function for Macintosh.
LActivate	Standard List Manager function for Macintosh.
ShowWindow	Standard Window Manager function for Macintosh.
SetWRefCon	Standard Dialog Manager function for Macintosh.
Called By	
Function	**Where Described**
showhelp	See Section 2.22.1.19.4.
htopicevents	See Section 2.22.1.19.14.
setuphtopic	See Section 2.22.1.19.16.

Table 2.22-49 setuphpick Information.

2.22.1.19.9 redrawhelp

redrawhelp redraws the chunks that are in the redraw area. The function call is redrawhelp(dptr, top, bottom). Table 2.22-50 describes the parameters used and functions called using this function. *dptr* is a pointer to the dialog containing the redraw area; *top* is the top of the area to redraw;*bottom* is the bottom of the area to redraw.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
dptr	DialogPtr	Development:THINK C: Mac #includes:DialogMgr.h
top	int	Standard C type.
bottom	int	Standard C type.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
curchunk	int	Standard C type.
curorigin	int	Standard C type.
destrect	Rect	Development:THINK C: Mac #includes:MacTypes.h
hpicthdl	helppicthdl	Development:SIMNET:libmac: helplocal.h
orgface	Style	Development:THINK C: Mac #includes:Quickdraw.h
orgfont	int	Standard C type.
orgsize	int	Standard C type.
picthdl	PicHandle	Development:THINK C: Mac #includes:Quickdraw.h
topichandle	Handle	Development:THINK C: Mac #includes:MacTypes.h
topicheight	int	Standard C type.
topicsptr	topicchunkptr	Development:SIMNET:libmac: helplocal.h
topictype	int	Standard C type.
Calls		
Function	**Where Described**	
TEUpdate	Standard TextEdit function for Macintosh.	
TextFont	Standard Quickdraw function for Macintosh.	
TextSize	Standard Quickdraw function for Macintosh.	
TextFace	Standard Quickdraw function for Macintosh.	
DrawPicture	Standard Quickdraw function for Macintosh.	
Called By		
Function	**Where Described**	
scrollto	See Section 2.22.1.19.11.	
drawtopic	See Section 2.22.1.19.15.	

Table 2.22-50 redrawhelp Information.

2.22.1.19.10 adjustchunks

adjustchunks adjusts the locations of all the chunks. *dist* is the amount by which to adjust the chunks. The function call is adjustchunks(dist). Table 2.22-51 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
dist	int	Standard C type.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
curchunk	int	Standard C type.
hpicthdl	helppicthdl	Development:SIMNET:libmac: helplocal.h
tehdl	TEHandle	Development:THINK C: Mac #includes:TextEdit.h
topichandle	Handle	Development:THINK C: Mac #includes:MacTypes.h
topicsptr	topicchunkptr	Development:SIMNET:libmac: helplocal.h
topictype	int	Standard C type.
Calls		
Function	**Where Described**	
OffsetRect	Standard Quickdraw function for Macintosh.	
Called By		
Function	**Where Described**	
scrollto	See Section 2.22.1.19.11.	

Table 2.22-51 adjustchunks Information.

2.22.1.19.11 scrollto

scrollto scrolls the help topic to the *neworigin*. *dptr* is the pointer to the dialog containing control and *neworigin* is the origin to scroll the topic to. The function call is scrollto(dptr, neworigin). Table 2.22-52 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
dptr	DialogPtr	Development:THINK C: Mac #includes:DialogMgr.h
neworigin	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
bottom	int	Standard C type.
itemhandle	Handle	Development:THINK C: Mac #includes:MacTypes.h
itemrect	Rect	Development:THINK C: Mac #includes:MacTypes.h
itemtype	int	Standard C type.
newvalue	unsigned long	Standard C type.
scrolldist	int	Standard C type.
testdist	int	Standard C type.
top	int	Standard C type.

Calls	
Function	**Where Described**
adjustchunks	See Section 2.22.1.19.10.
GetClip	Standard Quickdraw function for Macintosh.
ClipRect	Standard Quickdraw function for Macintosh.
EraseRect	Standard Quickdraw function for Macintosh.
redrawhelp	See Section 2.22.1.19.9.
SetClip	Standard Quickdraw function for Macintosh.
ScrollRect	Standard Quickdraw function for Macintosh.
SetCtlValue	Standard Control Manager function for Macintosh.
Called By	
Function	**Where Described**
trytoscroll	See Section 2.22.1.19.12.
htopicevents	See Section 2.22.1.19.14.

Table 2.22-52 scrollto Information.

2.22.1.19.12 trytoscroll

trytoscroll is called by TrackControl (indirectly) to try to scroll the topic by the amount specified. *dptr* is a pointer to the dialog containing control and *count* is the number of pixels to scroll the window. The function call is trytoscroll(dptr, count). Table 2.22-53 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
dptr	DialogPtr	Development:THINK C: Mac #includes:DialogPtr.h
count	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
neworigin	int	Standard C type.
Calls		
Function	**Where Described**	
scrollto	See Section 2.22.1.19.11.	
Called By		
Function	**Where Described**	
helpmove	See Section 2.22.1.19.13.	

Table 2.22-53 trytoscroll Information.

2.22.1.19.13 helpmove

helpmove is called by TrackControl to scroll the topic. *ctrlhdl* is the handle to the scroll bar and *part* is the part number where the user clicked. The function call is helpmove(ctrlhdl, part). Table 2.22-54 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
ctrlhdl	ControlHandle	Development:THINK C: Mac #includes:ControlMgr.h
part	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
dptr	DialogPtr	Development:THINK C: Mac #includes:DialogMg.h
Calls		
Function	**Where Described**	
trytoscroll	See Section 2.22.1.19.12.	

Table 2.22-54 helpmove Information.

2.22.1.19.14 htopicevents

htopicevents is the event handler for the htopic dialog box. *wptr* is the pointer to the dialog box window and *eventptr* is the pointer to the event record. The function call is htopicevents(wptr, eventptr). Table 2.22-55 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
wptr	WindowPtr	Development:THINK C: Mac #includes:WindowPtr.h
eventptr	pointer to EventRecord	Development:THINK C: Mac #includes:EventMgr.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
itemhit	int	Standard C type.
lastpart	int	Standard C type.
newindex	int	Standard C type.
neworigin	unsigned long	Standard C type.
newvalue	unsigned long	Standard C type.
topicptr	pointer to int	Standard C type.
userclick	Point	Development:THINK C: Mac #includes:MacTypes.h
whichcontrol	ControlHandle	Development:THINK C: Mac #includes:ControlMgr.h
whichpart	int	Standard C type.

Calls	
Function	**Where Described**
HiliteControl	Standard Control Manager function for Macintosh.
DialogSelect	Standard Dialog Manager function for Macintosh.
disposetopic	See Section 2.22.1.19.17.
SetRectRgn	Standard Quickdraw function for Macintosh.
GetClip	Standard Quickdraw function for Macintosh.
SetClip	Standard Quickdraw function for Macintosh.
DisposeControl	Standard Control Manager function for Macintosh.
setuphpick	See Section 2.22.1.19.8.
DisposDialog	Standard Dialog Manager function for Macintosh.
DisposeRgn	Standard Quickdraw function for Macintosh.
helpidindex	See Section 2.22.1.19.18.
setuphtopic	See Section 2.22.1.19.16.
GlobalToLocal	Standard Quickdraw function for Macintosh.
FindControl	Standard Control Manager function for Macintosh.
TrackControl	Standard Control Manager function for Macintosh.
GetCtlValue	Standard Control Manager function for Macintosh.
scrollto	See Section 2.22.1.19.11.

Table 2.22-55 htopicevents Information.

2.22.1.19.15 drawtopic

drawtopic is called by the dialog manager to display a topic. *dptr* is a pointer to the dialog box to use and *itemno* is the item number of the topic item. The function call is drawtopic(dptr, itemno). Table 2.22-56 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
dptr	DialogPtr	Development:THINK C: Mac #includes:DialogMgr.h
itemno	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
itemhandle	Handle	Development:THINK C: Mac #includes:MacTypes.h
itemrect	Rect	Development:THINK C: Mac #includes:MacTypes.h
itemtype	int	Standard C type.

Calls	
Function	**Where Described**
GetClip	Standard Quickdraw function for Macintosh.
GetDItem	Standard Dialog Manager function for Macintosh.
InsetRect	Standard Quickdraw function for Macintosh.
PenSize	Standard Quickdraw function for Macintosh.
FrameRect	Standard Quickdraw function for Macintosh.
DrawControls	Standard Control Manager function for Macintosh.
ClipRect	Standard Quickdraw function for Macintosh.
redrawhelp	See Section 2.22.1.19.9.
SetClip	Standard Quickdraw function for Macintosh.

Table 2.22-56 drawtopic Information.

2.22.1.19.16 setuphtopic

setuphtopic displays the specified help topic dialog box. *dptr* is the pointer to the dialog box to use. The function call is setuphtopic(dptr). Table 2.22-57 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
dptr	DialogPtr	Development:THINK C: Mac #includes:DialogMgr.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
itemhandle	Handle	Development:THINK C: Mac #includes:MacTypes.h
itemrect	Rect	Development:THINK C: Mac #includes:MacTypes.h
itemtype	int	Standard C type.
scrollrect	Rect	Development:THINK C: Mac #includes:MacTypes.h
scrollstate	int	Standard C type.
titlelen	unsigned long	Standard C type.

Calls	
Function	**Where Described**
GetResource	Standard Resource Manager function for Macintosh.
ShowCaution	See Section 2.22.1.4.1.
setuphpick	See Section 2.22.1.19.8.
HNoPurge	Standard Memory Manager function for Macintosh.
BlockMove	Standard Memory Manager function for Macintosh.
GetDItem	Standard Dialog Manager function for Macintosh.
SetCTitle	Standard Control Manager function for Macintosh.
HiliteControl	Standard Control Manager function for Macintosh.
SetIText	Standard Dialog Manager function for Macintosh.
EraseRect	Standard Quickdraw function for Macintosh.
SetDItem	Standard Dialog Manager function for Macintosh.
InsetRect	Standard Quickdraw function for Macintosh.
NewControl	Standard Control Manager function for Macintosh.
gethelp	See Section 2.22.1.19.19.
SetCtlValue	Standard Control Manager function for Macintosh.
ShowControl	Standard Control Manager function for Macintosh.
ShowWindow	Standard Window Manager function for Macintosh.
InvalRect	Standard Window Manager function for Macintosh.
SetWRefCon	Standard Window Manager function for Macintosh.
Called By	
Function	**Where Described**
showhelp	See Section 2.22.1.19.4.
hpickevents	See Section 2.22.1.19.6.
htopicevents	See Section 2.22.1.19.14.

Table 2.22-57 setuphtopic Information.

2.22.1.19.17 disposetopic

disposetopic cleans up after finishing with a topic. The function call is disposetopic(). Table 2.22-58 describes the parameters used and functions called using this function.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
curchunk	int	Standard C type.
topicsptr	topicchunkptr	Development:SIMNET:libmac: helplocal.h
topichandle	Handle	Development:THINK C: Mac #includes:MacTypes.h
topictype	int	Standard C type.
Calls		
Function	**Where Described**	
HPurge	Standard Memory Manager function for Macintosh.	
ReleaseResource	Standard Resource Manager function for Macintosh.	
TEDispose	Standard TextEdit function for Macintosh.	
DisposHandle	Standard Memory Manager function for Macintosh.	

Called By	
Function	**Where Described**
htopicevents	See Section 2.22.1.19.14.

Table 2.22-58 disposetopic Information.

2.22.1.19.18 helpidindex

helpidindex returns the index for a given id. *helpid* is the id of the help resource to find. The function call is helpidindex(helpid). Table 2.22-59 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
helpid	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
helpindex	int	Standard C type.
retindex	int	Standard C type.
topicptr	pointer to int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
retindex	int	Index number of the id.
Called By		
Function	**Where Described**	
setuphpick	See Section 2.22.1.19.8.	
htopicevents	See Section 2.22.1.19.14.	

Table 2.22-59 helpidindex Information.

2.22.1.19.19 gethelp

gethelp sets up the topic data, given a help handle. *helphdl* is the handle of the help resource to setup. The function call is gethelp(helphdl, destrect, controlstate, scrollmax). Table 2.22-60 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
helphdl	Handle	Development:THINK C: Mac #includes:MacTypes.h
destrect	pointer to Rect	Development:THINK C: Mac #includes:MacTypes.h
controlstate	pointer to int	Standard C type.
scrollmax	pointer to int	Standard C type.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
chunkptr	hchunkptr	Development:SIMNET:libmac: helplocal.h
curchunk	int	Standard C type.
curorigin	int	Standard C type.
hchunkid	int	Standard C type.
hchunktype	ResType	Development:THINK C: Mac #includes:MacTypes.h
helpptr	pointer to char	Standard C type.
skipcount	unsigned int	Standard C type.
topicshdl	Handle	Development:THINK C: Mac #includes:MacTypes.h
topicsptr	topicchunkptr	Development:SIMNET:libmac: helplocal.h
topichandle	Handle	Development:THINK C: Mac #includes:MacTypes.h
topicheight	int	Standard C type.
topictype	int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
topicshdl	Handle	Handle to topic structure.
Calls		
Function	**Where Described**	
NewHandle	Standard Memory Manager function for Macintosh.	
createte	See Section 2.22.1.19.22.	
createhpict	See Section 2.22.1.19.23.	
Called By		
Function	**Where Described**	
setuphtopic	See Section 2.22.1.19.16.	

Table 2.22-60 gethelp Information.

2.22.1.19.20 getcontenthdl

getcontenthdl returns the contents handle for a variable. *varname* is a pointer to the string containing the variable name and *varnamelen* is the length of the variable name. The function call is getcontenthdl(varname, varnamelen). Table 2.22-61 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
varname	pointer to char	Standard C type.
varnamelen	int	Standard C type.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
curchar	int	Standard C type.
curlen	int	Standard C type.
curpos	long	Standard C type.
curvar	pointer to char	Standard C type.
match	Boolean	Development:THINK C: Mac #includes:MacTypes.h
rethdl	Handle	Development:THINK C: Mac #includes:MacTypes.h
temp1ptr	pointer to char	Standard C type.
varfound	Boolean	Development:THINK C: Mac #includes:MacTypes.h
varslen	long	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
NIL	Handle	Variable not found.
rethdl	Handle	Handle to contents for variable.
Calls		
Function	**Where Described**	
GetHandleSize	Standard Memory Manager function for Macintosh.	
Called By		
Function	**Where Described**	
replacevars	See Section 2.22.1.19.21.	

Table 2.22-61 getcontenthdl Information.

2.22.1.19.21 replacevars

replacevars replaces variables with their contents. *txthdl* is the handle to the text for the chunk and *length* is a pointer to the length of text in the handle. The function call is replacevars(txthdl, length). Table 2.22-62 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
txthdl	Handle	Development:THINK C: Mac #includes:MacTypes.h
length	pointer to int	Standard C type.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
contenthdl	Handle	Development:THINK C: Mac #includes:MacTypes.h
contentlen	int	Standard C type.
curloc	int	Standard C type.
lastend	int	Standard C type.
laststart	int	Standard C type.
lengthchange	int	Standard C type.
orglength	unsigned long	Standard C type.
txtptr	pointer to char	Standard C type.
varcomplete	Boolean	Development:THINK C: Mac #includes:MacTypes.h
varlen	int	Standard C type.
varloc	int	Standard C type.
varstart	pointer to char	Standard C type.
Calls		
Function	**Where Described**	
getcontenthdl	See Section 2.22.1.19.20.	
GetHandleSize	Standard Memory Manager function for Macintosh.	
BlockMove	Standard Memory Manager function for Macintosh.	
SetHandleSize	Standard Memory Manager function for Macintosh.	
Called By		
Function	**Where Described**	
createte	See Section 2.22.1.19.22.	

Table 2.22-62 replacevars Information.

2.22.1.19.22 createte

createte creates a textedit record from an 'htxt' resource. *hchunkid* is the resource id of the 'htxt' resource. *destrect* is the rectangle of user area to fit it into and *ystart* is the y position to display the text relative to the topic. *height* returns the height of the textedit record here. The function call is createte(hchunkid, destrect, ystart, height) Table 2.22-63 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
hchunkid	int	Standard C type
destrect	pointer to Rect	Development:THINK C: Mac #includes:MacTypes.h
ystart	int	Standard C type.
height	pointer to int	Standard C type.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
fontinfo	FontInfo	Development:THINK C: Mac #includes:Quickdraw.h
fontnum	int	Standard C type.
length	int	Standard C type.
orgface	Style	Development:THINK C: Mac #includes:Quickdraw.h
orgfont	int	Standard C type.
orgsize	int	Standard C type.
realheight	int	Standard C type.
style	int	Standard C type.
tedrect	Rect	Development:THINK C: Mac #includes:MacTypes.h
tehdl	TEHandle	Development:THINK C: Mac #includes:TextEdit.h
tevrect	Rect	Development:THINK C: Mac #includes:MacTypes.h
txthandle	Handle	Development:THINK C: Mac #includes:macTypes.h
txtptr	pointer to char	Standard C type.

Return Values		
Return Value	**Type**	**Meaning**
tehdl	TEHandle	Handle to textedit record.

Calls	
Function	**Where *Described***
GetResource	Standard Resource Manager function for Macintosh.
replacevars	See Section 2.22.1.19.21.
OffsetRect	Standard Quickdraw function for Macintosh.
InsetRect	Standard Quickdraw function for Macintosh.
GetFNum	Standard Font Manager function for Macintosh.
TextFont	Standard Quickdraw function for Macintosh.
TextSize	Standard Quickdraw function for Macintosh.
TextFace	Standard Quickdraw function for Macintosh.
GetFontInfo	Standard Quickdraw function for Macintosh.
TENew	Standard TextEdit function for Macintosh.
HLock	Standard Memory Manager function for Macintosh.
TESetText	Standard TextEdit function for Macintosh.
HUnlock	Standard Memory Manager function for Macintosh.
ReleaseResource	Standard Resource Manager function for Macintosh.

Called By	
Function	**Where Described**
gethelp	See Section 2.22.1.19.19.

Table 2.22-63 createte Information.

2.22.1.19.23 createhpict

createhpict creates a quickdraw picture from an 'hpic' resource. *hchunkid* is the resource id of the 'hpic' resource. *destrect* is the rectangle of user area to fit it into and *ystart* is the y position to display the picture relative to the topic. *height* returns the height of the picture here. The function call is createhpict(hchunkid, destrect, ystart, height). Table 2.22-64 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
hchunkid	int	Standard C type.
destrect	pointer to Rect	Development:THINK C: Mac #includes:MacTypes.h
ystart	int	Standard C type.
height	pointer to int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
destwidth	int	Standard C type.
hoffset	int	Standard C type.
hpichandle	Handle	Development:THINK C: Mac #includes:MacTypes.h
hpicthdl	helppicthdl	Development:SIMNET:libmac: helplocal.h
oserr	OSErr	Development:THINK C: Mac #includes:MacTypes.h
pictdestrect	Rect	Development:THINK C: Mac #includes:MacTypes.h
pictrect	Rect	Development:THINK C: Mac #includes:MacTypes.h
pickwidth	int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
hpicthdl	helppicthdl	Handle to quickdraw picture.
Calls		
Function	**Where Described**	
GetResource	Standard Resource Manager function for Macintosh.	
HNoPurge	Standard Memory Manager function for Macintosh.	
NewHandle	Standard Memory Manager function for Macintosh.	
HLock	Standard Memory Manager function for Macintosh.	
HandToHand	Standard Operating System Utility function for Macintosh.	
HUnlock	Standard Memory Manager function for Macintosh.	
DisposHandle	Standard Memory Manager function for Macintosh.	
OffsetRect	Standard Quickdraw function for Macintosh.	
HPurge	Standard Memory Manager function for Macintosh.	
ReleaseResource	Standard Resource Manager function for Macintosh.	
Called By		
Function	**Where Described**	
gethelp	See Section 2.22.1.19.19.	

Table 2.22-64 createhpict Information.

2.22.1.20 init.c
Development:SIMNET:libmac:init.c

init.c contains a routine for initializing the Macintosh toolbox.

2.22.1.20.1 InitToolbox

InitToolbox initializes the Macintosh toolbox. The function call is InitToolbox(masters, restartProc). Table 2.22-65 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
masters	int	Standard C type.
restartProc	ProcPtr	Development:THINK C: Mac #includes:MacTypes.h
Calls		
Function	**Where Described**	
MaxApplZone	Standard Memory Manager function for Macintosh.	
MoreMasters	Standard Memory Manager function for Macintosh.	
InitGraf	Standard Quickdraw function for Macintosh.	
InitFonts	Standard Font Manager function for Macintosh.	
FlushEvents	Standard Operating System Event Manager function for Macintosh.	
InitWindows	Standard Window Manager function for Macintosh.	
TEInit	Standard TextEdit function for Macintosh.	
InitDialogs	Standard Dialog Manager function for Macintosh.	
InitCursor	Standard Quickdraw function for Macintosh.	
SetCursor	Standard Quickdraw function for Macintosh.	

Table 2.22-65 InitToolbox Information.

2.22.1.21 install.c
Development:SIMNET:libmac:install.c

install.c contains a routine for installing an entry into a scrolling table.

2.22.1.21.1 InstallScrollTableEntry

InstallScrollTableEntry either inserts or reinserts an entry, *entry*, into a scrolling table, specifed by *defn*. The function call is InstallScrollTableEntry(defn, entry). Table 2.22-66 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
defn	register pointer to ScrollTableFieldDefn	Development:SIMNET:libmac: scroll.h
entry	register ScrollTableEntryHandle	Development:SIMNET:libmac: scroll.h

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
key	unsigned long	Standard C type.
oldRow	int	Standard C type.
newRow	int	Standard C type.
i	int	Standard C type.
r	Rect	Development:THINK C: Mac #includes:MacTypes.h
e	ScrollTableEntryHandle	Development:SIMNET:libmac: scroll.h
savePort	GrafPtr	Development:THINK C: Mac #includes:Quickdraw.h

Calls	
Function	**Where Described**
ScrollTableEntryToRow	See Section 2.22.1.34.2.
NextPrev	Macro defined in select.c in Section 2.22.1.37.
PrevNext	Macro defined in select.c in Section 2.22.1.37.
SetCtlMax	Standard Control Manager function for Macintosh.
GetPort	Standard Quickdraw function for Macintosh.
SetPort	Standard Quickdraw function for Macintosh.
InvalRect	Standard Window Manager function for Macintosh.

Table 2.22-66 InstallScrollTableEntry Information.

2.22.1.22 longpt.h
Development:SIMNET:libmac:longpt.h

longpt.h defines data types for Cartesian points and rectangles represented using 32-bit integer coordinates.

2.22.1.23 longpt.c
Development:SIMNET:libmac:longpt.c

longpt.c contains routines for operating on points represented using 32-bit integer coordinates.

2.22.1.23.1 isqrt

isqrt is an integer square root algorithm. The alogrithm works by noting that the square root of an integer is the smallest of the two most nearly equal factors of that number, i.e. the 2 most equal roots of 255 are 15 and 17, and so 15 is the square of 255. The function call is isqrt(numb). Table 2.22-67 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
numb	register long	Standard C type.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
guess1	register long	Standard C type.
guess2	register long	Standard C type.
error	register long	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
guess1	register long	Square root.
numb	register long	numb is returned when negative because square root can not be computed.
Called By		
Function	**Where Described**	
DistBetween2Pts	See Section 2.22.1.23.2.	

Table 2.22-67 isqrt Information.

2.22.1.23.2 DistBetween2Pts

DistBetween2Pts computes the distance between two points, *pt1* and *pt2*, using only integers. The function call is DistBetween2Pts(pt1, pt2). Table 2.22-68 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
pt1	LongPt	Development:SIMNET:libmac:longpt.h
pt2	LongPt	Development:SIMNET:libmac:longpt.h
Return Values		
Return Value	**Type**	**Meaning**
	long	Return value from called to isqrt.
Calls		
Function	**Where Described**	
isqrt	See Section 2.22.1.23.1.	

Table 2.22-68 DistBetween2Pts Information.

2.22.1.23.3 PercentPt

PercentPt locates a point, *retPt*, a given percentage distant , *percent*, between a start and an end point, points *ptl* and *pt2*. The function call is PercentPt(pt1, pt2, percent, retPt). Table 2.22-69 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
pt1	LongPt	Development:SIMNET:libmac: longpt.h
pt2	LongPt	Development:SIMNET:libmac: longpt.h
percent	long	Standard C type.
retPt	pointer to LongPt	Development:SIMNET:libmac: longpt.h

Table 2.22-69 PercentPt Information.

2.22.1.23.4 InterpolatePoints

InterpolatePoints locates the point some fraction of the way between two other points. The function call is InterpolatePoints(pt1, pt2, num, denom, ptInt). Table 2.22-70 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
pt1	pointer to LongPt	Development:SIMNET:libmac: longpt.h
pt2	pointer to LongPt	Development:SIMNET:libmac: longpt.h
num	long	Standard C type.
denom	long	Standard C type.
ptInt	pointer to LongPt	Development:SIMNET:libmac: longpt.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
fract	float	Standard C type.

Table 2.22-70 InterpolatePoints Information.

2.22.1.23.5 SetLongPt

SetLongPt initializes a LongPt value. The function call is SetLongPt(pt, x, y). Table 2.22-71 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
pt	pointer to LongPt	Development:SIMNET:libmac: longpt.h
x	long	Standard C type.
y	long	Standard C type.

Table 2.22-71 SetLongPt Information.

2.22.1.24 lookup.c
Development:SIMNET:libmac:lookup.c

lookup.c contains a routine for locating an entry in a scrolling table.

2.22.1.24.1 LookupScrollTableEntry

LookupScrollTableEntry locates a scrolling table entry by its key. The function call is LookupScrollTableEntry(defn, key). Table 2.22-72 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
defn	pointer to ScrollTableFieldDefn	Development:SIMNET:libmac:scroll.h
key	unsigned long	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
e	register ScrollTableEntryHandle	Development:SIMNET:libmac:scroll.h
Return Values		
Return Value	**Type**	**Meaning**
e	register ScrollTableEntryHandle	Handle of scrolling table entry.

Table 2.22-72 LookupScrollTableEntry Information.

2.22.1.25 map.h
Development:SIMNET:libmac:map.h

map.h defines data types for representing the mapping between UTM coordinates and Cartesian world coordinates. Table 2.22-73 describes the variables used by map.h.

Variables		
Variable	**Type**	**Where Typedef Declared**
terrainMap	extern TerrainMap	Development:SIMNET:libmac:map.h

Table 2.22-73 map.h Variable Information.

2.22.1.26 map.c
Development:SIMNET:libmac:map.c

map.c contains routines for converting between UTM coordinates and Cartesian world coordinates. Table 2.22-74 describes the variables used by map.c.

Variables		
Variable	**Type**	**Where Typedef Declared**
terrainMap	TerrainMap	Development:SIMNET:libmac:map.h
scale[]	long int	Standard C type.

Table 2.22-74 map.c Variable Information.

2.22.1.26.1 StringToMapCoordinates

StringToMapCoordinates parses a string, *str*, and converts it to a MapCoordinates object, *map*. The function call is StringToMapCoordinates(map, str). Table 2.22-75 describes the parameters used and errors returned using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
map	pointer to MapCoordinates	Development:SIMNET:libmac:map.h
str	register pointer to char	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
atol()	extern function returning long	Standard C type.
i	register short	Standard C type.
ch	register short	Standard C type.
zone	register pointer to GridZone	Development:SIMNET:libmac:map.h
prec	short	Standard C type.
pt	LongPt	Development:SIMNET:libmac:longpt.h
xStr[6]	char	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
-1	int	badMapStringErr
-2	int	badMapLetterErr
-3	int	outOfGridZoneErr
0	int	Successful
Errors		
Error Name	**Reason for Error**	
badMapStringErr	String too long or contains wrong number of digits.	
badMapLetterErr	String letter pair doesn't designate a correct grid zone.	
outOfGridZoneErr	Coordinates are not within given grid zone.	

Called By	
Function	**Where Described**
CheckLastField	See Section 2.22.1.11.5.

Table 2.22-75 StringToMapCoordinates Information.

2.22.1.26.2 PointToMapCoordinates

PointToMapCoordinates builds a MapCoordinates object from a set of Cartesian coordinates identifying a point on the terrain. The function call is PointToMapCoordinates(map, pt, prec). Table 2.22-76 describes the parameters used and errors returned using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
map	register pointer to MapCoordinates	Development:SIMNET:libmac: map.h
pt	*pointer to LongPt*	Development:SIMNET:libmac: longpt.h
prec	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	register short	Standard C type.
n	register long	Standard C type.
cp	register pointer to char	Standard C type.
p	LongPt	Development:SIMNET:libmac: *longpt.h*
zone	pointer to GridZone	Development:SIMNET:libmac: map.h
punctuate	short	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
-4	int	*noGridZoneErr*
0	int	Successful
Errors		
Error Name	**Reason for Error**	
noGridZoneErr	Grid zone not present.	

Table 2.22-76 PointToMapCoordinates Information.

2.22.1.27 outline.c

Development:SIMNET:libmac:outline.c

outline.c contains routines for outlining a dialog's default button.

2.22.1.27.1 DrawItemOutline

DrawItemOutline is the draw procedure, called by the Dialog Manager. *w* is the pointer to the window the item is in; *itemNo* is the number of the item to be outlined. The function call is DrawItemOutline(w, itemNo). Table 2.22-77 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
w	WindowPtr	Development:THINK C: Mac #includes:WindowMgr.h
itemNo	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	int	Standard C type.
theItem	Handle	Development:THINK C: Mac #includes:MacTypes.h
r	Rect	Development:THINK C: Mac #includes:MacTypes.h
Calls		
Function	**Where Described**	
GetDItem	Standard Dialog Manager function for Macintosh.	
SetPort	Standard Quickdraw function for Macintosh.	
PenNormal	Standard Quickdraw function for Macintosh.	
PenSize	Standard Quickdraw function for Macintosh.	
FrameRoundRect	Standard Quickdraw function for Macintosh.	

Table 2.22-77 DrawItemOutline Information.

2.22.1.27.2 OutlineItem

OutlineItem is called to tie a drawing function to userItem. The function call is OutlineItem(dialog, itemNo). Table 2.22-78 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
dialog	register DialogPtr	Development:THINK C: Mac #includes:DialogMgr.h
itemNo	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	int	Standard C type.
theItem	Handle	Development:THINK C: Mac #includes:MacTypes.h
r	Rect	Development:THINK C: Mac #includes:MacTypes.h
cp	register pointer to char	Standard C type.

Calls	
Function	**Where Described**
GetDItem	Standard Dialog Manager function for Macintosh.
InsetRect	Standard Quickdraw function for Macintosh.
GetHandleSize	Standard Memory Manager function for Macintosh.
SetHandleSize	Standard Memory Manager function for Macintosh.
MemError	Standard Memory Manager function for Macintosh.
Called By	
Function	**Where Described**
ShowCaution	See Section 2.22.1.4.1.
ShowDialog	See Section 2.22.1.11.1.
showhelp	See Section 2.22.1.19.4.
ShowSimpleDialog	See Section 2.22.1.40.1.

Table 2.22-78 OutlineItem Information.

2.22.1.28 remove.c

Development:SIMNET:libmac:remove.c

remove.c contains a routine for removing an entry from a scrolling table.

2.22.1.28.1 RemoveScrollTableEntry

RemoveScrollTableEntry removes an entry, *e*, from a table, specified by *defn*, and discards it. The function call is RemoveScrollTableEntry(defn, e). Table 2.22-79 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
defn	register pointer to ScrollTableFieldDefn	Development:SIMNET:libmac: scroll.h
e	register ScrollTableEntryHandle	Development:SIMNET:libmac: scroll.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
savePort	GrafPtr	Development:THINK C: Mac #includes:Quickdraw.h
row	int	Standard C type.
i	int	Standard C type.
r	Rect	Development:THINK C: Mac #includes:MacTypes.h

Calls	
Function	**Where Described**
DisposHandle	Standard Memory Manager function for Macintosh.
GetPort	Standard Quickdraw function for Macintosh.
SetPort	Standard Quickdraw function for Macintosh.
SetCtlMax	Standard Control Manager function for Macintosh.
InvalRect	Standard Window Manager function for Macintosh.
ScrollTableEntryToRow	See Section 2.22.1.34.2.
NextPrev	Macro defined in remove.c in Section 2.22.1.28.
PrevNext	Macro defined in remove.c in Section 2.22.1.28.
Calls	
Function	**Where Described**
EmptyScrollTable	See Section 2.22.1.28.2.

Table 2.22-79 RemoveScrollTableEntry Information.

2.22.1.28.2 EmptyScrollTable

EmptyScrollTable removes all entries from a scrolling table, *defn*. The function call is EmptyScrollTable(defn). Table 2.22-80 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
defn	register pointer to ScrollTableFieldDefn	Development:SIMNET:libmac: scroll.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
e	register ScrollTableEntryHandle	Development:SIMNET:libmac: scroll.h
Calls		
Function	**Where Described**	
RemoveScrollTableEntry	See Section 2.22.1.28.1.	

Table 2.22-80 EmptyScrollTable Information.

2.22.1.29 scroll.h

Development:SIMNET:libmac:scroll.h

scroll.h is the header file associated with the following files which collectively provide support for scrolling tables within dialog boxes.

2.22.1.30 scroll.c

Development:SIMNET:libmac:scroll.c

scroll.c contains routines for creating and discarding scrolling tables. Table 2.22-81 describes the variables used by scroll.c.

Variables		
Variable	**Type**	**Where Typedef Declared**
DrawScrollTable()	extern pascal void	

Table 2.22-81 scroll.c Variable Information.

2.22.1.30.1 ShowScrollTable

ShowScrollTable prepares a scrolling table field, specified by *defn*, of a new dialog box, *dialog*. The function call is ShowScrollTable(dialog, defn). Table 2.22-82 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
dialog	pointer to DialogState	Development:SIMNET:libmac: dialog.h
defn	register pointer to ScrollTableFieldDefn	Development:SIMNET:libmac: scroll.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
savePort	GrafPtr	Development:THINK C: Mac #includes:Quickdraw.h
col	register pointer to ScrollTableColunDefn	Development:SIMNET:libmac: scroll.h
c	short	Standard C type.
w	short	Standard C type.
hSize	short	Standard C type.
vSize	short	Standard C type.
i	int	Standard C type.
box	Rect	Development:THINK C: Mac #includes:MacTypes.h
theitem	Handle	Development:THINK C: Mac #includes:MacTypes.h
entry	ScrollTableEntryHandle	Development:SIMNET:libmac: scroll.h

Calls	
Function	**Where Described**
GetPort	Standard Quickdraw function for Macintosh.
SetPort	Standard Quickdraw function for Macintosh.
TextFont	Standard Quickdraw function for Macintosh.
TextSize	Standard Quickdraw function for Macintosh.
GetFontInfo	Standard Quickdraw function for Macintosh.
GetDItem	Standard Dialog Manager function for Macintosh.
StringWidth	Standard Quickdraw function for Macintosh.
SetDItem	Standard Dialog Manager function for Macintosh.
NewControl	Standard Control Manager function for Macintosh.
NewHandle	Standard Memory Manager function for Macintosh.
Called By	
Function	**Where Described**
ShowDialog	See Section 2.22.1.11.1.

Table 2.22-82 ShowScrollTable Information.

2.22.1.30.2 DisposeScrollTable

DisposeScrollTable discards the storage occupied by a scrolling table, *defn*. The function call is DisposeScrollTable(defn). Table 2.22-83 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
defn	register pointer to ScrollTableFieldDefn	Development:SIMNET:libmac: scroll.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
e1	register ScrollTableEntryHandle	Development:SIMNET:libmac: scroll.h
e2	register ScrollTableEntryHandle	Development:SIMNET:libmac: scroll.h
Calls		
Function	**Where Described**	
DisposHandle	Standard Memory Manager function for Macintosh.	
DisposeControl	Standard Control Manager function for Macintosh.	

Table 2.22-83 DisposeScrollTable Information.

2.22.1.31 scrolldraw.c

Development:SIMNET:libmac:scrolldraw.c

scrolldraw.c contains routines for drawing scrolling tables.

2.22.1.31.1 UpdateScrollTableEntry

UpdateScrollTableEntry forces a redraw of an entry, *entry*, in a scrolling table, specified by *defn*. The function call is UpdateScrollTableEntry(defn, entry). Table 2.22-84 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
defn	pointer to ScrollTableFieldDefn	Development:SIMNET:libmac: scroll.h
entry	ScrollTableEntryHandle	Development:SIMNET:libmac: scroll.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
row	int	Standard C type.
savePort	GrafPtr	Development:THINK C: Mac #includes:Quickdraw.h
box	Rect	Development:THINK C: Mac #includes:MacTypes.h
Calls		
Function	**Where Described**	
ScrollTableEntryToRow	See Section 2.22.1.34.2.	
ScrollTableRowRect	See Section 2.22.1.34.3.	
GetPort	Standard Quickdraw function for Macintosh.	
SetPort	Standard Quickdraw function for Macintosh.	
InvalRect	Standard Window Manager function for Macintosh.	

Table 2.22-84 UpdateScrollTableEntry Information.

2.22.1.31.2 DrawScrollTable

DrawScrollTable draws a scrolling table field of a dialog box. The function call is DrawScrollTable(w, itemNo). Table 2.22-85 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
w	WindowPtr	Development:THINK C: Mac #includes:WindowMgr.h
itemNo	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
field	short	Standard C type.
Calls		
Function	**Where Described**	
DrawScrollTableC	SeeSection 2.22.1.31.3.	

Table 2.22-85 DrawScrollTable Information.

2.22.1.31.3 DrawScrollTableC

DrawScrollTableC draws a scrolling table field of a dialog box. The function call is DrawScrollTableC(defn). Table 2.22-86 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
defn	register pointer to ScrollTableFieldDefn	Development:SIMNET:libmac: scroll.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
row	int	Standard C type.
pos	int	Standard C type.
topMargin	int	Standard C type.
bottomMargin	int	Standard C type.
leftMargin	int	Standard C type.
rightMargin	int	Standard C type.
c	register int	Standard C type.
i	register int	Standard C type.
col	register pointer to ScrollTableColumnDefn	Development:SIMNET:libmac: scroll.h
entry	ScrollTableEntryHandle	Development:SIMNET:libmac: scroll.h
box	Rect	Development:THINK C: Mac #includes:MacTypes.h
cBox	Rect	Development:THINK C: Mac #includes:MacTypes.h
controlDef	Handle	Development:THINK C: Mac #includes:MacTypes.h
pnState	PenState	Development:THINK C: Mac #includes:Quickdraw.h

Calls	
Function	**Where Described**
GetPenState	Standard Quickdraw function for Macintosh.
PenNormal	Standard Quickdraw function for Macintosh.
TextFont	Standard Quickdraw function for Macintosh.
TextSize	Standard Quickdraw function for Macintosh.
TextFace	Standard Quickdraw function for Macintosh.
EraseRect	Standard Quickdraw function for Macintosh.
InsetRect	Standard Quickdraw function for Macintosh.
FrameRect	Standard Quickdraw function for Macintosh.
CallPascalL	
MoveTo	Standard Quickdraw function for Macintosh.
PenPat	Standard Quickdraw function for Macintosh.
Line	Standard Quickdraw function for Macintosh.
DrawString	Standard Quickdraw function for Macintosh.
LineTo	Standard Quickdraw function for Macintosh.
HLock	Standard Memory Manager function for Macintosh.
RectInRgn	Standard Quickdraw function for Macintosh.
HUnlock	Standard Memory Manager function for Macintosh.
SetPenState	Standard Quickdraw function for Macintosh.
Called By	
Function	**Where Described**
DrawScrollTable	See Section 2.22.1.31.2.
ScrollTableBoxScroll	See Section 2.22.1.32.7.

Table 2.22-86 DrawScrollTableC Information.

2.22.1.31.4 HiliteScrollTableSelection

HiliteScrollTableSelection is a hilite function that inverts a row. *defn* is the pointer to ScrollTable; *entry* is the entry of ScrollTable; *box* is the coordinates of ScrollTable entry box to be hilited. The function call is HiliteScrollTableSelection(defn, entry, box). Table 2.22-87 describes the parameters used and function called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
defn	pointer to ScrollTableFieldDefn	Development:SIMNET:libmac: scroll.h
entry	ScrollTableEntryHandle	Development:SIMNET:libmac: scroll.h
box	pointer to Rect	Development:THINK C: Mac #includes:MacTypes.h
Calls		
Function	**Where Described**	
InvertRect	Standard Quickdraw function for Macintosh.	

Table 2.22-87 HiliteScrollTableSelection Information.

2.22.1.32 scrollevt.c
Development:SIMNET:libmac:scrollevt.c

scrollevt.c contains routines for handling events related to scrolling tables. Table 2.22-88 describes the variables used by scrollevt.c.

Variables		
Variable	**Type**	**Where Typedef Declared**
scrollDefn	pointer to ScrollTableFieldDefn	Development:SIMNET:libmac: scroll.h

Table 2.22-88 scrollevt.c Variable Information.

2.22.1.32.1 ScrollTableActivate

ScrollTableActivate handles an activate event in a scrolling table. The function call is ScrollTableActivate(defn, theEvent). Table 2.22-89 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
defn	register pointer to ScrollTableFieldDefn	Development:SIMNET:libmac: scroll.h
theEvent	pointer to EventRecord	Development:THINK C: Mac #includes:EventMgr.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
savePort	GrafPtr	Development:THINK C: Mac #includes:Quickdraw.h
Calls		
Function	**Where Described**	
GetPort	Standard Quickdraw function for Macintosh.	
SetPort	Standard Quickdraw function for Macintosh.	
InvalRect	Standard Window Manager function for Macintosh.	
ShowControl	Standard Control Manager function for Macintosh.	
HideControl	Standard Control Manager function for Macintosh.	
Called By		
Function	**Where Described**	
DialogEvent	See Section 2.22.1.11.4.	

Table 2.22-89 ScrollTableActivate Information.

2.22.1.32.2 ScrollTableEvent

ScrollTableEvent handles a mouse-down event in a scrolling table. The function call is ScrollTableEvent(defn, theEvent). Table 2.22-90 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
defn	register pointer to ScrollTableFieldDefn	Development:SIMNET:libmac: scroll.h
theEvent	pointer to EventRecord	Development:THINK C: Mac #includes:EventMgr.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
p	Point	Development:THINK C: Mac #includes:MacTypes.h
i	int	Standard C type.
box	Rect	Development:THINK C: Mac #includes:MacTypes.h
Calls		
Function	**Where Described**	
SetPort	Standard Quickdraw function for Macintosh.	
GlobalToLocal	Standard Quickdraw function for Macintosh.	
ScrollTableScrollBarEvent	See Section 2.22.1.32.3.	
Called By		
Function	**Where Described**	
DialogEvent	See Section 2.22.1.11.4.	

Table 2.22-91 ScrollTableEvent Information.

2.22.1.32.3 ScrollTableScrollBarEvent

ScrollTableScrollBarEvent handles a mouse-down in the scroll bar. The function call is ScrollTableScrollBarEvent(defn, where). Table 2.22-92 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
defn	register pointer to ScrollTableFieldDefn	Development:SIMNET:libmac: scroll.h
where	Point	Development:THINK C: Mac #includes:MacTypes.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
theControl	ControlHandle	Development:THINK C: Mac #includes:ControlMgr.h

Calls	
Function	**Where Described**
FindControl	Standard Control Manager function for Macintosh.
TrackControl	Standard Control Manager function for Macintosh.
ScrollTablePage	SeeSection 2.22.1.32.6.
ScrollTableBoxScroll	See Section 2.22.1.32.7.
Called By	
Function	**Where Described**
ScrollTableEvent	See Section 2.22.1.32.2.

Table 2.22-92 ScrollTableScrollBarEvent Information.

2.22.1.32.4 ScrollTableUp

ScrollTableUp is called from TrackControl while the up-arrow is moused. *theControl* points to the scroll control; *partCode* says whether the mouse was hit in the up-arrow section of the control. The function call is ScrollTableUp(theControl, partCode). Table 2.22-93 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
theControl	ControlHandle	Development:THINK C: Mac #includes:ControlMgr.h
partCode	short	Standard C type.
Calls		
Function	**Where Described**	
SetCtlValue	Standard Control Manager function for Macintosh.	
ScrollTableBoxScroll	See Section 2.22.1.32.7.	

Table 2.22-93 ScrollTableUp Information.

2.22.1.32.5 ScrollTableDown

ScrollTableDown is called from TrackControl while the down-arrow is moused. The function call is ScrollTableDown(theControl, partCode). Table 2.22-94 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
theControl	ControlHandle	Development:THINK C: Mac #includes:ControlMgr.h
partCode	short	Standard C type.
Calls		
Function	**Where Described**	
SetCtlValue	Standard Control Manager function for Macintosh.	
ScrollTableBoxScroll	See Section 2.22.1.32.7.	

Table 2.22-94 ScrollTableDown Information.

2.22.1.32.6 ScrollTablePage

ScrollTablePage scrolls the table by a page. The function call is ScrollTablePage(partCode). Table 2.22-95 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
partCode	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
r	pointer to Rect	Development:THINK C: Mac #includes:MacTypes.h
i	int	Standard C type.
where	Point	Development:THINK C: Mac #includes:MacTypes.h
Calls		
Function	**Where Described**	
GetMouse	Standard Toolbox Event Manager function for Macintosh.	
TestControl	Standard Control Manager function for Macintosh.	
SetCtlValue	Standard Control Manager function for Macintosh.	
GetCtlValue	Standard Control Manager function for Macintosh.	
ScrollTableBoxScroll	See Section 2.22.1.32.7.	
StillDown	Standard Toolbox Event Manager function for Macintosh.	
Called By		
Function	**Where Described**	
ScrollTableScrollBarEvent	See Section 2.22.1.32.3.	

Table 2.22-95 ScrollTablePage Information.

2.22.1.32.7 ScrollTableBoxScroll

ScrollTableBoxScroll redraws a scrolled table. The function call is ScrollTableBoxScroll(defn). Table 2.22-96 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
defn	register pointer to ScrollTableFieldDefn	Development:SIMNET:libmac: scroll.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
oldTopRow	int	Standard C type.
box	Rect	Development:THINK C: Mac #includes:MacTypes.h
tempRgn	RgnHandle	Development:THINK C: Mac #includes:Quickdraw.h

Calls	
Function	**Where Described**
GetCtlValue	Standard Control Manager function for Macintosh.
NewRgn	Standard Quickdraw function for Macintosh.
SetPort	Standard Quickdraw function for Macintosh.
ScrollRect	Standard Quickdraw function for Macintosh.
InvalRgn	Standard Window Manager function for Macintosh.
BeginUpdate	Standard Window Manager function for Macintosh.
DrawScrollTableC	See Section 2.22.1.31.3.
EndUpdate	Standard Window Manager function for Macintosh.
Called By	
Function	**Where Described**
ScrollTableScrollBarEvent	See Section 2.22.1.32.3.
ScrollTableUp	See Section 2.22.1.32.4.
ScrollTableDown	See Section 2.22.1.32.4.
ScrollTablePage	See Section 2.22.1.32.6.
SelectScrollTableEntry	See Section 2.22.1.37.1.

Table 2.22-96 ScrollTableBoxScroll Information.

2.22.1.33 scrollmap.c

Development:SIMNET:libmac:scrollmap.c

scrollmap.c contains a routine that invokes a specified function on each entry in a scrolling table.

2.22.1.33.1 MapScrollTableSelected

MapScrollTableSelected calls a supplied function for each selected entry in a scrolling table. *defn* is a pointer to the structure of scroll table; *inc* points to a function to call for each selection; *backward* is true if selection extends backwards. The function call is MapScrollTableSelected(defn, fnc, backward). Table 2.22-97 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
defn	pointer to ScrollTableFieldDefn	Development:SIMNET:libmac:scroll.h
(fnc)()	pointer to function that returns int	Standard C type.
backward	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
entry	ScrollTableEntryHandle	Development:SIMNET:libmac:scroll.h
selection	ScrollTableEntryHandle	Development:SIMNET:libmac:scroll.h
i	int	Standard C type.

Table 2.22-97 MapScrollTableSelected Information

2.22.1.34 scrollrow.c

Development:SIMNET:libmac:scrollrow.c

scrollrow.c contains routines for mapping between scrolling table entries and the rows at which they are displayed.

2.22.1.34.1 ScrollTableRowToEntry

ScrollTableRowToEntry maps a table row number to an entry. The function call is ScrollTableRowToEntry(defn, row). Table 2.22-98 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
defn	pointer to ScrollTableFieldDefn	Development:SIMNET:libmac: scroll.h
row	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
e	register ScrollTableEntryHandle	Development:SIMNET:libmac: scroll.h
Return Values		
Return Value	**Type**	**Meaning**
0	ScrollTableEntryHandle	Unsuccessful
e	ScrollTableEntryHandle	Handle of entry.
Called By		
Function	**Where Described**	
SelectScrollTableEntry	See Section 2.22.1.37.1.	

Table 2.22-98 ScrollTableRowToEntry Information.

2.22.1.34.2 ScrollTableEntryToRow

ScrollTableEntryToRow maps an entry to a table row number. If the entry is not displayed, -1 is returned. The function call is ScrollTableEntryToRow(defn, e). Table 2.22-99 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
defn	pointer to ScrollTableFieldDefn	Development:SIMNET:libmac: scroll.h
e	ScrollTableEntryHandle	Development:SIMNET:libmac: scroll.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
row	int	Standard C type.

Return Values		
Return Value	**Type**	**Meaning**
row	int	Table row number.
-1	int	Entry not displayed.
Called By		
Function	**Where Described**	
InstallScrollTableEntry	See Section 2.22.1.21.1.	
RemoveScrollTableEntry	See Section 2.22.1.28.1.	
UpdateScrollTableEntry	See Section 2.22.1.31.1.	
ZoomScrollTableEntry	See Section 2.22.1.36.1.	
SetScrollTableSelection	See Section 2.22.1.37.2.	

Table 2.22-99 ScrollTableEntryToRow Information.

2.22.1.34.3 ScrollTableRowRect

ScrollTableRowRect computes a rectangle surrounding a row of a table. The function call is ScrollTableRowRect(defn, row,box). Table 2.22-100 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
defn	pointer to ScrollTableFieldDefn	Development:SIMNET:libmac: scroll.h
row	int	Standard C type.
box	pointer to Rect	Development:THINK C: Mac #includes:MacTypes.h
Called By		
Function	**Where Described**	
UpdateScrollTableEntry	See Section 2.22.1.31.1.	
ZoomScrollTableEntry	See Section 2.22.1.36.1.	
InvertScrollTableRow	See Section 2.22.1.37.4.	

Table 2.22-100 ScrollTableRowRect Information.

2.22.1.35 scrollto.c

Development:SIMNET:libmac:scrollto.c

scrollto.c contains a routine for scrolling a table to bring a particular row into view.

2.22.1.35.1 ScrollToShow

ScrollToShow scrolls a table to show a particular entry. It returns the new row number of the entry, after any necessary scrolling. The function call is ScrollToShow(defn, entry). Table 2.22-101 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
defn	pointer to ScrollTableFieldDefn	Development:SIMNET:libmac: scroll.h
entry	register ScrollTableEntryHandle	Development:SIMNET:libmac: scroll.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
newTop	register short	Standard C type.
n	register short	Standard C type.
maxTop	short	Standard C type.
savePort	GrafPtr	Development:THINK C: Mac #includes:Quickdraw.h
Return Values		
Return Value	**Type**	**Meaning**
n-defn->topRow	int	New row number of entry.
Calls		
Function	**Where Described**	
SetCtlValue	Standard Control Manager function for Macintosh.	
GetPort	Standard Quickdraw function for Macintosh.	
SetPort	Standard Quickdraw function for Macintosh.	
InvalRect	Standard Window Manager function for Macintosh.	

Table 2.22-101 ScrollToShow Information.

2.22.1.36 scrollzoom.c

Development:SIMNET:libmac:scrollzoom.c

scrollzoom.c contains a routine for "zoom" animating between a scrolling table entry and a pop-up window representing its contents.

2.22.1.36.1 ZoomScrollTableEntry

ZoomScrollTableEntry zooms between a table entry and a pop-up window. *defn* is a pointer to scroll table definition; *entry* is the entry that pop-up will describe; *window* is a pointer to a pop-up window; *zoomUp* is true if the pop-up window should be on top and false if the scroll table should be on top. The function call is ZoomScrollTableEntry(defn, entry, window, zoomUp). Table 2.22-102 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
defn	pointer to ScrollTableFieldDefn	Development:SIMNET:libmac: scroll.h
entry	register ScrollTableEntryHandle	Development:SIMNET:libmac: scroll.h
window	WindowPtr	Development:THINK C: Mac #includes:WindowMgr.h
zoomUp	int	Standard C type.
Internal Variables		

Variable	Type	Where Typedef Declared
row	int	Standard C type.
box	Rect	Development:THINK C: Mac #includes:MacTypes.h
Calls		
Function	**Where Described**	
HideWindow	Standard Window Manager function for Macintosh.	
ScrollTableEntryToRow	See Section 2.22.1.34.2.	
ScrollTableRowRect	See Section 2.22.1.34.3.	
ZoomToWindow	See Section 2.22.1.47.4.	
ShowWindow	Standard Window Manager function for Macintosh.	

Table 2.22-102 ZoomScrollTableEntry Information.

2.22.1.37 select.c

Development:SIMNET:libmac:select.c

select.c contains routines for processing mouse events in a scrolling table.

2.22.1.37.1 SelectScrollTableEntry

SelectScrollTableEntry is called when the mouse is depressed within a scrolling table. *defn* is a pointer to scroll table structure; *row* is the row of the scroll table that was pressed; *modifiers* indicates which, if any, keys were pressed at the same time as the mouse down. The function call is SelectScrollTableEntry(defn, row, modifiers). Table 2.22-103 describes the parameters used, errors returned and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
defn	register pointer to ScrollTableFieldDefn	Development:SIMNET:libmac: scroll.h
row	int	Standard C type.
modifiers	int	Standard C type.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
entry	register ScrollTableEntryHandle	Development:SIMNET:libmac: scroll.h
e1	ScrollTableEntryHandle	Development:SIMNET:libmac: scroll.h
minExtentEntry	ScrollTableEntryHandle	Development:SIMNET:libmac: scroll.h
maxExtentEntry	ScrollTableEntryHandle	Development:SIMNET:libmac: scroll.h
i	int	Standard C type.
sense	int	Standard C type.
inSelection	int	Standard C type.
oldInSelection	int	Standard C type.
minExtentRow	int	Standard C type.
maxExtentRow	int	Standard C type.
originalRow	int	Standard C type.
limit	int	Standard C type.
pt	Point	Development:THINK C: Mac #includes:MacTypes.h

Calls	
Function	**Where Described**
ScrollTableRowToEntry	See Section 2.22.1.34.1.
SetScrollTableSelection	See Section 2.22.1.37.2.
InvertScrollTableRow	See Section 2.22.1.37.4.
StillDown	Standard Toolbox Event Manager function for Macintosh.
GetMouse	Standard Toolbox Event Manager function for Macintosh.
GetCtlMin	Standard Control Manager function for Macintosh.
SetCtlValue	Standard Control Manager function for Macintosh.
ScrollTableBoxScroll	See Section 2.22.1.32.7.

Table 2.22-103 SelectScrollTableEntry Information.

2.22.1.37.2 SetScrollTableSelection

SetScrollTableSelection sets the current selection to a particular entry. The function call is SetScrollTableSelection(defn, entry). Table 2.22-104 describes the parameters used, errors returned and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
defn	register pointer to ScrollTableFieldDefn	Development:SIMNET:libmac: scroll.h
entry	ScrollTableEntryHandle	Development:SIMNET:libmac: scroll.h

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
row	short	Standard C type.

Calls	
Function	**Where Described**
CancelScrollTableSelection	See Section 2.22.1.37.3.
ScrollTableEntryToRow	See Section 2.22.1.34.2.
InvertScrollTableRow	See Section 2.22.1.37.4.
Called By	
Function	**Where Described**
SelectScrollTableEntry	See Section 2.22.1.37.1.

Table 2.22-104 SetScrollTableSelection Information.

2.22.1.37.3 CancelScrollTableSelection

CancelScrollTableSelection cancels any current selections in a scrolling table, *defn*. The function call is CancelScrollTableSelection(defn). Table 2.22-105 describes the parameters used, and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
defn	register pointer to ScrollTableFieldDefn	Development:SIMNET:libmac: scroll.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
e	register ScrollTableEntryHandle	Development:SIMNET:libmac: scroll.h
row	int	Standard C type.
Calls		
Function	**Where Described**	
InvertScrollTableRow	See Section 2.22.1.37.4.	
Called By		
Function	**Where Described**	
SetScrollTableSelection	See Section 2.22.1.37.2.	

Table 2.22-105 CancelScrollTableSelection Information.

2.22.1.37.4 InvertScrollTableRow

InvertScrollTableRow inverts the hiliting of a scrolling table row. The function call is InvertScrollTableRow(defn, row). Table 2.22-106 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
defn	pointer to ScrollTableFieldDefn	Development:SIMNET:libmac: scroll.h
row	int	Standard C type.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
saveport	GrafPtr	Development:THINK C: Mac #includes:Quickdraw.h
r	Rect	Development:THINK C: Mac #includes:MacTypes.h
Calls		
Function	**Where Described**	
ScrollTableRowRect	See Section 2.22.1.34.3.	
GetPort	Standard Quickdraw function for Macintosh.	
SetPort	Standard Quickdraw function for Macintosh.	
InvertRect	Standard Quickdraw function for Macintosh.	
Called By		
Function	**Where Described**	
SelectScrollTableEntry	See Section 2.22.1.37.1.	
SetScrollTableSelection	See Section 2.22.1.37.2.	
CancelScrollTableSelection	See Section 2.22.1.37.3.	

Table 2.22-106 InvertScrollTableRow Information.

2.22.1.37.5 FirstScrollTableSelection

FirstScrollTableSelection returns the handle to the first selected entry in a scrolling table. The function call is FirstScrollTableSelection(defn). Table 2.22-107 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
defn	pointer to ScrollTableFieldDefn	Development:SIMNET:libmac: scroll.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
e	register ScrollTableEntryHandle	Development:SIMNET:libmac: scroll.h
Return Values		
Return Value	**Type**	**Meaning**
0	ScrollTableEntryHandle	No selected entry.
e	ScrollTableEntryHandle	Handle of selected entry.

Table 2.22-107 FirstScrollTableSelection Information.

2.22.1.37.6 RemoveScrollTableSelected

RemoveScrollTableSelected removes currently selected entries from a scrolling table. The function call is RemoveScrollTableSelected(defn). Table 2.22-108 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
defn	*register pointer to* ScrollTableFieldDefn	Development:SIMNET:libmac: scroll.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
e1	register ScrollTableEntryHandle	Development:SIMNET:libmac: scroll.h
e2	register ScrollTableEntryHandle	Development:SIMNET:libmac: scroll.h
i	int	Standard C type.
row	int	Standard C type.
redrawTriggered	int	Standard C type.
r	Rect	Development:THINK C: Mac #includes:MacTypes.h
savePort	GrafPtr	Development:THINK C: Mac #includes:Quickdraw.h
Calls		
Function	**Where Described**	
GetPort	Standard Quickdraw function for Macintosh.	
SetPort	Standard Quickdraw function for Macintosh.	
SetCtlMax	Standard Control Manager function for Macintosh.	
SetCtlValue	Standard Control Manager function for Macintosh.	
InvalRect	Standard Window Manager function for Macintosh.	
NextPrev	Macro defined in select.c in Section 2.22.1.37.	
PrevNext	Macro defined in select.c in Section 2.22.1.37.	
DisposHandle	Standard Memory Manager function for Macintosh.	

Table 2.22-108 RemoveScrollTableSelected Information.

2.22.1.38 sequence.h
Development:SIMNET:libmac:sequence.h

sequence.h header file associated with the following file, sequence.c(Section 2.22.1.39).

2.22.1.39 sequence.c
Development:SIMNET:libmac:sequence.c

sequence.c contains routines providing sequencing among dialogs with provisions for backing up in a sequence and taking alternate branches.

2.22.1.39.1 StartDialogSeq

StartDialogSeq puts up the first dialog in a sequence of dialogs. *sStorage* is a pointer to dialog storage; *first* is a pointer to the first dialog in the sequence; *terminate* is a function that is called when the dialog sequence is finished. The function is a pointer to DialogSeqState. The function call is StartDialogSeq(sStorage, first, terminate). Table 2.22-109 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
sStorage	Ptr	Development:THINK C: Mac #includes:MacTypes.h
first	pointer to DialogNodeDefn	Development:SIMNET:libmac: sequence.h
(terminate)()	pointer to function returning int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
state	pointer to DialogSeqState	Development:SIMNET:libmac: sequence.h
Return Values		
Return Value	**Type**	**Meaning**
state	pointer to DialogSeqState	State of dialog sequence.
Calls		
Function	**Where Described**	
NewPtr	Standard Memory Manager function for Macintosh.	
ShowDialogSeqNode	See Section 2.22.1.39.2.	

Table 2.22-109 StartDialogSeq Information.

2.22.1.39.2 ShowDialogSeqNode

ShowDialogSeqNode puts up a dialog box in the context of a sequence of dialog boxes. *state* is the structure defining the state of this sequence; *node* is the current node in this sequence. The function call is ShowDialogSeqNode(state, node). Table 2.22-110 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
state	pointer to DialogSeqState	Development:SIMNET:libmac: sequence.h
node	pointer to DialogNodeDefn	Development:SIMNET:libmac: sequence.h
Calls		
Function	**Where Described**	
ShowDialog	See Section 2.22.1.11.1.	
ShowWindow	Standard Window Manager function for Macintosh.	
SetWRefCon	Standard Window Manager function for Macintosh.	

Called By	
Function	**Where Described**
StartDialogSeq	See Section 2.22.1.39.1.

Table 2.22-110 ShowDialogSeqNode Information.

2.22.1.39.3 DialogSeqEvent

DialogSeqEvent handles an event in a dialog box member of a sequence. The function call is DialogSeqEvent(window, theEvent). Table 2.22-111 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
window	WindowPtr	Development:THINK C: Mac #includes:WindowMgr.h
theEvent	pointer to EventRecord	Development:THINK C: Mac #includes:EventMgr.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
state	pointer to DialogSeqState	Development:SIMNET:libmac: sequence.h
itemNo	int	Standard C type.
nextNode	pointer to DialogNodeDefn	Development:SIMNET:libmac: sequence.h
Return Values		
Return Value	**Type**	**Meaning**
0	int	Successful.
Calls		
Function	**Where Described**	
DialogEvent	See Section 2.22.1.11.4.	
DialogSeqNext	See Section 2.22.1.39.4.	
DialogSeqPrev	See Section 2.22.1.39.5.	
showhelp	See Section 2.22.1.19.4.	
Called By		
Function	**Where Described**	
DialogSeqNext	See Section 2.22.1.39.4.	
DialogSeqPrev	See Section 2.22.1.39.5.	

Table 2.22-111 DialogSeqEvent Information.

2.22.1.39.4 DialogSeqNext

DialogSeqNext goes forward in a dialog sequence. The function call is DialogSeqNext(state, node). Table 2.22-112 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
state	register pointer to DialogSeqState	Development:SIMNET:libmac: sequence.h
node	pointer to DialogNodeDefn	Development:SIMNET:libmac: sequence.h
Calls		
Function	**Where Described**	
ThrowDialog	See Section 2.22.1.11.3.	
DialogSeqEvent	See Section 2.22.1.39.3.	
Called By		
Function	**Where Described**	
DialogSeqEvent	See Section 2.22.1.39.3.	

Table 2.22-112 DialogSeqNext Information.

2.22.1.39.5 DialogSeqPrev

DialogSeqPrev backs up in a dialog sequence. The function call is DialogSeqPrev(state). Table 2.22-113 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
state	register pointer to DialogSeqState	Development:SIMNET:libmac: sequence.h
Calls		
Function	**Where Described**	
ShowCaution	See Section 2.22.1.4.1.	
ThrowDialog	See Section 2.22.1.11.3.	
DialogSeqEvent	See Section 2.22.1.39.3.	
Called By		
Function	**Where Described**	
DialogSeqEvent	See Section 2.22.1.39.3.	

Table 2.22-113 DialogSeqPrev Information.

2.22.1.40 simple.c
Development:SIMNET:libmac:simple.c

simple.c contains routines implementing dialogs used to display informational messages.

2.22.1.40.1 ShowSimpleDialog

ShowSimpleDialog displays a dialog described by a DLOG resource. The function call is ShowSimpleDialog(resourceID). Table 2.22-114 describes the parameters used functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
resourceID	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
d	DialogPtr	Development:THINK C: Mac #includes:DialogMgr.h
Return Values		
Return Value	**Type**	**Meaning**
d	DialogPtr	The dialog specified by the resource id.
Calls		
Function	**Where Described**	
GetNewDialog	Standard Dialog Manager function for Macintosh.	
LastCaution	See Section 2.22.1.4.3.	
SetWRefCon	Standard Window Manager function for Macintosh.	
OutlineItem	See Section 2.22.1.27.2.	
Called By		
Function	**Where Described**	
ShowCaution	See Section 2.22.1.4.1.	
ShowVersions	See Section 2.22.1.44.1.	

Table 2.22-114 ShowSimpleDialog Information.

2.22.1.40.2 SimpleDialogEvent

SimpleDialogEvent is called when a simple dialog gets an event. The function call is SimpleDialogEvent(window, theEvent). Table 2.22-115 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
window	WindowPtr	Development:THINK C: Mac #includes:WindowMgr.h
theEvent	pointer to EventRecord	Development:THINK C: Mac #includes:EventMgr.h

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
itemHit	int	Standard C type.
Calls		
Function	**Where Described**	
DialogSelect	Standard Dialog Manager function for Macintosh.	
DisposDialog	Standard Dialog Manager function for Macintosh.	
SysBeep	Standard Operating System Utility function for Macintosh.	

Table 2.22-115 SimpleDialogEvent Information.

2.22.1.41 table.h
Development:SIMNET:libmac:table.h

table.h header file associated with the following file, table.c(Section 2.22.1.42).

2.22.1.42 table.c
Development:SIMNET:libmac:table.c

table.c contains routines implementing fixed-length (non-scrolling) tables within dialog boxes.

2.22.1.42.1 ShowFixTable

ShowFixTable prepares a fixed length table field of a new dialog box. The function call is ShowFixTable(dialog, defn). Table 2.22-116 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
dialog	pointer to DialogState	Development:SIMNET:libmac:dialog.h
defn	register pointer to FixTableFieldDefn	Development:SIMNET:libmac:table.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
savePort	GrafPtr	Development:THINK C:Mac #includes:Quickdraw.h
col	register pointer to FixTableColumnDefn	Development:SIMNET:libmac:table.h
c	short	Standard C type.
w	short	Standard C type.
hSize	short	Standard C type.
vSize	short	Standard C type.
i	int	Standard C type.
box	Rect	Development:THINK C:Mac #includes:MacTypes.h
theItem	Handle	Development:THINK C:Mac #includes:MacTypes.h

Calls	
Function	**Where Described**
GetPort	Standard Quickdraw function for Macintosh.
SetPort	Standard Quickdraw function for Macintosh.
TextFont	Standard Quickdraw function for Macintosh.
TextSize	Standard Quickdraw function for Macintosh.
GetFontInfo	Standard Quickdraw function for Macintosh.
StringWidth	Standard Quickdraw function for Macintosh.
GetDItem	Standard Dialog Manager function for Macintosh.
SetDItem	Standard Dialog Manager function for Macintosh.
Called By	
Function	**Where Described**
ShowDialog	See Section 2.22.1.11.1.

Table 2.22-116 ShowFixTable Information.

2.22.1.42.2 UpdateFixTableRow

UpdateFixTableRow forces a redraw of a row in a fixed-length table. The functior call is UpdateFixTableRow(defn, row). Table 2.22-117 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
defn	register pointer to FixTableFieldDefn	Development:SIMNET:libmac: table.h
row	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
savePort	GrafPtr	Development:THINK C: Mac #includes:Quickdraw.h
box	Rect	Development:THINK C: Mac #includes:MacTypes.h
Calls		
Function	**Where Described**	
GetPort	Standard Quickdraw function for Macintosh.	
SetPort	Standard Quickdraw function for Macintosh.	
InvalRect	Standard Window Manager function for Macintosh.	

Table 2.22-117 UpdateFixTableRow Information.

2.22.1.42.3 FixTableEvent

FixTableEvent handles a mouse-down event in a fixed-length table. The function call is FixTableEvent(defn, theEvent). Table 2.22-118 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
defn	register pointer to FixTableFieldDefn	Development:SIMNET:libmac: table.h
theEvent	pointer to EventRecord	Development:THINK C: Mac #includes:EventMgr.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
p	Point	Development:THINK C: Mac #includes:MacTypes.h
savePort	GrafPtr	Development:THINK C: Mac #includes:Quickdraw.h
i	int	Standard C type.
theItem	Handle	Development:THINK C: Mac #includes:MacTypes.h
box	Rect	Development:THINK C: Mac #includes:MacTypes.h
Calls		
Function	**Where Described**	
GetPort	Standard Quickdraw function for Macintosh.	
SetPort	Standard Quickdraw function for Macintosh.	
GlobalToLocal	Standard Quickdraw function for Macintosh.	
GetDItem	Standard Dialog Manager function for Macintosh.	
Called By		
Function	**Where Described**	
DialogEvent	See Section 2.22.1.11.4.	

Table 2.22-118 FixTableEvent Information.

2.22.1.42.4 DrawFixTable

DrawFixTable draws a fixed-length table field of a dialog box. The function call is DrawFixTable(w, itemNo). Table 2.22-119 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
w	WindowPtr	Development:THINK C: Mac #includes:WindowMgr.h
itemNo	int	Standard C type.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
ht	int	Standard C type.
i	int	Standard C type.
row	int	Standard C type.
pos	int	Standard C type.
c	int	Standard C type.
field	int	Standard C type.
defn	register pointer to FixTableFieldDefn	Development:SIMNET:libmac: table.h
col	pointer to FixedTableColumnDefn	Development:SIMNET:libmac: table.h
theItem	Handle	Development:THINK C: Mac #includes:MacTypes.h
r	Rect	Development:THINK C: Mac #includes:MacTypes.h
box	Rect	Development:THINK C: Mac #includes:MacTypes.h
pnState	PenState	Development:THINK C: Mac #includes:Quickdraw.h

Calls	
Function	**Where Described**
GetDItem	Standard Dialog Manager function for Macintosh.
DialogLookupField	See Section 2.22.1.12.1.
GetPenState	Standard Quickdraw function for Macintosh.
PenNormal	Standard Quickdraw function for Macintosh.
TextFont	Standard Quickdraw function for Macintosh.
TextFaceSize	Standard Quickdraw function for Macintosh.
EraseRect	Standard Quickdraw function for Macintosh.
InsetRect	Standard Quickdraw function for Macintosh.
FrameRect	Standard Quickdraw function for Macintosh.
MoveTo	Standard Quickdraw function for Macintosh.
PenPat	Standard Quickdraw function for Macintosh.
Line	Standard Quickdraw function for Macintosh.
StringWidth	Standard Quickdraw function for Macintosh.
DrawString	Standard Quickdraw function for Macintosh.
LineTo	Standard Quickdraw function for Macintosh.
SetRect	Standard Quickdraw function for Macintosh.
SetPenState	Standard Quickdraw function for Macintosh.

Table 2.22-119 DrawFixTable Information.

2.22.1.42.5 FixTableRowRect

FixTableRowRect computes a rectangle surrounding a single row of a table. The function call is FixTableRowRect(defn, row, box). Table 2.22-120 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
defn	register pointer to FixTableFieldDefn	Development:SIMNET:libmac: table.h
row	int	Standard C type.
box	register pointer to Rect	Development:THINK C: Mac #includes:MacTypes.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	int	Standard C type.
theItem	Handle	Development:THINK C: Mac #includes:MacTypes.h
Calls		
Function	**Where Described**	
GetDItem	Standard Dialog Manager function for Macintosh.	

Table 2.22-120 FixTableRowRect Information.

2.22.1.43 title.c

Development:SIMNET:libmac:title.c

title.c contains a routine for displaying a title string in the menu bar.

2.22.1.43.1 MenuBarTitle

MenuBarTitle writes a title string, *str*, centered, in the menu bar. The function call is MenuBarTitle(str). Table 2.22-121 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
str	pointer to char	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
savePort	GrafPtr	Development:THINK C: Mac #includes:Quickdraw.h
screenPort	GrafPtr	Development:THINK C: Mac #includes:Quickdraw.h
fInfo	FontInfo	Development:THINK C: Mac #includes:Quickdraw.h
r	Rect	Development:THINK C: Mac #includes:MacTypes.h

Calls	
Function	**Where Described**
GetPort	Standard Quickdraw function for Macintosh.
OpenPort	Standard Quickdraw function for Macintosh.
GetFontInfo	Standard Quickdraw function for Macintosh.
EraseRect	Standard Quickdraw function for Macintosh.
MoveTo	Standard Quickdraw function for Macintosh.
StringWidth	Standard Quickdraw function for Macintosh.
DrawString	Standard Quickdraw function for Macintosh.
SetPort	Standard Quickdraw function for Macintosh.

Table 2.22-121 MenuBarTitle Information.

2.22.1.44 version.c

Development:SIMNET:libmac:version.c

version.c contains routines for displaying a dialog identifying the versions of files comprising the current application.

2.22.1.44.1 ShowVersions

ShowVersions puts up a dialog listing the open resource files and the version string from each. The function call is ShowVersions(). Table 2.22-122 describes the functions called using this function.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
hdl	Handle	Development:THINK C: Mac #includes:MacTypes.h
i	int	Standard C type.
d	DialogPtr	Development:THINK C: Mac #includes:DialogMgr.h
box	Rect	Development:THINK C: Mac #includes:macTypes.h
versions[256]	char	Standard C type.
Calls		
Function	**Where Described**	
GetIndResource	Standard Resource Manager function for Macintosh.	
AppendString	See Section 2.22.1.44.2.	
ShowSimpleDialog	See Section 2.22.1.40.1.	
GetDItem	Standard Dialog Manager function for Macintosh.	
SetIText	Standard Dialog Manager function for Macintosh.	
ShowWindow	Standard Window Manager function for Macintosh.	

Table 2.22-122 ShowVersions Information.

2.22.1.44.2 AppendString

AppendString appends two character strings, *s1* and *s2*. The function call is AppendString(s1, s2). Table 2.22-123 describes the parameters used and function called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
s1	pointer to char	Standard C type.
s2	pointer to char	Standard C type.
Calls		
Function	**Where Described**	
BlockMove	Standard Memory Manager function for Macintosh.	
Called By		
Function	**Where Described**	
ShowVersions	See Section 2.22.1.44.1.	

Table 2.22-123 AppendString Information.

2.22.1.45 wait.c

Development:SIMNET:libmac:wait.c

wait.c contains routines to put up alert boxes with wait messages. Table 2.22-124 describes the variables used by wait.c.

Variables		
Variable	**Type**	**Where Typedef Declared**
waitDialog	WindowRecord	Development:THINK C: Mac #includes:WindowMgr.h

Table 2.22-124 wait.c Variable Information.

2.22.1.45.1 ShowWait

ShowWait puts up a dialog with a wait message, *message*. The function call is ShowWait(message). Table 2.22-125 describes the parameter used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
message	pointer to char	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
box	Rect	Development:THINK C: Mac #includes:MacTypes.h
savePort	GrafPtr	Development:THINK C: Mac #includes:Quickdraw.h
buffer[4096]	char	Standard C type.

Calls	
Function	**Where Described**
SetCursor	Standard Quickdraw function for Macintosh.
GetCursor	Standard Toolbox Utility function for Macintosh.
SetRect	Standard Quickdraw function for Macintosh.
NewWindow	Standard Window Manager function for Macintosh.
GetPort	Standard Quickdraw function for Macintosh.
SetPort	Standard Quickdraw function for Macintosh.
TextFont	Standard Quickdraw function for Macintosh.
SetRect	Standard Quickdraw function for Macintosh.
TextBox	Standard TextEdit function for Macintosh.

Table 2.22-125 ShowWait Information.

2.22.1.45.2 ThrowWait

ThrowWait closes the dialog window with the wait message and puts up the arrow cursor. The function call is ThrowWait(). Table 2.22-126 describes the functions called using this function.

Calls	
Function	**Where Described**
CloseWindow	Standard Menu Manager function for Macintosh.
SetCursor	Standard Quickdraw function for Macintosh.

Table 2.22-126 ThrowWait Information.

2.22.1.46 window.c

Development:SIMNET:libmac:window.c

window.c contains routines for generic handling of events associated with windows. Table 2.22-127 describes the variables used by window.c.

Variables		
Variable	**Type**	**Where Typedef Declared**
menuHandler	ProcPtr	Development:THINK C: Mac #includes:MacTypes.h

Table 2.22-127 window.c Variable Information.

2.22.1.46.1 SetMenuHandler

SetMenuHandler provides a routine, *hdler*, to field menu events. The function call is SetMenuHandler(hdler). Table 2.22-128 describes the parameter used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
hdler	ProcPtr	Development:THINK C: Mac #includes:MacTypes.h

Table 2.22-128 SetMenuHandler Information.

2.22.1.46.2 WindowEvent

Window Event figures out which window an event has occured in, and calls the event handler for that window. If the event represents the selection of a menu item, a long word is returned defined as for the Toolbox function MenuSelect, otherwise zero is returned. The function call is WindowEvent(theEvent). Table 2.22-129 describes the parameter used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
theEvent	register pointer to EventRecord	Development:THINK C: Mac #includes:EventMgr.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
menuItem	unsigned long	Standard C type.
window	WindowPtr	Development:THINK C: Mac #includes:WindowMgr.h
Calls		
Function	**Where Described**	
FindWindow	Standard Window Manager function for Macintosh.	
MenuSelect	Standard Menu Manager function for Macintosh.	
SystemClick	Standard Desk Manager function for Macintosh.	
MenuKey	Standard Menu Manager function for Macintosh.	
FrontWindow	Standard Window Manager function for Macintosh.	
GetWRefCon	Standard Window Manager function for Macintosh.	

Table 2.22-129 WindowEvent Information.

2.22.1.47 zoom.c

Development:SIMNET:libmac:zoom.c

zoom.c contains routines for "zoom" animating between a rectangle and a window. Table 2.22-130 describes the variables used by zoom.c.

Variables		
Variable	**Type**	**Where Typedef Declared**
deskPort	GrafPort	Development:THINK C: Mac #includes:Quickdraw.h

Table 2.22-130 zoom.c Variable Information.

2.22.1.47.1 ZoomInit

ZoomInit is called once before any calls to ZoomRect. The function call is ZoomInit(). Table 2.22-131 describes the functions called using this function.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
savePort	GrafPtr	Development:THINK C: Mac #includes:Quickdraw.h
Calls		
Function	**Where Described**	
GetPort	Standard Quickdraw function for Macintosh.	
OpenPort	Standard Quickdraw function for Macintosh.	
PenPat	Standard Quickdraw function for Macintosh.	
PenMode	Standard Quickdraw function for Macintosh.	
SetPort	Standard Quickdraw function for Macintosh.	

Table 2.22-131 ZoomInit Information.

2.22.1.47.2 ZoomRect

ZoomRect is given two rectangles in global coordinates and interpolates on into the other, making a zooming rectangle image on the screen. The rectangles and the screen image are not altered. *smallRect* is the coordinates of the smaller box; *bigRect* is the coordinates of the bigger box; *zoomUp* is *true* if zooming from small box to big box. The function call is ZoomRect(smallRect, bigRect, zoomUp). Table 2.22-132 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
smallRect	pointer to Rect	Development:THINK C: Mac #includes:MacTypes.h
bigRect	pointer to Rect	Development:THINK C: Mac #includes:MacTypes.h
zoomUp	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
savePort	GrafPtr	Development:THINK C Mac #includes:Quickdraw.h
rect1	Rect	Development:THINK C: Mac #includes:MacTypes.h
rect2	Rect	Development:THINK C: Mac #includes:MacTypes.h
rect3	Rect	Development:THINK C: Mac #includes:MacTypes.h
rect4	Rect	Development:THINK C: Mac #includes:MacTypes.h
fract	Fixed	Development:THINK C: Mac #includes:MacTypes.h
factor	Fixed	Development:THINK C: Mac #includes:MacTypes.h
i	int	Standard C type.

Calls	
Function	**Where Described**
GetPort	Standard Quickdraw function for Macintosh.
SetPort	Standard Quickdraw function for Macintosh.
FixRatio	Standard Toolbox Utility function for Macintosh.
FrameRect	Standard Quickdraw function for Macintosh.
SetRect	Standard Quickdraw function for Macintosh.
Blend	See Section 2.22.1.47.3.
FixMul	Macro defined in zoom.c.
Called By	
Function	**Where Described**
ZoomToWindow	See Section 2.22.1.47.4.

Table 2.22-132 ZoomRect Information.

2.22.1.47.3 Blend

Blend interpolates one step in zooming from one rectangle to another. *smallCoord* is the coordinates of the smaller rectangle; *bigCoord* is the coordinates of the bigger rectangle. The function call is Blend(smallCoord, bigCoord, fract). Table 2.22-133 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
smallCoord	int	Standard C type.
bigCoord	int	Standard C type.
fract	Fixed	Development:THINK C: Mac #includes:MacTypes.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
smallFix	Fixed	Development:THINK C: Mac #includes:MacTypes.h
bigFix	Fixed	Development:THINK C: Mac #includes:MacTypes.h
tempFix	Fixed	Development:THINK C: Mac #includes:MacTypes.h
Return Values		
Return Value	**Type**	**Meaning**
s	int	DDP socket number
Calls		
Function	**Where Described**	
FixMul	Macro defined in zoom.c.	
FixRound	Standard Toolbox Utility function for Macintosh.	
Called By		
Function	**Where Described**	
ZoomRect	See Section 2.22.1.47.2.	

Table 2.22-133 Blend Information.

2.22.1.47.4 ZoomToWindow

ZoomToWindow zooms between a rectangle within one window and the frame of another window. The function call is ZoomToWindow(window1, rect1, window2, zoomUp). Table 2.22-134 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
window1	WindowPtr	Development:THINK C: Mac #includes:WindowMgr.h
rect1	pointer to Rect	Development:THINK C: Mac #includes:MacTypes.h
window2	register WindowPtr	Development:THINK C: Mac #includes:WindowMgr.h
zoomUp	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
smallRect	Rect	Development:THINK C: Mac #includes:MacTypes.h
bigRect	Rect	Development:THINK C: Mac #includes:MacTypes.h
i	register int	Standard C type.
Calls		
Function	**Where Described**	
ZoomRect	See Section 2.22.1.47.2.	
Called		
Function	**Where Described**	
ZoomScrollTableEntry	See Section 2.22.1.36.1.	

Table 2.22-134 ZoomToWindow Information.

2.22.2 libsim

Folder: "Development:SIMNET:MCC:libsim"
This folder contains a library of C functions supporting the placement and reconstitution of vehicle simulators. The library is used by both the SCC and Vehicle Placement Console Macintosh applications. The library is composed from the following files:

2.22.2.1 libsim.h
Development:SIMNET:MCC:libsim:libsim.h

libsim.h defines the external interface to the libsim library. Table 2.22-135 describes the variables used by libsim.h.

Variables		
Variable	**Type**	**Where Typedef Declared**
terrainBounds	extern LongRect	Development:SIMNET:libmac:longpt.h
battleScheme	extern char	Standard C type.
mccForceID	extern char	Standard C type.
companyRole	extern array of char	Standard C type.
simulators	extern array of SimDescriptor	Development:SIMNET:MCC:libsim:libsim.h
pastSimulators	extern array of SavedDefaults	Development:SIMNET:MCC:libsim:libsim.h

Table 2.22-135 libsim.h Variable Information.

2.22.2.2 libsim_int.h

Development:SIMNET:MCC:libsim:libsim_int.h

limsim_int.h contains definitions used within the libsim library. Table 2.22-136 describes the variables used by libsim_int.h.

Variables		
Variable	**Type**	**Where Typedef Declared**
libsimStandaloneVersion	extern char	Standard C type.
simulatorTypes	extern array of SimulatorTypeData	Development:SIMNET:MCC:libsim:libsim_int.h
placeBuffer	extern SimPlaceData	Development:SIMNET:MCC:libsim:libsim_int.h
detailBuffer	extern SimVehicleStatus	Development:SIMNET:MCC:include:sim_xact.h
placeDialog	extern pointer to DialogState	Development:SIMNET:libmac:dialog.h
detailDialog	extern pointer to DialogState	Development:SIMNET:libmac:dialog.h
vehCancelField	extern PushButtonFieldDefn	Development:SIMNET:libmac:dialog.h
vehBumperField	extern NumberFieldDefn	Development:SIMNET:libmac:dialog.h
vehBowField	extern NumberFieldDefn	Development:SIMNET:libmac:dialog.h
vehTeamAField	extern RBFieldDefn	Development:SIMNET:libmac:dialog.h
vehTeamBField	extern RBFieldDefn	Development:SIMNET:libmac:dialog.h
vehObserverField	extern RBFieldDefn	Development:SIMNET:libmac:dialog.h
vehRectField	extern RectFieldDefn	Development:SIMNET:libmac:dialog.h
vehLocationField	extern CoordinatesFieldDefn	Development:SIMNET:libmac:dialog.h

Table 2.22-136 libsim_int.h Variable Information.

2.22.2.3 atalk.c
Development:SIMNET:MCC:libsim:atalk.c

atalk.c contains routines for transferring information about vehicle simulators between the MCC host and a Macintosh.

2.22.2.3.1 DownloadSimulators

DownloadSimulators obtains information about the vehicle simulators available for inclusion in the exercise. The function call is DownloadSimulators(). Table 2.22-137 describes the internal variables used and functions called using this function.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	register int	Standard C type.
sim	register pointer to SimDescriptor	Development:SIMNET:MCC: libsim:libsim.h
req	SimExistsRequest	Development:SIMNET:MCC: include:sim_xact.h
rsp	SimExistsResponse	Development:SIMNET:MCC: include:sim_xact.h
rtn	int	Standard C type.
Calls		
Function	**Where Described**	
ATPPut	See Section 2.22.1.3.3.	
PointToMapCoordinates	See Section 2.22.1.26.2.	

Table 2.22-137 DownloadSimulators Information.

2.22.2.3.2 UploadVehicleParameters

UploadVehicleParameters tells the host about a vehicle placement. Vehicle parameters are obtained from the structure supplied. The function call is UploadVehicleParameters(sim, data). Table 2.22-138 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
sim	register pointer to SimDescriptor	Development:SIMNET:MCC: libsim:libsim.h
data	register pointer to SimPlaceData	Development:SIMNET:MCC: libsim:libsim_int.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
req	SimInitRequest	Development:SIMNET:MCC: include:sim_xact.h
rsp	SimInitResponse	Development:SIMNET:MCC: include:sim_xact.h
i	int	Standard C type.
errCode	int	Standard C type.

Return Values		
Return Value	**Type**	**Meaning**
0	int	Unsuccessful.
1	int	Successful.
Calls		
Function	**Where Described**	
ATPPut	See Section 2.22.1.3.3.	
PointToMapCoordinates	See Section 2.22.1.26.2.	
ShowCaution	See Section 2.22.1.4.1.	
Called By		
Function	**Where Described**	
CompleteVehicleDialog	See Section 2.22.2.9.6.	

Table 2.22-138 UploadVehicleParameters Information.

2.22.2.3.3 DownloadVehicleParameters

DownloadVehicleParameters obtains the latest vehicle data from the host. The vehicle parameters are placed in the structure supplied. The function call is DownloadVehicleParameters(sim, data). Table 2.22-139 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
sim	register pointer to SimDescriptor	Development:SIMNET:MCC: libsim:libsim.h
data	register pointer to SimPlaceData	Development:SIMNET:MCC: libsim:libsim_int.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
req	SimQueryRequest	Development:SIMNET:MCC: include:sim_xact.h
rsp	SimQueryResponse	Development:SIMNET:MCC: include:sim_xact.h
i	short	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
0	int	Unsuccessful.
1	int	Successful.

Calls	
Function	**Where Described**
ATPPut	See Section 2.22.1.3.3.
ShowCaution	See Section 2.22.1.4.1.
PointToMapCoordinates	See Section 2.22.1.26.2.
Called By	
Function	**Where Described**
ShowVehicleDialog	See Section 2.22.2.9.5.

Table 2.22-139 DownloadVehicleParameters Information.

2.22.2.3.4 ProcessSimPlacedRequest

ProcessSimPlacedRequest processes a transaction from the host indicating that a simulator has been placed by another console. The function call is ProcessSimPlacedRequest (req). Table 2.22-140 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
req	register pointer to SimPlacedRequest	Development_SIMNET:MCC: include:sim_xact.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
sim	register pointer to SimDescriptor	Development:SIMNET:MCC: libsim:libsim.h
Calls		
Function	**Where Described**	
PointToMapCoordinates	See Section 2.22.1.26.2.	

Table 2.22-140 ProcessSimPlacedRequest Information.

2.22.2.4 data.c

Development:SIMNET:MCC:libsim:data.c

data.c defines data structures associated with the libsim library, such as tables of information about types and instances of vehicle simulators. Table 2.22-141 describes the variables used by data.c.

Variables		
Variable	**Type**	**Where Typedef Declared**
libsimStandaloneVersion	char	Standard C type.
terrainBounds	LongRect	Development:SIMNET:libmac:longpt.h
battleScheme	char	Standard C type.
mccForceID	char	Standard C type.
companyRole	array of maxNumberCompanies char	Standard C type.
simulators	array of maxVehicleSimulators SimDescriptor	Development:SIMNET:MCC:libsim:libsim.h
pastSimulators	array of maxVehicleSimulators SavedDefaults	Development:SIMNET:MCC:libsim:libsim.h
placeDialog	pointer to DialogState	Development:SIMNET:libmac:dialog.h
detailDialog	pointer to DialogState	Development:SIMNET:libmac:dialog.h
placeBuffer	SimPlaceData	Development:SIMNET:MCC:libsim:libsim_int.h
detailBuffer	SimVehicleStatus	Development:SIMNET:MCC:include:sim_xact.h
m1PlaceDialog	extern DialogDefn	Development:SIMNET:libmac:dialog.h
m2PlaceDialog	extern DialogDefn	Development:SIMNET:libmac:dialog.h
fredPlaceDialog	extern DialogDefn	Development:SIMNET:libmac:dialog.h
simulatorTypes	array of SimulatorTypeData	Development:SIMNET:MCC:libsim:libsim_int.h

Table 2.22-141 data.c Variable Information.

2.22.2.5 fred.c

Development:SIMNET:MCC:libsim:fred.c

fred.c contains routines implementing a user interface for placing or reconstituting a SIMNET FRED (rotary wing aircraft) simulator. Table 2.22-142 describes the variables used by fred.c.

Variables		
Variable	**Type**	**Where Typedef Declared**
fredPlaceDialog	extern DialogDefn	Development:SIMNET:libmac:dialog.h
fredFuelField	NumberFieldDefn	Development:SIMNET:libmac:dialog.h
fredAttackField	RBFieldDefn	Development:SIMNET:libmac:dialog.h
fredScoutField	RBFieldDefn	Development:SIMNET:libmac:dialog.h
fredAmmoFields	array of NumberFieldDefn	Development:SIMNET:libmac:dialog.h
fredPlaceFieldList	pointer to array of FieldDefn	Development:SIMNET:libmac:dialog.h
fredPlaceDialog	DialogDefn	Development:SIMNET:libmac:dialog.h

Table 2.22-142 fred.c Variable Information.

2.22.2.5.1 FREDSetDefaults

FREDSetDefaults sets the FRED placement fields to their default values. The function call is FREDSetDefaults(). This function calls only standard functions and is not called by any other function in this library, so no table is included for it.

2.22.2.5.2 FREDLoadSavedData

FREDLoadSavedData loads the FRED placement dialog with canned parameters. The function call is FREDLoadSavedData(company, bumper). Table 2.22-143 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
company	char	Standard C type.
bumper	char	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
sim	pointer to SavedDefaults	Development:SIMNET:MCC:libsim:libsim.h
Calls		
Function	**Where Described**	
FindOldSimulator	See Section 2.22.2.9.9.	
UpdateDialog	See Section 2.22.1.11.2.	

Called By	
Function	**Where Described**
VehicleBumperNumber	See Section 2.22.2.9.7.

Table 2.22-143 FREDLoadSavedData Information.

2.22.2.5.3 FREDPlaceEvent

FREDPlaceEvent is not used in the version 6.6 release.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
dialog	pointer to DialogState	Development:SIMNET:libmac:dialog.h
itemNo	int	Standard C type.
Calls		
Function	**Where Described**	
showhelp	See Section 2.22.1.19.4.	
CheckMandatoryFields	See Section 2.22.1.13.1.	
CompleteVehicleDialog	See Section 2.22.2.9.6.	

Table 2.22-144 FREDPlaceEvent Information.

2.22.2.6 load.c

Development:SIMNET:MCC:libsim:load.c

load.c contains routines for loading "canned" data into the library's data structures, to permit operation of a standalone, demo versions of the SCC and Vehicle Placement Console.

2.22.2.6.1 LoadCannedSimulators

LoadCannedSimulators loads canned simulator data, for the demo version. The parameter, *company*, specifies whether the simulators should be unassigned (if it is assignedNothing) or assigned to a particular company. The function call is LoadCannedSimulators(company). Table 2.22-145 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
company	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
ps	register pointer to SavedDefaults	Development:SIMNET:MCC:libsim:libsim.h
str	array of 50 char	Standard C type.

Table 2.22-145 LoadCannedSimulators Information.

2.22.2.7 m1.c
Development:SIMNET:MCC:libsim:m1.c

m1.c contains routines implementing a user interface for placing or reconstituting a SIMNET M1 simulator. Table 2.22-146 describes the variables used by m1.c.

Variables		
Variable	**Type**	**Where Typedef Declared**
m1PlaceDialog	extern DialogDefn	Development:SIMNET:libmac:dialog.h
m1DetailDialog	extern DialogDefn	Development:SIMNET:libmac:dialog.h
fuelLoadBuffer	int	Standard C type.
apdsLoadBuffer	int	Standard C type.
heatLoadBuffer	int	Standard C type.
m1FuelField	NumberFieldDefn	Development:SIMNET:libmac:dialog.h
m1APDSField	NumberFieldDefn	Development:SIMNET:libmac:dialog.h
m1HEATField	NumberFieldDefn	Development:SIMNET:libmac:dialog.h
m1MaintFields	array of RBFieldDefn	Development:SIMNET:libmac:dialog.h
m1PlaceFieldList	pointer to array of FieldDefn	Development:SIMNET:libmac:dialog.h
m1PlaceDialog	DialogDefn	Development:SIMNET:libmac:dialog.h
m1TurretField	NumberFieldDefn	Development:SIMNET:libmac:dialog.h
m1FuelFields	array of NumberFieldDefn	Development:SIMNET:libmac:dialog.h
m1APDSFields	array of NumberFieldDefn	Development:SIMNET:libmac:dialog.h
m1HEATFields	array of NumberFieldDefn	Development:SIMNET:libmac:dialog.h
m1DetailFieldList	pointer to array of FieldDefn	Development:SIMNET:libmac:dialog.h
m1DetailDialog	DialogDefn	Development:SIMNET:libmac:dialog.h

Table 2.22-146 m1.c Variable Information.

2.22.2.7.1 M1SetDefaults

M1SetDefaults sets the M1 placement fields to their default values. The function call is M1SetDefaults(). This function calls only standard functions and is not called by any other function in this library, so no table is included for it.

2.22.2.7.2 M1LoadSavedData

M1LoadSavedData loads the M1 placement dialog with canned parameters. The function call is M1LoadSavedData(company, bumper). Table 2.22-147 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
company	char	Standard C type.
bumper	char	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
sim	pointer to SavedDefaults	Development:SIMNET:MCC: libsim:libsim.h
Calls		
Function	**Where Described**	
FindOldSimulator	See Section 2.22.2.9.9.	
M1PlaceFetch	See Section 2.22.2.7.4.	
UpdateDialog	See Section 2.22.1.11.2.	
Called By		
Function	**Where Described**	
ValidBumperNumber	See Section 2.22.2.9.7.	

Table 2.22-147 M1LoadSavedData Information.

2.22.2.7.3 M1FillDetail

M1FillDetail updates the detailed allocation of fuel and ammunition according to the total fuel and ammunition quantities entered. The function call is M1FillDetail(). Table 2.22-148 describes the internal variables used and functions called using this function.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
remainder1	register int	Standard C type.
remainder2	register int	Standard C type.
maximum	register int	Standard C type.
apdsChanged	int	Standard C type.
heatChanged	int	Standard C type.
Calls		
Function	**Where Described**	
min	Macro defined in Development:SIMNET:MCC:libsim:m1.c.	
Called By		
Function	**Where Described**	
M1PlaceEvent	See Section 2.22.2.7.5.	

Table 2.22-148 M1FillDetail Information.

2.22.2.7.4 M1PlaceFetch

M1PlaceFetch reads the current values of the fuel and ammo load buffers into the M1 Placement dialog to display to the user. The function call is M1PlaceFetch(dialog). Table 2.22-149 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
dialog	pointer to DialogState	Development:SIMNET:libmac:dialog.h
Called By		
Function	**Where Described**	
M1LoadSavedData	See Section 2.22.2.7.2.	
M1DetailEvent	See Section 2.22.2.7.6.	

Table 2.22-149 M1PlaceFetch Information.

2.22.2.7.5 M1PlaceEvent

M1PlaceEvent fields events in the M1 Placement dialog. The function call is M1PlaceEvent(dialog, itemNo). Table 2.22-150 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
dialog	pointer to DialogState	Development:SIMNET:libmac:dialog.h
itemNo	int	Standard C type.
Calls		
Function	**Where Described**	
showhelp	See Section 2.22.1.19.4.	
CheckMandatoryFields	See Section 2.22.1.13.1.	
ShowCaution	See Section 2.22.1.4.1	
M1FillDetail	See Section 2.22.2.7.3.	
CompleteVehicleDialog	See Section 2.22.2.9.6.	
ShowDialog	See Section 2.22.1.11.1.	
ShowWindow	Standard Window Manager function for Macintosh.	

Table 2.22-150 M1PlaceEvent Information.

2.22.2.7.6 M1DetailEvent

M1DetailEvent fields events in the M1 Detail dialog The function call is M1DetailEvent(dialog, itemNo). Table 2.22-151 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
dialog	pointer to DialogState	Development:SIMNET:libmac:dialog.h
itemNo	int	Standard C type.

Calls	
Function	**Where Described**
showhelp	See Section 2.22.1.19.4.
CheckMandatoryFields	See Section 2.22.1.13.1.
ShowCaution	See Section 2.22.1.4.1
M1PlaceFetch	See Section 2.22.2.7.4.
UpdateDialog	See Section 2.22.1.11.2.
ThrowDialog	See Section 2.22.1.11.3.

Table 2.22-151 M1DetailEvent Information.

2.22.2.8 m2.c

Development:SIMNET:MCC:libsim:m2.c

m2.c contains routines implementing a user interface for placing or reconstituting a SIMNET M2/3 simulator. Table 2.22-152 describes the variables used by m2.c.

Variables		
Variable	**Type**	**Where Typedef Declared**
m2PlaceDialog	extern DialogDefn	Development:SIMNET:libmac:dialog.h
m2DetailDialog	extern DialogDefn	Development:SIMNET:libmac:dialog.h
fuelLoadBuffer	int	Standard C type.
towLoadBuffer	int	Standard C type.
m2FuelField	NumberFieldDefn	Development:SIMNET:libmac:dialog.h
m2BinFields	array of NumberFieldDefn	Development:SIMNET:libmac:dialog.h
m2BinTypeFields	array of RBFieldDefn	Development:SIMNET:libmac:dialog.h
m2AmmoFields	array of NumberFieldDefn	Development:SIMNET:libmac:dialog.h
m2MaintFields	array of RBFieldDefn	Development:SIMNET:libmac:dialog.h
m2ConfigurationFields	array of RBFieldDefn	Development:SIMNET:libmac:dialog.h
m2PlaceFieldList	array of pointer to FieldDefn	Development:SIMNET:libmac:dialog.h
m2PlaceDialog	DialogDefn	Development:SIMNET:libmac:dialog.h
m2TurretField	NumberFieldDefn	Development:SIMNET:libmac:dialog.h
m2FuelFields	array of NumberFieldDefn	Development:SIMNET:libmac:dialog.h
m2LauncherFields	array of CBFieldDefn	Development:SIMNET:libmac:dialog.h
m2MissilFields	array of NumberFieldDefn	Development:SIMNET:libmac:dialog.h
m2DetailFieldList	pointer to array of FieldDefn	Development:SIMNET:libmac:dialog.h
m2DetailDialog	DialogDefn	Development:SIMNET:libmac:dialog.h

Table 2.22-152 m2.c Variable Information.

2.22.2.8.1 LimitCheck

LimitCheck is used to locally check capacity limits. The function call is LimitCheck(type, value, limit, units). Table 2.22-153 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
type	pointer to char	Standard C type.
value	int	Standard C type.
limit	int	Standard C type.
units	pointer to char	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
buf	array of 100 char	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
0	int	Over the limits.
1	int	Within the limits.
Calls		
Function	**Where Described**	
ShowCaution	See Section 2.22.1.4.1.	
Called By		
Function	**Where Described**	
M2PlaceEvent	See Section 2.22.2.8.6.	
M2DetailEvent	See Section 2.22.2.8.8.	

Table 2.22-153 LimitCheck Information.

2.22.2.8.2 M2SetDefaults

M2SetDefaults sets the M2 placement fields to their default values. The function call is M2SetDefaults(). This function calls only standard functions and is not called by any other function in this library, so no table is included for it.

2.22.2.8.3 M2LoadSavedData

M2LoadSavedData loads the M2 placement dialog with canned parameters. The function call is M2LoadSavedData(company, bumper). Table 2.22-154 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
company	char	Standard C type.
bumper	char	Standard C type.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
sim	pointer to SavedDefaults	Development:SIMNET:MCC:libsim:libsim.h
Calls		
Function	**Where Described**	
FindOldSimulator	See Section 2.22.2.9.9.	
M2PlaceFetch	See Section 2.22.2.8.5.	
UpdateDialog	See Section 2.22.1.11.2.	
Called By		
Function	**Where Described**	
ValidBumperNumber	See Section 2.22.2.9.7.	

Table 2.22-154 M2LoadSavedData Information.

2.22.2.8.4 M2FillDetail

M2FillDetail updates the detailed allocation of fuel and ammunition according to the total fuel and ammunition quantities entered. The function call is M2FillDetail(). Table 2.22-155 describes the internal variables used and functions called using this function.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
remainder	register int	Standard C type.
Calls		
Function	**Where Described**	
min	Macro defined in Development:SIMNET:MCC:libsim:m2.c.	
Called By		
Function	**Where Described**	
M2PlaceEvent	See Section 2.22.2.8.6.	

Table 2.22-155 M2FillDetail Information.

2.22.2.8.5 M2PlaceFetch

M2PlaceFetch reads the current values of the fuel and tow load buffers into the M2 Placement dialog to display to the user. The function call is M2PlaceFetch(dialog). Table 2.22-156 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
dialog	pointer to DialogState	Development:SIMNET:libmac:dialog.h

Called By	
Function	**Where Described**
M2LoadSavedData	See Section 2.22.2.8.3.
M2DetailEvent	See Section 2.22.2.8.8.

Table 2.22-156 M2PlaceFetch Information.

2.22.2.8.6 M2PlaceEvent

M2PlaceEvent fields events in the M2 Placement dialog. The function call is M2PlaceEvent(dialog, itemNo). Table 2.22-157 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
dialog	pointer to DialogState	Development:SIMNET:libmac:dialog.h
itemNo	int	Standard C type.
Calls		
Function	**Where Described**	
showhelp	See Section 2.22.1.19.4.	
CheckMandatoryFields	See Section 2.22.1.13.1.	
LimitCheck	See Section 2.22.2.8.1.	
ShowCaution	See Section 2.22.1.4.1.	
M2FillDetail	See Section 2.22.2.8.4.	
CompleteVehicleDialog	See Section 2.22.2.9.6.	
ShowDialog	See Section 2.22.1.11.1.	
ShowWindow	Standard Window Manager function for Macintosh.	

Table 2.22-157 M2PlaceEvent Information.

2.22.2.8.7 M2DetailFetch

M2DetailFetch reads in the current stowed missile limits into the M2 Detail dialog for display to the user. The function call is M2DetailFetch(dialog). Table 2.22-158 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
dialog	pointer to DialogState	Development:SIMNET:libmac:dialog.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
str	array of 10 char	Standard C type.
Calls		
Function	**Where Described**	
SetStaticText	See Section 2.22.1.11.8.	

Table 2.22-158 M2DetailFetch Information.

2.22.2.8.8 M2DetailEvent

M2DetailEvent fields events in the M2 Detail dialog. The function call is M2DetailEvent(dialog, itemNo). Table 2.22-159 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
dialog	pointer to DialogState	Development:SIMNET:libmac:dialog.h
itemNo	int	Standard C type.

Calls	
Function	**Where Described**
showhelp	See Section 2.22.1.19.4.
CheckMandatoryFields	See Section 2.22.1.13.1.
LimitCheck	See Section 2.22.2.8.1.
ShowCaution	See Section 2.22.1.4.1.
M2PlaceFetch	See Section 2.22.2.8.5.
UpdateDialog	See Section 2.22.1.11.2.
ThrowDialog	See Section 2.22.1.11.3.

Table 2.22-159 M2DetailEvent Information.

2.22.2.9 sim.c

Development:SIMNET:MCC:libsim:sim.c

sim.c contains various routines supporting dialogs for placing and reconstituting vehicle simulators. Table 2.22-160 describes the variables used by sim.c.

Variables		
Variable	**Type**	**Where Typedef Declared**
currentSim	pointer to SimDescriptor	Development:SIMNET:MCC:libsim:libsim.h
vehCancelField	PushButtonFieldDefn	Development:SIMNET:libmac:dialog.h
vehBumperField	NumberFieldDefn	Development:SIMNET:libmac:dialog.h
vehTeamAField	RBFieldDefn	Development:SIMNET:libmac:dialog.h
vehTeamBField	RBFieldDefn	Development:SIMNET:libmac:dialog.h
vehObserverField	RBFieldDefn	Development:SIMNET:libmac:dialog.h
vehRectField	RectFieldDefn	Development:SIMNET:libmac:dialog.h
vehLocationField	CoordinatesFieldDefn	Development:SIMNET:libmac:dialog.h
vehBowField	NumberFieldDefn	Development:SIMNET:libmac:dialog.h

Table 2.22-160 sim.c Variable Information.

2.22.2.9.1 SetUpSimulators

SetUpSimulators initializes the simulator data structures. The function call is SetUpSimulators(). Table 2.22-161 describes the internal variables used and functions called using this function.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	register int	Standard C type.
sim	register pointer to SimDescriptor	Development:SIMNET:MCC: libsim:libsim.h
Calls		
Function	**Where Described**	
NewPtr	Standard Memory Manager function for Macintosh.	
OpenResFile	Standard Resource Manager function for Macintosh.	
DeepShit	See Section 2.22.1.8.2.	
ExitToShell	Standard Segment Loader function for Macintosh.	

Table 2.22-161 SetUpSimulators Information.

2.22.2.9.2 SimulatorTypeCStr

SimulatorTypeCStr returns a C string naming a simulator type. The function call is SimulatorTypeCStr(simulatorType). Table 2.22-162 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
simulatorType	int	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
simulatorTypes[simulatorType].nameCstr	pointer to char	C string naming simulator type.

Table 2.22-162 SimulatorTypeCStr Information.

2.22.2.9.3 SimulatorTypePStr

SimulatorTypePStr returns a Pascal string naming a simulator type. The function call is SimulatorTypePStr(simulatorType). Table 2.22-163 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
simulatorType	int	Standard C type.

Return Values		
Return Value	**Type**	**Meaning**
simulatorTypes[simulatorType].namePstr	pointer to char	Pascal string naming simulator type.

Table 2.22-163 SimulatorTypePStr Information.

2.22.2.9.4 CompanyName

CompanyName returns a string naming a specified company, *company*. This routine uses and modifies a static buffer. The string returned is only valid until the next call to CompanyName. The function call is CompanyName(company). Table 2.22-164 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
company	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
unitString	pointer to array of char	Standard C type.
coString	pointer to char	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
unitString[company]	pointer to char	The name of the unit
coString	pointer to char	The name of the company
Called By		
Function	**Where Described**	
SimDrawCompany	Se Section 2.22.2.10.5.	

Table 2.22-164 CompanyName Information.

2.22.2.9.5 ShowVehicleDialog

ShowVehicleDialog displays a dialog for placing or reconstituting a vehicle. The function call is ShowVehicleDialog(sim, reconstFlag, completion). Table 2.22-165 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
sim	register pointer to SimDescriptor	Development:SIMNET:MCC: libsim:libsim.h
reconstFlag	int	Standard C type.
(completion)()	void function call	Standard

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
simData	register pointer to SimulatorTypeData	Development:SIMNET:MCC:libsim:libsim_int.h
str	array of 10 char	Standard C type.
tempPlaceBuffer	SimPlaceData	Development:SIMNET:MCC:libsim:libsim_int.h
Return Values		
Return Value	**Type**	**Meaning**
0	int	Unsuccessful.
1	int	Successful.
Calls		
Function	**Where Described**	
ShowCaution	See Section 2.22.1.4.1.	
DownloadVehicleParameters	See Section 2.22.2.3.3.	
ShowDialog	See Section 2.22.1.11.1.	
SetStaticText	See Section 2.22.1.11.8.	
LabelForceButtons	See Section 2.22.1.16.1.	

Table 2.22-165 ShowVehicleDialog Information.

2.22.2.9.6 CompleteVehicleDialog

CompleteVehicleDialog closes a dialog for placing or reconstituting a vehicle. The function call is CompleteVehicleDialog(success). Table 2.22-166 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
success	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
pastSim	pointer to SavedDefaults	Development:SIMNET:MCC:libsim:libsim.h
Calls		
Function	**Where Described**	
UploadVehicleParameters	See Section 2.22.2.3.2.	
FindOldSimulators	See Section 2.22.2.9.9.	
NextDefaultSlot	See Section 2.22.2.9.10.	
ShowCaution	See Section 2.22.1.4.1.	
ThrowDialog	See Section 2.22.1.11.3.	
Called By		
Function	**Where Described**	
FREDPlaceEvent	See Section 2.22.2.5.2.	
M1PlaceEvent	See Section 2.22.2.7.5.	
M2PlaceEvent	See Section 2.22.2.8.6.	

Table 2.22-166 CompleteVehicleDialog Information.

2.22.2.9.7 ValidBumperNumber

ValidBumperNumber checks that a bumper number entered in a vehicle placement dialog is unique. The function call is ValidBumperNumber(dialog, fp, str). Table 2.22-167 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
dialog	pointer to DialogState	Development:SIMNET:libmac:dialog.h
fp	pointer to TypeInFieldDefn	Development:SIMNET:libmac:dialog.h
str	pointer to char	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
sim	register pointer to SimDescriptor	Development:SIMNET:MCC:libsim:libsim.h
i	register short	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
0	int	Another vehicle already has that bumper number.
1	int	Unique bumper number.
Calls		
Function	**Where Described**	
ShowCaution	See Section 2.22.1.4.1.	
M1LoadSavedData	See Section 2.22.2.7.2.	
M2LoadSavedData	See Section 2.22.2.8.3.	
FREDLoadSavedData	See Section 2.22.2.5.2.	

Table 2.22-167 ValidBumperNumber Information.

2.22.2.9.8 LocationInExerciseArea

LocationInExerciseArea checks that a location is within the bounds of the terrain database. The function call is LocationInExerciseArea(dialog, fp,str). Table 2.22-168 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
dialog	pointer to DialogState	Development:SIMNET:libmac:dialog.h
fp	pointer to TypeInFieldDefn	Development:SIMNET:libmac:dialog.h
str	pointer to char	Standard C type.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
c	register pointer to MapCoordinates	Development:SIMNET:libmac: map.h
msg	array of 60 char	Standard C type.
Return Values		
Return Value	**Type**	**Meaning**
0	int	Location outside bounds.
1	int	Location inside bounds.
Calls		
Function	**Where Described**	
ShowCaution	See Section 2.22.1.4.1.	

Table 2.22-168 LocationInExerciseArea Information.

2.22.2.9.9 FindOldSimulator

FindOldSimulator looks up a simulator by company and bumper number in the table of simulator default settings. The function call is FindOldSimulator(company, bumper, simulatorType). Table 2.22-169 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
company	char	Standard C type.
bumper	char	Standard C type.
simulatorType	char	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	register int	Standard C type.
sim	register pointer to SavedDefaults	Development:SIMNET:MCC: libsim:libsim.h
Return Values		
Return Value	**Type**	**Meaning**
0	pointer to SavedDefaults	Simulator not found.
sim	pointer to SavedDefaults	Simulator found.
Called By		
Function	**Where Described**	
FREDLoadSavedData	See Section 2.22.2.5.2.	
M1LoadSavedData	See Section 2.22.2.7.2.	
M2LoadSavedData	See Section 2.22.2.8.3.	
CompleteVehicleDialog	See Section 2.22.2.9.6.	

Table 2.22-169 FindOldSimulator Information.

2.22.2.9.10 NextDefaultSlot

NextDefaultSlot returns a pointer to the next entry for storing simulator default settings. The function call is NextDefaultSlot(). Table 2.22-170 describes the internal variables used by this function.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	register int	Standard C type.
sim	register pointer to SavedDefaults	Development:SIMNET:MCC: libsim:libsim.h
Return Values		
Return Value	**Type**	**Meaning**
0	pointer to SavedDefaults	No entry found.
sim	pointer to SavedDefaults	Next entry.
Called By		
Function	**Where Described**	
CompleteVehicleDialog	See Section 2.22.2.9.6.	

Table 2.22-170 NextDefaultSlot Information.

2.22.2.10 table.c

Development:SIMNET:MCC:libsim:table.c

table.c contains routines displaying a table of vehicle simulators available for placement.

2.22.2.10.1 SimTableSetup

SimTableSetup populates a scrolling table of simulators with all those in a particular company, *company*. The function call is SimTableSetup(defn, company). Table 2.22-171 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
defn	pointer to ScrollTableFieldDefn	Development:SIMNET:libmac: scroll.h
company	unsigned char	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
sim	pointer to regiser SimDescriptor	Development:SIMNET:MCC: libsim:libsim.h
entry	register SimListEntryHandle	Development:SIMNET:MCC: libsim:libsim_int.h
i	short	Standard C type.

Calls	
Function	**Where Described**
NewHandle	Standard Memory Manager function for Macintosh.
InstallScrollTableEntry	See Section 2.22.1.21.1.

Table 2.22-171 SimTableSetup Information.

2.22.2.10.2 SimTableUpdate

SimTableUpdate inserts or updates an entry in a table of simulators. The function call is SimTableUpdate(defn, sim). Table 2.22-172 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
defn	pointer to ScrollTableFieldDefn	Development:SIMNET:libmac: scroll.h
sim	register pointer to SimDescriptor	Development:SIMNET:MCC: libsim:libsim.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
key	long	Standard C type.
entry	register ScrollTableEntryHandle	Development:SIMNET:libmac: scroll.h
Calls		
Function	**Where Described**	
LookupScrollTableEntry	See Section 2.22.1.24.1.	
UpdateScrollTableEntry	See Section 2.22.1.31.1.	
NewHandle	Standard Memory Manager function for Macintosh.	
InstallScrollTableEntry	See Section 2.22.1.21.1.	

Table 2.22-172 SimTableUpdate Information.

2.22.2.10.3 SimTableEntry

SimTableEntry finds the record for the simulator represented by a particular simulator table entry, *entry*. The function call is SimTableEntry(entry). Table 2.22-173 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
entry	ScrollTableEntryHandle	Development:SIMNET:libmac: scroll.h
Return Values		
Return Value	**Type**	**Meaning**
&simulators[((SimListEntryHandle)entry)->sim]	pointer to SimDescriptor	Development:SIMNET:MCC: libsim:libsim.h

Table 2.22-173 SimTableEntry Information.

2.22.2.10.4 SimDrawBumper

SimDrawBumper fills in the fields of the bumper column in the simulator table to display to the user. The function call is SimDrawBumper(defn, col, entry, box). Table 2.22-174 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
defn	pointer to ScrollTableFieldDefn	Development:SIMNET:libmac: scroll.h
col	pointer to ScrollTableColumnDefn	Development:SIMNET:libmac: scroll.h
entry	ScrollTableEntryHandle	Development:SIMNET:libmac: scroll.h
box	pointer to Rect	Development:THINK C: Mac #includes:MacTypes.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
sim	register pointer to SimDescriptor	Development:SIMNET:MCC: libsim:libsim.h
str[10]	char	Standard C type.
Calls		
Function	**Where Described**	
NumToString	Standard Binary to Decimal Conversion Package function for Macintosh.	
Move	Standard Quickdraw function for Macintosh.	
StringWidth	Standard Quickdraw function for Macintosh.	
DrawString	Standard Quickdraw function for Macintosh.	

Table 2.22-174 SimDrawBumper Information.

2.22.2.10.5 SimDrawCompany

SimDrawCompany fills in the fields of the company column in the simulator table to display to the user. The function call is SimDrawCompany(defn, col, entry, box). Table 2.22-175 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
defn	pointer to ScrollTableFieldDefn	Development:SIMNET:libmac: scroll.h
col	pointer to ScrollTableColumnDefn	Development:SIMNET:libmac: scroll.h
entry	ScrollTableEntryHandle	Development:SIMNET:libmac: scroll.h
box	pointer to Rect	Development:THINK C: Mac #includes:MacTypes.h

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
sim	register pointer to SimDescriptor	Development:SIMNET:MCC: libsim:libsim.h
cp	register pointer to char	Standard C type.
Calls		
Function	**Where Described**	
CompanyName	See Section 2.22.2.9.4.	
DrawText	Standard Quickdraw function for Macintosh.	

Table 2.22-175 SimDrawCompany Information.

2.22.2.10.6 SimDrawLocation

SimDrawLocation fills in the fields of the location column in the simulator table to display to the user. The function call is SimDrawLocation(defn, col, entry, box). Table 2.22-176 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
defn	pointer to ScrollTableFieldDefn	Development:SIMNET:libmac: scroll.h
col	pointer to ScrollTableColumnDefn	Development:SIMNET:libmac: scroll.h
entry	ScrollTableEntryHandle	Development:SIMNET:libmac: scroll.h
box	pointer to Rect	Development:THINK C: Mac #includes:MacTypes.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
sim	register pointer to SimDescriptor	Development:SIMNET:MCC: libsim:libsim.h
Calls		
Function	**Where Described**	
DrawText	Standard Quickdraw function for Macintosh.	

Table 2.22-176 SimDrawLocation Information.

2.22.2.10.7 SimDrawPlaced

SimDrawPlaced fills in the fields of the placed column in the simulator table to display to the user. The function call is SimDrawPlaced(defn, col, entry, box). Table 2.22-177 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
defn	pointer to ScrollTableFieldDefn	Development:SIMNET:libmac: scroll.h
col	pointer to ScrollTableColumnDefn	Development:SIMNET:libmac: scroll.h
entry	ScrollTableEntryHandle	Development:SIMNET:libmac: scroll.h
box	pointer to Rect	Development:THINK C: Mac #includes:MacTypes.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
sim	register pointer to SimDescriptor	Development:SIMNET:MCC: libsim:libsim.h
Calls		
Function	**Where Described**	
Move	Standard Quickdraw function for Macintosh.	
StringWidth	Standard Quickdraw function for Macintosh.	
DrawString	Standard Quickdraw function for Macintosh.	

Table 2.22-177 SimDrawPlaced Information.

2.22.2.10.8 SimDrawSide

SimDrawSide fills in the fields of the side column in the simulator table to display to the user. The function call is SimDrawSide(defn, col, entry, box). Table 2.22-178 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
defn	pointer to ScrollTableFieldDefn	Development:SIMNET:libmac: scroll.h
col	pointer to ScrollTableColumnDefn	Development:SIMNET:libmac: scroll.h
entry	ScrollTableEntryHandle	Development:SIMNET:libmac: scroll.h
box	pointer to Rect	Development:THINK C: Mac #includes:macTypes.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
sim	register pointer to SimDescriptor	Development:SIMNET:MCC: libsim:libsim.h
Calls		
Function	**Where Described**	
DrawString	Standard Quickdraw function for Macintosh.	
ForceNamePStr	See Section 2.22.1.16.2.	

Table 2.22-178 SimDrawSide Information.

2.22.2.10.9 SimDrawSimulator

SimDrawSimulator fills in the fields of the simulator column in the simulator table to display to the user. The function call is SimDrawSimulator(defn, col, entry, box). Table 2.22-179 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
defn	pointer to ScrollTableFieldDefn	Development:SIMNET:libmac: scroll.h
col	pointer to ScrollTableColumnDefn	Development:SIMNET:libmac: scroll.h
entry	ScrollTableEntryHandle	Development:SIMNET:libmac: scroll.h
box	pointer to Rect	Development:THINK C: Mac #includes:MacTypes.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
sim	register pointer to SimDescriptor	Development:SIMNET:MCC: libsim:libsim.h
str[10]	char	Standard C type.
Calls		
Function	**Where Described**	
NumToString	Standard Binary to Decimal Conversion Package function for Macintosh.	
Move	Standard Quickdraw function for Macintosh.	
StringWidth	Standard Quickdraw function for Macintosh.	
DrawString	Standard Quickdraw function for Macintosh.	
DrawChar	Standard Quickdraw function for Macintosh.	

Table 2.22-179 SimDrawSimulator Information.

2.22.2.10.10 SimDrawType

SimDrawType fills in the fields of the type column in the simulator table to display to the user. The function call is SimDrawType(defn, col, entry, box). Table 2.22-180 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
defn	pointer to ScrollTableFieldDefn	Development:SIMNET:libmac: scroll.h
col	pointer to ScrollTableColumnDefn	Development:SIMNET:libmac: scroll.h
entry	ScrollTableEntryHandle	Development:SIMNET:libmac: scroll.h
box	pointer to Rect	Development:THINK C: Mac #includes:MacTypes.h

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
sim	register pointer to SimDescriptor	Development:SIMNET:MCC: libsim:libsim.h
Calls		
Function	**Where Described**	
DrawString	Standard Quickdraw function for Macintosh.	

Table 2.22-180 SimDrawType Information.

2.22.2.11 Sim Pictures

Sim Pictures contains resources that determine the appearance of the user interface presented for placing and reconstituting vehicle simulators.

2.22.2.12 resource.h

Development:SIMNET:MCC:libsim:resource.h

resource.h defines the numbers of the resources present in the Sim Pictures resource file.

2.22.3 libsupply

Folder: "Development:SIMNET:MCC:libsupply"
This folder contains a library of C definitions representing the characteristics of various kinds of ammunition known to the MCC system. The library is used by both the SCC and Admin/Log Macintosh applications. The library is composed from the following files:

2.22.3.1 libsupply.h

Development:SIMNET:MCC:libsupply:libsupply.h

libsupply.h defines the external interface to the libsupply library. Table 2.22-181 describes the variables used by libsupply.h.

Variables		
Variable	**Type**	**Where Typedef Declared**
ammoCharacteristics	extern array of AmmoCharacteristics	Development:SIMNET:MCC: libsupply:libsupply.h
ammoTableColumns	extern array of ScrollTableColumnDefn	Development:SIMNET:libmac: scroll.h

Table 2.22-181 libsupply.h Variable Information.

2.22.3.2 data.c
Development:SIMNET:MCC:libsupply:data.c

data.c defines a table containing the name, weight and volume of each type of ammunition. Table 2.22-182 describes the variables used by data.c.

Variables		
Variable	**Type**	**Where Typedef Declared**
ammoCharacteristics	array of[numberAmmoVarieties AmmoCharacteristics	Development:SIMNET:MCC: libsupply:libsupply.h

Table 2.22-182 data.c Variable Information.

2.22.3.3 version.h
Development:SIMNET:MCC:libsupply:version.h

version.h defines a constant determining whether the libsupply library is compiled to represent the characteristics of US or Soviet ammunition.

2.22.3.4 transfer.c
Development:SIMNET:MCC:libsupply:transfer.c

transfer.c implements a dialog for transferring ammunition between one truck or stockpile and another. Table 2.22-183 describes the variables used by transfer.c.

Variables		
Variable	**Type**	**Where Typedef Declared**
parties	array of 2 TransferParty	Development:SIMNET:MCC: libsupply:libsupply.h
(completionCallback)()	void function call	Standard
transferQuantity	int	Standard C type.
direction	char	Standard C type.
transferAmmoArrowBtns	array of 2 RBFieldDefn	Development:SIMNET:libmac: dialog.h
transferAmmoOKBtn	staticPushButtonFieldDefn	Development:SIMNET:libmac: dialog.h
transferAmmoTransferBtn	staticPushButtonFieldDefn	Development:SIMNET:libmac: dialog.h
transferAmmoQuantityField	NumberFieldDefn	Development:SIMNET:libmac: dialog.h
transferAmmoTables	array of 2 ScrollTableFieldDefn	Development:SIMNET:libmac: scroll.h
transferAmmoFieldList	pointer to array of FieldDefn	Development:SIMNET:libmac: dialog.h
transferAmmoDialog	DialogDefn	Development:SIMNET:libmac: dialog.h

Table 2.22-183 transfer.c Variable Information.

2.22.3.4.1 ShowTransferAmmoDialog

ShowTransferAmmoDialog puts up a dialog for transferring ammunition. The function call is ShowTransferAmmoDialog(dialog, resourceID, leftParty, rightParty, callback). Table 2.22-184 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
dialog	Ptr	Development:THINK C: Mac #includes:MacTypes.h
resourceID	int	Standard C type.
leftParty	pointer to TransferParty	Development:SIMNET:MCC: libsupply:libsupply.h
rightParty	pointer to TransferParty	Development:SIMNET:MCC: libsupply:libsupply.h
(callback)()	void function call	Standard
Return Values		
Return Value	**Type**	**Meaning**
(DialogState) dialog	pointer to DialogState	Ammo transfer dialog.
Calls		
Function	**Where Described**	
ShowDialog	See Section 2.22.1.11.1.	
SellText	Standard Dialog Manager function for Macintosh.	

Table 2.22-184 ShowTransferAmmoDialog Information.

2.22.3.4.2 TransferAmmoFetch

TransferAmmoFetch loads values into an ammo transfer dialog, *dialog*. The function call is TransferAmmoFetch(dialog). Table 2.22-185 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
dialog	pointer to DialogState	Development:SIMNET:libmac: dialog.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
weight	long	Standard C type.
volume	long	Standard C type.
theType	int	Standard C type.
theItem	Handle	Development:THINK C: Mac #includes:MacTypes.h
rect	Rect	Development:THINK C: Mac #includes:MacTypes.h

Calls	
Function	**Where Described**
SetStaticText	See Section 2.22.1.11.8.
UpdateTransferAmmoLoads	See Section 2.22.3.4.4.

Table 2.22-185 TransferAmmoFetch Information.

2.22.3.4.3 TransferAmmoEvent

TransferAmmoEvent fields an event in a transfer ammo dialog. The function call is TransferAmmoEvent(dialog, itemNo). Table 2.22-186 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
dialog	pointer to DialogState	Development:SIMNET:libmac: dialog.h
itemNo	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
party	register pointer to TransferParty	Development:SIMNET:MCC: libsupply:libsupply.h
ammoType	register short	Standard C type.
weight	long	Standard C type.
volume	long	Standard C type.
str	array of 100 char	Standard C type.
savePort	GrafPtr	Development:THINK C: Mac #includes:Quickdraw.h
theType	int	Standard C type.
theItem	Handle	Development:THINK C: Mac #includes:MacTypes.h
rect	Rect	Development:THINK C: Mac #includes:MacTypes.h
Return Values		
Return Value	**Type**	**Meaning**
0	int	Unsuccessful.
itemNo	int	

Calls	
Function	**Where Described**
SellText	Standard Dialog Manager function for Macintosh.
ShowCaution	See Section 2.22.1.4.1.
TotalAmmoCapacity	See Section 2.22.3.9.1.
TransferBetweenAmmoLists	See Section 2.22.3.5.11.
EnableControl	See Section 2.22.1.7.2.
DisableControl	See Section 2.22.1.7.1.
SetRadioButton	See Section 2.22.1.11.6.
UpdateCapacityDisplay	See Section 2.22.3.4.5.
DisposeScrollTable	See Section 2.22.1.30.2.
ThrowDialog	See Section 2.22.1.11.3.

Table 2.22-186 TransferAmmoEvent Information.

2.22.3.4.4 UpdateTransferAmmoLoads

UpdateTransferAmmoLoads updates the display of the loads on board the two parties. The function call is UpdateTransferAmmoLoads(dialog). Table 2.22-187 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
dialog	pointer to DialogState	Development:SIMNET:libmac: dialog.h
Calls		
Function	**Where Described**	
EmptyScrollTable	See Section 2.22.1.28.2.	
FillAmmunitionList	See Section 2.22.3.5.1.	
SetScrollTableSelection	See Section 2.22.1.37.2.	
EnableControl	See Section 2.22.1.7.2.	
DisableControl	See Section 2.22.1.7.1.	
SetRadioButton	See Section 2.22.1.11.6.	
UpdateCapacityDisplay	See Section 2.22.3.4.5.	
Called By		
Function	**Where Described**	
TransferAmmoFetch	See Section 2.22.3.4.2.	

Table 2.22-187 UpdateTransferAmmoLoads Information.

2.22.3.4.5 UpdateCapacityDisplay

UpdateCapacityDisplay updates the display of the weight and volume of ammunition on board the two parties. The function call is UpdateCapacityDisplay(dialog). Table 2.22-188 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
dialog	pointer to DialogState	Development:SIMNET:libmac:dialog.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
party	register pointer to TransferParty	Development:SIMNET:MCC:libsupply:libsupply.h
i	register short	Standard C type.
weight	long	Standard C type.
volume	long	Standard C type.
weightString	array of 50 char	Standard C type.
volumeString	array of 50 char	Standard C type.
Calls		
Function	**Where Described**	
TotalAmmoCapacity	See Section 2.22.3.9.1.	
SetStaticText	See Section 2.22.1.11.8.	
Called By		
Function	**Where Described**	
TransferAmmoEvent	See Section 2.22.3.4.3.	
UpdateTransferAmmoLoads	See Section 2.22.3.4.4.	

Table 2.22-188 UpdateCapacityDisplay Information.

2.22.3.5 ammolist.c

Development:SIMNET:MCC:libsupply:ammolist.c

ammolist.c implements the scrolling lists of ammunition types and quantities. Table 2.22-189 describes the variables used by ammolist.c.

Variables		
Variable	**Type**	**Where Typedef Declared**
ammoTableColumns	array of numberAmmoTableColumns ScrollTableColumnDefn	Development:SIMNET:libmac:scroll.h

Table 2.22-189 ammolist.c Variable Information.

2.22.3.5.1 FillAmmunitionList

FillAmmunitionList populates a scrolling list of ammunition quantities from a manifest, *manifest*. The function call is FillAmmunitionList(defn, manifest). Table 2.22-190 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
defn	pointer to ScrollTableFieldDefn	Development:SIMNET:libmac:scroll.h
manifest	Manifest	Development:SIMNET:MCC:include:resupply.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	register short	Standard C type.
weight	long	Standard C type.
volume	long	Standard C type.
entry	AmmoListEntryHandle	Development:SIMNET:MCC:libsupply:libsupply.h
Calls		
Function	**Where Described**	
NewAmmoListEntry	See Section 2.22.3.5.2.	
InstallAmmoEntry	See Section 2.22.3.5.3.	
Called By		
Function	**Where Described**	
UpdateTransferAmmoLoads	See Section 2.22.3.4.4.	

Table 2.22-190 FillAmmunitionList Information.

2.22.3.5.2 NewAmmoListEntry

NewAmmoListEntry allocates a new entry for a scrolling list of ammunition quantities. The function call is NewAmmoListEntry(type, quantity, weight, volume). Table 2.22-191 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
type	short	Standard C type.
quantity	short	Standard C type.
weight	long	Standard C type.
volume	long	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
ammo	register AmmoListEntryHandle	Development:SIMNET:MCC:libsupply:libsupply.h

Return Values		
Return Value	**Type**	**Meaning**
ammo	AmmoListEntryHandle	New ammo list entry.
Calls		
Function	**Where Described**	
NewHandle	Standard Memory Manager function for Macintosh.	
Called By		
Function	**Where Described**	
FillAmmunitionList	See Section 2.22.3.5.1.	
TransferAmmoBetweenLists	See Section 2.22.3.5.11.	
UpdateAmmoList	See Section 2.22.3.5.12.	

Table 2.22-191 NewAmmoListEntry Information.

2.22.3.5.3 InstallAmmoEntry

InstallAmmoEntry installs a new entry into a scrolling list of ammuniton quantities. The function call is InstallAmmoEntry(ammo, defn). Table 2.22-192 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
ammo	AmmoListEntryHandle	Development:SIMNET:MCC:libsupply:libsupply.h
defn	pointer to ScrollTableFieldDefn	Development:SIMNET:libmac:scroll.h
Calls		
Function	**Where Described**	
InstallScrollTableEntry	See Section 2.22.1.21.1.	
Called By		
Function	**Where Described**	
FillAmmunitionList	See Section 2.22.3.5.1.	
TransferAmmoBetweenLists	See Section 2.22.3.5.11.	
UpdateAmmoList	See Section 2.22.3.5.12.	

Table 2.22-192 InstallAmmoEntry Information.

2.22.3.5.4 RemoveAmmoEntry

RemoveAmmoEntry removes an entry, *entry*, from a scrolling list of ammunition quantities. The function call is RemoveAmmoEntry(entry, defn). Table 2.22-193 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
entry	AmmoListEntryHandle	Development:SIMNET:MCC: libsupply:libsupply.h
defn	pointer to ScrollTableFieldDefn	Development:SIMNET:libmac: scroll.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
h	ScrollTableEntryHandle	Development:SIMNET:libmac: scroll.h
Calls		
Function	**Where Described**	
ScrollToShow	See Section 2.22.1.35.1.	
SetScrollTableSelection	See Section 2.22.1.37.2.	
RemoveScrollTableEntry	See Section 2.22.1.28.1.	
Called By		
Function	**Where Described**	
TransferAmmoBetweenLists	See Section 2.22.3.5.11.	

Table 2.22-193 RemoveAmmoEntry Information.

2.22.3.5.5 SelectAmmoEntry

SelectAmmoEntry is called on a mouse click in a scrolling list of ammunition quantities. The function call is SelectAmmoEntry(defn, row, rect, theEvent). Table 2.22-194 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
defn	pointer to ScrollTableFieldDefn	Development:SIMNET:libmac: scroll.h
row	int	Standard C type.
rect	pointer to Rect	Development:THINK C: Mac #includes:MacTypes.h
theEvent	pointr to EventRecord	Development:THINK C: Mac #includes:EventMgr.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
entry	ScrollTableEntryHandle	Development:SIMNET:libmac: scroll.h
Calls		
Function	**Where Described**	
ScrollTableRowToEntry	See Section 2.22.1.34.1.	
SetScrollTableSelection	See Section 2.22.1.37.2.	

Table 2.22-194 SelectAmmoEntry Information.

2.22.3.5.6 AmmoDrawName

AmmoDrawName draws the ammo type column. The function call is AmmoDrawName(defn, col, entry, box). Table 2.22-195 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
defn	pointer to ScrollTableFieldDefn	Development:SIMNET:libmac: scroll.h
col	pointer to ScrollTableColumnDefn	Development:SIMNET:libmac: scroll.h
entry	ScrollTableEntryHandle	Development:SIMNET:libmac: scroll.h
box	pointer to Rect	Development:THINK C: Mac #includes:MacTypes.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
cp	pointer to char	Standard C type.
Calls		
Function	**Where Described**	
DrawString	Standard Quickdraw function for Macintosh.	

Table 2.22-195 AmmoDrawName Information.

2.22.3.5.7 AmmoDrawNumber

AmmoDrawNumber draws a numeric value in an ammo list. The function call is AmmoDrawNumber(col, quantity, n). Table 2.22-196 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
col	pointer to ScrollTableColumnDefn	Development:SIMNET:libmac: scroll.h
quantity	int	Standard C type.
n	long	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
string	array of 20 char	Standard C type.

Calls	
Function	**Where Described**
Move	Standard Quickdraw function for Macintosh.
DrawString	Standard Quickdraw function for Macintosh.
NumToString	Standard Binary to Decimal Conversion Package function for Macintosh.
Called By	
Function	**Where Described**
AmmoDrawQuantity	See Section 2.22.3.5.8.
AmmoDrawWeight	See Section 2.22.3.5.9.
AmmoDrawVolume	See Section 2.22.3.5.10.

Table 2.22-196 AmmoDrawNumber Information.

2.22.3.5.8 AmmoDrawQuantity

AmmoDrawQuantity is the ammo quantity column draw function. The function call is AmmoDrawQuantity(defn, col, entry, box). Table 2.22-197 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
defn	pointer to ScrollTableFieldDefn	Development:SIMNET:libmac: scroll.h
col	pointer to ScrollTableColumnDefn	Development:SIMNET:libmac: scroll.h
entry	ScrollTableEntryHandle	Development:SIMNET:libmac: scroll.h
box	pointer to Rect	Development:THINK C: Mac #includes:MacTypes.h
Calls		
Function	**Where Described**	
AmmoDrawNumber	See Section 2.22.3.5.7.	

Table 2.22-197 AmmoDrawQuantity Information.

2.22.3.5.9 AmmoDrawWeight

AmmoDrawWeight is the ammo weight column draw function. The function call is AmmoDrawWeight(defn, col, entry, box). Table 2.22-198 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
defn	pointer to ScrollTableFieldDefn	Development:SIMNET:libmac: scroll.h
col	pointer to ScrollTableColumnDefn	Development:SIMNET:libmac: scroll.h
entry	ScrollTableEntryHandle	Development:SIMNET:libmac: scroll.h
box	pointer to Rect	Development:THINK C: Mac #includes:MacTypes.h
Calls		
Function	**Where Described**	
AmmoDrawNumber	See Section 2.22.3.5.7.	

Table 2.22-198 AmmoDrawWeight Information.

2.22.3.5.10 AmmoDrawVolume

AmmoDrawVolume is the ammo volume column draw function. The function call is AmmoDrawVolume(defn, col, entry, box). Table 2.22-199 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
defn	pointer to ScrollTableFieldDefn	Development:SIMNET:libmac: scroll.h
col	pointer to ScrollTableColumnDefn	Development:SIMNET:libmac: scroll.h
entry	ScrollTableEntryHandle	Development:SIMNET:libmac: scroll.h
box	pointer to Rect	Development:THINK C: Mac #includes:MacTypes.h
Calls		
Function	**Where Described**	
AmmoDrawNumber	See Section 2.22.3.5.7.	

Table 2.22-199 AmmoDrawVolume Information.

2.22.3.5.11 TransferAmmoBetweenLists

TransferAmmoBetweenLists moves a quantity, *quantity*, of ammunition, *type*, from one list to another, *fromTable* and*toTable*. The function call is TransferAmmoBetweenLists(fromTable, toTable, type, quantity). Table 2.22-200 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
fromTable	pointer to ScrollTableFieldDefn	Development:SIMNET:libmac: scroll.h
toTable	pointer to ScrollTableFieldDefn	Development:SIMNET:libmac: scroll.h
type	int	Standard C type.
quantity	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
ammoChar	pointer to AmmoCharacteristics	Development:SIMNET:MCC: libsupply:libsupply.h
entry	register AmmoListEntryHandle	Development:SIMNET:MCC: libsupply:libsupply.h
weight	long	Standard C type.
volume	long	Standard C type.
Calls		
Function	**Where Described**	
LookupScrollTableEntry	See Section 2.22.1.24.1.	
UpdateScrollTableEntry	See Section 2.22.1.31.1.	
RemoveAmmoEntry	See Section 2.22.3.5.4.	
NewAmmoListEntry	See Section 2.22.3.5.2.	
InstallAmmoEntry	See Section 2.22.3.5.3.	
ScrollToShow	See Section 2.22.1.35.1.	
SetScrollTableSelection	See Section 2.22.1.37.2.	
Called By		
Function	**Where Described**	
TransferAmmoEvent	See Section 2.22.3.4.3.	

Table 2.22-200 TransferAmmoBetweenLists Information.

2.22.3.5.12 UpdateAmmoList

UpdateAmmoList updates an ammo list to match the contents of a manifest, *manifest*.. The function call is UpdateAmmoList(manifest, table). Table 2.22-201 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
manifest	pointer to Manifest	Development:SIMNET:MCC: include:resupply.h
table	pointer to ScrollTableFieldDefn	Development:SIMNET:libmac: scroll.h

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	register short	Standard C type.
entry	register AmmoListEntryHandle	Development:SIMNET:MCC: libsupply:libsupply.h
n	int	Standard C type.
weight	long	Standard C type.
volume	long	Standard C type.
Calls		
Function	**Where Described**	
LookupScrollTableEntry	See Section 2.22.1.24.1.	
UpdateScrollTableEntry	See Section 2.22.1.31.1.	
RemoveScrollTableEntry	See Section 2.22.1.28.1.	
NewAmmoListEntry	See Section 2.22.3.5.2.	
InstallAmmoEntry	See Section 2.22.3.5.3.	

Table 2.22-201 UpdateAmmoList Information.

2.22.3.6 libsupply_int.h

Development:SIMNET:MCC:libsupply:libsupply_int.h

libsupply_int.h supplies defintions internal to the libsupply library.

2.22.3.7 summary.c

Development:SIMNET:MCC:libsupply:summary.c

summary.c contains a routine which displays a summary of the load onboard a truck.

2.22.3.7.1 DisplayAmmoLoadSummary

DisplayAmmoLoadSummary displays a summary of the load aboard an ammo truck. The function call is DisplayAmmoLoadSummary(manifest, rect, byAmmoType). Table 2.22-202 describes the parameters used and functions called using this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
manifest	register pointer to Manifest	Development:SIMNET:MCC: include:resupply.h
rect	pointer to Rect	Development:THINK C: Mac #includes:MacTypes.h
byAmmoType	int	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
string	array of char	Standard C type.
weight	long	Standard C type.
volume	long	Standard C type.
v	short	Standard C type.
halfWidth	short	Standard C type.

Calls	
Function	**Where Described**
APPEND	Macro defined in Development:SIMNET:MCC:libsupply:summary.c.
DrawString	Standard Quickdraw function for Macintosh.
TotalAmmoCapacity	See Section 2.22.3.9.1.
MoveTo	Standard Quickdraw function for Macintosh.
StringWidth	Standard Quickdraw function for Macintosh.
PenPat	Standard Quickdraw function for Macintosh.
Line	Standard Quickdraw function for Macintosh.

Table 2.22-202 DisplayAmmoLoadSummary Information.

2.22.3.8 supply_version.h
Development:SIMNET:MCC:libsupply:supply_version.h

supply_version.h defines the version of the libsupply library to be compiled.

2.22.3.9 supply.c
Development:SIMNET:MCC:libsupply:supply.c

supply.c contains a routine related to the supply trucks.

2.22.3.9.1 TotalAmmoCapacity

TotalAmmoCapacity sums the weight and volume of a load of ammunition. The function call is TotalAmmoCapacity(manifest, weight, volume). Table 2.22-203 describes the parameters used by this function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
manifest	pointer to Manifest	Development:SIMNET:MCC: include:resupply.h
weight	pointer to long	Standard C type.
volume	pointer to long	Standard C type.
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	register short	Standard C type.
n	register int	Standard C type.
Called By		
Function	**Where Described**	
TransferAmmoEvent	See Section 2.22.3.4.3.	
UpdateCapacityDisplay	See Section 2.22.3.4.5.	
DisplayAmmoLoadSummary	See Section 2.22.3.7.1.	

Table 2.22-203 TotalAmmoCapacity Information.

2.22.4 Top-level Includes

Folder: "Development:SIMNET:include"
This folder contains C header files shared across various SIMNET Macintosh libraries and applications. They match header files of the same names found in the /simnet/include and/simnet/include/ protocol directories of MCC host software. These header files are:

2.22.4.1 basic.h
Development:SIMNET:include:basic.h

basic.h defines various types and constants used in the representation of SIMNET protocol data units.

2.22.4.2 battles.h
Development:SIMNET:include:battles.h

battles.h defines constants used to identify various kinds of battle schemes.

2.22.4.3 repair.h
Development:SIMNET:include:repair.h

repair.h defines constants for the various types of repairs simulated by the MCC system.

2.22.4.4 obj_type.h
Development:SIMNET:include:obj_type.h

obj_type.h defines constants used in SIMNET protocol data units to represent various types of simulated objects.

2.22.4.5 veh_role.h
Development:SIMNET:include:veh_role.h

veh_role.h defines constants for the various roles that can be served by a vehicle that the MCC system controls or simulates.

2.22.4.6 veh_type.h
Development:SIMNET:include:veh_type.h

veh_type.h defines constants used in SIMNET protocol data units to represent various types of vehicles.

2.22.4.7 address.h
Development:SIMNET:include:address.h

address.h defines constants associated with addressing the simulator sites.

2.22.4.6 repair_m1.h
Development:SIMNET:include:repair_m1.h

repair_m1.h defines the codes for various repairs simulated by the SIMNET M1 simulator.

2.22.4.6 repair_m2.h
Development:SIMNET:include:repair_m2.h

repair_m2.h defines the codes for various repairs simulated by the SIMNET M2 simulator.

File: "Development:SIMNET:SIMNET Resources"
This resource file contains various resources required by routines in the libmac library, such as definitions for the dialog boxes displayed by the caution and help routines.

Folder: "Development:SIMNET:Icon Button"
This folder contains source code implementing an "icon button" control, which behaves like a radio button or check box, but appears as an icon with an associated text label. This code is compiled to produce a CDEF code resource, which is then copied into the "SIMNET Resources" resource file.

Folder: "Development:SIMNET:MCC"
This folder forms the root of the hierarchy of folders containing SIMNET MCC software for the Macintosh.

2.22.5 MCC Includes

Folder: "Development:SIMNET:MCC:include"
This folder contains C header files shared across various SIMNET MCC Macintosh applications. They match header files of the same names found in the /simnet/mcc/include directory of MCC host software. These header files are:

2.22.5.1 ammo.h
Development:SIMNET:MCC:include:ammo.h

ammo.h defines constants for the various kinds of ammunition known to the MCC system.

2.22.5.2 arty.h
Development:SIMNET:MCC:include:arty.h

arty.h defines constants associated with the MCC system's artillery simulation.

2.22.5.3 console.h
Development:SIMNET:MCC:include:console.h

console.h contains definitions related to AppleTalk communication between the MCC host and its Macintosh consoles. Table 2.22-204 describes the varaibles used by console.h.

Variables		
Variable	**Type**	**Where Typedef Declared**
hostSocket	extern int	Standard C type.

Table 2.22-204 console.h Variable Information.

2.22.5.4 MCC_limits.h
Development:SIMNET:MCC:include:MCC_limits.h

MCC_limits.h defines various limits of the MCC system (e.g., the capacity of a fuel or ammunition supply truck).

2.22.5.5 options.h
Development:SIMNET:MCC:include:options.h

options.h defines a structure for representing the presence of various optional elements that the MCC system may simulate.

2.22.5.6 manifest.h
Development:SIMNET:MCC:include:manifest.h

manifest.h defines a structure for representing quantities of various kinds of ammunition and fuel.

2.22.5.7 sim_xact.h
Development:SIMNET:MCC:include:sim_xact.h

sim_xact.h defines AppleTalk communication, between the MCC host and its Macintosh consoles, for the allocation, placement and reconstitution of vehicle simulators.

2.22.5.8 fleet_type.h
Development:SIMNET:MCC:include:fleet_type.h

fleet_type.h defines the contstants distinguishing the various truck fleets simulated by the MCC system.

2.22.5.9 resupply.h
Development:SIMNET:MCC:include:resupply.h

resupply.h contains definitions associated with the MCC system's simulation of resupply trucks.

2.22.5.10 veh_assign.h
Development:SIMNET:MCC:include:veh_assign.h

veh_assign.h defines constants representing the organizational units to which a vehicle may be assigned.

INDEX BY SECTION NUMBER

AbortEvent	2.17.3.1.8
AbortFetch	2.17.3.1.7
access_147	2.20.2.20.3
access_cmc	2.20.2.20.2
ActivateConsole	2.21.1.7.2
ActivateForExerciseStart	2.1.4.2.2
ActivateForReconstitution	2.1.4.2.3
ActivateForTowingArrival	2.1.4.2.4
ActivateReplyReceived	2.1.4.2.6
ActivateReplyTimedOut	2.1.4.2.7
ActivateVehicle	2.1.4.2.8
Activating Combat Vehicle Simulators	2.1.4
actloc.c	2.1.4.1
Act_List_Add	2.1.4.1.4
Act_List_Alloc	2.1.4.1.2
Act_List_Delete	2.1.4.1.5
Act_List_Hash	2.1.4.1.1
Act_List_Init	2.1.4.1.3
Act_List_Loc_Exists	2.1.4.1.6
AdditionalSorties	2.18.2.3.2
AddMenuFile	2.17.8.4.2
address.c	2.20.1.9
address.h	2.22.4.7
AddSubscription	2.20.1.1.9
adjust.c	2.17.1.1
adjustchunks	2.22.1.19.10
AdjustEvent	2.17.1.1.2
AdjustFire	2.6.3.1.2
AdjustScreenState	2.17.8.3.4
Admin Console Definitions	2.15.4
Admin Console Software	2.15.5
Admin Pictures	2.15.6.1
Admin.h	2.15.4.1
adminlog.c	2.13.1.1
AdminMac.h	2.15.4.2
aggregate.c	2.20.1.3
alarm.c	2.21.2.4
alarm.h	2.21.2.3
AlarmsEnabled	2.21.2.4.4
alloc.c	2.13.2.1

Allocation, Placement and Reconstitution of Vehicle Simulators	2.13.2
AllocEntryEvent	2.13.2.1.6
AllocEntryFetch	2.13.2.1.5
AllocShMem	2.1.1.5.1
AllocTableEvent	2.13.2.1.4
AllocTableFetch	2.13.2.1.2
AllocTableSelect	2.13.2.1.3
Alloc_Free	2.21.1.13.2
Alloc_Minefield	2.1.8.3.2
AllotSorties	2.7.1.1.4
ALOCEvent	2.13.1.1.5
ALOCInitFetch	2.13.1.1.3
ALOCReconstFetch	2.13.1.1.4
ammo.h	2.22.5.1
AmmoDrawName	2.22.3.5.6
AmmoDrawNumber	2.22.3.5.7
AmmoDrawQuantity	2.22.3.5.8
AmmoDrawVolume	2.22.3.5.10
AmmoDrawWeight	2.22.3.5.9
ammolist.c	2.22.3.5
AmmoTableSelect	2.13.3.3.5
AmmoTransferComplete	2.13.3.3.6
AmmoTruckLoadComplete	2.15.1.1.2
AmmoTruckTableEvent	2.13.3.4.6
AmmoTruckTableFetch	2.13.3.4.5
Appearance of Admin User Interface	2.15.6
Appearance of CAS Console User Interface	2.18.3
Appearance of FSE User Interface	2.17.6
Appearance of Maint User Interface	2.16.5
Appearance of User Interface	2.14.3
AppendByte	2.11.1.12
AppendString	2.22.1.44.2
Appletalk Network Software	2.12
ApplyToBatteries	2.17.5.1.5
ApproximateCos	2.17.1.1.5
ApproximateSin	2.17.1.1.4
arty.h	2.22.5.2
AssembleMissionString	2.19.3.9.1
AssembleResourceString	2.19.3.9.2
asset.c	2.8.2.2
asset.h	2.8.2.1
AssetKilled	2.8.2.2.5
AssetReconstituted	2.8.2.2.4

assoc.h	2.20.1.26
AssocAddToBucket	2.20.1.25.1
AssocAddToEndOfTimeList	2.20.1.16.2
AssocAddToStartOfTimeList	2.20.1.16.1
AssocAddTransaction	2.20.1.23.2
AssocAttach	2.20.1.5.2
AssocBucketLookup	2.20.1.25.3
AssocCacheResponse	2.20.1.18.2
AssocClose	2.20.1.11.1
AssocCreateFreeList	2.20.1.24.1
AssocCreateMCA	2.20.1.1.7
AssocCreateMCAWithMask	2.20.1.1.8
AssocCurrentlySubscribed	2.20.1.1.3
AssocCurrentlySubscribedWithMask	2.20.1.1.6
AssocDef	2.20.1.29
AssocDeleteCachedResponse	2.20.1.18.3
AssocDeleteFromBucket	2.20.1.25.2
AssocDeleteFromTimeList	2.20.1.16.3
AssocDeleteTransaction	2.20.1.23.3
AssocError	2.20.1.10.1
AssocFindResponse	2.20.1.18.5
AssocFindTransaction	2.20.1.23.4
AssocFreeDescriptor	2.20.1.24.4
AssocGetDescriptor	2.20.1.24.3
AssocGetLastAddress	2.20.1.15.1
AssocGetRspMask	2.20.1.13.2
AssocGetSimAddress	2.20.1.9.1
AssocGrowFreeList	2.20.1.24.2
AssocInitResponse	2.20.1.18.1
AssocInitTransactions	2.20.1.23.1
AssocMoveToEndOfTimeList	2.20.1.16.4
AssocOpen	2.20.1.5.1
AssocPadBuffer	2.20.1.2.2
AssocProcessDatagramPDU	2.20.1.21.1
AssocProcessRequestPDU	2.20.1.20.1
AssocProcessResponsePDU	2.20.1.19.1
AssocReadParams	2.20.1.22.2
AssocReceiveAssocPDU	2.20.1.14.1
AssocReceivePDU	2.20.1.6.1
AssocRescheduleTransaction	2.20.1.23.5
AssocSendAggregate	2.20.1.3.1
AssocSendDatagram	2.20.1.2.1
AssocSendResponse	2.20.1.4.2

AssocSendTransact 2.20.1.4.1
AssocSetProtocolFamily 2.20.1.12.1
AssocSetRspMask 2.20.1.13.3
AssocSetSendMask 2.20.1.13.1
AssocSubscribe 2.20.1.1.1
AssocSubscribeWithMask 2.20.1.1.4
AssocTickAssocLayer 2.20.1.8.1
AssocTimeOutOldResponses 2.20.1.18.4
AssocUnsubscribe 2.20.1.1.2
AssocUnsubscribeWithMask 2.20.1.1.5
AssocWaitForPDU 2.20.1.7.1
assoc_lcl.h 2.20.1.27
atalk.c 2.13.8.6
atalk.c 2.14.2.1
atalk.c 2.15.2.1
atalk.c 2.16.1.1
atalk.c 2.22.1.3
atalk.c 2.22.2.3
ATAllocBuffer 2.21.1.2.1
ATAllocSocket 2.21.1.4.1
atatp.c 2.21.1.1
atbuf.c 2.21.1.2
ATFreeBuffer 2.21.1.2.2
ATFreeSocket 2.21.1.4.2
atnbp.c 2.21.1.3
ATPError 2.22.1.3.6
ATPGetRequest 2.21.1.1.5
ATPPut 2.21.1.7.4
ATPPut 2.22.1.3.3
ATPPutComplete 2.21.1.7.5
ATPRequest 2.21.1.1.4
ATPResponse 2.21.1.1.6
ATPSendRequest 2.21.1.1.1
ATPSendResponse 2.21.1.1.2
ATPTimeout 2.21.1.1.3
ATPTransact 2.21.1.7.3
ATReceive 2.21.1.1.7
ATRecv Process 2.10.2
ATSend Process 2.10.1
atskt.c 2.21.1.4
AttachLocks 2.21.2.6.1
AttachMsgQueue 2.21.2.8.1
AttachSharedMem 2.21.2.7.1

AttackCCV	2.1.2.6.4
Auxiliary Software	2.13.8
azimuth.c	2.21.1.5
Background	1.1
basic.h	2.22.4.1
Battalion	2.1.1.3.1
battery.c	2.17.2.1
BatteryEventHandler	2.17.2.1.4
BatterySelectionViable	2.17.5.1.9
BatteryViable	2.17.5.1.10
battles.h	2.22.4.2
BigProb	2.22.1.8.2
bigprob.c	2.22.1.8
BinNibble	2.11.1.2
Blend	2.22.1.47.3
block.c	2.20.1.7
Blow_Vehicle	2.1.8.3.12
BMOverviewEvent	2.13.8.16.4
BMOverviewFetch	2.13.8.16.3
BombRequest	2.7.1.1.5
BreachRequest	2.1.1.2.3
Bridge	2.1.1.3.2
Bridge	2.11.1.17
Bridge Program	2.11.1
BroadcastAppearance	2.1.2.3.3
BroadcastCCVBatch	2.1.2.3.2
BroadcastDeactivate	2.1.2.3.4
BroadcastToChildren	2.2.1.4.5
BtryReconstEvent	2.13.4.2.11
bucket.c	2.20.1.25
Build_ActivatePD	2.1.4.2.5
Build_Exercise_Status	2.1.7.1.6
Build_Simulation_Status	2.1.7.1.7
Build_Vehicle_Status	2.1.7.1.8
cache.h	2.21.7.1
cache_alloc.c	2.21.7.7
cache_and_file_terminate	2.21.7.7.2
cache_cntl.c	2.21.7.8
cache_data.c	2.21.7.9
cache_init	2.21.7.7.1
cache_init.c	2.21.7.10
cache_query.c	2.21.7.11
cache_queue.c	2.21.7.12

cancel.c	2.16.4.9
CancelAlarm	2.21.2.4.3
CancelAtCommand	2.17.1.3.5
CancelMission	2.18.2.6.4
CancelOffers	2.4.3.1.6
CancelRepair	2.16.4.9.5
CancelResupply	2.4.3.1.4
CancelScrollTableSelection	2.22.1.37.3
CannedStuff.c	2.19.3.1
CAS Console Definitions	2.18.1
CAS Console Software	2.18.2
CAS Pictures	2.18.3.1
cas.c	2.1.2.1
cas.c	2.2.1.1
cas.c	2.7.1.1
cas.c	2.13.4.1
CAS.h	2.18.1.1
CASMac.h	2.18.1.2
caution.c	2.22.1.4
ccv.c	2.2.3.1
ccv.c	2.21.1.6
ccv_change.c	2.1.2.6
ccv_model.c	2.1.2.3
CEDTGToString	2.19.3.10.8
centerdialog	2.22.1.19.5
CEStringToDTG	2.19.3.10.7
CEW Asset Management	2.8.2
CEW Console Definitions	2.19.1
CEW Console Software	2.19.2
CEW Objects	2.19.5
cew.c	2.2.1.6
cew.c	2.13.4.4
cew.c	2.21.1.8
cew.h	2.13.4.3
CEW.h	2.19.1.3
CEWAllocEvent	2.13.4.4.3
CEWAllocFetch	2.13.4.4.2
CEWConfirmEvent	2.13.4.4.6
CEWConfirmFetch	2.13.4.4.5
CEWHost.h	2.8.1.2
CEWMac.h	2.19.1.2
CEWReconstEvent	2.13.4.4.8
CEWReconstFetch	2.13.4.4.7

CheckBattery 2.17.5.1.8
CheckBreachDialog 2.19.3.2.7
CheckConfig 2.17.8.1.6
CheckDialogs.c 2.19.3.2
CheckDTGString 2.19.3.3.2
CheckEmplacementDialog 2.19.3.2.5
CheckFiring 2.17.1.3.6
CheckForCondition 2.18.2.1.2
CheckLastField 2.22.1.11.5
CheckList.c 2.19.5.1
CheckList.h 2.19.5.2
CheckList::AddRow 2.19.5.1.2
CheckList::GetRow 2.19.5.1.4
CheckList::GetState 2.19.5.1.6
CheckList::init 2.19.5.1.1
CheckList::SetRow 2.19.5.1.3
CheckList::SetState 2.19.5.1.5
CheckList::ToggleState 2.19.5.1.7
CheckList_Handler 2.19.3.7.2
CheckMandatoryEvent 2.13.8.8.1
CheckMandatoryFields 2.22.1.13.1
CheckMissionTypeDialog 2.19.3.2.9
CheckMoveDialog 2.19.3.2.8
CheckReviewDialog 2.19.3.2.13
CheckSchedMission 2.17.8.1.19
CheckSimulatorActivity 2.1.4.2.12
CheckSortieLimits 2.18.2.3.4
CheckStatBreachDialog 2.19.3.2.11
CheckStatEmplacementDialog 2.19.3.2.10
CheckStatMoveDialog 2.19.3.2.12
CheckTimers 2.15.5.7.2
CheckTimers 2.16.4.11.2
CheckTimers 2.17.5.1.1
CheckUTMString 2.19.3.3.1
CheckVolleyTable 2.1.2.4.3
CheckvsCurrentTime 2.19.3.3.3
CheckvsCurrentTimeStr 2.19.3.3.4
Check_Net 2.1.1.2.7
check_x_ints 2.19.4.3.2
check_x_ints 2.21.1.24.2
check_y_ints 2.19.4.3.3
check_y_ints 2.21.1.24.3
ChkSerIn 2.11.1.14

ChkSerOut	2.11.1.15
CleanupDeadSockets	2.21.2.12.6
CleanupSockets	2.21.2.12.4
ClearDialogsForMission	2.18.2.2.3
ClearField	2.19.3.9.5
ClearHot	2.18.2.6.5
Clock Reset Command	2.9.2
clock.c	2.9.2.1
clock.c	2.22.1.6
clock.h	2.22.1.5
ClockCommand	2.9.2.1.1
close.c	2.20.1.11
CloseCancelDialog	2.16.4.9.4
CloseDispatchDialog	2.15.3.1.6
CloseDispatchDialog	2.16.4.6.5
CloseHaltDialog	2.15.5.5.6
CloseHaltDialog	2.16.4.10.5
CloseLoadDialog	2.15.1.1.6
CloseNetworkInterface	2.21.1.22.4
CloseRecoverDialog	2.16.2.1.5
CloseSocket	2.21.2.10.4
CloseStartDialog	2.16.4.7.11
cmd.c	2.9.1.2
cmd_clock	2.9.1.2.2
cmd_disable	2.9.1.2.3
cmd_exit	2.9.1.2.4
cmd_list	2.9.1.2.5
cmd_perf	2.9.1.2.6
cmd_restart	2.9.1.2.7
cmd_restore	2.9.1.2.8
cmd_save	2.9.1.2.9
cmd_send_breach	2.9.1.2.1
cmd_status_all	2.9.1.2.10
cmd_status_bumper	2.9.1.2.11
cmd_status_ipc	2.9.1.2.12
cmd_status_net	2.9.1.2.13
cmd_status_process	2.9.1.2.14
cmd_status_sim	2.9.1.2.15
cmd_status_sims	2.9.1.2.16
cmd_status_vid	2.9.1.2.17
cmd_status_yumm	2.9.1.2.18
cmd_trace	2.9.1.2.19
Command Line Interface	2.9.1

CommenceFFE	2.17.1.3.4
Communication with World	2.2.4
CompanyName	2.22.2.9.4
CompleteDisplacement	2.13.8.10.1
CompleteVehicleDialog	2.22.2.9.6
ComputeAdjustment	2.17.1.1.3
ComputeETA	2.13.8.10.18
ComputeETA	2.15.3.1.4
ComputeETA	2.16.2.1.4
ComputeETA	2.16.4.6.4
comp_pct	2.13.8.10.19
ComWindow::AddHotSpot	2.19.5.3.2
ComWindow::CheckHotSpot	2.19.5.3.4
ComWindow::initial	2.19.5.3.3
ComWindow::show	2.19.5.3.1
ComWindows.c	2.19.5.3
ComWindows.h	2.19.5.4
condition.c	2.18.2.1
ConditionEvent	2.18.2.1.5
ConditionFetch	2.18.2.1.4
Configuration and Configuration Management	1.4
ConsiderVehicle	2.1.2.1.3
consistent.c	2.21.7.13
Console	2.1.1.3.3
console.c	2.21.1.7
console.h	2.19.1.6
console.h	2.22.5.3
ConsoleReleaseArrived	2.21.1.7.7
control.c	2.22.1.7
CoRoleFetch	2.13.8.11.13
CountCaution	2.22.1.4.4
CountRoundsFired	2.1.6.2.10
count_objects_in_patch	2.21.7.26.1
count_treelines_in_patch	2.21.7.29.1
count_trees_in_patch	2.21.7.30.1
createhpict	2.22.1.19.23
createte	2.22.1.19.22
Crossing_Minefield	2.1.8.3.14
CSC Descriptions	2
css.c	2.2.1.2
css.c	2.13.3.1
CSSConfirmEvent	2.13.3.1.8
CSSConfirmFetch	2.13.3.1.7

CSSTypeBranch	2.13.3.1.3
CSSTypeFetch	2.13.3.1.2
CurBattleScheme	2.1.1.3.14
Current_Minefield_Count	2.1.8.3.1
data.c	2.1.1.1
data.c	2.6.1.2
data.c	2.13.8.7
data.c	2.14.1.4
data.c	2.15.5.6
data.c	2.16.4.12
data.c	2.17.7.4
data.c	2.18.1.4
data.c	2.21.1.9
data.c	2.21.7.14
data.c	2.22.2.4
data.c	2.22.3.2
DeactivateAllVehicles	2.1.5.1.2
DeactivateReplyReceived	2.1.4.2.9
DeactivateReplyTimedOut	2.1.4.2.10
DeactivateVehicle	2.1.4.2.11
decode.c	2.21.1.10
DecodeFrame	2.11.1.8
DecodeHullToWorldMatrix	2.21.1.10.1
DecodeRotationVector	2.21.1.10.2
DecodeVehicleStatusFAAD	2.21.1.12.2
DecodeVehicleStatusFRED	2.21.1.14.2
DecodeVehicleStatusM1	2.21.1.20.2
DecodeVehicleStatusM2	2.21.1.21.2
defaults.h	2.20.1.28
DefaultTrainsEvent	2.13.3.1.4
DefineBattleScheme	2.1.1.3.15
DeleteResourceLists	2.19.3.7.3
DeleteSubscription	2.20.1.1.10
Delivery of Fire	2.6.3
demo.c	2.15.5.3
demo.c	2.16.4.4
DepotConfirmEvent	2.13.3.2.2
DepotConfirmFetch	2.13.3.2.1
depots.c	2.13.3.2
dequeue_terrain_patch	2.21.7.12.1
DetermineMines	2.19.3.2.14
dialine.c	2.22.1.10
Dialog Lists of Assets	2.19.3

Dialog Lists of Scheduled Missions	2.17.4
dialog.c	2.13.8.8
dialog.c	2.14.2.2
dialog.c	2.18.2.2
dialog.c	2.22.1.11
dialog.h	2.22.1.9
DialogEvent	2.22.1.11.4
DialogFieldSupport.c	2.19.3.3
DialogLookupField	2.22.1.12.1
DialogOutlineItem	2.19.5.5.18
DialogSeqEvent	2.22.1.39.3
DialogSeqNext	2.22.1.39.4
DialogSeqPrev	2.22.1.39.5
DialogWindow::activate	2.19.5.5.3
DialogWindow::BringToFront	2.19.5.5.17
DialogWindow::close	2.19.5.5.6
DialogWindow::equal	2.19.5.5.15
DialogWindow::GetDialogPtr	2.19.5.5.16
DialogWindow::GetItemRect	2.19.5.5.14
DialogWindow::GetText	2.19.5.5.7
DialogWindow::GetValue	2.19.5.5.10
DialogWindow::handler	2.19.5.5.2
DialogWindow::HiliteItem	2.19.5.5.12
DialogWindow::init	2.19.5.5.1
DialogWindow::quit	2.19.5.5.4
DialogWindow::SelectText	2.19.5.5.9
DialogWindow::SetText	2.19.5.5.8
DialogWindow::SetUserItem	2.19.5.5.13
DialogWindow::SetValue	2.19.5.5.11
Dialog_Windows.c	2.19.5.5
Dialog_Windows.h	2.19.5.6
dialookup.c	2.22.1.12
diamand.c	2.22.1.13
DisableControl	2.22.1.7.1
DisableTeam	2.16.4.11.5
DisableTruck	2.15.5.7.4
DiscardPopUpDialog	2.15.5.7.9
DiscardPopUpDialog	2.16.4.11.9
dispatch.c	2.15.3.1
dispatch.c	2.16.4.6
DispatchAlarm	2.21.2.4.5
displace.c	2.4.2.1
displace.c	2.5.2.1

displace.c 2.13.8.10
displace.c 2.17.3.1
displace.h 2.13.8.9
DisplaceBranch 2.13.8.10.4
DisplaceDispatchEvent 2.13.8.10.15
DisplaceDispatchFetch 2.13.8.10.14
DisplaceEvent 2.13.8.10.3
DisplaceEvent 2.17.3.1.5
DisplaceFetch 2.13.8.10.2
DisplaceFetch 2.17.3.1.3
DisplaceHaltEvent 2.13.8.10.17
DisplaceHaltFetch 2.13.8.10.16
Displacement of Artillery Batteries 2.17.3
DisplaceSelectElement 2.13.8.10.5
DisplaceSelectUnit 2.13.8.10.6
DisplayAmmoLoadSummary 2.22.3.7.1
DisplayCCV 2.21.1.6.1
DisplayMsgQStatus 2.9.3.1.4
DisplaySemStatus 2.9.3.1.3
DisplayShmStatus 2.9.3.1.2
DisposeScrollTable 2.22.1.30.2
disposetopic 2.22.1.19.17
DistBetween2Pts 2.22.1.23.2
DistributeBursts 2.6.3.1.8
ditl.h 2.18.1.6
DoAbout 2.19.3.4.2
DoAbout.c 2.19.3.4
DoMenu 2.19.2.2.2
DoneStatus 2.19.3.8.5
DoResourceStatus 2.19.3.8.1
DownloadBtryParameters 2.13.4.2.9
DownloadGunData 2.17.8.3.2
DownloadInitialData 2.15.5.1.2
DownloadInitialData 2.16.4.1.2
DownloadOptions 2.13.8.6.4
DownloadParameters 2.14.2.1.2
DownloadSimulators 2.22.2.3.1
DownloadTerrain 2.13.8.6.3
DownloadTerrainMap 2.22.1.3.4
DownloadTruckParameters 2.13.3.6.?
DownloadVehicleParameters 2.22.2.3.3
Do_Generic_StatusQuery 2.1.7.1.9
do_mode_cmd 2.20.2.?.5

do_mode_cmd_147 2.20.2.7.7
do_mode_cmd_cmc 2.20.2.7.6
draw.h 2.22.1.2
DrawBatteryRect 2.17.2.1.6
DrawBatteryText 2.17.2.1.5
DrawClock 2.22.1.6.1
DrawDispElement 2.13.8.10.9
DrawDispETA 2.13.8.10.12
DrawDispStatus 2.13.8.10.11
DrawDispUnit 2.13.8.10.10
DrawFixTable 2.22.1.42.4
DrawItemOutline 2.22.1.27.1
DrawLine 2.22.1.10.1
drawlist 2.22.1.19.7
DrawReconstElement 2.13.5.2.3
DrawReconstVehicle 2.13.5.2.4
DrawScrollTable 2.22.1.31.2
DrawScrollTableC 2.22.1.31.3
DrawStatusWindow 2.17.2.2.3
drawtopic 2.22.1.19.15
DropBombs 2.7.1.1.6
dtg.c 2.22.1.15
dtg.h 2.22.1.14
DTGElapsed 2.22.1.15.3
DTGToString 2.22.1.15.1
dump.c 2.21.7.15
dump_terrain 2.21.7.15.4
elevation.c 2.21.7.16
EmplaceRequest 2.1.1.2.2
Emplace_Generic 2.1.8.3.6
EmptyScrollTable 2.22.1.28.2
EnableControl 2.22.1.7.2
EnableMenus 2.17.8.4.4
EnableTeam 2.16.4.11.6
EnableTruck 2.15.5.7.5
EncodeFrame 2.11.1.6
EncodeVehicleStatusFAAD 2.21.1.12.1
EncodeVehicleStatusFRED 2.21.1.14.1
EncodeVehicleStatusM1 2.21.1.20.1
EncodeVehicleStatusM2 2.21.1.21.1
EndFireMission 2.17.1.2.10
EndMission 2.17.8.5.2
EndMissionBattery 2.17.5.1.7

EndOfFrame	2.11.1.13
enqueue_terrain_patch	2.21.7.12.2
error.c	2.20.1.10
error.c	2.21.1.11
error.c	2.21.2.5
error.c	2.21.7.17
EstablishBatterySelection	2.17.5.2.1
Exercise	2.1.1.3.4
exercise.c	2.13.8.11
ExerciseStopped	2.2.4.2.3
External Interfaces	1.2
Ex_Log	2.1.1.3.3
faad.c	2.21.1.12
family.c	2.20.1.12
ffe.c	2.17.1.3
FFEEvent	2.17.1.3.3
file.c	2.13.8.12
file.c	2.17.8.1
FillAmmunitionList	2.22.3.5.1
fill_bytes	2.21.1.30.1
FindAnOpenSpot	2.1.2.5.1
FindOldSimulator	2.22.2.9.9
find_height_on_poly	2.21.7.16.5
find_support	2.21.7.16.3
fire.c	2.1.2.4
fire.c	2.6.3.1
fire.c	2.17.1.2
FireEntryEvent	2.17.4.1.12
FireEvent	2.17.1.2.5
FireForEffect	2.6.3.1.3
FireRequest	2.6.3.1.1
FireShell	2.1.2.3.5
FireTableDrawDescription	2.17.4.1.8
FireTableDrawLocation	2.17.4.1.7
FireTableDrawRemarks	2.17.4.1.9
FireTableSelect	2.17.4.1.5
firetarget.c	2.17.4.1
FireVolley	2.1.2.4.2
FireVolley	2.17.5.1.2
FirstScrollTableSelection	2.22.1.37.5
fixedpt_to_mil	2.21.1.5.2
FixTableEvent	2.22.1.42.3
FixTableRowRect	2.22.1.42.5

fleet_type.h	2.22.5.8
force.c	2.22.1.16
ForceNamePStr	2.22.1.16.2
fpf.c	2.17.1.4
FPFEntryEvent	2.17.4.2.11
FPFEvent	2.17.1.4.6
FPFRequest	2.6.3.1.4
FPFTableDrawDescription	2.17.4.2.8
FPFTableDrawLeft	2.17.4.2.6
FPFTableDrawNumber	2.17.4.2.5
FPFTableDrawRight	2.17.4.2.7
FPFTableSelect	2.17.4.2.4
fpftarget.c	2.17.4.2
fqueue.c	2.21.1.13
FQueue_Analyze	2.21.1.13.9
FQueue_Create	2.21.1.13.3
FQueue_Dump_Bucket	2.21.1.13.8
FQueue_Insert	2.21.1.13.4
FQueue_Remove	2.21.1.13.5
FQueue_Retrieve	2.21.1.13.7
FQueue_Scan	2.21.1.13.6
fred.c	2.21.1.14
fred.c	2.22.2.5
FREDLoadSavedData	2.22.2.5.2
FREDPlaceEvent	2.22.2.5.3
FREDSetDefaults	2.22.2.5.1
fred_encode_failures	2.21.1.14.3
free_list.c	2.20.1.24
Free_Minefield	2.1.8.3.3
FSE Console Definitions	2.17.7
FSE Console Software	2.17.8
FSE Pictures	2.17.6.1
fse.c	2.2.1.3
fse.c	2.13.4.2
FSE.h	2.17.7.1
FSEConfirmEvent	2.13.4.2.7
FSEConfirmFetch	2.13.4.2.6
FSEMac.h	2.17.7.2
FuelTruckLoadEvent	2.15.1.1.4
FuelTruckLoadFetch	2.15.1.1.3
FuelTruckTableFetch	2.13.3.4.7
FutureEvent	2.18.2.2.7
FutureMissionTest	2.17.8.4.3

generic.c 2.21.1.15
geometry.c 2.1.8.1
GetCEWParameters 2.21.1.8.1
GetCompanyFromOrg 2.21.1.25.5
GetConfig 2.17.8.1.4
getcontenthdl 2.22.1.19.20
GetCurrentMonth 2.19.3.10.6
GetDisTime 2.19.3.3.6
GetFireDialogInfo 2.17.1.2.2
GetFPFDialogInfo 2.17.1.4.2
GetGuises 2.21.1.17.1
GetGunAzimuth 2.13.4.2.12
gethelp 2.22.1.19.19
GetInternProcID 2.21.2.12.7
GetOldStyleCompany 2.21.1.25.6
GetSCCVehicleIndex 2.2.1.6.4
GetSocketName 2.21.2.10.11
GetTerrainBounds 2.19.2.1.6
GetTruckParameters 2.21.1.29.1
GetVehicleLocation 2.1.1.5.4
get_closest_object_in_patch 2.21.7.26.5
get_closest_treeline_in_patch 2.21.7.29.5
get_closest_tree_in_patch 2.21.7.30.5
get_device_number 2.20.2.10.7
Get_DTG 2.22.1.15.4
get_grid_number 2.21.7.16.6
get_locks 2.20.2.21.1
get_nth_object_in_patch 2.21.7.26.3
get_nth_treeline_in_patch 2.21.7.29.3
get_nth_tree_in_patch 2.21.7.30.3
get_object_name_list 2.21.7.15.23
get_obstr_object_in_patch 2.21.7.26.7
get_patch.c 2.21.7.18
get_texture_name_list 2.21.7.15.22
get_type 2.20.2.10.6
GrabReqEntry 2.21.2.12.2
grid.c 2.21.1.16
gr_loc_num.c 2.21.7.19
GuideBomber 2.1.2.1.2
guise.c 2.21.1.17
halt.c 2.15.5.5
halt.c 2.16.4.10
hash.c 2.21.1.18

Hash_Get_Entry	2.21.1.13.1
header.c	2.21.7.21
HeadingSetUp	2.19.3.8.4
HeldEvent	2.18.2.2.11
HeldFetch	2.18.2.2.10
help.c	2.22.1.19
help.h	2.22.1.17
helpidindex	2.22.1.19.18
helpinit	2.22.1.19.1
helplocal.h	2.22.1.18
helpmove	2.22.1.19.13
helpshutdown	2.22.1.19.2
HelpStatus	2.19.3.8.6
HexDigit	2.11.1.1
HideCCV	2.21.1.6.2
HiliteMission	2.18.2.6.3
HiliteRect	2.19.3.10.4
HiliteScrollTableSelection	2.22.1.31.4
HotSpots.c	2.19.3.5
hpickevents	2.22.1.19.6
htopicevents	2.22.1.19.14
h_to_w.c	2.21.7.20
IgnoreStatusQuery	2.1.7.1.5
include.c	2.21.7.22
included_unit	2.1.7.1.3
including_unit	2.1.7.1.4
IndirectFireDamage	2.1.2.4.6
indone	2.19.3.5.1
inflate_rect	2.1.8.1.4
init.c	2.13.8.13
init.c	2.21.1.19
init.c	2.22.1.20
InitAlarms	2.21.2.4.1
InitArtyBranch	2.13.4.2.4
InitArtyEvent	2.13.4.2.3
InitArtyFetch	2.13.4.2.2
InitCASEvent	2.13.4.1.3
InitCCV	2.1.2.3.1
InitErrorHandling	2.21.1.11.1
Initialization	2.2.1
Initialization and Communication	2.4.1
Initialization and Communication	2.5.1
Initialization and Communication	2.6.1

Initialization and Communication 2.7.1
Initialization and Communication 2.8.1
Initialization Software 2.3.1
InitializeAllTrucks 2.4.1.1.7
InitializeAllTrucks 2.5.1.1.7
InitializeAsset 2.8.2.2.3
InitializeAssets 2.8.2.2.2
InitializeBattery 2.6.1.1.4
InitializeProcess 2.21.1.19.1
InitializeResupply 2.4.3.1.1
InitializeTruck 2.2.1.2.2
InitializeTruck 2.4.2.1.1
InitializeTruck 2.5.2.1.1
Initializing MCC Software 2.1.1
InitIndirectFire 2.1.2.4.1
InitOverviewBranch 2.13.8.13.3
InitOverviewEvent 2.13.8.13.2
InitOverviewFetch 2.13.8.13.1
InitPopDenMap 2.1.2.2.1
InitProcessTable 2.1.1.4.1
InitRadios 2.21.1.23.1
InitShMem 2.21.2.12.5
InitTeamState 2.16.4.11.1
InitToolbox 2.22.1.20.1
InitTruckState 2.15.5.7.1
InitUnit 2.21.1.19.4
InitVehicleTables 2.1.1.5.2
init_cache_map 2.21.7.10.3
Init_MCC_Processing 2.1.1.2.8
init_object_and_texture_names 2.21.7.15.21
init_patch_guards 2.21.7.10.4
init_patch_indices 2.21.7.10.1
Init_PDU_Dispatch 2.1.1.2.6
init_rect 2.1.8.1.3
init_terrain_cache 2.21.7.10.2
inside_poly 2.19.4.1.3
install.c 2.22.1.21
InstallAmmoEntry 2.22.3.5.3
InstallClock 2.22.1.6.4
InstallDrawRoutine 2.17.2.1.1
InstallFireTarget 2.17.4.1.10
InstallFPFTarget 2.17.4.2.9
InstallScrollTableEntry 2.22.1.21.1

Interface between Admin Console and MCC Host	2.15.2
Interface between Maint Console and MCC Host	2.16.1
Internal Structure	1.3
InterpolatePoints	2.17.3.2.5
InterpolatePoints	2.22.1.23.4
InterProcess Communication Status Command	2.9.3
intersect_rect	2.1.8.1.2
Introduction : MCC CSCI Description	1
InvertScrollTableRow	2.22.1.37.4
ipc.c	2.9.3.1
ipc.h	2.21.2.1
ipcerrinit	2.21.2.13.2
IPC_errinit	2.21.2.13.1
IPC_SystemError	2.21.2.5.1
IsCombatVehicle	2.1.1.5.6
IsGroundVehicle	2.1.1.5.5
isLegalGEMSSDensity	2.19.3.2.1
isLegalMinefield	2.19.3.2.2
IsMortar	2.6.2.1.1
isqrt	2.22.1.23.1
keys.h	2.21.2.2
KillServiceProcesses	2.1.5.1.3
LabelForceButtons	2.22.1.16.1
LastCaution	2.21.1.4.3
libassoc	2.20.1
libbbd	2.21.3
libipc	2.21.2
libipcerrs.c	2.21.2.13
libipcerrs.err	2.21.2.14
libmac	2.22.1
libmac.h	2.22.1.1
libmatrix	2.21.4
libmcc	2.21.1
libmove	2.21.5
libnet.h	2.20.2.26
libnetif	2.20.2
libshm	2.21.6
libsim	2.22.2
libsim.h	2.22.2.1
libsim_int.h	2.22.2.2
libsupply	2.22.3
libsupply.h	2.22.3.1
libsupply_int.h	2.22.3.6

libtdb 2.21.7
LimitCheck 2.22.2.8.1
limits.c 2.18.2.3
line_crosses_rect 2.1.8.1.1
Listening to the SIMNET Local Area Network 2.1.6
LkUpSocket 2.21.2.10.10
load.c 2.15.1.1
load.c 2.17.8.2
load.c 2.18.2.4
load.c 2.22.2.6
LoadCannedALOC 2.13.1.1.1
LoadCannedCAS 2.13.4.1.1
LoadCannedCSS 2.13.3.1.1
LoadCannedData 2.13.8.15.2
LoadCannedData 2.15.5.3.1
LoadCannedData 2.16.4.4.1
LoadCannedExercise 2.13.8.11.1
LoadCannedFireTargets 2.17.8.2.3
LoadCannedFPFTargets 2.17.8.2.4
LoadCannedFSE 2.13.4.2.1
LoadCannedLimits 2.18.2.4.2
LoadCannedResources 2.19.3.1.2
LoadCannedSchedule 2.18.2.4.3
LoadCannedSimulators 2.22.2.6.1
LoadCannedStatus 2.17.8.2.2
LoadCannedTerrainMap 2.14.2.3.2
LoadCannedTerrainMap 2.17.8.2.1
LoadCannedTerrainMap 2.18.2.4.1
LoadCannedTerrainMap 2.19.3.1.1
LoadCannedTOC 2.13.1.2.1
LoadDefaultCEWParameters 2.13.4.4.1
LoadFireBuffers 2.17.1.2.6
LoadFPFBuffers 2.17.1.4.8
LoadGunneryTargets 2.13.8.12.1
LoadOptionsPermitted 2.13.8.11.7
LoadPresets 2.13.8.12.6
LoadPresets 2.17.8.1.2
LocalArriveRequest 2.6.2.1.4
LocalDispatchRequest 2.6.2.1.3
LocalRectToGlobal 2.19.3.10.1
LocalToGMTTime 2.9.2.1.2
LocalUnitArrived 2.17.3.2.2
LocalUnitDispatched 2.17.3.2.1

LocationInExerciseArea	2.22.2.9.8
Lock	2.21.2.6.3
lock.c	2.13.8.14
lock.c	2.21.2.6
lock.c	2.21.7.23
LockDialogEvent	2.13.8.14.2
LogExerciseEnd	2.1.5.1.4
longpt.c	2.22.1.23
longpt.h	2.22.1.22
lookup.c	2.22.1.24
LookupAssetNumber	2.8.2.2.1
LookupCCVNumber	2.4.1.1.9
LookupCCVNumber	2.5.1.1.9
LookupFireTarget	2.17.4.1.3
LookupProcessIdentifier	2.21.1.26.2
LookupProcessNumber	2.21.1.26.3
LookupScrollTableEntry	2.22.1.24.1
LookupTruckNumber	2.4.1.1.8
LookupTruckNumber	2.5.1.1.8
m1.c	2.21.1.20
m1.c	2.22.2.7
M1DetailEvent	2.22.2.7.6
M1FillDetail	2.22.2.7.3
M1LoadSavedData	2.22.2.7.2
M1PlaceEvent	2.22.2.7.5
M1PlaceFetch	2.22.2.7.4
M1SetDefaults	2.22.2.7.1
m1_encode_failures	2.21.1.20.4
m2.c	2.21.1.21
m2.c	2.22.2.8
M2DetailEvent	2.22.2.8.8
M2DetailFetch	2.22.2.8.7
M2FillDetail	2.22.2.8.4
M2LoadSavedData	2.22.2.8.3
M2PlaceEvent	2.22.2.8.6
M2PlaceFetch	2.22.2.8.5
M2SetDefaults	2.22.2.8.2
m2_encode_failures	2.21.1.21.4
MacInits	2.19.2.1.3
MacintoshTime	2.21.1.27.1
MacQuit	2.19.2.1.4
main	2.1.1.2.9
main	2.2.1.4.1

main 2.3.1.1.1
main 2.4.1.1.1
main 2.5.1.1.1
main 2.6.1.1.1
main 2.7.1.1.1
main 2.8.1.1.1
main 2.9.1.1.3
main 2.10.1.1.1
main 2.10.1.2.1
main 2.13.8.15.1
main 2.14.2.3.1
main 2.15.5.1.1
main 2.16.4.1.1
main 2.17.8.3.1
main 2.18.2.5.1
main 2.19.2.1.1
main.c 2.2.1.4
main.c 2.3.1.1
main.c 2.4.1.1
main.c 2.5.1.1
main.c 2.6.1.1
main.c 2.8.1.1
main.c 2.9.1.1
main.c 2.13.8.15
main.c 2.14.2.3
main.c 2.15.5.1
main.c 2.16.4.1
main.c 2.17.8.3
main.c 2.18.2.5
main.c 2.19.2.1
MainEventLoop 2.13.8.15.3
MainEventLoop 2.14.2.3.3
MainEventLoop 2.15.5.1.3
MainEventLoop 2.16.4.1.3
MainEventLoop 2.17.8.3.3
MainEventLoop 2.18.2.5.2
MainEventLoop 2.19.2.1.2
MainHelpButton 2.19.3.5.5
Maint Console Definitions 2.16.3
Maint Console Software 2.16.4
Maint Pictures 2.16.5.1
maint.c 2.1.6.1
Maint.h 2.16.x.1

MaintMac.h	2.16.3.2
MaintTruckTableFetch	2.13.3.4.8
Make_Breach_Status	2.1.8.3.16
Make_Minefield_Status	2.1.8.3.17
make_polygon	2.1.8.1.5
manifest.h	2.22.5.6
map.c	2.21.7.24
map.c	2.22.1.26
map.h	2.21.7.2
map.h	2.22.1.25
MapAmmoCode	2.6.3.1.5
MapScrollTableSelected	2.22.1.33.1
map_buffers	2.20.2.21.2
map_enp	2.20.2.21.3
marker.c	2.1.8.2
MarkerInterval	2.1.1.3.18
MarkerSpacing	2.1.1.3.17
Marker_Add_Line	2.1.8.2.2
Marker_Broadcast_Markers	2.1.8.2.3
Marker_Emplace_Minefield	2.1.8.2.1
Marker_Get_Next_Marker	2.1.8.2.4
Marker_Point_In_Terrain	2.1.8.2.5
MarkField	2.19.3.9.4
mask.c	2.20.1.13
Masscomp Communication Software	2.1
master.c	2.13.8.16
MCC Includes	2.22.5
mccerrinit_	2.21.1.33.2
MCCRoleEvent	2.13.8.11.3
MCCRoleFetch	2.13.8.11.2
MCC_errinit	2.21.1.33.1
MCC_errs.c	2.21.1.33
MCC_errs.err	2.21.1.32
MCC_limits.h	2.22.5.4
MCC_SystemError	2.21.1.11.3
memory.c	2.21.2.7
memory.c	2.21.7.25
memory_init	2.21.7.25.2
memory_terminate	2.21.7.25.3
menu.c	2.17.8.4
MenuBarTitle	2.22.1.43.1
MenuCommand	2.17.8.4.5
menus.c	2.19.2.2

mil_to_fixedpt	2.21.1.5.1
Minefield Geometry	2.19.4
Minefield Simulation	2.1.8
minefield.c	2.1.8.3
MineScanInterval	2.1.1.3.16
Mines_Breach_Lane	2.1.8.3.8
Mines_Check_Minefields	2.1.8.3.15
Mines_Emplace_Breach	2.1.8.3.9
Mines_Emplace_Minefield	2.1.8.3.7
Mines_Get_Info	2.1.8.3.11
Mines_Max_Minefields	2.1.8.3.4
Mines_Max_Vertices	2.1.8.3.5
Mines_Point_In_Minefield	2.1.8.3.13
Mines_Process_Breach_Datagram	2.1.8.3.10
Mines_Send_Minefield_Status	2.1.8.3.18
mission.c	2.17.8.5
mission.c	2.18.2.6
MissionButton	2.19.3.5.2
MissionObj.c	2.19.5.7
MissionObj.h	2.19.5.8
MissionObj::ArrivalResources	2.19.5.7.19
MissionObj::CheckResStatus	2.19.5.7.15
MissionObj::DispatchResources	2.19.5.7.18
MissionObj::edit	2.19.5.7.2
MissionObj::GetEndTime	2.19.5.7.8
MissionObj::GetNumber	2.19.5.7.12
MissionObj::GetOrderType	2.19.5.7.23
MissionObj::GetStartActivityTime	2.19.5.7.7
MissionObj::GetStartMoveTime	2.19.5.7.6
MissionObj::GetStatus	2.19.5.7.21
MissionObj::GetType	2.19.5.7.22
MissionObj::init	2.19.5.7.1
MissionObj::NewMission	2.19.5.7.3
MissionObj::ResetMissionTime	2.19.5.7.17
MissionObj::SendArrival	2.19.5.7.30
MissionObj::SendBreach	2.19.5.7.28
MissionObj::SendDispatch	2.19.5.7.29
MissionObj::SendEmplacement	2.19.5.7.27
MissionObj::SendEndMission	2.19.5.7.26
MissionObj::SendStartActivity	2.19.5.7.25
MissionObj::SendStartMovement	2.19.5.7.24
MissionObj::SetEndTime	2.19.5.7.11
MissionObj::SetResStatus	2.19.5.7.13

MissionObj::SetStartActivityTime 2.19.5.7.10
MissionObj::SetStartMoveTime 2.19.5.7.9
MissionObj::SetStatus 2.19.5.7.20
MissionObj::show 2.19.5.7.4
MissionObj::UpdateMissionString 2.19.5.7.5
MissionObj::UpdateResAssigned 2.19.5.7.14
MissionObj::UpdateResPosition 2.19.5.7.16
MissionSelection 2.19.3.5.3
Model Operation of Artillery Batteries 2.17.5
Model Travel of Maint Teams 2.16.2
Model Travel of Supply Trucks 2.15.3
model.c 2.16.4.11
model.c 2.17.5.1
Modeling Computer-Controlled Vehicles (CCVs) 2.1.2
models.c 2.15.5.7
Monitoring of the MCC System 2.1.7
MoreSortiesEvent 2.13.4.1.5
MoreSortiesFetch 2.13.4.1.4
mother.c 2.1.1.2
move.c 2.17.3.2
msgqueue.c 2.21.2.8
MySound 2.19.3.4.5
NBPLookup 2.21.1.3.1
NBPReceive 2.21.1.3.2
NearbyCCVs 2.1.2.2.4
NearbyVehicle 2.1.2.5.2
NearestCombatVehicle 2.1.2.5.3
net.c 2.21.1.22
netlock.h 2.20.2.25
network.def 2.20.2.27
network.h 2.20.2.24
NetworkArrived 2.13.8.10.22
NetworkDispatched 2.13.8.10.21
NetworkEventHandler 2.22.1.3.5
net_acce.c 2.20.2.20
net_access 2.20.2.20.1
net_addr.c 2.20.2.2
net_addr_bin_to_str 2.20.2.2.2
net_addr_compare 2.20.2.2.1
net_addr_format_convert 2.20.2.2.6
net_addr_str_to_bin 2.20.2.2.3
net_add_mca 2.20.2.11.1
net_add_type 2.20.2.18.1

net_alive 2.20.2.1.1
net_bufs 2.20.2.6.3
net_clos.c 2.20.2.3
net_close 2.20.2.3.1
net_ctl.c 2.20.2.1
net_current_time 2.20.2.8.3
net_data.c 2.20.2.22
net_del_mca 2.20.2.11.2
net_device_base 2.20.2.8.7
net_extloop 2.20.2.7.3
net_flus.c 2.20.2.4
net_flush 2.20.2.4.1
net_getaddr 2.20.2.2.4
net_gettime 2.20.2.8.1
net_get_next_packet 2.20.2.14.4
net_get_parameters 2.20.2.10.2
net_get_rcv 2.20.2.17.9
net_get_rcv_from_addr 2.20.2.17.4
net_get_rcv_to_addr 2.20.2.17.3
net_get_rcv_type 2.20.2.17.5
net_get_recv 2.20.2.14.6
net_get_send 2.20.2.15.4
net_get_snd 2.20.2.19.7
net_get_statistics 2.20.2.5.1
net_get_timestamp 2.20.2.12.3
net_heartbeat 2.20.2.8.6
net_hostbuf_info 2.20.2.6.1
net_info.c 2.20.2.6
net_init_mca 2.20.2.11.3
net_init_time 2.20.2.8.4
net_init_type 2.20.2.18.2
net_interface_type 2.20.2.6.4
net_intloop 2.20.2.7.2
net_iocontrol 2.20.2.1.4
net_load 2.20.2.9.1
net_load.c 2.20.2.9
net_loopback 2.20.2.1.6
net_mca.c 2.20.2.11
net_mode.c 2.20.2.7
net_nopened 2.20.2.1.5
net_norm 2.20.2.7.1
net_open 2.20.2.10.1
net_open.c 2.20.2.10

net_orecv.c	2.20.2.14
net_osend.c	2.20.2.15
net_print_statistics	2.20.2.5.4
net_prom	2.20.2.7.4
net_put_timestamp	2.20.2.12.4
net_rcv	2.20.2.17.2
net_recv	2.20.2.14.1
net_recv.c	2.20.2.17
net_reg.c	2.20.2.16
net_reg_read	2.20.2.16.1
net_reg_write	2.20.2.16.2
net_release_next_packet	2.20.2.14.5
net_release_rcv	2.20.2.17.10
net_release_recv	2.20.2.14.7
net_release_send	2.20.2.15.5
net_release_snd	2.20.2.19.8
net_res	2.20.2.1.2
net_reset_lock	2.20.2.17.1
net_run	2.20.2.13.1
net_run.c	2.20.2.13
net_send	2.20.2.15.1
net_send.c	2.20.2.19
net_settime	2.20.2.8.2
net_settimeout	2.20.2.1.3
net_set_parameterss	2.20.2.10.3
net_set_snd_from_addr	2.20.2.19.5
net_set_snd_type	2.20.2.19.6
net_sharebuf_info	2.20.2.6.2
net_snd	2.20.2.19.2
net_stam.c	2.20.2.12
net_stamp_disable	2.20.2.12.2
net_stamp_enable	2.20.2.12.1
net_stat.c	2.20.2.5
net_stat_string	2.20.2.5.3
net_stomp_time	2.20.2.8.5
net_stop	2.20.2.13.2
net_stuf.c	2.20.2.21
net_syserror_info	2.20.2.6.5
net_time.c	2.20.2.8
net_type.c	2.20.2.18
net_unload	2.20.2.9.2
net_version	2.20.2.6.6
net_zeroaddr	2.20.2.2.5

net_zero_statistics 2.20.2.5.2
new.c 2.17.1.5
NewAmmoListEntry 2.22.3.5.2
NewDispElement 2.13.8.10.7
NewDispUnit 2.13.8.10.8
NewFireTarget 2.17.4.1.1
NewFPFTarget 2.17.4.2.1
NewMission 2.17.8.5.1
NewMission 2.18.2.6.1
NewMissionEvent 2.17.1.5.2
NewPresets 2.17.8.1.1
NewReconstElement 2.13.5.2.1
NewReconstVehicle 2.13.5.2.2
NewRepair 2.16.4.7.12
NewSchedMission 2.17.4.3.1
NewTarget 2.13.6.2.1
NextDefaultSlot 2.22.2.9.10
Notify 2.19.3.4.4
NotifyStateChange 2.16.4.11.8
NotifyStatusChange 2.15.5.7.7
NotImpDialog 2.19.3.4.3
objects.c 2.21.7.26
objects.h 2.21.7.3
object_include 2.21.7.22.2
object_intersected 2.21.7.26.10
obj_type.h 2.22.4.4
OnCallFetch 2.18.2.2.6
open.c 2.20.1.5
OpenFireMission 2.17.1.2.3
OpenFPFMission 2.17.1.4.4
OpenHostSocket 2.21.1.7.1
OpenMission 2.17.8.5.3
OpenNetworkInterface 2.21.1.22.3
OpenSocket 2.21.2.10.3
open_147 2.20.2.10.5
open_cmc 2.20.2.10.4
OptElement1Fetch 2.13.8.11.10
OptElement2Event 2.13.8.11.12
OptElement2Fetch 2.13.8.11.11
options.h 2.22.5.5
OrgTrainsBranch 2.13.3.1.5
origin.c 2.20.1.23
OtherEventLoop 2.19.3.8.1

outline.c	2.22.1.27
OutlineItem	2.22.1.27.2
OutLineRect	2.19.3.10.5
param.c	2.1.1.3
params.c	2.20.1.22
parser.c	2.9.1.3
parser_init	2.9.1.3.1
parser_restore_term	2.9.1.3.2
Password	2.1.1.3.5
PasswordEvent	2.13.8.16.2
PasswordFetch	2.13.8.16.1
PastEvent	2.18.2.2.13
PastFetch	2.18.2.2.12
PDMDelete	2.1.2.2.3
PDMInsert	2.1.2.2.2
PDU_Dispatch_NOP	2.1.1.2.5
PercentPt	2.22.1.23.3
perf.c	2.9.4.1
Performance Monitor Command	2.9.4
PerformRepair	2.5.3.1.2
PermitOptionalElements	2.13.8.11.9
pktq.h	2.20.2.23
Place Console Definitions	2.14.1
Place Console Software	2.14.2
Place Pictures	2.14.3.1
place.c	2.13.2.2
place.c	2.14.2.4
Place.h	2.14.1.1
PlaceBombs	2.1.2.1.1
PlaceCCV	2.1.2.6.1
PlaceComplete	2.13.2.2.5
PlaceComplete	2.14.2.4.5
PlaceHalf	2.6.2.1.2
PlaceIndirectFire	2.1.2.4.4
PlaceMac.h	2.14.1.2
Placement of Combat Vehicle Simulators	2.3.2
placement.c	2.1.2.5
PlaceParallelSheaf	2.6.3.1.7
PlaceSelectEvent	2.13.2.2.3
PlaceSelectEvent	2.14.2.4.3
PlaceSelectFetch	2.13.2.2.2
PlaceSelectFetch	2.14.2.4.2
PlaceTableEvent	2.13.2.2.7

PlaceTableEvent 2.14.2.4.7
PlaceTableFetch 2.13.2.2.4
PlaceTableFetch 2.14.2.4.4
PlaceTableSelect 2.13.2.2.6
PlaceTableSelect 2.14.2.4.6
Placing Combat Vehicles 2.2.2
Placing Static Vehicles 2.2.3
PointToMapCoordinates 2.22.1.26.2
point_in_polygon 2.1.8.1.6
polygon_include 2.21.7.22.1
poly_area 2.19.4.1.1
poly_area.c 2.19.4.1
poly_area.h 2.19.4.2
poly_area_loc 2.19.4.1.2
popden.c 2.1.2.2
position.c 2.6.2.1
Positioning of Howitzers and Mortars 2.6.2
Positioning Vehicles 2.1.3
PrintWindow 2.19.3.6.1
print_c_hdr 2.21.7.15.15
print_c_poly 2.21.7.15.16
print_grid_locator 2.21.7.15.11
print_int_code 2.19.4.3.5
print_statistics 2.21.7.15.18
print_terrain_map 2.21.7.15.13
print_trl_hdr 2.21.7.15.9
Process Monitor Command 2.9.5
process.c 2.1.1.4
process.c 2.9.5.1
ProcessAddlDescriptors 2.20.1.22.7
ProcessALOCDispatchRequest 2.2.3.1.4
ProcessALOCRequest 2.2.3.1.3
ProcessArrivalRequest 2.8.1.1.9
ProcessArtyArriveRequest 2.2.1.3.5
ProcessArtyDispatchRequest 2.2.1.3.4
ProcessArtyQueryRequest 2.2.1.3.2
ProcessArtyReconstRequest 2.2.1.3.3
ProcessBoundsRequest 2.8.1.1.5
ProcessBreachRequest 2.8.1.1.7
ProcessCASRequest 2.2.1.1.1
ProcessCEWInitRequest 2.2.1.6.1
ProcessCEWQueryRequest 2.2.1.6.2
ProcessCEWReconstRequest 2.2.1.6.5

ProcessCEWStartRequest 2.2.1.6.3
ProcessCommandShiftOptionSequence 2.13.8.15.4
ProcessCSSRequest 2.2.1.2.6
ProcessDataColTransaction 2.1.6.2.3
ProcessDatColDatagram 2.1.6.2.2
ProcessDepotRequest 2.2.1.2.5
ProcessDispatchRequest 2.8.1.1.8
ProcessEmplaceRequest 2.8.1.1.6
ProcessExerciseRequest 2.2.1.5.3
ProcessExitMessage 2.2.4.2.2
ProcessFSERequest 2.2.1.3.1
ProcessGridInfoRequest 2.21.1.16.1
ProcessHost 2.20.1.22.4
ProcessInitDescriptors 2.20.1.22.6
ProcessMapSheetsRequest 2.2.1.5.2
ProcessMaxSubscriptions 2.20.1.22.5
ProcessMessage 2.1.1.2.1
ProcessMessage 2.2.1.4.2
ProcessMessage 2.3.1.1.2
ProcessMessage 2.4.1.1.2
ProcessMessage 2.5.1.1.2
ProcessMessage 2.6.1.1.2
ProcessMessage 2.7.1.1.2
ProcessMessage 2.8.1.1.2
ProcessMessage 2.10.1.1.2
ProcessMgmtDatagram 2.1.6.2.4
ProcessMgmtTransaction 2.1.6.2.5
ProcessParameters 2.1.1.3.19
ProcessPermitOptionsRequest 2.2.1.4.6
ProcessProtocolFamily 2.20.1.22.8
ProcessRequest 2.13.8.6.1
ProcessRequest 2.14.2.1.1
ProcessRequest 2.15.2.1.1
ProcessRequest 2.16.1.1.1
ProcessRequest 2.17.8.3.6
ProcessRequest 2.18.2.5.3
ProcessRequest 2.19.2.1.5
ProcessResponse 2.15.2.1.2
ProcessResponse 2.16.1.1.2
ProcessSimAllocRequest 2.2.2.1.1
ProcessSimDatagram 2.1.6.2.9
ProcessSimExistsRequest 2.21.1.25.1
ProcessSimInitRequest 2.21.1.25.2

ProcessSimPlacedRequest 2.22.2.3.4
ProcessSimProblemRequest 2.13.8.6.2
ProcessSimQueryRequest 2.21.1.25.4
ProcessSimTransaction 2.1.6.2.6
ProcessSite 2.20.1.22.3
ProcessStartRequest 2.3.1.1.4
ProcessStatus 2.9.5.1.1
ProcessStatusQueryDatagram 2.1.7.1.10
ProcessStatusQueryTransaction 2.1.7.1.11
ProcessStopRequest 2.2.4.2.1
ProcessSupplyRequest 2.8.1.1.10
ProcessTargetSetRequest 2.2.3.2.1
ProcessTerrainRequest 2.2.1.5.1
ProcessTOCDispatchRequest 2.2.3.1.2
ProcessTOCRequest 2.2.3.1.1
ProcessTruckInitRequest 2.2.1.2.1
ProcessTruckQueryRequest 2.2.1.2.7
ProcessUMCPDisplaceRequest 2.2.1.2.4
proc_dgram.c 2.20.1.21
proc_req.c 2.20.1.20
proc_rsp.c 2.20.1.19
protocol.c 2.1.6.2
Pstrcpy 2.13.8.18.1
PutCompanyInSim 2.21.1.25.7
p_poly_provides_support 2.21.7.16.4
QueryFrame 2.11.1.9
QueueARead 2.11.1.3
QueueAWrite 2.11.1.4
QueueSWrite 2.11.1.5
Quit_Handler 2.9.1.1.2
radio.c 2.21.1.23
RaiseCondition 2.18.2.1.3
RandomDelay 2.6.3.1.10
RandomDistance 2.6.3.1.9
raw.c 2.20.1.14
RcvMsg 2.21.2.12.1
ReadFireTargets 2.17.8.1.12
ReadFPFTargets 2.17.8.1.15
ReadGunneryTargets 2.13.8.12.4
ReadSchedMissions 2.17.8.1.18
ReadStatus 2.17.8.1.10
ReadTerrainMap 2.17.8.1.8
ReadyTimeout 2.4.3.1.9

ReadyTimeout 2.5.3.1.3
receive.c 2.20.1.6
reconscr.c 2.13.5.2
reconscr.h 2.13.5.1
reconst.c 2.13.5.3
ReconstElementSelect 2.13.5.3.5
ReconstEvent 2.13.5.3.7
ReconstFetch 2.13.5.3.4
ReconstituteBattery 2.6.1.1.5
ReconstituteTruck 2.2.1.2.3
ReconstituteTruck 2.4.2.1.2
ReconstituteTruck 2.5.2.1.2
Reconstitution of Simulated Vehicles 2.13.5
ReconstVehicleSelect 2.13.5.3.6
Recording Statistics 2.1.5
recover.c 2.16.2.1
RecoverEvent 2.16.2.1.3
RecoverFetch 2.16.2.1.2
rectangle_intersected 2.21.7.26.9
recv.c 2.10.2.1
recv_147 2.20.2.14.3
recv_147_8023 2.20.2.17.7
recv_cmc 2.20.2.14.2
recv_cmc_8023 2.20.2.17.6
RedrawBatteryIcon 2.17.2.2.6
RedrawClock 2.22.1.6.3
redrawhelp 2.22.1.19.9
RedrawMissionIcon 2.17.2.2.5
RedrawUnitLocation 2.17.3.1.9
Refresh.c 2.19.3.6
RefreshWindow 2.19.3.8.3
Remap_Ammo 2.1.6.2.7
Remap_Fuel 2.1.6.2.8
RemoteArrive 2.6.2.1.6
RemoteDispatch 2.6.2.1.5
RemoteUnitArrived 2.17.3.2.4
RemoteUnitDispatched 2.17.3.2.3
remove.c 2.22.1.28
RemoveAmmoEntry 2.22.3.5.4
RemoveCCV 2.1.2.6.3
RemoveClient 2.4.3.1.7
RemoveLocks 2.21.2.6.5
RemoveMsgQueue 2.21.2.8.2

RemoveRadios	2.21.1.23.2
RemoveScrollTableEntry	2.22.1.28.1
RemoveScrollTableSelected	2.22.1.37.6
RemoveSharedMem	2.21.2.7.2
RemoveTarget	2.13.6.2.2
repair.h	2.22.4.3
RepairCancelEvent	2.16.4.9.3
RepairCancelFetch	2.16.4.9.2
RepairCompleted	2.16.4.11.4
RepairReplyReceived	2.1.6.1.1
RepairReplyTimedOut	2.1.6.1.2
repairtable.c	2.16.4.8
RepairTableDrawDescription	2.16.4.8.9
RepairTableDrawDisabled	2.16.4.8.8
RepairTableDrawETC	2.16.4.8.10
RepairTableDrawLocation	2.16.4.8.7
RepairTableDrawState	2.16.4.8.11
RepairTableDrawVehicle	2.16.4.8.6
RepairTableEvent	2.16.4.8.4
RepairTableFetch	2.16.4.8.3
RepairTableSelect	2.16.4.8.5
repair_m1.h	2.22.4.6
repair_m2.h	2.22.4.6
Repair_Request	2.1.6.1.3
ReplaceMissionField	2.19.3.9.3
replacevars	2.22.1.19.21
ReportDispatch	2.15.2.1.3
ReportDispatch	2.16.1.1.3
ReportHalt	2.15.2.1.4
ReportHalt	2.16.1.1.4
ReportIPCStatus	2.9.3.1.1
ReportMCCSimulationStatus	2.1.7.1.1
ReportPerformance	2.9.4.1.1
ReportRepair	2.16.1.1.5
ReportSimProblem	2.2.2.1.5
ReportYUMMStatus	2.21.2.10.12
RequestArrived	2.2.1.4.3
RequestArrived	2.3.1.1.3
RequestArrived	2.4.1.1.4
RequestArrived	2.5.1.1.4
RequestArrived	2.6.1.1.3
RequestArrived	2.7.1.1.3
RequestArrived	2.8.1.1.4

ResCleanUp 2.19.3.8.8
ResetClock 2.2.1.4.4
ResetClock 2.3.1.1.5
ResetClock 2.4.1.1.3
ResetClock 2.5.1.1.3
ResetClock 2.6.1.1.7
ResetClock 2.7.1.1.7
ResetClock 2.8.1.1.3
ResetLocks 2.21.2.6.2
ResetMCC 2.1.5.1.1
ResetTargets 2.13.6.2.4
ResItem.c 2.19.3.7
resource.h 2.13.7.2
resource.h 2.14.1.5
resource.h 2.16.4.3
resource.h 2.17.6.2
resource.h 2.18.1.5
resource.h 2.19.1.5
resource.h 2.22.2.12
ResourceItems 2.19.3.7.1
ResourceObj.c 2.19.5.9
ResourceObj.h 2.19.5.10
ResourceObj::DistanceToSecs 2.19.5.9.9
ResourceObj::GetCCVNumber 2.19.5.9.3
ResourceObj::GetKind 2.19.5.9.2
ResourceObj::GetKph 2.19.5.9.8
ResourceObj::GetMission 2.19.5.9.5
ResourceObj::GetName 2.19.5.9.10
ResourceObj::GetPosition 2.19.5.9.7
ResourceObj::GetStatus 2.19.5.9.12
ResourceObj::init 2.19.5.9.1
ResourceObj::SetMission 2.19.5.9.6
ResourceObj::SetPosition 2.19.5.9.4
ResourceObj::SetStatus 2.19.5.9.11
ResourceObj::show 2.19.5.9.13
Respawn 2.2.4.1.2
respondent.c 2.20.1.18
ResStatusWindow.c 2.19.3.8
restart.c 2.2.4.1
RestartConsole 2.2.4.1.1
RestoreTruck 2.4.2.1.6
RestoreTruck 2.5.2.1.6
resupply.h 2.22.5.9

ResupplyOffer 2.1.1.2.4
reverse_param 2.19.4.3.4
reverse_param 2.21.1.24.4
ReviewDialogHandler 2.19.3.2.3
ReviewSelection 2.19.3.2.15
rotate_queue 2.21.7.12.3
SAF_NET_SND_KLUDGE 2.21.1.22.1
same_unit 2.1.7.1.2
SaveConfig 2.17.8.1.5
SaveFireTargets 2.17.8.1.11
SaveFPFTargets 2.17.8.1.14
SaveGunneryTargets 2.13.8.12.2
SavePresets 2.13.8.12.7
SavePresets 2.17.8.1.3
SaveSchedMissions 2.17.8.1.17
SaveStatus 2.17.8.1.9
SaveTerrainMap 2.17.8.1.7
SaveVehicleStatus 2.1.6.2.1
ScanString 2.1.1.3.20
SCC #includes.c 2.13.8.3
SCC Pictures 2.13.7.1
SCC.h 2.13.8.1
SCCMac.h 2.13.8.2
SCCoptions.h 2.13.8.4
SchedEntryEvent 2.17.4.3.12
SchedTableDrawDescription 2.17.4.3.7
SchedTableDrawRounds 2.17.4.3.9
SchedTableDrawStatus 2.17.4.3.10
SchedTableDrawTarget 2.17.4.3.6
SchedTableDrawTime 2.17.4.3.5
SchedTableDrawUnits 2.17.4.3.8
SchedTableSelect 2.17.4.3.4
schedule.c 2.17.4.3
schedule.c 2.18.2.7
ScheduleDisableEvent 2.15.5.7.6
ScheduleDisableEvent 2.16.4.11.7
ScheduleDrawDescription 2.18.2.7.8
ScheduleDrawLocation 2.18.2.7.7
ScheduleDrawResults 2.18.2.7.10
ScheduleDrawSorties 2.18.2.7.9
ScheduleDrawStatus 2.18.2.7.11
ScheduleDrawTOT 2.18.2.7.6
ScheduleDrawType 2.18.2.7.5

ScheduleEvent	2.18.2.7.12
ScheduleHilite	2.18.2.7.4
ScheduleSelect	2.18.2.7.3
scroll.c	2.22.1.30
scroll.h	2.22.1.29
scrolldraw.c	2.22.1.31
scrollevt.c	2.22.1.32
scrollmap.c	2.22.1.33
scrollrow.c	2.22.1.34
ScrollTableActivate	2.22.1.32.1
ScrollTableBoxScroll	2.22.1.32.7
ScrollTableDown	2.22.1.32.5
ScrollTableEntryToRow	2.22.1.34.2
ScrollTableEvent	2.22.1.32.2
ScrollTablePage	2.22.1.32.6
ScrollTableRowRect	2.22.1.34.3
ScrollTableRowToEntry	2.22.1.34.1
ScrollTableScrollBarEvent	2.22.1.32.3
ScrollTableUp	2.22.1.32.4
scrollto	2.22.1.19.11
scrollto.c	2.22.1.35
ScrollToShow	2.22.1.35.1
scrollzoom.c	2.22.1.36
segintr.c	2.21.1.24
seg_intrsct	2.19.4.3.1
seg_intrsct	2.21.1.24.1
seg_intrsct.c	2.19.4.3
seg_intrsct.h	2.19.4.4
select.c	2.22.1.37
SelectAmmoEntry	2.22.3.5.5
SelectScrollTableEntry	2.22.1.37.1
SelectTerrainEvent	2.13.8.11.6
SelectTerrainFetch	2.13.8.11.5
SelectTerrainUpdate	2.13.8.11.4
SelectTruckTableRow	2.15.5.4.16
send.c	2.10.1.1
send.c	2.20.1.2
SendAnswer	2.11.1.10
SendConsoleResponse	2.21.1.7.6
SendData	2.21.2.12.3
SendDepotRequest	2.4.1.1.6
SendDepotRequest	2.5.1.1.6
SendErrorReport	2.21.1.11.2

SendFireRequest 2.17.5.1.3
SendForceRequest 2.4.1.1.5
SendForceRequest 2.5.1.1.5
SendFPFRequest 2.17.5.1.4
SendIndirectFirePDU 2.1.2.4.5
SendMsg 2.21.2.10.6
SendPacket 2.21.1.4.3
SendReq 2.21.2.10.7
SendRsp 2.21.2.10.8
send_147 2.20.2.15.3
send_147_8023 2.20.2.19.4
send_cmc 2.20.2.15.2
send_cmc_8023 2.20.2.19.3
sequence.c 2.22.1.39
sequence.h 2.22.1.38
service.c 2.4.3.1
service.c 2.5.3.1
ServiceObj.c 2.19.5.11
ServiceObj.h 2.19.5.12
ServiceObj::AddService 2.19.5.11.2
ServiceObj::CheckQs 2.19.5.11.4
ServiceObj::DeleteService 2.19.5.11.3
ServiceObj::init 2.19.5.11.1
ServiceObj::ShowQ 2.19.5.11.5
ServiceObj::UpdateQs 2.19.5.11.6
ServiceRequest 2.4.3.1.2
ServiceRequest 2.5.3.1.1
ServiceTimeout 2.4.3.1.8
Servicing 2.4.3
Servicing 2.5.3
SetAlarm 2.21.2.4.2
setBreachTime 2.19.3.2.6
SetChannelDefaults 2.20.1.5.3
SetCheckBox 2.22.1.11.7
SetClock 2.22.1.6.5
setEmplacementTime 2.19.3.2.4
sethelpvar 2.22.1.19.3
SetLongPt 2.22.1.23.5
SetMenuHandler 2.22.1.46.1
SetParamText 2.19.3.4.1
SetRadioButton 2.22.1.11.6
SetSchedMission 2.17.4.3.16
SetScrollTableSelection 2.22.1.37.2

SetSortieLimits 2.18.2.3.1
SetText 2.22.1.11.8
SetUp 2.11.1.16
SetUp 2.13.8.17.1
SetUp 2.14.2.5.1
SetUp 2.15.5.2.1
SetUp 2.16.4.2.1
SetUp 2.17.8.6.1
SetUp 2.18.2.8.1
setup.c 2.13.8.17
setup.c 2.14.2.5
setup.c 2.15.5.2
setup.c 2.16.4.2
setup.c 2.17.8.6
setup.c 2.18.2.8
SetUpAlloc 2.13.2.1.1
SetUpAppleTalk 2.22.1.3.1
SetUpATPRequest 2.22.1.3.2
SetUpBattery 2.17.2.1.2
SetUpCondition 2.18.2.1.1
SetUpDialog 2.18.2.2.1
SetUpDisplacement 2.13.8.10.13
SetUpFireMission 2.17.1.2.1
SetUpFireTable 2.17.4.1.2
SetUpFPFMission 2.17.1.4.1
SetUpFPFTable 2.17.4.2.2
setuphpick 2.22.1.19.8
setuphtopic 2.22.1.19.16
SetUpMenus 2.17.8.4.1
SetUpMenus 2.19.2.2.1
SetUpOptionalElements 2.13.8.11.8
SetUpPlace 2.13.2.2.1
SetUpPlace 2.14.2.4.1
SetUpRepairTable 2.16.4.8.1
SetupResWindow 2.19.3.8.2
SetUpSchedTable 2.17.4.3.2
SetUpSchedule 2.18.2.7.1
SetUpSimulators 2.22.2.9.1
SetUpStatusWindow 2.17.2.2.1
SetUpSynchReq 2.21.2.12.8
SetUpTargets 2.13.6.4.1
SetUpTeamTable 2.16.4.5.1
SetUpTrucks 2.13.3.4.1

SetUpTruckTables 2.15.5.4.1
set_g_ethernet_two_packets 2.21.1.22.2
ShowAdjust 2.17.1.1.1
ShowBattery 2.17.2.1.3
ShowBtryReconstDialog 2.13.4.2.10
ShowCancelDialog 2.16.4.9.1
ShowCaution 2.22.1.4.1
ShowCEWReconstDialog 2.13.4.4.10
ShowDialog 2.22.1.11.1
ShowDialogSeqNode 2.22.1.39.2
ShowDispatchDialog 2.15.3.1.1
ShowDispatchDialog 2.16.4.6.1
ShowDisplace 2.17.3.1.1
ShowFFE 2.17.1.3.1
ShowFFEWithAdjust 2.17.1.3.2
ShowFireMission 2.17.1.2.4
ShowFireTable 2.17.4.1.4
ShowFireTarget 2.17.4.1.11
ShowFixTable 2.22.1.42.1
ShowFPFMission 2.17.1.4.5
ShowFPFTable 2.17.4.2.3
ShowFPFTarget 2.17.4.2.10
ShowHaltDialog 2.15.5.5.1
ShowHaltDialog 2.16.4.10.1
ShowHelp 2.17.8.3.5
showhelp 2.22.1.19.4
ShowLoadDialog 2.15.1.1.1
ShowMission 2.18.2.2.2
ShowNewFireMission 2.17.1.5.1
ShowNewFPFMission 2.17.1.4.3
ShowRecoverDialog 2.16.2.1.1
ShowRepairTable 2.16.4.8.2
ShowSchedMission 2.17.4.3.11
ShowSchedTable 2.17.4.3.3
ShowSchedule 2.18.2.7.2
ShowScrollTable 2.22.1.30.1
ShowSimpleDialog 2.22.1.40.1
ShowStartDialog 2.16.4.7.1
ShowStatusWindow 2.17.2.2.2
ShowSummary 2.18.2.3.3
ShowTarget 2.13.6.3.1
ShowTargetTable 2.13.6.4.2
ShowTeamTable 2.16.4.5.2

ShowTransferAmmoDialog 2.22.3.4.1
ShowTruckDialog 2.13.3.3.1
ShowTruckTables 2.15.5.4.2
ShowVehicleDialog 2.22.2.9.5
ShowVersions 2.22.1.44.1
ShowWait 2.22.1.45.1
shutdown.c 2.1.5.1
ShutdownChildren 2.2.4.2.4
SignalChild 2.21.1.26.4
Sim Pictures 2.22.2.11
sim.c 2.1.4.2
sim.c 2.2.2.1
sim.c 2.3.2.1
sim.c 2.21.1.25
sim.c 2.22.2.9
SimAllocedRemotely 2.2.2.1.3
SimAllocedRemotely 2.3.2.1.1
SimDrawBumper 2.22.2.10.4
SimDrawCompany 2.22.2.10.5
SimDrawLocation 2.22.2.10.6
SimDrawPlaced 2.22.2.10.7
SimDrawSide 2.22.2.10.8
SimDrawSimulator 2.22.2.10.9
SimDrawType 2.22.2.10.10
SimGetSimType 2.21.1.25.3
SimListSaves 2.9.7.1.1
SimnetTypes.h 2.19.1.4
SimPlacedLocally 2.2.2.1.2
SimPlacedLocally 2.3.2.1.2
SimPlacedRemotely 2.2.2.1.4
SimPlacedRemotely 2.3.2.1.3
simple.c 2.22.1.40
SimpleCautionEvent 2.21.1.4.2
SimpleDialogEvent 2.22.1.40.2
Simple_Windows.c 2.19.5.13
Simple_Windows.h 2.19.5.14
SimReconstComplete 2.13.5.3.1
SimRestoreAll 2.9.7.1.3
SimSaveAll 2.9.7.1.2
SimStatusReport 2.9.6.1.1
SimTableEntry 2.22.2.10.3
SimTableSetup 2.22.2.10.1
SimTableUpdate 2.22.2.10.2

Simulating Command Post Vehicles 2.13.1
Simulating Gunnery Targets 2.13.6
Simulation of Close Air and Fire Support 2.13.4
Simulation of Combat Service Support 2.13.3
Simulator Save/Restore 2.9.7
SimulatorAlloced 2.14.2.4.8
SimulatorPlaced 2.13.2.2.8
SimulatorPlaced 2.14.2.4.9
SimulatorTypeCStr 2.22.2.9.2
SimulatorTypePStr 2.22.2.9.3
sim_xact.h 2.22.5.7
Smoke 2.1.1.3.6
soil.c 2.1.3.1
SoilOkay 2.1.3.1.1
SortiesScheduled 2.18.2.3.5
spawn.c 2.21.1.26
SpawnProcess 2.21.1.26.1
start.c 2.16.4.7
StartConfirmEvent 2.13.8.11.15
StartDialogSeq 2.22.1.39.1
StartDrawClass 2.16.4.7.7
StartDrawDescription 2.16.4.7.6
StartDrawDuration 2.16.4.7.8
StartEvent 2.16.4.7.3
StartFetch 2.16.4.7.2
StartHilite 2.16.4.7.5
StartMissionBattery 2.17.5.1.6
StartOfFrame 2.11.1.11
StartSchedMission 2.17.4.3.15
StartSelect 2.16.4.7.4
Status of Artillery Batteries 2.17.2
Status Report Command 2.9.6
status.c 2.1.7.1
status.c 2.9.6.1
status.c 2.17.2.2
StatusButton 2.19.3.5.4
StatusEventHandler 2.17.2.2.4
StatusHashSize 2.1.1.3.12
StatusSupport.c 2.19.3.9
stccpy 2.19.3.9.6
stop.c 2.2.4.2
StopEvent 2.13.8.16.5
stpcpy 2.19.3.10.3

string.c	2.13.8.18
StringToDTG	2.22.1.15.2
StringToMapCoordinates	2.22.1.26.1
strtok	2.20.1.17.1
strtok.c	2.20.1.17
StuffHeldMission	2.18.2.2.9
StuffTimeOnTarget	2.18.2.2.4
stxcpy	2.19.3.10.2
stxcpy	2.19.5.15.13
subscribe.c	2.20.1.1
summary.c	2.22.3.7
supply.c	2.22.3.9
supply_version.h	2.22.3.8
Swindow::BringToFront	2.19.5.13.9
Swindow::close	2.19.5.13.7
Swindow::equal	2.19.5.13.8
Swindow::GetWindowPtr	2.19.5.13.10
Swindow::hide	2.19.5.13.5
Swindow::init	2.19.5.13.1
Swindow::IsVisible	2.19.5.13.6
Swindow::quit	2.19.5.13.3
Swindow::show	2.19.5.13.2
Swindow::update	2.19.5.13.4
SystemFailure	2.22.1.8.1
table.c	2.15.5.4
table.c	2.22.1.42
table.c	2.22.2.10
table.h	2.22.1.41
TableDrawNumber	2.17.4.1.6
target.c	2.2.3.2
target.c	2.13.6.2
target.c	2.17.1.6
target.h	2.13.6.1
TargetDrawAppearance	2.13.6.4.7
TargetDrawLocation	2.13.6.4.8
TargetDrawName	2.13.6.4.5
TargetDrawType	2.13.6.4.6
TargetEntryEvent	2.13.6.3.3
TargetEntryFetch	2.13.6.3.2
TargetHit	2.13.6.2.3
TargetKilled	2.2.3.2.2
TargetTableEvent	2.13.6.4.9
TargetTableHilite	2.13.6.4.4

TargetTableSelect 2.13.6.4.3
tdb.h 2.21.7.4
tdb_cache_disable 2.21.7.8.2
tdb_cache_enable 2.21.7.8.1
tdb_close_object 2.21.7.26.6
tdb_close_thing 2.21.7.28.1
tdb_close_tree 2.21.7.30.6
tdb_close_trline 2.21.7.29.6
tdb_consistent 2.21.7.13.1
tdb_dump_terrain 2.21.7.15.3
tdb_error 2.21.7.17.1
tdb_get_db_format 2.21.7.31.5
tdb_get_db_name 2.21.7.15.19
tdb_get_db_version 2.21.7.15.20
tdb_get_dumpfile 2.21.7.15.2
tdb_get_grid_number 2.21.7.19.1
tdb_get_hull_to_world 2.21.7.20.1
tdb_get_stripe 2.21.7.11.2
tdb_get_tdb_info 2.21.7.27.3
tdb_get_terrain 2.21.7.18.1
tdb_get_z 2.21.7.16.2
tdb_giv_utm_get_xy 2.21.7.24.1
tdb_giv_xy_get_utm 2.21.7.24.3
tdb_init.c 2.21.7.27
tdb_init_cache 2.21.7.27.1
tdb_init_memory 2.21.7.25.1
tdb_local.h 2.21.7.5
tdb_lock_patch 2.21.7.23.1
tdb_map_utm_to_xy 2.21.7.24.2
tdb_map_xy_to_utm 2.21.7.24.4
tdb_nth_object 2.21.7.26.4
tdb_nth_tree 2.21.7.30.4
tdb_nth_trline 2.21.7.29.4
tdb_object_count 2.21.7.26.2
tdb_obstr_object 2.21.7.26.8
tdb_place_vehicle 2.21.7.20.5
tdb_print_cache_status 2.21.7.15.17
tdb_print_canopy 2.21.7.15.14
tdb_print_db_format 2.21.7.31.3
tdb_print_db_info 2.21.7.15.12
tdb_print_edge 2.21.7.15.6
tdb_print_format_compatible 2.21.7.31.2
tdb_print_object 2.21.7.15.7

tdb_print_polygon	2.21.7.15.5
tdb_print_tree	2.21.7.15.10
tdb_print_trline	2.21.7.15.8
tdb_print_version	2.21.7.31.1
tdb_p_cache_enabled	2.21.7.8.3
tdb_p_on_database	2.21.7.17.2
tdb_read_header	2.21.7.21.1
tdb_right_format	2.21.7.31.4
tdb_set_dumpfile	2.21.7.15.1
tdb_shade_get_z	2.21.7.16.1
tdb_shade_place_vehicle	2.21.7.20.4
tdb_terminate	2.21.7.27.2
tdb_thing_string	2.21.7.28.2
tdb_tree_count	2.21.7.30.2
tdb_trline_count	2.21.7.29.2
tdb_unlock_patch	2.21.7.23.2
TeamDispatchEvent	2.16.4.6.3
TeamDispatchFetch	2.16.4.6.2
TeamDrawAssignment	2.16.4.5.7
TeamDrawETA	2.16.4.5.10
TeamDrawLocation	2.16.4.5.9
TeamDrawNumber	2.16.4.5.6
TeamDrawStatus	2.16.4.5.8
TeamHaltEvent	2.16.4.10.3
TeamHaltFetch	2.16.4.10.2
teamtable.c	2.16.4.5
TeamTableEvent	2.16.4.5.4
TeamTableFetch	2.16.4.5.3
TeamTableHilite	2.16.4.5.11
TeamTableSelect	2.16.4.5.5
Terminal_Handler	2.9.1.1.1
Terminology and Documentation	1.5
Terrain	2.1.1.3.7
terrain.c	2.2.1.5
terrain.h	2.21.7.6
terrain_cache_inquire	2.21.7.11.1
terrain_memory_inquire	2.21.7.25.4
TextList.c	2.19.5.15
TextList.h	2.19.5.16
TextList::AddRow	2.19.5.15.6
TextList::DeleteRow	2.19.5.15.7
TextList::GetCellRect	2.19.5.15.4
TextList::GetCount	2.19.5.15.5

TextList::GetRow	2.19.5.15.10
TextList::init	2.19.5.15.1
TextList::List1Click	2.19.5.15.12
TextList::ListDispose	2.19.5.15.3
TextList::ListProc	2.19.5.15.11
TextList::ListUpdate	2.19.5.15.2
TextList::MarkRow	2.19.5.15.8
TextList::SetRow	2.19.5.15.9
tgtentry.c	2.13.6.3
tgttable.c	2.13.6.4
The Admin Console	2.15
The Admin Process	2.4
The Bridge Console	2.11
The CAS (Close Air Support) Console	2.18
The CAS (Close Air Support) Process	2.7
The CEW (Combat Engineering Workstation) Console	2.19
The CEW (Combat Engineering Workstation) Process	2.8
The FSE (Fire Support Element) Console	2.17
The FSE (Fire Support Element) Process	2.6
The Macintosh Libraries and Headers	2.22
The Maint (Maintenance) Console	2.16
The Maint (Maintenance) Process	2.5
The MCC Libraries	2.21
The Mother Process	2.1
The Network Communication Libraries	2.2
The Place (Placement) Console	2.14
The Place (Placement) Process	2.3
The SCC (SIMNET Control Console) Process	2.2
The SCC Console	2.13
The Terminal Process	2.9
things.c	2.21.7.28
ThrowDialog	2.22.1.11.3
ThrowDisplace	2.17.3.1.2
ThrowWait	2.22.1.45.2
tick.c	2.20.1.8
time.c	2.21.1.27
TimeOfFlight	2.6.3.1.6
TimeOfFlight	2.17.5.1.11
TimeString	2.1.5.1.5
time_list.c	2.20.1.16
title.c	2.22.1.43
toc.c	2.13.1.2
TOCEvent	2.13.1.2.5

TOCInitFetch	2.13.1.2.3
TOCReconstFetch	2.13.1.2.4
ToggleBattery	2.17.8.5.4
ToggleConsoleLock	2.13.8.14.1
Top-level Includes	2.22.4
total.c	2.21.1.28
TotalAmmoCapacity	2.22.3.9.1
TotalFuelFAAD	2.21.1.12.3
TotalFuelGeneric	2.21.1.15.1
TotalFuelM1	2.21.1.20.3
TotalFuelM2	2.21.1.21.3
TotalVehicleFuel	2.21.1.28.1
TraceHandler	2.21.1.19.2
TraceMessage	2.21.1.19.3
tracks_calc_unit_normal	2.21.7.20.3
tracks_set_support_plane	2.21.7.20.2
Transact	2.21.2.10.9
transact.c	2.20.1.4
transfer.c	2.22.3.4
TransferAmmoBetweenLists	2.22.3.5.11
TransferAmmoEvent	2.22.3.4.3
TransferAmmoFetch	2.22.3.4.2
treelines.c	2.21.7.29
trees.c	2.21.7.30
Truck Dispatching	2.4.2
Truck Dispatching	2.5.2
Truck Load and Unload at Supply Depots	2.15.1
truck.c	2.13.3.6
truck.c	2.21.1.29
truck.h	2.13.3.5
TruckArrived	2.4.2.1.3
TruckArrived	2.5.2.1.3
TruckDefault	2.13.3.4.3
TruckDefaultAlignment	2.13.3.4.4
TruckDefaultAll	2.13.3.4.2
TruckDispatched	2.4.2.1.4
TruckDispatched	2.5.2.1.4
TruckDispatchEvent	2.15.3.1.3
TruckDispatchFetch	2.15.3.1.2
TruckDrawAmmoLoad	2.13.3.4.16
TruckDrawAmmoLoad	2.15.5.4.8
TruckDrawAssignment	2.13.3.4.13
TruckDrawAssignment	2.15.5.4.7

TruckDrawETA 2.15.5.4.12
TruckDrawFuelLoad 2.13.3.4.15
TruckDrawFuelLoad 2.15.5.4.9
TruckDrawLocation 2.13.3.4.14
TruckDrawLocation 2.15.5.4.11
TruckDrawNumber 2.13.3.4.12
TruckDrawNumber 2.15.5.4.6
TruckDrawStatus 2.15.5.4.10
truckentry.c 2.13.3.3
TruckEntryEvent 2.13.3.3.4
TruckEntryFetch 2.13.3.3.3
TruckHaltEvent 2.15.5.5.3
TruckHaltFetch 2.15.5.5.2
TruckKilled 2.4.2.1.5
TruckKilled 2.5.2.1.5
TruckReconstComplete 2.13.5.3.3
TruckReconstStart 2.13.5.3.2
trucktable.c 2.13.3.4
TruckTableComplete 2.13.3.4.9
TruckTableEvent 2.15.5.4.4
TruckTableFetch 2.15.5.4.3
TruckTableHilite 2.13.3.4.11
TruckTableHilite 2.15.5.4.13
TruckTableSelect 2.13.3.4.10
TruckTableSelect 2.15.5.4.5
trytoscroll 2.22.1.19.12
TubeKilled 2.6.1.1.6
TurnFleetIntoCapabilities 2.4.2.1.7
TurnFleetIntoCapabilities 2.5.2.1.7
TurnRadioOff 2.21.1.23.4
TurnRadioOn 2.21.1.23.3
UndoDisplace 2.17.3.1.6
UndoFireTarget 2.17.4.1.14
UndoFPFTarget 2.17.4.2.13
UndoSchedMission 2.17.4.3.14
unget_locks 2.20.2.21.6
UnhookFireTarget 2.17.4.1.15
units.c 2.17.5.2
Unlock 2.21.2.6.4
unmap_buffers 2.20.2.21.5
unmap_enp 2.20.2.21.4
UpdateAmmoDisplay 2.17.2.1.8
UpdateAmmoList 2.22.3.5.12

UpdateAmmoLoadTotals 2.13.3.3.2
UpdateAppearance 2.1.4.2.1
UpdateBatteryDisplay 2.17.2.1.7
UpdateBatterySelection 2.17.5.2.2
UpdateButtons 2.15.5.4.15
UpdateCapacityDisplay 2.22.3.4.5
UpdateCEWAssetLocation 2.13.4.4.9
UpdateClock 2.22.1.6.2
UpdateDialog 2.22.1.11.2
UpdateDispatchLoad 2.15.3.1.5
UpdateDisplacement 2.13.8.10.20
UpdateFireDialog 2.17.1.2.7
UpdateFireState 2.17.1.2.8
UpdateFixTableRow 2.22.1.42.2
UpdateFPFDialog 2.17.1.4.9
UpdateHaltLoad 2.15.5.5.5
UpdateHaltLocation 2.15.5.5.4
UpdateHaltLocation 2.16.4.10.4
UpdateLoadDialog 2.15.1.1.5
UpdateMission 2.18.2.6.2
UpdateMissionDisplay 2.17.8.5.5
UpdateMissionState 2.17.8.5.6
UpdateObserver 2.17.8.5.7
UpdateOTDirection 2.17.1.2.9
UpdatePopUpLoad 2.15.5.7.8
UpdateReleasePoint 2.17.3.1.4
UpdateRepairTableRow 2.16.4.8.12
UpdateScrollTableEntry 2.22.1.31.1
UpdateSharedMemory 2.6.3.1.11
UpdateStartButton 2.16.4.7.10
UpdateStartDialog 2.16.4.7.9
UpdateTeamLocation 2.16.4.11.3
UpdateTeamTableButtons 2.16.4.5.13
UpdateTeamTableRow 2.16.4.5.12
UpdateTransactions 2.20.1.8.2
UpdateTransferAmmoLoads 2.22.3.4.4
UpdateTruckLocation 2.15.5.7.3
UpdateTruckTableRow 2.15.5.4.14
UpdateUnitLocation 2.17.3.2.6
UploadALOCParameters 2.13.1.1.2
UploadBtryParameters 2.13.4.2.8
UploadCASParameters 2.13.4.1.2
UploadCEWParameters 2.13.4.4.4

UploadCSSParameters	2.13.3.1.6
UploadExerciseParameters	2.13.8.11.14
UploadFSEParameters	2.13.4.2.5
UploadTarget	2.13.6.2.5
UploadTOCParameters	2.13.1.2.2
UploadTruckParameters	2.13.3.6.1
UploadVehicleParameters	2.22.2.3.2
upshift	2.20.1.22.1
User Interface for Controlling Indirect Fire Missions	2.17.1
Utils.c	2.19.3.10
UTMToLongPt	2.19.3.3.5
ValidBumperNumber	2.9.7.1.7
ValidBumperNumber	2.22.2.9.7
ValidFPFTarget	2.17.1.4.7
ValidFrame	2.11.1.7
ValidNewFireTarget	2.17.4.1.13
ValidNewFPFTarget	2.17.4.2.12
ValidSimNum	2.9.7.1.6
ValidTgtLocation	2.17.1.6.1
ValidTime	2.17.4.3.13
ValidTOT	2.18.2.2.8
VehFromBumper	2.9.7.1.5
VehFromSimNum	2.9.7.1.4
VehHashTableSize	2.1.1.3.11
Vehicle	2.1.1.3.9
vehicle.c	2.1.1.5
vehicle.c	2.9.7.1
VehicleIDToSim	2.1.1.5.3
VehicleIDToTrailer	2.21.1.11.4
VehicleIsGunneryTarget	2.1.2.6.2
VehicleMarkingR	2.1.1.3.13
VehicleResupplied	2.4.3.1.3
VehicleStatusReport	2.9.6.1.2
vehsubsys.c	2.21.1.30
veh_assign.h	2.22.5.10
Veh_Log	2.1.1.3.8
veh_role.h	2.22.4.5
veh_subsys_encode_air_failures	2.21.1.30.3
veh_subsys_encode_ground_failures	2.21.1.30.2
veh_type.h	2.22.4.6
version.c	2.21.7.31
version.c	2.22.1.44
version.h	2.13.8.5

version.h	2.14.1.3
version.h	2.17.7.3
version.h	2.18.1.3
version.h	2.19.1.1
version.h	2.22.3.3
Visual Appearance of SCC User Interface	2.13.7
VList	2.1.1.3.10
vList_Hash	2.21.1.18.1
vStatusList_Hash	2.21.1.18.2
vtimeout.c	2.1.7.2
VTimeout_Appearance_Timeout	2.1.7.2.1
wait.c	2.22.1.45
wait_for_empty_ring_element	2.20.2.19.1
wait_for_full_ring_element	2.20.2.17.8
who.c	2.20.1.15
widebdry.c	2.21.1.31
wide_bdry	2.21.1.31.1
window.c	2.22.1.46
WindowEvent	2.22.1.46.2
WipeFireTable	2.17.8.1.13
WipeFPFTable	2.17.8.1.16
WipeSchedTable	2.17.8.1.20
WipeTargetTable	2.13.8.12.5
WithdrawBattery	2.17.5.2.3
WithdrawOffer	2.4.3.1.5
WriteGunneryTargets	2.13.8.12.3
yumm.h	2.21.2.9
YUMMCleanUp	2.21.2.10.5
yummextern.c	2.21.2.10
YUMMInit	2.21.2.10.1
yumminit.h	2.21.2.11
yummintern.c	2.21.2.12
YUMMProcess	2.21.2.10.2
zoom.c	2.22.1.47
ZoomBatteryIcon	2.17.2.2.8
ZoomDownToTableEntry	2.13.8.8.3
ZoomDownToTableEntry	2.14.2.2.2
ZoomInit	2.22.1.47.1
ZoomMissionDown	2.18.2.2.5
ZoomMissionIcon	2.17.2.2.7
ZoomRect	2.22.1.47.2
ZoomScrollTableEntry	2.22.1.36.1
ZoomToWindow	2.22.1.47.4

ZoomUpFromTableEntry 2.13.8.8.2
ZoomUpFromTableEntry 2.14.2.2.1