ADST/WDL/TR--93-003036

ADST
Software Design Document, Volume III
AIRNET
Digital Message Communications Console
(CSCI 11)

Loral Western Development Labs
Electronic Defense Systems Software Department
Software Engineering Laboratory
3200 Zanker Road
San Jose, California 95161-9041

March 31, 1993

Contract No. N61339-91-D-0001
CDRL A009

Prepared for

Simulation Training and Instrumentation Command
Naval Training Systems Center
12350 Research Parkway
Orlando, FL 32826-3275

LORAL
Western Development Labs

REPORT DOCUMENTATION PAGE

Form approved
OMB No.

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	31 MARCH 1993	Version 1.0

4. TITLE AND SUBTITLE
ADST Software Design Document: Volume III: AIRNET Digital Message Communications Console

5. FUNDING NUMBERS
Contract No. N61339-91-D-0001

6. AUTHOR(S)
Aiken, John; Desmeules, Peter; Au-Yeung, Anna; McNerney, Tim; Stephan, Jeff

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Loral Systems Company
ADST Program Office
12443 Research Parkway, Suite 303
Orlando, FL 32826

8. PERFORMING ORGANIZATION REPORT NUMBER
ADST/WDL/TR-93-003036

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Simulation, Training and Instrumentation Command
STRICOM
Naval Training Systems Center
12350 Research Parkway
Orlando, FL 32826-3275

10. SPONSORING ORGANIZATION REPORT

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Final version for public release.

12b. DISTRIBUTION CODE
A

13. ABSTRACT (Maximum 200 words)

The Software Design Document describes the design of the AIRNET Digital Message Communications software.

14. SUBJECT TERMS

15. NUMBER OF PAGES
629

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	17. SECURITY CLASSIFICATION OF THIS PAGE	17. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

LIST of FIGURES

LIST of TABLES

1. Scope.

1.1 Identification.

Airnet Digital Message Communications Console CSCI.

1.2 System overview.

The AIRNET system is a network of simulators to help to teach Army aviation crews and battlefield support personnel to fight in a combined arms battlefield environment.

The AIRNET Digital Message Communications Console (DMCC) provides a capability for pre-formatted and free text digital messaging between simulation entities which are internal and external to the AIRNET network.

The DMCC software is a multi-threaded, X-windows client/server application suite.

The DMCC software contains a custom protocol for communications in the context of the SIMNET and Draft DIS 2.0 distributed simulation protocols. This protocol is fully interoperable across the SIMNET/DIS 2.0 Protocol Translator Gateway.

The DMCC uses a graphical user interface to emulate vehicle crew station soldier machine interfaces.

The Digital Message Communications Console allows transmission, reception, and storage of, and access to, pre-formatted and free text tactical messages between ground support, Tactical Operations Centers, Fire Support Elements, and manned vehicle simulators, via the SIMNET and DIS simulation networks.

The DMCC software is platformed on a Sun Microsystems Sparc-10 workstation running SunOS, Sun Microsystems Software's implementation of the UNIX operating system. The runtime environment includes Version 11 of the X-Windows windowing system and OSF Motif version 1.1 graphical user interface.

Up to 8 Wyse X-terminals may be connected to the workstation via the DMCC dedicated Ethernet local area network. Multiple DMCC client executables may be operational concurrently under the X client/server model.

DMCC client processes run on the Sun hardware. "Local client" operation is not used in this implementation.

The following diagram shows the top-level architecture of the Digital Message Communications Console System.

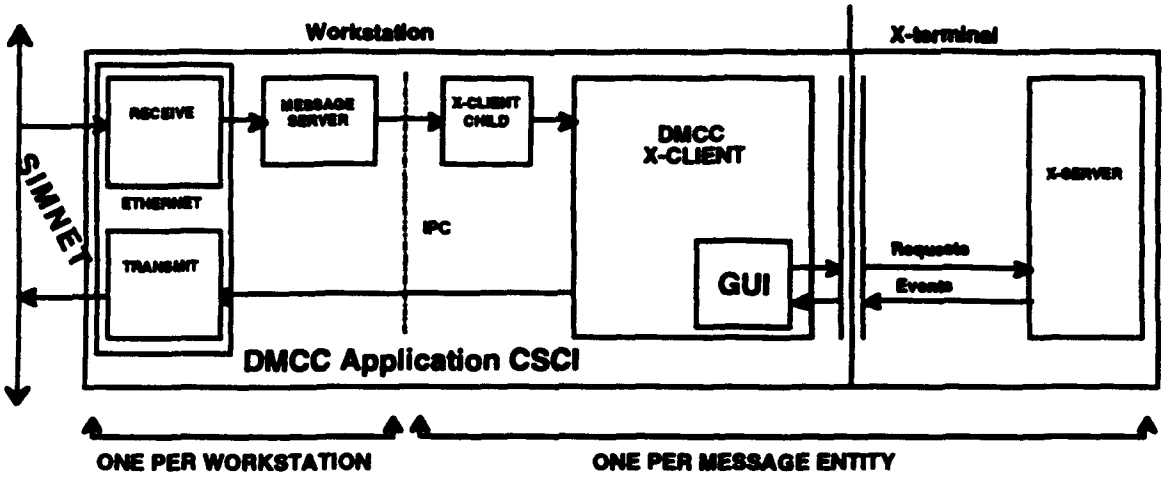


Figure 1 DMCS Software Architecture

The Receive process monitors the Ethernet for incoming messages and informs the Message Server of their arrival.

The Message Server checks the message to see if it is addressed to any of the resident DMCC X-client applications. If the message is addressed to one or more of the DMCC X-client applications, then the Message Server places the message in Shared Memory and informs the affected DMCC X-client applications of its location through a message queue which is monitored by the Client Child processes.

The Client Child processes are spawned at logon by the X-Client processes and wait for indication of incoming messages addressed to it's parent. The Client Child processes then inform the Client of the message arrival through a pipe.

Th DMCC X-Client process is instantiated for each DMCC entity active in the workstation. The DMCC X-Client process contains the message queue, user interface, and message preparation and dispatch software. The X-Client also contains the software which allows the user to set up addresses to which the user wishes to transmit messages, groups to which the user wishes to receive messages, and locations which are referred to in creating messages for transmission.

The DMCC X-Client software has windows for six different purposes, including:

(1) Logon and Setup
(2) Message Queue Access
(3) Message Access/Deletion
(4) Message Retrieval
(5) Message Forwarding
(6) Report Preparation and Transmission

Each instantiation of the DMCC maintains a list of people or teams to whom it can send messages called the **Address List**. The user sets up the list of addresses prior to the start of the exercise, in order to quickly send messages during the exercise. He may change the list during the exercise, but normally he would not want to do this.

For example, the user may want to be able to send messages during the exercise to his battalion commander, his wing man, the Fire Support Element, etc. The OPORDS for his mission will include this list. He will probably want to enter their CEOIs (call signs) in the Address List.

Groups are collections of DMCC simulation entities. For example, several entities may be members of the same team, say TEAM A. If they wish to receive messages transmitted to TEAM A, they must tell their DMCCs to receive messages addressed to that group. Each DMCC allows its operator to be a member of up to seven groups.

The following figure details the nature of the instantiated-client configuration in the context of a X-terminal client server environment:

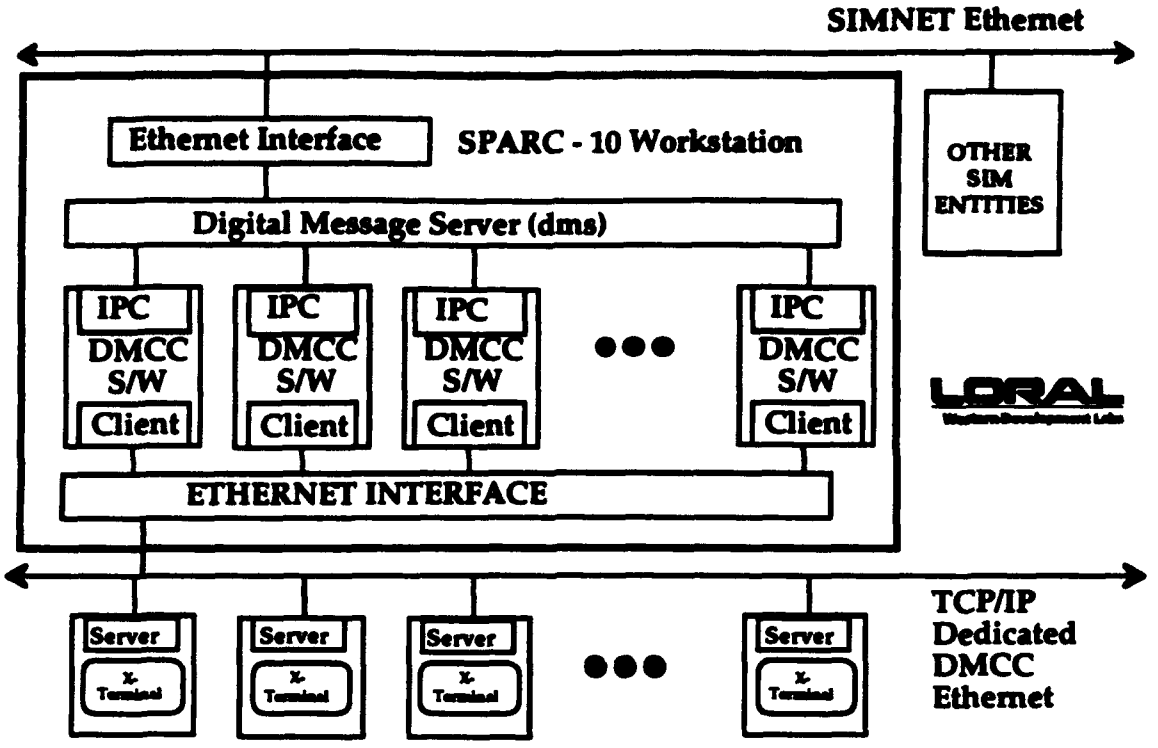


Figure 2 DMCC X-windows Client Server Environment

1.3 Document overview.

This document describes the system design of the Digital Message Communications Console CSCI. It outlines the structure and composition of the CSCI sub-functions (CSC's and CSU's) and provides a detailed description of each.

Section 2 describes the documents referenced in this specification.

Section 3 outlines the preliminary design overview of the CSCI, and a description of each CSC's purpose.

Section 4 describes the detailed design of each CSC.

Section 5 provides a detailed breakdown of all data elements defined for each CSC and for each associated CSU.

Section 6 provides a data file cross reference to aid in locating functions.

Sections 7 and 8 provide requirements and general design notes.

Section 9 contains structure charts for the overall DMCC CSCI and its component CSCs.

Section 10 details the datastructures.

Section 11 contains state transition diagrams for the various concurrent process CSCs in the DMCC CSCI.

Section 12 contains diagrams of the Protocol Data Units used by the Digital Message Communications Console to send and receive digital messages.

Section 13 covers the Functional Tests used to verify proper operation of the DMCC software.

2. Referenced and Relevant documents.

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

2.1 Government documents.

SPECIFICATIONS:

None.

STANDARDS:

DI-MISC-80711, Scientific and Technical Reports.

The SIMNET Network and Protocols, Report No. 7627, Arthur R. Pope; Prepared for DARPA by Bolt, Barenek and Newman, Inc. June 1991.

IST-CR-92-12

Military Standard: Draft DIS 2.0: Protocol Data Units for Entity Information and Entity Interaction in a Distributed Interactive Simulation, Institute for Simulation and Training, 12424 Research Parkway, Suite 300, Orlando, Florida, September 4, 1992

2.2 Non-Government documents.

Rotary Wing Aircraft AIRNET Aeromodel and Weapons Model Conversion Statement of Work, April 6, 1992.

Software Requirements & Interface Specification for the AIRNET MCC Comanche Support and Digital Message/Communications Upgrade,

System Specification for the RWA AIRNET Aeromodel and Weapons Conversion, LORAL Western Development Labs, 3200 Zanker Road, POBox 49041, San Jose, California, WDL/TR--92-003011, June 5, 1992.

The X Window System: Programming and Applications with Xt, OSF/Motif® editi[illegible], Douglas A. Young, Hewlett-Packard Laboratories, Palo Alto, Calif[illegible]ia, Prentice Hall, Englewood Cliffs, New Jersey, 1990

The Motif® Programming Manual for OSF/Motif version 1.1, volumes 1 - 6, (The Definitive Guides to the X-Window System) Dan Heller, O'Reilly & Associates, Inc., July 1992.

OSF/Motif® Programmer's Reference, Release 1.1, Open Software Foundation, Prentice Hall, Englewood Cliffs, New Jersey 1991

Builders Xcessory User's Guide, Version 2.0, Integrated Computer Solutions, Incorporated, 201 Broadway, Cambridge, Massachusetts, 1991.

Information technology--Local and metropolitan area networks--Part 3: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications, International Standard ISO/IEC 8802-3 : 1992 ANSI/IEE Std 802.3, 1992 Edition, The Institute of Electronics and Electronics Engineers, Inc.

The Ethernet, Physical and Data Link Layer Specifications, Version 2.0. Digital Equipment Corporation, Intel Corporation, and Xerox Corporation, 1982.

A Standard for the Transmission of IP Datagram over IEEE 802 Networks (Internet and Address Resolution Protocol on IEEE 802 Networks) Network Working Group, Request for comments(RFC): 1042 J. Postel and J. Reynolds, ISI, February 1988.

3. Preliminary design.

3.1 CSCI overview.

The DMCC Computer Software Configuration Item includes all the software executeables to implement the Digital Message Communications Console. It includes CSC's for SIMNET communications; message routing; user interface, message storage, access, retrieval, preparation, and transmission; and message arrival notification.

Multiple X-client and X-client child processes may be instantiated to serve multiple X-terminal simulation entities.

The design of the Digital Message Communications Console software is detailed in this document.

3.1.1 CSCI architecture.

The structure charts shown in *Appendix A* outline the detailed organizational structure of the CSCI.

The following top level structure chart outlines the top level structure of the DMCC CSCI:

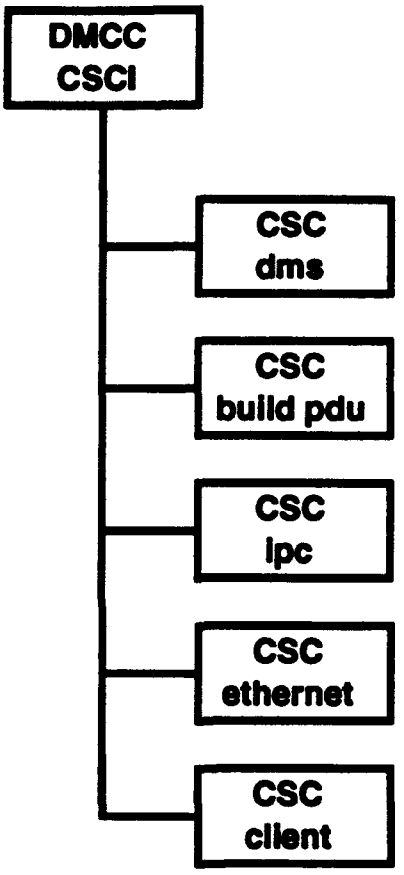


Figure 3 Top Level DMCC CSCs

The list of Computer Software Components (CSCs) and sub-level CSCs following this paragraph includes a summary of the purpose of each, the inter-CSC relationships and a CSC-to-CSC interface summary.

Table 1. CSC Summary table

CSC	Source File	Description/Interface Summary
dms	dms.c	DESCRIPTION: The dms function manages the distribution of messages to DMCC operators. It receives a login message from each operator, which contains the message CEOIs/Groups that should be sent to the operator. INPUTS: 1. DMCC messages 2. Login requests 3. Logout requests 4. CEOI/Group updates OUTPUTS: 1. Incoming shared memory segment 2. Fan-out message queue 3. Outgoing message queue 4. Digital Message arrival notifications
build pdu	bldpdu.c	DESCRIPTION: The build pdu function provides the X-window operator interface the means to construct Digital Messages and send them out to the SIMNET Ethernet process. INPUTS: 1. Digital Message data items OUTPUTS: 1. Digital Messages 2. DTG
ipc	dmslib.c	DESCRIPTION: The ipc function provides an interface for the X-window process, and the SIMNET Ethernet interface processes to send and receive DMCC messages. It initializes the X-window child process that receives new Digital Message arrival notifications from the dms process. The child process reads the new messages from shared memory and passes them to the X-window parent process via a pipe. INPUTS: 1. Digital Message from SIMNET interface process 2. Digital Message data items from X-window operator interface process 3. CEOI/Groups OUTPUTS: 1. Digital Message arrival notifications 2. Digital Messages 3. Login requests 4. Logout requests

MMI		DESCRIPTION: The MMI function allows the user to logon, create, read and send messages INPUTS: 1. DMCC messages 2. Formatted messages for reading OUTPUTS: 1. Login requests 2. Logout requests 3. CEOI/Group updates 4. DMCC messages 5. Digital Message data items
Address list screen creation	addrList.c	DESCRIPTION: Creates Address List Screen INPUTS: None OUTPUTS: None
Address list screen callbacks	addrListCB.c	DESCRIPTION: Address List Callbacks INPUTS: None OUTPUTS: None
Console screen creation	console.c	DESCRIPTION: Creates Console Screen INPUTS: None OUTPUTS: None
Console screen callbacks	consoleCB.c	DESCRIPTION: Console Screen callbacks INPUTS: None OUTPUTS: None
Form trim drawing	formTrim.c	DESCRIPTION: Crop forms INPUTS: None OUTPUTS: None

Free text report screen creation	freeTxt.c	DESCRIPTION: Creates free text msg prep screen INPUTS: None OUTPUTS: None
Free text report screen callbacks	freeTxtCB.c	DESCRIPTION: Free text report callbacks INPUTS: None OUTPUTS: None
Get DMCC messages	getMessageCB.c	DESCRIPTION: Retrieves messages from shared mem INPUTS: None OUTPUTS: None
Group list screen creation	groupList.c	DESCRIPTION: Creates group list screen INPUTS: None OUTPUTS: None
Group list screen callbacks	groupListCB.c	DESCRIPTION: Group list screen callbacks INPUTS: None OUTPUTS: None
List widget utilities	listUtilities.c	DESCRIPTION: List widget functionality INPUTS: None OUTPUTS: None
Location list screen creation	locList.c	DESCRIPTION: Creates location list screen INPUTS: None OUTPUTS: None

Location list screen callbacks	locListCB.c	DESCRIPTION: Location List Callbacks INPUTS: None OUTPUTS: None
Logon screen creation	logon.c	DESCRIPTION: Creates Logon screen INPUTS: None OUTPUTS: None
Logon screen callbacks	logonCB.c	DESCRIPTION: Logon screen callbacks INPUTS: None OUTPUTS: None
Main	main.c	DESCRIPTION: Main INPUTS: None OUTPUTS: None
Movcmd TMI screen callbacks	movcmdTmi_call.c	DESCRIPTION: MOVCMD screen callbacks INPUTS: None OUTPUTS: None
Movcmd TMI screen create	movcmdTmi_create.c	DESCRIPTION: Creates MOVCMD screen INPUTS: None OUTPUTS: None
Movcmd screen callbacks	movcmd_call.c	DESCRIPTION: MOVCMD screen callbacks INPUTS: None OUTPUTS: None

Movcmd screen create	movcmd_create.c	DESCRIPTION: Creates MOVDMD screen INPUTS: None OUTPUTS:None
Message box screen create	msg1.c	DESCRIPTION: Creates Message Box Screen INPUTS: None OUTPUTS: None
Message box screen callbacks	msg1CB.c	DESCRIPTION: Message Box Screen Callbacks INPUTS: None OUTPUTS: None
Message read screeen create	msgRead.c	DESCRIPTION: Creates READ screen INPUTS: None OUTPUTS: None
Message read screen callbacks	msgReadCB.c	DESCRIPTION: Read screen callbacks INPUTS: None OUTPUTS: None
MTO TMI screen callbacks	mtoTmi_call.c	DESCRIPTION: MTO screen callbacks INPUTS: None OUTPUTS: None
MTO TMI screen create	mtoTmi_create.c	DESCRIPTION: Creates MTO screen INPUTS: None OUTPUTS: None

MTO screen callbacks	mto_call.c	DESCRIPTION: MOT screen callbacks INPUTS: None OUTPUTS: None
MTO screen create	mto_create.c	DESCRIPTION: Creates MTO screen INPUTS: None OUTPUTS: None
Popup messages	popupMessage.c	DESCRIPTION: Executes popup messages INPUTS: None OUTPUTS: None
Reports screen create	report.c	DESCRIPTION: Creates Reports screen INPUTS: None OUTPUTS: None
Reports screen callback	reportCB.c	DESCRIPTION: Reports screen callbacks INPUTS: None OUTPUTS: None
Request TMI screen callbacks	reqtTmi_call.c	DESCRIPTION: Request screen callbac ks INPUTS: None OUTPUTS: None
Request TMI screen create	reqtTmi_create.c	DESCRIPTION: Creates Request screen TMIs INPUTS: None OUTPUTS: None

Request screen callbacks	reqt_call.c	DESCRIPTION: Request screen callbacks INPUTS: None OUTPUTS: None
Request screen create	reqt_create.c	DESCRIPTION: Creates Request screen INPUTS: None OUTPUTS: None
Shot TMI screen callbacks	shotTmi_call.c	DESCRIPTION: Shot TMI screen callbacks INPUTS: None OUTPUTS: None
Shot TMI screen create	shotTmi_create.c	DESCRIPTION: Creates Shot screen TMIs INPUTS: None OUTPUTS: None
Shot screen callbacks	shot_call.c	DESCRIPTION: Shot screen callbacks INPUTS: None OUTPUTS: None
Shot screen create	shot_create.c	DESCRIPTION: Creates Shot screen INPUTS: None OUTPUTS: None
Splash TMI screen callbacks	splashTmi_call.c	DESCRIPTION: Splash TMI screen callbacks INPUTS: None OUTPUTS: None

Splash TMI screen create	splashTmi_create.c	DESCRIPTION: Creates Splash screen INPUTS: None OUTPUTS: None
Splash screen callbacks	splash_call.c	DESCRIPTION: Splash screen callbacks INPUTS: None OUTPUTS: None
Splash screen create	splash_create.c	DESCRIPTION: Creates splash screen INPUTS: None OUTPUTS: None
Spot TMI screen callbacks	spotTmi_call.c	DESCRIPTION: Spot TMI screen callbacks INPUTS: None OUTPUTS: None
Spot TMI screen create	spotTmi_create.c	DESCRIPTION: Creates spot screen TMIs INPUTS: None OUTPUTS: None
Spot screen callbacks	spot_call.c	DESCRIPTION: Spot screen callbacks INPUTS: None OUTPUTS: None
Spot screens create	spot_create.c	DESCRIPTION: Creates Spot screen INPUTS: None OUTPUTS: None

Main screen create	sysMain.c	DESCRIPTION: Creates SYS MAIN screen INPUTS: None OUTPUTS: None
Main screen callbacks	sysMainCB.c	DESCRIPTION: Main screen callbacks INPUTS: None OUTPUTS: None
MMI utilities	util.c	DESCRIPTION: Various Man Machine Interface Utilities INPUTS: Various (See section 4) OUTPUTS: Various (See section 4)

3.1.2 System states and modes.

The DMCC software consists of five concurrent process types, to receive, route and transmit ethernet Protocol Data Units, and the X-Client and X-Client Child processes. All of these process types, with the exception of the X-Client, are modeless, that is, they wait for a single type of event and then handle that event in a single-threaded, one shot manner.

The X-Client Process, on the other hand, has many states or modes, and actions which this process takes in response to events depend on many factors. The basic modes or states which the DMCC Client Process can exhibit are closely bound to the concept of an "active" X-Window.

The X-Client Process State Event Matrix is shown in Appendix C. This diagram shows actions and transitions from various window states bound to event flags representing operator intervention, such as clicking on a button region of a window. Each cell shows what actions are taken as a result of an event followed by a semicolon (;) character, which is followed by an indication of the next window to transition to (make active), if any. The top row is a list of the window events and the left column is a list of window states.

3.1.3 Memory and processing time allocation.

Because the Digital Message Communication System is a near-real-time messaging system, there are no hard requirements for processing time, other than to respond fast enough such that seamless "realistic" simulation is maintained.. The Sparc-10 workstation platform provides processing speed (56,000,000 instructions per second) an order of magnitude in excess of that required for timely user interface activity and message handling.

The following table depicts the rough-order-of-magnitude, worst case memory requirements for the DMCC software suite:

Table 2 Memory Requirements

Process	Rough Order Of Magnitude, Worst Case Memory Requirement
System	16 MB
8 X window managers	12 MB
dms process	1 MB
8 Client Processes	12 MB
8 Client Children	1 MB
Ethernet Receive	1 MB
Ethernet Transmit	1 MB
Total	44 MB

3.2 CSCI design description

The DMCC CSCI consists of five concurrent processes to receive ethernet messages (Receive), rout them to client DMCC entities (Digital Message Server) , to transmit ethernet messages (Transmit), to handle user interface and manage user message queues (X-Client) and to inform the X-Client of message arrival (X-Client_Child). In a particular installation, only one Transmit, Receive and Digital Message Server process is active at any one time. The X-Client and X-Client Child processes are instantiated for each of the active Digital Message Communications Consoles (X-Terminals) active in the system. Thus, for an eight client system, a total of nineteen concurrent processes may be active.

3.2.1 CSC dms

DESCRIPTION: The dms function manages the distribution of messages to DMCC operators. It receives a login message from each operator, which contains the message CEOIs/Groups that should be sent to the operator.

The following diagram shows the relationship of the digital message server to the other concurrent processes in the DMCC software.

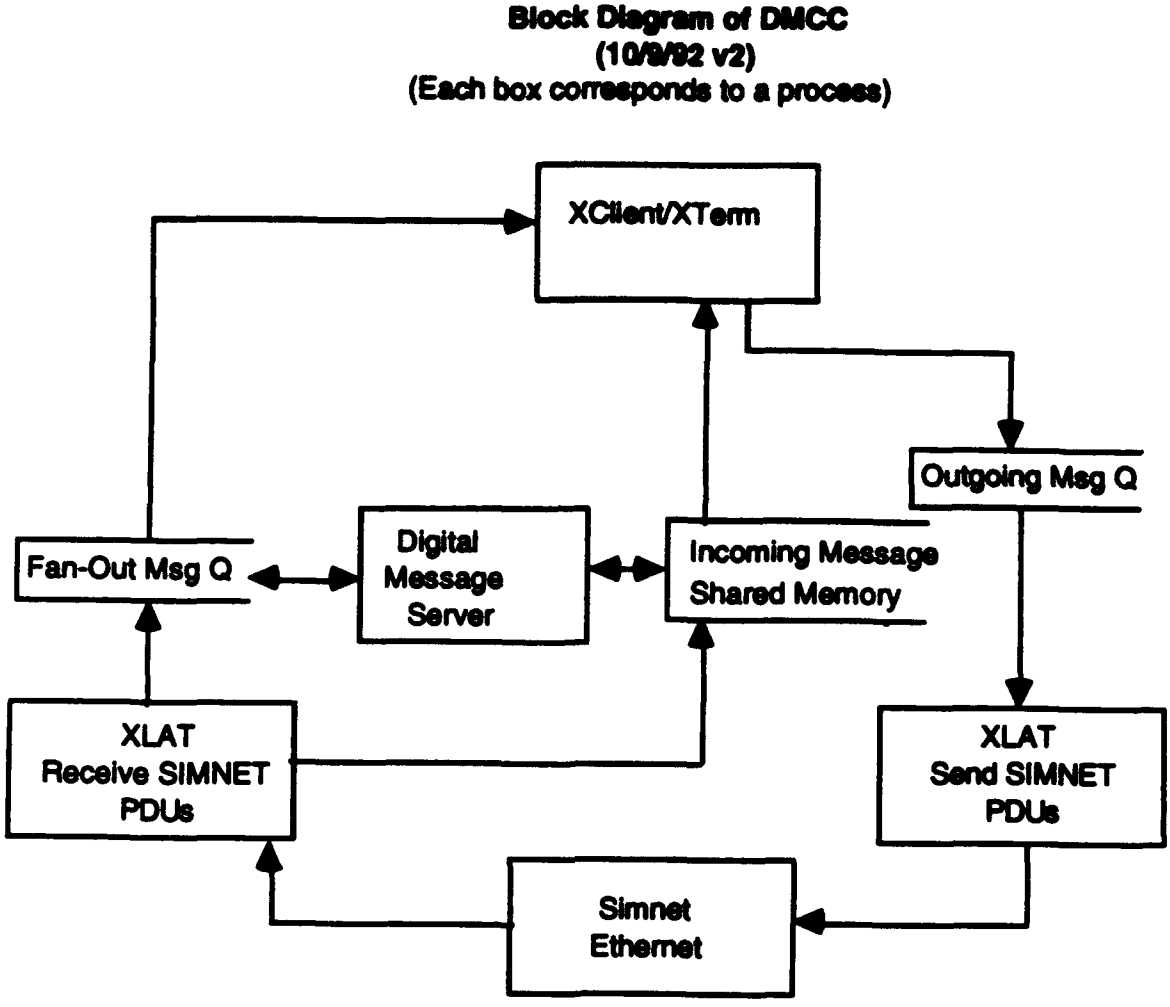


Figure 4 DMCC Concurrent Processes

Specific duties performed by the digital message server are:
1. Remove any existing copies of shared memory segment and message queues.
2. Create Incoming shared memory segment, Fan-Out message queue, and Outgoing message queue.
3. Process login and logout requests from X-Client CSC.
4. Store CEOIs/Groups received in login message or newdests message from X-Client CSC.
5. Process new DMCC PDU arrival notifications from ethernet CSC.
6. Determine which operators should be notified of the arrival of a new DMCC PDU based on a match between the Target CEOI in the message, and the CEOIs/Groups passed in the login message or newdests message, and the exercise id contained in the new DMCC PDU and the exercise that was passed from the operator in the login message.

7. Send kill signal via operating system interface to the X-client child when an operator logs-out and then remove all notifications sent to operator in Fan-Out queue, and decrement 'in use' (this is data item "nbr_readers" in the shared memory segment) flag for each message in shared memory segment.

The following State Transition Diagram details the activity sequencing for the digital message server:

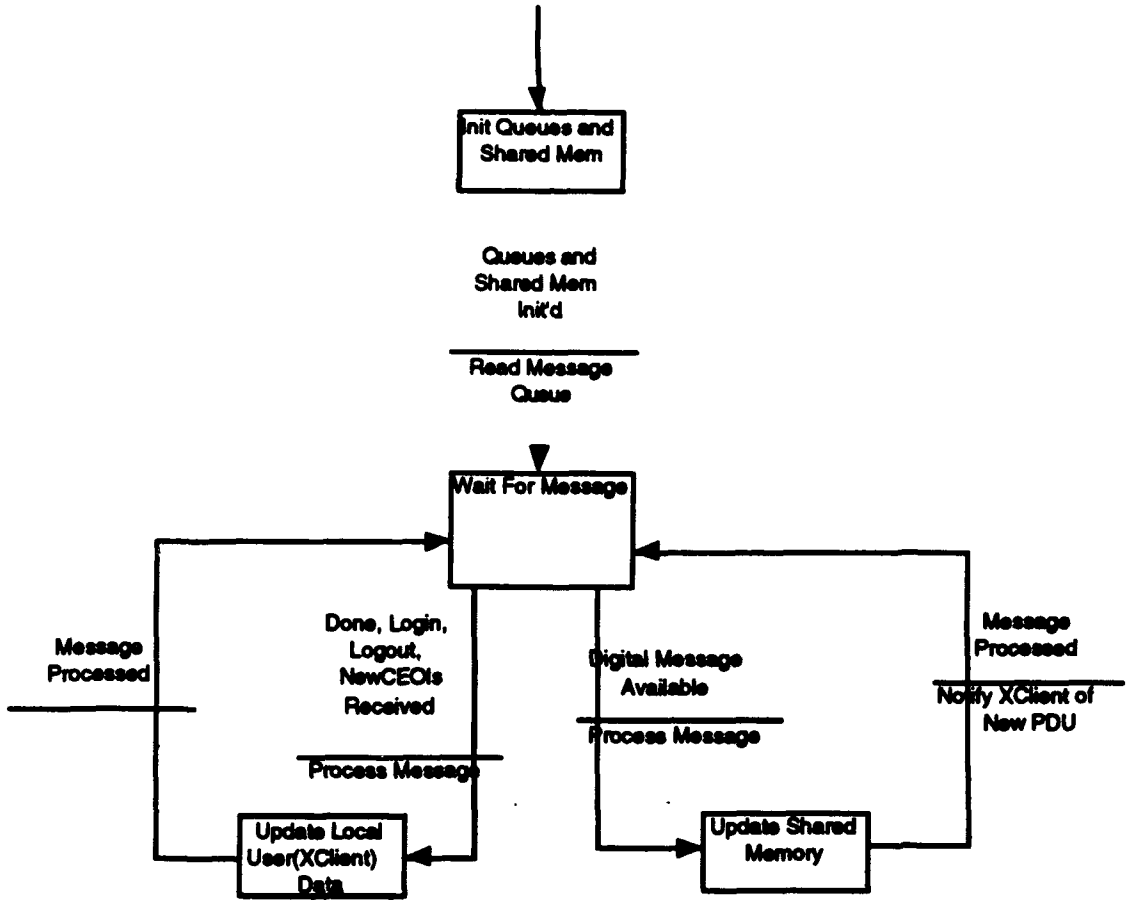


Figure 5 CSC dms State Transition Diagram

CSC INTERFACES:

1. "shmaddr" data item is initialized with address of shared memory segment. Offsets from this address contain messages and are passed to CSC client when a new message arrives.
2. Data item "msgin" with flag msg_type set to MSG_DMAVAIL is received from CSC ethernet and contains notification of the arrival of a new message, and its location in the shared memory segment.
3. Data item "msgin" with flag msg_type set to MSG_DONE is received from CSC client and indicates when a message has been read.

4. Data item "msgin" with flag msg_type set to MSG_LOGOUT is received from CSC client and indicates a logout.
5. Data item "msgin" with flag msg_type set to MSG_LOGIN is received from CSC client and indicates an operator is loging-in.
6. Data item "msgin" with flag msg_type set to MSG_NEWDESTS is received from CSC client and indicates that the operators list of CEOIs/Groups has changed.
7. Data item "msgout" is sent to CSC client when a new message is available.

3.2.2 CSC build pdu

DESCRIPTION: The build pdu function provides the X-window operator interface process the means to construct Digital Messages and send them out to the SIMNET Ethernet interface process.

Specific duties performed are:
1. Create the appropriate Digital Message.
2. Send the message to CSC ethernet.
3. Return the current Zulu DTG.
4. Forward a received message to another operator.

CSC INTERFACES:
1. Data item "pdu" is sent to the CSC ethernet via the Outgoing queue
2. Each PDU builder CSU has it's own interface which includes those unique data items needed to build the PDU. These are: TBD
3. Data item "dtg" is returned to caller CSC client containing Zulu date-time group.
4. Data item "dmcc_timezone_offset_g" contains the localtime to zulu time difference in minutes, and is received from CSC client via the file "/tmp/dmcc_timezone_offset".

3.2.3 CSC ipc

DESCRIPTION: The ipc function provides an interface for the X-window process, and the SIMNET Ethernet interface processes to send and receive DMCC messages. It initializes the X-window client child process that receives new Digital Message arrival notifications from the dms process. The child process reads the new messages from shared memory and passes them to the X-window parent process via a pipe.

Specific duties performed are:
1. Create a 'child' process of CSC client that is used to communicate with CSC dms.
2. Attach to shared memory segment and Fan-Out queue for CSC client 'child'.
3. Attach to Outgoing queue for CSC dmcc 'parent'.

4. Attach to Fan-Out queue for CSC dmcc_sim_tx.
5. Send login and logout message to CSC dms.
6. Put a message in the Outgoing queue.
7. Receive a message from the Fan-Out queue containing notification of the arrival of a new DMCC PDU.
8. Find the address of a free record in the shared memory segment.
9. Copy a DMCC PDU into the shared memory segment.
10. Send updated lists of operator CEOI's/Group's from CSC dmcc to CSC dms.
11. Read the pipe between the X-client process and the child process.
12. Return the exercise id of the operator to a caller.
13. Return the operators name to the caller.

CSC INTERFACES:
1. Data item "mode" received from CSC's client and ethernet.
2. Data item "buf" received from CSC client .
3. Data item "buflen" received from CSC client .
4. Data item "buf" passed to CSC client .
5. Data item "getfreerecord" passed to CSC ethernet.
6. Data item "new_dm_address" is received from CSC ethernet.
7. Data item "buf" is received from CSC ethernet
8. Data item "buflen" received from CSC ethernet.
9. Data item "dests" received from CSC ethernet.
10. Data item "exid" received from CSC ethernet.
11. Data item "dests" received from CSC client .
12. Data item "exid" received from CSC client .
13. Data item "ceoi" received from CSC client .
14. Data item "p" received from CSC client .
15. Data item "pdu" received from CSC client .
16. Data item "dms_getexid" passed to CSC build pdu.
17. Data item "dms_getopername" passed to CSC build pdu.
18. Data item "dmslogin" passed to CSC client .

The following diagram depicts the state transitions of the Client Child process:

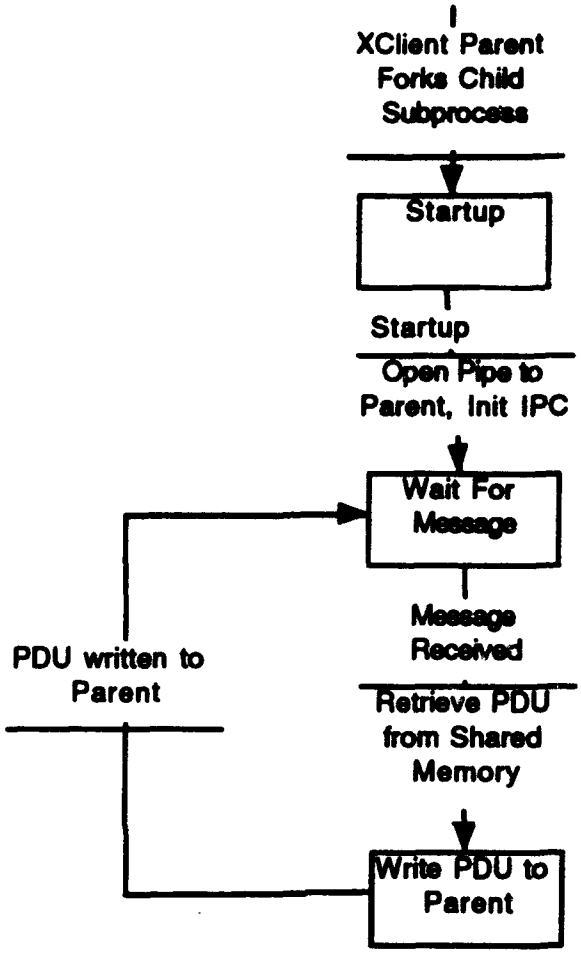


Figure 6 Client Child State Transition Diagram

3.2.4 CSC ethernet

DESCRIPTION: The ethernet CSC contains the functionality for receiving DMCC PDUs from and sending them to the SIMNET ethernet network. It consists of two concurrent processes, one for transmission and one for reception of messages.

Specific duties performed are:
1. Set up receive and transmit queues for the incoming and outgoing messages.
2. Perform encode/decode actions on the messages and completeness checks on the contents of those messages.

CSC INTERFACES:
1. Name of computer interface is passed to dmcc_sim_rx and dmcc_sim_tx via operator input:
dmcc_sim_rx *Ethernet_interface_name*
dmcc_sim_tx *Ethernet_interface_name*

3.2.4.1 Sub-level CSC dmcc_sim_rx process

DESCRIPTION: The dmcc_sim_rx function initializes the message queue, invokes functions to initialize ethernet interface to receive messages in promiscuous mode on Ethernet, to read in message, to timestamp the input message, to validate the SNAP and association layer headers of the input message, to validate exercise identification number, to check for valid association pdu format, to split up the input message packet into individual association PDUs if applicable, and to copy parsed/checked message onto CSC dms shared memory.

Specific duties performed are:
1. get ethernet interface name from command line argument and copy it onto the global variable SIM_interface_name.
2. initialize message queue with dmsinit().
3. invoke SIM_rx_netif().

CSC INTERFACES:
1. on line command argument "le0" or "le1" is the interface name and is copied onto the global variable SIM_interface_name.
2. Data Item"DMS_ENET_TRANSMIT" is passed to CSU dmsinit.

3.2.4.2 Sub-level CSC dmcc_sim_tx

DESCRIPTION: The dmcc sim_tx function controls the flow of messages from the DMCC onto the Ethernet. It parses the interface name from on-line command argument and invokes functions to read pdu from initialized message queue, to add association layer and padding bytes to the input message if applicable, to initialize ethernet interface to send messages onto the network as broadcast messages, to set the IEEE 802.3 MAC sublayer frame control header with the destination broadcast address, to fill the IEEE 802.2 LLC sublayer frame control header, to add padding bytes if necessary, and to output message onto Ethernet.

Specific duties performed are:
1. Initialize message queues for each of the input Ethernet interface names.
2. Zero the message buffer and read a new message into the buffer.
3. Send the message to the pdu encode function.

CSC INTERFACES:
1. on line command argument "le0" or "le1" is the interface name and is copied onto the global variable SIM_interface_name.
2. Data item"DMS_ENET_TRANSMIT" is passed to CSU dmsinit.
3. Data item "input buffer- buff" is passed to CSU bzero.
4. Data item "input buffer-buff" is passed to CSU dms_recvdm.
5. Data items "NEW_ALPDU-pdu sequence", "input buffer-buff" and "number bytes read-nbyte_read" passes to CSU ALPDU_SIM_encode.

3.2.5 CSC client

DESCRIPTION: The client function provides the interface with the user, allowing the user to create and read Digital Messages.

Specific duties performed are:
1. Allow the user to login and send the login information to the dms CSC.
2. Allow the user to enter CEOI/Group information and send it to the dms CSC.
3. Receive PDU from the dms CSC. The client will then send the PDU to the PDU decode CSC for translation to a user readable form. This message is then placed in a global structure and an entry is added to the MSG1 screen and the New Mail icon is updated.
4. Allows the user to read the message.
5. Allows the user to reply to the message.
6. Allows the user to forward the message.
7. Allows the user to forward the message and include an additional message.
8. Allows the user to delete the message from the message queue.
9. Allows the user to compose a Digital Message and send it either as an urgent or a routine message.

CSC INTERFACES:
1. Data item "readBuf" is received from dms CSC via a pipe with the label pipeFD. This data item is then placed in the global data item "messages" as the "pdu" field for use later.
2. The "pdu" field of the data item "messages" is passed to pdu_decode.
3. The data item "msgBuf" is returned from the call to pdu_decode.
4. Data item "dmcc_timezone_offset_g" contains the localtime to zulu time difference in minutes and is received from a read of the file "/tmp/dmcc_timezone_offset_g".
5. The data item "reusePDU" is sent to the bldpdu CSC through a call to fwdpdu.
6. The data item "SenderCEOI" containing the name of the sender of the Digital Message is passed to the bldpdu CSC through a call to bldAck. This call sends an acknowledgment pdu to the sender.

7. Data items (TBD) are sent to the bldpdu CSC via one of the bldpdu functions, depending on what type of Message.

3.2.6 CSC pdu_decode

DESCRIPTION: The pdu decode function decodes the input pdu based on its specific message type and returns the decoded message to caller. It outputs error message for unrecognizable message type.

Specific duties performed are:
1. Decode the input message header.
2. Decode the input message based on its message type.
3. Concatenate the decoded message header and body.
4. Return the decoded message to caller.

CSC INTERFACES:
1. Data item "pdu" pass-in to CSC pdu_decode.
2. Data item "pdu" is passed to the CSU pdu_spec_decode for decoding the header information.
3. Data item "pdu" is casted with its specific message type before passing to the appropriate message decode function to decode the message body.
Possible message decode functions:

```
decode_ack_pdu( (Ack_type *) pdu)
decode_spot_pdu( (Spot_type *) pdu)
decode_bda_pdu( (BDA_type *) pdu);
decode_free_text_pdu( (FreeText_type *) pdu)
decode_gndroute_pdu( (Recon_type *) pdu)
decode_airroute_pdu( (Recon_type *) pdu)
decode_bridge_pdu( (Recon_type *) pdu)
decode_lzpz_pdu( (Recon_type *) pdu)
decode_bpop_pdu( (Recon_type *) pdu)
decode_crossing_pdu( (Recon_type *) pdu)
decode_repeat_pdu( (Artillery_type *) pdu)
decode_cancel_pdu( (Artillery_type *) pdu)
decode_check_pdu( (Artillery_type *) pdu)
decode_cno_pdu( (Artillery_type *) pdu)
decode_shift_pdu( (Artillery_type *) pdu)
decode_nwmsn_pdu( (Artillery_type *) pdu)
decode_mto_pdu( (Artillery_type *) pdu)
decode_shot_pdu( (Artillery_type *) pdu)
decode_splash_pdu( (Artillery_type *) pdu)
decode_eom_pdu( (Artillery_type *) pdu)
decode_status_pdu( (Status_type *) pdu)
decode_request_pdu( (Request_type *) pdu)
decode_nbc1_pdu( (NBC_type *) pdu)
decode_nbc4_pdu( (NBC_type *) pdu)
```

```
decode_nbc5_pdu( (NBC_type *) pdu)
decode_miji_pdu( (MIJI_type *) pdu)
decode_pirep_pdu( (PIREP_type *) pdu)
decode_dnav_pdu( (DNAV_type *) pdu)
decode_movcmd_pdu( (Move_type *) pdu)
```

4. Data item "dmc_buffer" which contains the decoded message header and body is being returned to caller.

4. Detailed design.

The following section identifies and describes the relationship of each CSU to it's calling CSC. The table below provides a reference of CSU's and their calling CSC's.

Table 3. CSC/CSU Cross Reference Table.

CSC	CSU
dms	main
	init_queue
	init_shared_mem
	wait_for_msg
	cmpfixedstr
	trimup
	cleanup_xchild
build pdu	setdtg
	bldAck
	bldFreeText
	bldSpot
	bldRequest
	bldMove
	bldMTO
	bldShot
	bldSplash
	fwdpdu
ipc	dmsinit
	dms_senddm
	dms_recvdm
	dms_getfreerecord
	dms_seekrecord
	dms_notify
	dms_bcopy
	dmslogin
	dmsnewdests
	dmslogout
	dms_getdm
	dms_getexid
	dms_getopername

4.1 CSC dms.

The following paragraphs under this section describe the relationship of the dms CSC in terms of data flow between the CSU's of this CSC and identifies all CSU interfaces that are external to the dms CSC.

CSC dms satisfies System Segment Specification requirements 3.2.1.2.1 and 3.2.1.2.2.

The following structure chart outlines the top level structure of the dms CSC.

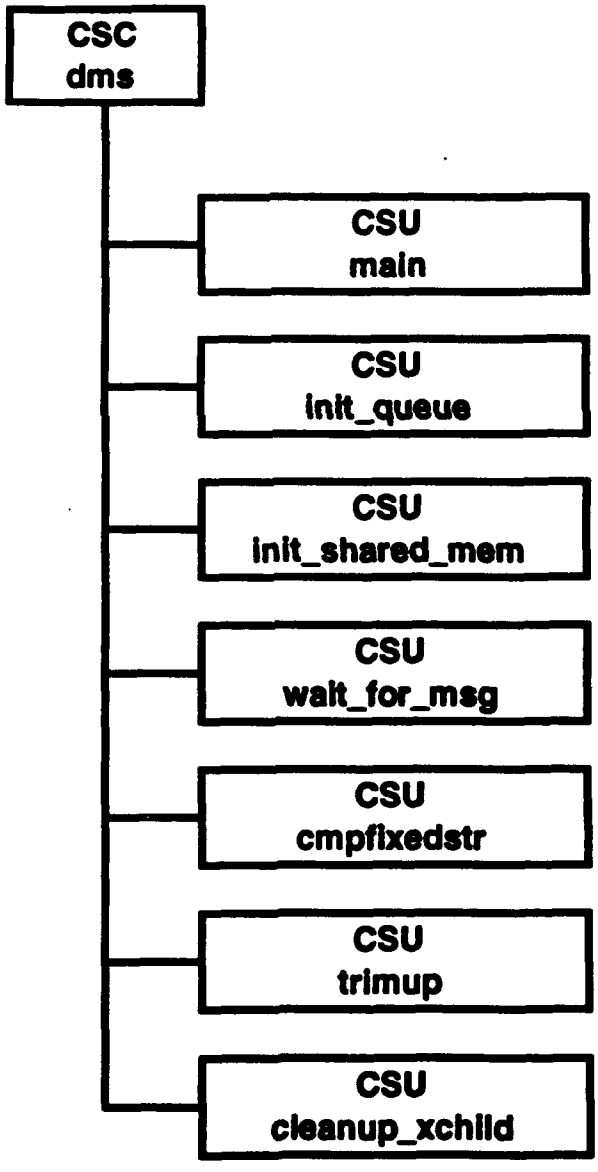


Figure 6a CSC dms Structure

4.1.1 CSU main.

The main CSU is a driver for the dms function, and calls routines to initialize message queues and shared memory, and then calls a routine that infinitely waits for message notifications to arrive. The following paragraphs provide design information for this CSU.

4.1.1.1 CSU main design specification/constraints.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.1.1.2 CSU main design.

The information identified below represents the detailed design of the main CSU.

a. Input/output data elements.

INPUTS:
None

OUTPUTS: None

b. Local data elements.

None

c. Interrupts and signals. None

d. Algorithms. None

e. Error handling. None

f. Data conversion. None

g. Use of other elements. Other elements that are used by the main CSU:
"fanout_qid" stores the id of the Fanout message queue.
"outgoing_qid" stores the id of the Outgoing queue.
"shmaddr" stores the address of the shared memory segment.

h. Logic flow. Single pass.

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

4.1.2 CSU init_queue.

The init_queue CSU creates the message queues used in the DMCC. It will first delete any queues left from a previous run of the software. The following paragraphs provide design information for this CSU.

4.1.2.1 CSU init_queue requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.1.2.2 CSU init_queue design.

The information identified below represents the detailed design of the init_queue CSU.

a. Input/output data elements.

INPUTS:
"key" is a data item containing the operating system if of the queue to be created.

OUTPUTS: None.

b. Local data elements.

"qid" is a data item used to store the id of the created message queue.

c. Interrupts and signals. None

d. Algorithms. None

e. Error handling. logs an error if the queues cannot be attached, and returns 1

f. Data conversion. None

g. Use of other elements. Other elements that are used by the init_queue CSU: None

h. Logic flow. Single pass.

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

4.1.3. CSU init_shared_mem.

The init_shared_mem CSU creates the shared memory segment used in the DMCC. It will first delete the segment if one is left from a previous run of the software. The following paragraphs provide design information for this CSU.

4.1.3.1 CSU init_shared_mem requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.1.3.2 CSU init_shared_mem design.

The information identified below represents the detailed design of the init_shared_mem CSU.

a. Input/output data elements.

INPUTS:
"key" is the operating system id for the shared memory segment.

"size" is the length in bytes of the segment.

OUTPUTS:
"init_shared_mem" is the address of the shared memory segment.

b. Local data elements.

"mid" stores the id of the shared memory segment.

c. Interrupts and signals. None

d. Algorithms. None

e. Error handling. logs an error if the shared memory segment cannot be attached, and returns 1.

f. Data conversion. None

g. Use of other elements. Other elements that are used by the bzero CSU: None

h. Logic flow. Single pass.

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

4.1.4. CSU wait_for_msg.

The wait_for_msg CSU waits for new messages in the Fan-Out queue. These messages are from the CSC dmcc_sim_rx with notifications of a new DMCC PDU arrival, or from CSC dmcc with login, logout, message read, or new CEOIs/Groups. The following paragraphs provide design information for this CSU.

4.1.4.1 CSU wait_for_msg requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.1.4.2 CSU wait_for_msg design.

The information identified below represents the detailed design of the wait_for_msg CSU.

a. Input/output data elements.

INPUTS: None

OUTPUTS: None

b. Local data elements.

"maxshmem" contains the highest address in the shared memory segment

c. Interrupts and signals. None

d. Algorithms. None

e. Error handling. None

f. Data conversion. None

g. Use of other elements. Other elements that are used by the wait_for_msg CSU:
"fanout_qid"
"msgin"
"dm_rqsts"
"shmaddr"

h. Logic flow. Single pass.

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

4.1.5 CSU cmpfixedstr.

The cmpfixedstr CSU compares 2 vectors of strings and returns true if any string in one matches one in the other. The following paragraphs provide design information for this CSU.

4.1.5.1 CSU cmpfixedstr requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.1.5.2 CSU cmpfixedstr design.

The information identified below represents the detailed design of the cmpfixedstr CSU.

a. Input/output data elements.

INPUTS:
"ain" is a pointer to a vector of strings
"bin" is a pointer to a vector of strings
"size" is the maximum length of each string
"nbtocompare" is a count of the number of strings to compare

OUTPUTS:
"cmpfixedstr" is 1 if there is a match, or 0 if there is no match

b. Local data elements.

"i","j" are local indices
"alen","blen" are the lengths of each string being compared
"a","b" are pointers that hold the address of the current strings being compared

c. Interrupts and signals. None

d. Algorithms. None

e. Error handling. Returns 0 if no match, otherwise 1

f. Data conversion. None

g. Use of other elements. Other elements that are used by the cmpfixedstr CSU: None

h. Logic flow. 2 nested for loops running over each vector of strings.

i. Data structures. None

j. Local data files or database. None

k. Limitations. CEOI strings must be no larger than "size" or terminate with a NULL

4.1.6 CSU trimup.

The trimup CSU trims trailing, leading and embedded spaces, and upcases all letters, in a fixed size string. The following paragraphs provide design information for this CSU.

4.1.6.1 CSU trimup requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.1.6.2 CSU trimup design.

The information identified below represents the detailed design of the trimup CSU.

a. Input/output data elements.

INPUTS:
"a" is a pointer to a string
"size" is the size of the string

OUTPUTS:
"a" is a pointer to a string--the trimmed and upcased string overwrites the input string

b. Local data elements.

None

c. Interrupts and signals. None

d. Algorithms. None

e. Error handling. None

f. Data conversion. None

g. Use of other elements. Other elements that are used by the trimup CSU: None

h. Logic flow. For loop over each character in string.

i. Data structures. None

j. Local data files or database. None

k. Limitations. CEOI strings must be no larger than "size" or terminate with a NULL

4.1.7 CSU cleanup_xchild.

The cleanup_xchild CSUcleans up after an XClient-child logs-out or is noticed to have disappeared. It sends SIGHUP and SIGKILL to the child process, and then tries to retrieve any messages left in the Fan-Out queue for that process. If any messages are retrieved then the "shmem_addr" pointer is used to locate messages destined for the process, and decrements the "nbr_readers" counter. The following paragraphs provide design information for this CSU.

4.1.7.1 CSU cleanup_xchild requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.1.7.2 CSU cleanup_xchild design.

The information identified below represents the detailed design of the cleanup_xchild CSU.

a. Input/output data elements.

INPUTS:
"id" is the process id of the X-Client child process

OUTPUTS: None

b. Local data elements.

None

c. Interrupts and signals. None

d. Algorithms. None

e. Error handling. None

f. Data conversion. None

g. Use of other elements. Other elements that are used by the cleanup_xchild CSU:
"fanout_qid"

h. Logic flow. While loop over all messages retrieved from queue.

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

4.2. CSC build pdu.

The following paragraphs under this section describe the relationship of the build pdu CSC in terms of data flow between the CSU's of this CSC and identifies all CSU interfaces that are external to the build pdu CSC.

The following diagram details the top level structure of the build_pdu CSC:

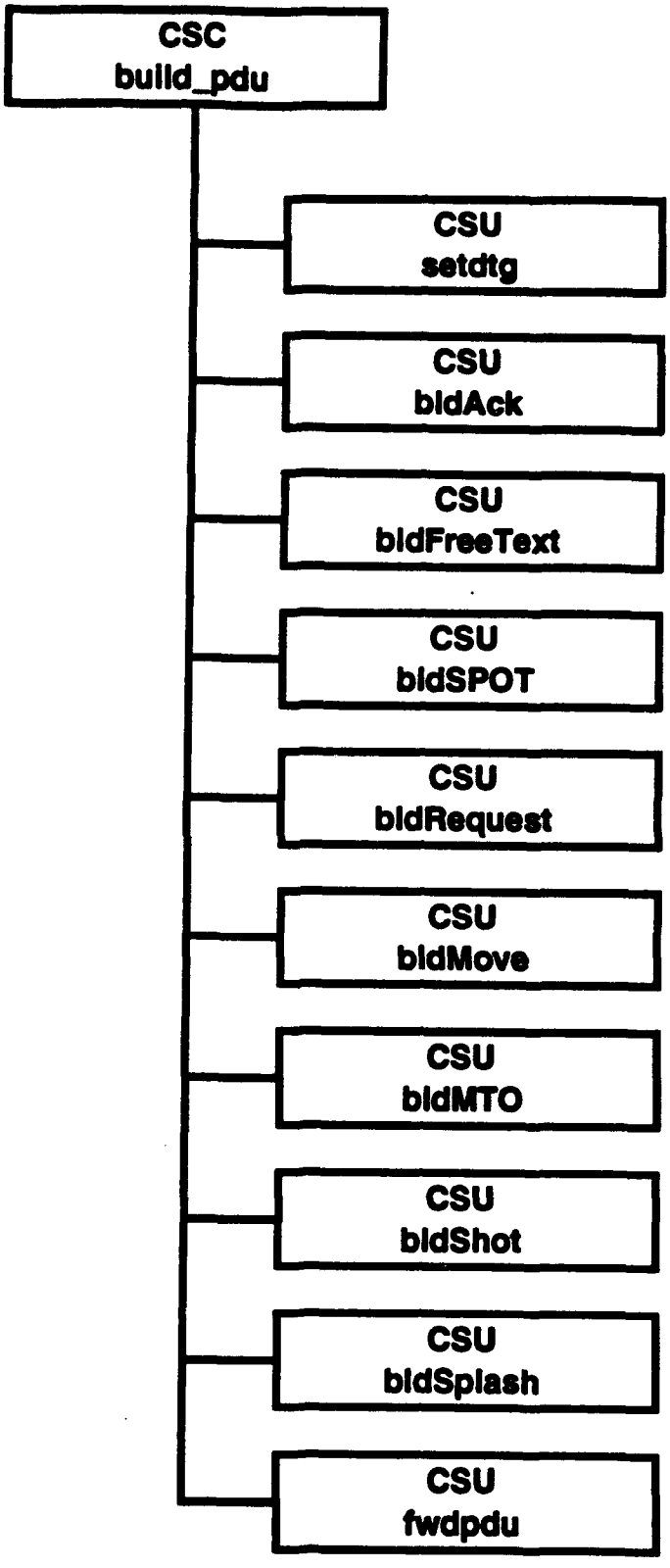


Figure 7 build_pdu Structure

4.2.1 CSU setdtg.

The setdtg CSU returns the current Zulu date-time group to the caller. It The following paragraphs provide design information for this CSU.

4.2.1.1 CSU setdtg requirements

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.2.1.2 CSU setdtg design.

The information identified below represents the detailed design of the setdtg CSU.

a. Input/output data elements.

INPUTS:
"dtg" is a pointer to a data store for the dtg

OUTPUTS:
"dtg" is a pointer to a data store for the dtg

b. Local data elements.

"tp" contains current time
"tm" contains localtime
"buf" contains formatted dtg string

c. Interrupts and signals. None

d. Algorithms. None

e. Error handling. None

f. Data conversion. None

g. Use of other elements. Other elements that are used by the setdtg CSU: None

h. Logic flow. Single pass.

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

4.2.2 CSU bldAck.

The bldAck CSU builds and sends an Ack Digital Message. The following paragraphs provide design information for this CSU.

4.2.2.1 CSU bldAck requirements

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.2.2.2 CSU bldAck design.

The information identified below represents the detailed design of the bldAck CSU.

a. Input/output data elements.

INPUTS:
"targetCEOI" is a pointer to destination CEOI

OUTPUTS: None

b. Local data elements.

None

c. Interrupts and signals. None

d. Algorithms. None

e. Error handling. None

f. Data conversion. None

g. Use of other elements. Other elements that are used by the bldAck CSU:
"pdu"

h. Logic flow. Single pass.

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

4.2.3 CSU bldFreeText.

The bldFreeText CSU builds and sends a FreeText Digital Message. The following paragraphs provide design information for this CSU.

4.2.3.1 CSU bldFreeText requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.2.3.2 CSU bldFreeText design.

The information identified below represents the detailed design of the bldFreeText CSU.

a. Input/output data elements.

INPUTS:
"targetCEOI" is a pointer to the destination CEOI strings
"annotation" is a pointer to the annotation string
"prio" is the PDU priority
"FreeText" is a pointer to the free text string

OUTPUTS: None

b. Local data elements.

None

c. Interrupts and signals. None

d. Algorithms. None

e. Error handling. None

f. Data conversion. None

g. Use of other elements. Other elements that are used by the bldFreeText CSU:
"pdu"

h. Logic flow. Single pass.

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

4.2.4 CSU bldSpot.

The bldSpot CSU builds and sends the SPOT Digital Message. The following paragraphs provide design information for this CSU.

4.2.4.1 CSU bldSpot requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.2.4.2 CSU bldSpot design.

The information identified below represents the detailed design of the bldSpot CSU.

a. Input/output data elements.

INPUTS:
"targetCEOI" is a pointer to the destination CEOI strings
"annotation" is a pointer to the annotation string
"prio" is the PDU priority
"TimeSighted" is a pointer to the time sighted dtg
"ObeserverLocation" is a pointer to the observers UTM
"TargetQuantity" is a count of the targets
"ObserverIntentions" is a byte indicating the observers intentions
"TargetType" is a byte indicating the type of the target

"TargetActivity" is a byte indicating the activity of the target
"TargetDirection" is a byte indicating the direction of the target
"TargetSpeed" is a pointer to the speed of the target
"TargetLocation" is a pointer to the target's UTM
"TargetUnit" is a byte indicating the unit of the target

OUTPUTS: None

b. Local data elements.

None

c. Interrupts and signals. None

d. Algorithms. None

e. Error handling. None

f. Data conversion. None

g. Use of other elements. Other elements that are used by the bldSpot CSU:
"pdu"

h. Logic flow. Single pass.

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

4.2.5 CSU bldRequest.

The bldRequest CSU builds and sends the Request Digital Message. The following paragraphs provide design information for this CSU.

4.2.5.1 CSU bldRequest requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.2.5.2 CSU bldRequest design.

The information identified below represents the detailed design of the bldRequest CSU.

a. Input/output data elements.

INPUTS:
"targetCEOI" is a pointer to the destination CEOI strings
"annotation" is a pointer to the annotation string
"prio" is the PDU priority
"ReportType" is a byte indicating the type of report
"ReconType" is a byte indicating the type of RECON

OUTPUTS: None

b. Local data elements.

None

c. Interrupts and signals. None

d. Algorithms. None

e. Error handling. None

f. Data conversion. None

g. Use of other elements. Other elements that are used by the bldRequest CSU:
"pdu"

h. Logic flow. Single pass.

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

4.2.6 CSU bldMove.

The bldMove CSU builds and sends Move Digital Messages. The following paragraphs provide design information for this CSU.

4.2.6.1 CSU bldMove requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.2.6.2 CSU bldMove design.

The information identified below represents the detailed design of the bldMove CSU.

a. Input/output data elements.

INPUTS:
"targetCEOI" is a pointer to the destination CEOI strings
"annotation" is a pointer to the annotation string
"prio" is the PDU priority
"TargetID" is a pointer to the id of the target
"Task" is a byte indicating the task
"Who" is a byte indicating who
"When" is a byte indicating when
"DTG" is a pointer to a dtg

OUTPUTS: None

b. Local data elements.

None

c. Interrupts and signals. None

d. Algorithms. None

e. Error handling. None

f. Data conversion. None

g. Use of other elements. Other elements that are used by the bldMove CSU:
"pdu"

h. Logic flow. Single pass.

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

4.2.7 CSU bldMTO.

The bldMTO CSU builds and sends MTO Digital Messages. The following paragraphs provide design information for this CSU.

4.2.7.1 CSU bldMTO requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.2.7.2 CSU bldMTO design.

The information identified below represents the detailed design of the bldMTO CSU.

a. Input/output data elements.

INPUTS:
"targetCEOI" is a pointer to the destination CEOI strings
"annotation" is a pointer to the annotation string
"prio" is the PDU priority
"MissionStatus" is a byte indicating the status of the mission
"RequestAdjustment" is a byte indicating an adjustment request
"EnterTarget" is a byte indicating the target to enter
"EndMission" is a byte indicating the mission end

OUTPUTS: None

b. Local data elements.

None

c. Interrupts and signals. None

d. Algorithms. None

e. Error handling. None

f. Data conversion. None

g. Use of other elements. Other elements that are used by the bldMTO CSU:
"pdu"

h. Logic flow. Single pass.

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

4.2.8 CSU bldShot.

The bldShot CSU builds and sends Shot Digital Messages. The following paragraphs provide design information for this CSU.

4.2.8.1 CSU bldShot requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.2.8.2 CSU bldShot design.

The information identified below represents the detailed design of the bldShot CSU.

a. Input/output data elements.

INPUTS:
"targetCEOI" is a pointer to the destination CEOI strings
"annotation" is a pointer to the annotation string
"prio" is the PDU priority
"MissionStatus" is a byte indicating the status of the mission
"RoundsFired" is a byte indicating the rounds fired
"ImpactTime" is a byte indicating the time of impact

OUTPUTS: None

b. Local data elements.

None

c. Interrupts and signals. None

d. Algorithms. None

e. Error handling. None

f. Data conversion. None

g. Use of other elements. Other elements that are used by the bldShot CSU:
"pdu"

h. Logic flow. Single pass.

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

4.2.9. CSU bldSplash.

The following paragraphs provide design information for this CSU.

4.2.9.1 CSU bldSplash requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.2.9.2 CSU bldSplash design.

The information identified below represents the detailed design of the bldSplash CSU.

a. Input/output data elements.

INPUTS:
"targetCEOI" is a pointer to the destination CEOI strings
"annotation" is a pointer to the annotation string
"prio" is the PDU priority
"MissionStatus" is a byte indicating the status of the mission
"RoundsFired" is a byte indicating the rounds fired
"ImpactTime" is a byte indicating the time of impact

OUTPUTS: None

b. Local data elements.

None

c. Interrupts and signals. None

d. Algorithms. None

e. Error handling. None

f. Data conversion. None

g. Use of other elements. Other elements that are used by the bldSplash CSU:
"pdu"

h. Logic flow. Single pass.

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

4.2.10 CSU fwdpdu.

The fwdpdu CSU builds and sends forwarded Digital Messages. The following paragraphs provide design information for this CSU.

4.2.10.1 CSU fwdpdu requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.2.10.2 CSU fwdpdu design.

The information identified below represents the detailed design of the fwdpdu CSU.

a. Input/output data elements.

INPUTS:
"fwdpdu" is a pointer to the destination CEOI string
"tpdu" is a pointer to a pdu

OUTPUTS: None

b. Local data elements.

"pdu_size" contains the size of the pdu

c. Interrupts and signals. None

d. Algorithms. None

e. Error handling. None

f. Data conversion. None

g. Use of other elements. Other elements that are used by the fwdpdu CSU:
"pdu"

h. Logic flow. Single pass.

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

4.3 CSC ipc

The following diagram details the top level structure of the ipc CSU:

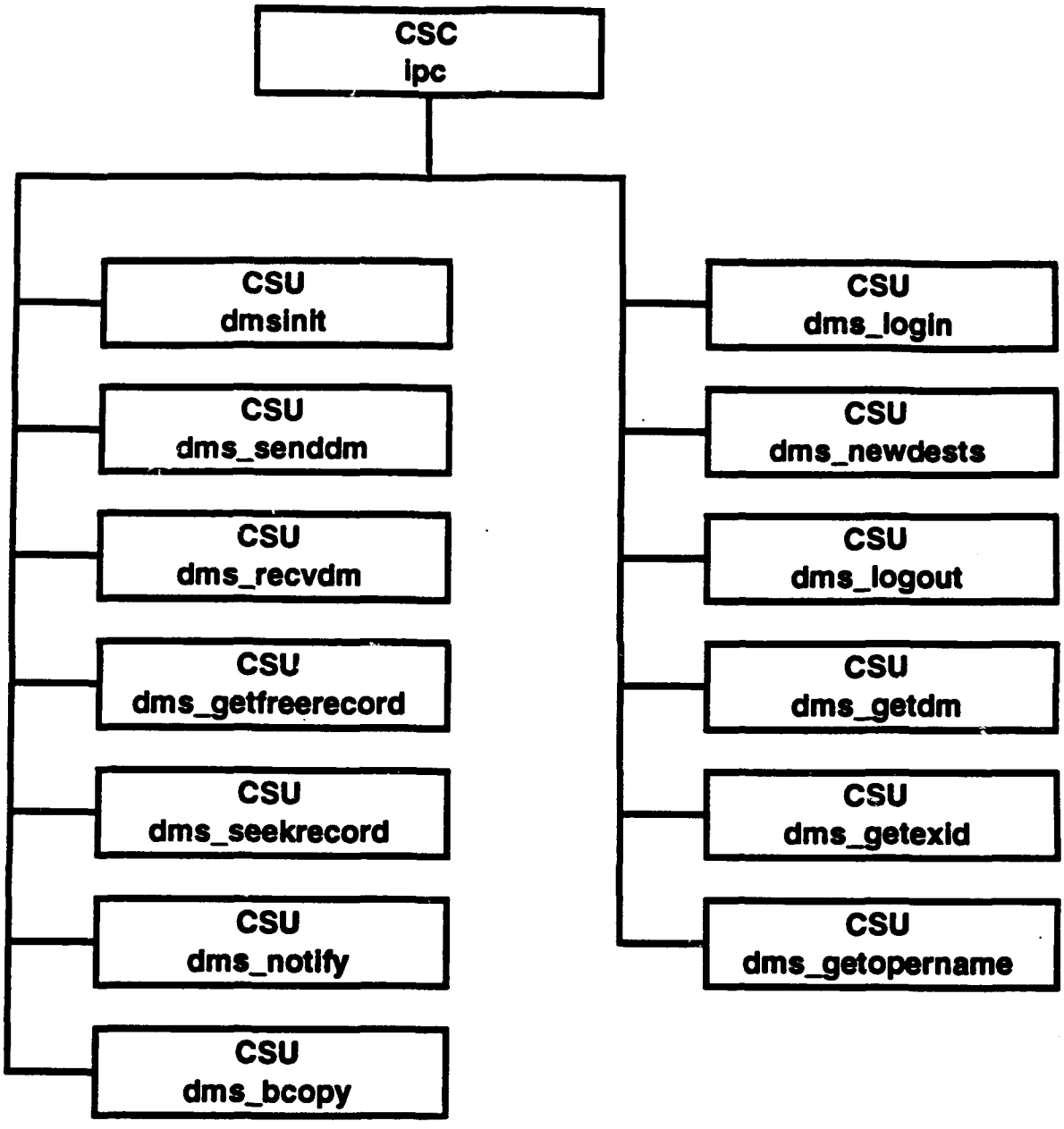


Figure 8 CSC ipc Structure

CSC ipc partially satisfies System Segment Specification requirements 3.2.1.2.1 and 3.2.1.2.2.1.

The following paragraphs under this section describe the relationship of the ipc CSC in terms of data flow between the CSU's of this CSC and identifies all CSU interfaces that are external to the ipc CSC.

4.3.1 CSU dmsinit.

The dmsinit CSU handles attaching to the message queues and to the shared memory segment. The following paragraphs provide design information for this CSU.

4.3.1.1 CSU dmsinit requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.3.1.2 CSU dmsinit design.

The information identified below represents the detailed design of the dmsinit CSU.

a. Input/output data elements.

INPUTS:
"mode" indicates whether the caller is receiving or transmitting to the Ethernet

OUTPUTS: None

b. Local data elements.

"shmem_size" is the maximum size of the shared memory segment

c. Interrupts and signals. None

d. Algorithms. None

e. Error handling. returns 0 if unable to attach to queues and shared memory

f. Data conversion. None

g. Use of other elements. Other elements that are used by the dmsinit CSU:
"fanoutqid"
"incoming_shmid"
"outgoingqid"

h. Logic flow. Single pass.

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

4.3.2 CSU dms_senddm.

The dms_senddm puts a message in the Outgoing queue. The following paragraphs provide design information for this CSU.

4.3.2.1 CSU dms_senddm requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.3.2.2 CSU dms_senddm design.

The information identified below represents the detailed design of the dms_senddm CSU.

a. Input/output data elements.

INPUTS:
"buf" is a pointer to the message
"buflen" indicates the size of the message

OUTPUTS: None

b. Local data elements.

None

c. Interrupts and signals. None

d. Algorithms. None

e. Error handling. returns 0 if unable to put the message in the queue

f. Data conversion. None

g. Use of other elements. Other elements that are used by the dms_senddm CSU:
"outgoingqid"

h. Logic flow. Single pass.

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

4.3.3 CSU dms_recvdm.

The dms_recvdm waits for any message to appear in the Outgoing queue. The following paragraphs provide design information for this CSU.

4.3.3.1 CSU dms_recvdm requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.3.3.2 CSU dms_recvdm design.

The information identified below represents the detailed design of the dms_recvdm CSU.

a. Input/output data elements.

INPUTS:
"buf" is a pointer to a buffer to contain the received message

OUTPUTS:
"buf" is a pointer to a buffer to contain the received message

b. Local data elements.

"msgin" contains the received message

c. Interrupts and signals. None

d. Algorithms. None

e. Error handling. Returns 0 if unable to retrieve the message from the queue

f. Data conversion. None

g. Use of other elements. Other elements that are used by the dms_recvdm CSU:
"outgoingqid"

h. Logic flow. Single pass.

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

4.3.4 CSU dms_getfreerecord.

The dms_getfreerecord CSU tries to find a free record in shared memory by calling dms_seekrecord CSU, and on error, it will sleep and try again, for a limit of MAXRETRY. The following paragraphs provide design information for this CSU.

4.3.4.1 CSU dms_getfreerecord requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.3.4.2 CSU dms_getfreerecord design.

The information identified below represents the detailed design of the dms_getfreerecord CSU.

a. Input/output data elements.

INPUTS: None

OUTPUTS:
"dms_getfreerecord" contains the address of a free record in shared memory

b. Local data elements.

"recaddr" contains address of free record

c. Interrupts and signals. None

d. Algorithms. None

e. Error handling. Return 0 if unable to find a free record

f. Data conversion. None

g. Use of other elements. Other elements that are used by the dms_getfreerecord CSU: None

h. Logic flow. Single pass.

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

4.3.5 CSU dms_seekrecord.

The dms_seekrecord CSU finds the next free record in the shared memory segment and returns the address of the record. The following paragraphs provide design information for this CSU.

4.3.5.1 CSU dms_seekrecord requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.3.5.2 CSU dms_seekrecord design.

The information identified below represents the detailed design of the dms_seekrecord CSU.

a. Input/output data elements.

INPUTS: None

OUTPUTS:
"dms_seekrecord" contains the address of a free record in shared memory

b. Local data elements.

"maxshmem" contains the highest address in the shared memory segment

c. Interrupts and signals. None

d. Algorithms. None

e. Error handling. Returns 0 if unable to find a free record

f. Data conversion. None

g. Use of other elements. Other elements that are used by the dms_seekrecord CSU:
"shmaddr"

h. Logic flow. Single pass.

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

4.3.6 CSU dms_notify.

The dms_notify CSU sends a notification of a new message arrival to dms CSU. The following paragraphs provide design information for this CSU.

4.3.6.1 CSU dms_notify requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.3.6.2 CSU dms_notify design.

The information identified below represents the detailed design of the dms_notify CSU.

a. Input/output data elements.

INPUTS:
"new_dm_address" contains the address of the new message in the shared memory segment

OUTPUTS: None

b. Local data elements.

"msgout" contains the MSG_DMAVAIL message

c. Interrupts and signals. None

d. Algorithms. None

e. Error handling. Returns 0 if unable to put the message in the queue

f. Data conversion. None

g. Use of other elements. Other elements that are used by the dms_notify CSU:
"fanoutqid"

h. Logic flow. Single pass.

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

4.3.7 CSU dms_bcopy.

The dms_bcopy CSU copies a buffer from the caller into shared memory, and then notifies dms CSU. The following paragraphs provide design information for this CSU.

4.3.7.1 CSU dms_bcopy requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.3.7.2 CSU dms_bcopy design.

The information identified below represents the detailed design of the dms_bcopy CSU.

a. Input/output data elements.

INPUTS:
"buf" is a pointer to a buffer containing the message
"buflen" contains the length of the buffer
"dests" is a pointer to a vector of CEOI strings
"exid" is byte containing the id of the exercise

OUTPUTS: None

b. Local data elements.

"nextrec" contains address of next free record in shared memory segment

c. Interrupts and signals. None

d. Algorithms. None

e. Error handling. Returns 0 if unable to copy new message into shared memory segment

f. Data conversion. None

g. Use of other elements. Other elements that are used by the dms_bcopy CSU: None

h. Logic flow. Single pass.

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

4.3.8 CSU dmslogin.

The dmslogin CSU creates a child process for the caller to handle communication with dms CSU. The child then sends a login message to dms CSU. The child process enters an infinite for loop waiting for a new message arrival notification from the dms CSU. The following paragraphs provide design information for this CSU.

4.3.8.1 CSU dmslogin requirements

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.3.8.2 CSU dmslogin design.

The information identified below represents the detailed design of the dmslogin CSU.

a. Input/output data elements.

INPUTS:
"dests" is a pointer to a vector of CEOI strings
"exid" contains the exercise id of the operator

OUTPUTS: None

b. Local data elements.

"p" contains id of pipes
"msgout" contains message sent out to fan-out queue
"msgin" contains messages received from fan-out queue
"shmem_size" contains the size of the shared memory segment

c. Interrupts and signals. None

d. Algorithms. None

e. Error handling. Returns 0 if unable to create pipes

f. Data conversion. None

g. Use of other elements. Other elements that are used by the dmslogin CSU:
"incoming_shmem"
"fanoutqid"
"outgoingqid"
"childpid"
"cur_exerciseid"
"cur_opername"

h. Logic flow. Single pass.

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

4.3.9 CSU dmsnewdests.

The dmsnewdests CSU sends a new set of CEOIs and Groups that the operator wishes to receive. The following paragraphs provide design information for this CSU.

4.3.9.1 CSU dmsnewdests requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.3.9.2 CSU dmsnewdests design.

The information identified below represents the detailed design of the dmsnewdests CSU.

a. Input/output data elements.

INPUTS:
"ceoi" contains the new vector of CEOI/Group strings

OUTPUTS: None

b. Local data elements.

"msgout" contains message sent out to fan-out queue

c. Interrupts and signals. None

d. Algorithms. None

e. Error handling. Returns 0 if unable to put message in queue

f. Data conversion. None

g. Use of other elements. Other elements that are used by the dmsnewdests CSU:
"fanoutqid"

h. Logic flow. Single pass.

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

4.3.10. CSU dmslogout.

The dmslogout CSU sends a logout message to the dms CSC. The following paragraphs provide design information for this CSU.

4.3.10.1 CSU dmslogout requirements

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.3.10.2 CSU dmslogout design.

The information identified below represents the detailed design of the dmslogout CSU.

a. Input/output data elements.

INPUTS: None

OUTPUTS: None

b. Local data elements.

"msgout" contains message sent out to fan-out queue

c. Interrupts and signals. None

d. Algorithms. None

e. Error handling. Returns 0 if unable to put message in queue

f. Data conversion. None

g. Use of other elements. Other elements that are used by the dmslogout CSU:
"fanoutqid"
"childpid"

h. Logic flow. Single pass.

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

4.3.11 CSU dms_getdm.

The dms_getdm CSU reads from the pipe between the X-Client parent and its child. The CSU hangs until the read completes. The following paragraphs provide design information for this CSU.

4.3.11.1. CSU dms_getdm requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.3.11.2 CSU dms_getdm design.

The information identified below represents the detailed design of the dms_getdm CSU.

a. Input/output data elements.

INPUTS:
"p" contains the id of the pipe
"pdu" is a pointer to buffer that will contain the message read from the pipe

OUTPUTS:
"pdu" is a pointer to buffer that will contain the message read from the pipe

b. Local data elements.

"nbytes" contains number of bytes read

c. Interrupts and signals. None

d. Algorithms. None

e. Error handling. None

f. Data conversion. None

g. Use of other elements. Other elements that are used by the dms_getdm CSU:
"p"

h. Logic flow. Single pass.

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

4.3.12 CSU dms_getexid.

The dms_getexid CSU returns the exercise id of the session to the caller. The following paragraphs provide design information for this CSU.

4.3.12.1 CSU dms_getexid requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.3.12.2 CSU dms_getexid design.

The information identified below represents the detailed design of the dms_getexid CSU.

a. Input/output data elements.

INPUTS: None

OUTPUTS:
"dms_getexid" contains the exercise id of the session

b. Local data elements.

None

c. Interrupts and signals. None

d. Algorithms. None

e. Error handling. None

f. Data conversion. None

g. Use of other elements. Other elements that are used by the dms_getexid CSU:
"cur_exerciseid"

h. Logic flow. Single pass.

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

4.3.13 CSU dms_getopername

The dms_getopername CSU returns the operators name to the caller. The following paragraphs provide design information for this CSU.

4.3.13.1 CSU dms_getopername requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.3.13.2 CSU dms_getopername design.

The information identified below represents the detailed design of the dms_getopername CSU.

a. Input/output data elements.

INPUTS: None

OUTPUTS:
"dms_getopername" contains the operators name

b. Local data elements.

None

c. Interrupts and signals. None

d. Algorithms. None

e. Error handling. None

f. Data conversion. None

g. Use of other elements. Other elements that are used by the dms_getopername CSU:
"cur_opername"

h. Logic flow. Single pass.

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

4.4. CSC Ethernet.

The following paragraphs under this section describe the relationship of the Ethernet CSC in terms of data flow between the CSU's of this CSC and identifies all CSU interfaces that are external to the Ethernet CSC.

CSC Ethernet partialy satisfies System Segment Specifiction 3.2.1.2.2.2 and 3.2.1.2.2.1.

4.4.1. Sub-Level CSC dmcc_sim_rx.c.

The following paragraphs under this section describe the relationship of the dmcc_sim_rx Sub-Level CSC in terms of data flow between the CSU's of this Sub-level CSC.

The following diagram depicts the structure of the dmcc_sim_tx sublevel CSC.

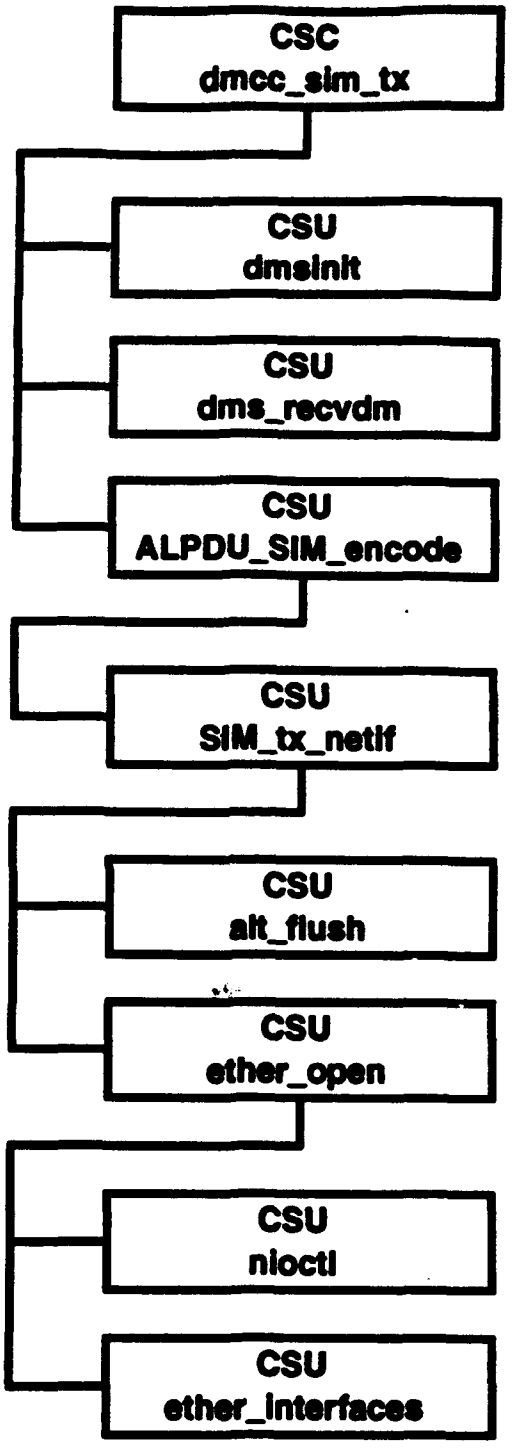


Figure 9 Sublevel CSU dmcc_sim_rx Structure

4.4.1.1 CSU main

The main CSU is a driver for the dmcc_sim_rx function, and calls routines to attach shared memory, set ethernet interface mode for receiving packets, log message status counters, validate Ethernet type, validate the LLC sublayer header, and decode the input message all within a while-loop.

4.4.1.1.1 CSU main requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.4.1.1.2. CSU main design.

The information identified below represents the detailed design of the main CSU.

Syntax of invoking this CSU:

dmcc_sim_rx

a. Input/output data elements.

INPUTS:
"argv" is the command line input argument containing the Ethernet card name.

OUTPUTS:
None.

b. Local data elements.
"i" is an integer for-loop counter.

c. Interrupts and signals. None

d. Algorithms. None

e. Error handling.
display message with printf.

f. Data conversion. None

g. Use of other elements.
Data elements:
SIM_interface_name
Exercise_ID
Library routines:
if - else
printf
exit
CSUs:
dmsinit
SIM_rx_netif

h. Logic flow.
set Exercise ID to pass all type of messages;
if command line argument is less than two
print the proper command usage;
else
get the SIMNET interface name from command line argument;
invoke dmsinit to initialize message queue;
if dmsinit return error
output error message and exit;
invoke SIM_rx_netif;

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

4.4.1.2 CSU SIM_rx_netif

The SIM_rx_netif CSU call function to set the receiving Ethernet interface mode then within an infinite loop, it reads in the input message, validates its Ethernet type, validates the LLC sublayer header, calls function to update the message status counters and to decode the input message.

4.4.1.2.1 CSU SIM_rx_netif requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.4.1.2.2. CSU SIM_rx_netif design.

The information identified below represents the detailed design of the SIM_rx_netif CSU.

Syntax of invoking this CSU:
SIM_rx_netif();

a. Input/output data elements.

INPUTS:
None.
OUTPUTS:
None.

b. Local data elements.
"i" is an integer for-loop counter.
"pType" is an unsigned short integer containing the input message's Ethernet type.
"frame_seq" is long or 64 bit integer indicating the frame sequence.
"timervalues" is an instance of struct timeval.
"timerzone" is an instance of struct timezone.
"iov[]" is an instance of struct iovec.
"byte_read" is an integer storage containing the number of message byte read.
"prom" is of type ether_addr.
"protocol_id" is a local global unsigned integer storage containing the protocol id of the received message.
"etherPacket" is a static ether_packet type containing the input message structure.

c. Interrupts and signals. None

d. Algorithms. None.

e. Error handling.
1. invoke shutdown_SIM_rx to display messages.
2. return 1 for error.

f. Data conversion. None

g. Use of other elements.

Data elements:
SIM_sigrec,
SIM_rx_etherFd,
SIM_interface_name,
SIM_rx_index,
frame_buff[][].

Library routines:
if - else,
while - loop,
printf,
gettimeofday,
memcpy,
readv.

CSUs:
ether_open,
shutdown_SIM_rx,
ALPDU_SIM_decode,
logger.

h. Logic flow.

initialize SIM_sigrec and frame_seq to zero;
set timerzone.tz_dsttime = DST_NONE(with no specific time zone);

invoke ether_open to set the assigned Ethernet card with promiscuous receive mode;

if ether_open return error
invoke shutdown_SIM_rx to output error
message and return with one;

for i < the max of MAX_DATAGRAM +(1 for discard data storage)
set etherPacket.pktbuf[i] points to the address of frame_buff[i][0];

initialize header buffer's length field to the number of possible MAC header length and base field points to the beginning of the input message;

While - forever loop
if the number of non-processed input message < maximum datagram is allowed in the input buffer
initialize data buffer's len field to
maximum
number of byte a message is allowed and
base field to the current index of message

```
input buffer;

invoke readv to read in message;
if number of byte read is <= zero
   invoke shutdown_SIM_rx to output the
   proper error message then return to
   dmcc_sim_rx with an error;
else
   invoke gettimeofday to stamp the
   incoming message;

   if the input message's Ethernet type field
   is not an valid SIMNET message
      invoke logger to log the Bad Ether
      Type error count;
   else
      if protocol_id is not valid;
        invoke logger to log the
      BAD_PROTOCOL_ID counter;
      else if DSAP is not valid
        invoke logger to log the BAD_DSAP          counter;
      else if SSAP is not valid
        invoke logger to log the BAD_SSAP          counter;
      else if control is not valid
        invoke logger to log the                   BAD_CONTROL
      counter;
      else
        invoke ALPDU_SIM_decode to                 decode the input
      message;
        increment the circular SIM_rx_index;
        increment SIM_sigrec to indicate the              number of
      messages that have sent              to ALPDU_SIM_decode but
      not yet              been processed;
else
  invoke readv to read in the message to the
  last datagram storage are in input buffer;
  invoke logger to log the Trash Received
  message counter;
return to caller;
```

i. Data structures. None

j. Local data files or database. None

k. Limitations.

1. Superuser privilege is required to set ethernet function.
2. Only accept message if the number of not yet processed message does not exceed the limit of the input buffer.

4.4.1.3 CSU ALPDU_SIM_decode

The ALPDU_SIM_decode CSU split up the incoming user data into individual ALPDUs since there may be multiple ALPDUs in one user datagram and call function to validate the Exercise ID and validate the ALPDU format.

4.4.1.3.1 CSU ALPDU_SIM_decode requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.4.1.3.2. CSU ALPDU_SIM_decode design.

The information identified below represents the detailed design of the ALPDU_SIM_decode CSU.

Syntax of invoking this CSU:
ALPDU_SIM_decode(address of the input message, length of input message/packet, timestamp);

a. Input/output data elements.

INPUTS:
" assoc_pkt" is an unsigned character pointer that points to the input message/packet.
"max_length" is an unsigned short integer storage containing the input message/packet length.
"timervalues" is of struct timeval containing the input message/packet timestamp.

OUTPUTS: None.

b. Local data elements.
"assoc_length" is a static unsigned short integer storage containing the association length;
"PDU_kind" is a static unsigned character storage containing the pdu type
"next_assoc_pkt" is a static unsigned character pointer that points to the next association layer pdu to be split.
"first_frame" is an unsigned character flag indicating the first association layer pdu of the input packet.
"valid_frame" is a Boolean flag to indicate if an association layer pdu is in valid format.

c. Interrupts and signals. None

d. Algorithms. None

e. Error handling. return with an error integer of 1.

f. Data conversion. None

g. Use of other elements.
Data elements:
SIM_Assoc_Layer.
Library routines:
if - else,
do - while,
if.
CSUs:
check_exercise_ID_SIM,
Rcv_SIM_APDU,
check_SIM_ALPDU,
logger.

h. Logic flow.
initialize valid_frame to FALSE;
if SIM_Assoc_Layer is set to FALSE
if return of check_exercise_ID_SIM is valid
invoke Rcv_SIM_APDU with the NEW_ALPDU parameter;
else
invoke logger to increment the BAD_EXERCISE_ID counter;
else
set next_assoc_pkt points to assoc_pkt;
invoke valid_frame = check_SIM_ALPDU
to check for valid ALPDU format;

```
if valid_frame
  set first_frame = TRUE;
  do - while max_length > 0 and      check_SIM_ALPDU() of the
     next   ALPDU is in valid format

     if PDU_kind is not equal to padding
        invoke check_exercise_ID_SIM;                  if valid exercise
     ID;
         if this is the first frame
            invoke Rcv_SIM_APDU with              the NEW_ALPDU
     parameter;
            set first_frame to FALSE;
        else
            invoke Rcv_SIM_APDU with              the CONT_ALPDU
     parameter;
       else invalid exercise ID
          invoke logger to increment the
     BAD_EXERCISE_ID counter;

     subtract the last ALPDU length from   the original length by
     max_length =       max_length - assoc_length;

     advance the next_assoc_pkt pointer by
     next_assoc_pkt = next_assoc_pkt +     assoc_length;

if the entire packet is valid where valid_frame is TRUE
   invoke Rcv_SIM_APDU to send the packet
   onto SIMNET by passing the END_ALPDU
   parameter;

return valid_frame status to caller.
```

i. Data structures. None

j. Local data files or database. None

k. Limitations.

The global variable SIM_Assoc_Layer must be set TRUE to perform the ALPDU split up.

4.4.1.4 CSU check_exercise_ID_SIM

The check_exercise_ID_SIM CSU

4.4.1.4.1 CSU check_exercise_ID_SIM requirements

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.4.1.4.2. CSU check_exercise_ID_SIM design.

The information identified below represents the detailed design of the check_exercise_ID_SIM CSU.

Syntax of invoking this CSU:

a. Input/output data elements.

INPUTS:
"new_PDU" is an unsigned character pointer that points to the input association layer pdu.

OUTPUTS:
"valid_frame" is a Boolean flag to indicate the status of the input pdu's exercise ID

b. Local data elements. None

c. Interrupts and signals. None

d. Algorithms. None

e. Error handling. None

f. Data conversion. None

g. Use of other elements.
Data elements:
pass_all,
Exercise_ID.
Library routines:
if.
CSUs:
None.

h. Logic flow.
set valid_frame = TRUE;

if pass_all is FALSE and the exercise ID of new_PDU does not equal to
Exercise_ID
set valid_frame = FALSE;

return valid_frame;

i. Data structures. None

j. Local data files or database. None

k. Limitations.

4.4.1.5 CSU check_SIM_ALPDU

The check_SIM_ALPDU CSU checks for a valid ALPDU format. It returns FALSE if invalid and logs the error of TRUE if valid.

4.4.1.5.1 CSU check_SIM_ALPDU design requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.4.1.5.2. CSU check_SIM_ALPDU design.

The information identified below represents the detailed design of the check_SIM_ALPDU CSU.

Syntax of invoking this CSU:
check_SIM_ALPDU(assoc_pkt, max_length, assoc_length, PDU_kind);

a. Input/output data elements.

INPUTS:
"assoc_pkt" is an unsigned character pointer that points to the input association layer pdu address.
"max_length" is an unsigned short integer containing the number of input packet bytes left for splitting.
"assoc_length" is an unsigned short pointer that points to the number bytes in the input association layer pdu.

"PDU_kind" is an unsigned character pointer that points to the association layer pdu's type.

OUTPUTS:
"valid_frame" is an integer storage containing the Boolean value to be returned. It indicates whether or not the input association layer pdu is in a valid ALPDU format.

b. Local data elements.
"assoc_PDU" is a pointer of AssocPDU_type that points to the input association pdu.

c. Interrupts and signals. None

d. Algorithms.
set up association parameters.
check to see if length is correct and still bytes to process.
check to see if its a valid association pdu.
check to see if its a Digital Message Protocol PDU.
return the status of check.

e. Error handling.
invoke logger() to log all errors in a counter format.

f. Data conversion. None

g. Use of other elements.
Data elements:
ASSOC_hdr_leng.
Library routines:
if - else if
CSUs:
logger.

h. Logic flow.
initialize valid_frame to TRUE;
initialize assoc_PDU points to assoc_pkt;

assign the input assoc_PDU->PDUkind to PDU_kind;
set association length by *assoc_length =
ASSOC_hdr_leng[*PDU_kind] +
(assoc_PDU->DataLength * Assoc_MULTIPLIER);

if (*assoc_length > max_length) or (max_length <= zero)
set valid_frame to FALSE;
invoke logger to increment the BAD_ASSOC_LENGTH counter;

else if (*PDU_kind equals to an unknown kind) or (*PDU_kind > the possible association kind)
set valid_frame to FALSE;
invoke logger to increment the BAD_ASSOC_KIND counter;

else if (assoc_PDU->Protocol does not equals to Digital Message Protocol PDU)
set valid_frame to FALSE;
invoke logger to increment the
BAD_ASSOC_PROTOCOL counter;

return valid_frame to caller;

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

4.4.1.6 CSU Rcv_SIM_ALPDU

The Rcv_SIM_ALPDU CSU strips out the incoming PDU and releases the receive buffer by decrementing the global variable sigrec when through processing that PDU and END_ALPDU is received.

4.4.1.6.1 CSU Rcv_SIM_ALPDU design requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.4.1.6.2 CSU Rcv_SIM_ALPDU design.

The information identified below represents the detailed design of the Rcv_SIM_ALPDU CSU.

Syntax of invoking this CSU:

Rcv_SIM_ALPDU(end_ALPDU, assoc_pkt, assoc_length, timervalues);

a. Input/output data elements.

INPUTS:
"end_ALPDU" is an unsigned character containing the sequence of input association pdu.
"assoc_pkt" is an unsigned character pointer that points to the input association layer pdu address.
"assoc_length" is an unsigned short integer storage containing the number bytes in the input association layer pdu.
"timervalues" is of struct timeval containing the input packet timestamp.

OUTPUTS: None

b. Local data elements.

"assoc_PDU" is a pointer of AssocPDU_type that points to the input association pdu.
"ALPDU_kind" is an unsigned character containing the input association pdu type.
"PDU_length" is an unsigned short storage containing the input association pdu length in number of bytes.

c. Interrupts and signals. None

d. Algorithms. None

e. Error handling. None

f. Data conversion. None

g. Use of other elements.

Data elements:
ASSOC_hdr_leng,
SIM_Assoc_Layer.

Library routines:
if
switch - case

CSUs:
decode_SIM_pkt.

h. Logic flow.

```
initialize ALPDU_kind to an unknown kind;
initialize assoc_PDU to point to assoc_pkt;
initialize PDU_length to assoc_length;

if SIM_Assoc_Layer is TRUE
  set ALPDU_kind = assoc_PDU->PDUKind;
  set PDU_length less the
   ASSOC_hdr_leng[ALPDU_kind];

switch (end_ALPDU)
  case continue ALPDU
    invoke decode_SIM_pkt to decode the
   input pdu;
  case end ALPDU
    decrement the SIM_sigrec to indicate the
   input packet is processed and is about to
   sent out to the net;
  case new ALPDU
    invoke decode_SIM_pkt to decode the
   input pdu;

return to caller;
```

i. Data structures. None

j. Local data files or database. None

k. Limitations.

This CSU depends on the global variable SIM_Assoc_Layer to know whether the Association Layer is present and should be processed or not.

This CSU assumes the calling routine has filtered out all Padding APDUs and APDUs which carry on data.

4.4.1.7 CSU dms_decode_SIM_pkt

The dms_decode_SIM_pkt CSU checks for a valid ALPDU format. It returns FALSE if invalid and logs the error of TRUE if valid.

4.4.1.7.1 CSU dms_decode_SIM_pkt design requirements

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.4.1.7.2. CSU dms_decode_SIM_pkt design.

The information identified below represents the detailed design of the dms_decode_SIM_pkt CSU.

Syntax of invoking this CSU:

dms_decode_SIM_pkt(time, old_pkt, new_pkt, max_length, ALPDU_kind);

a. Input/output data elements.

INPUTS:
"time" is of structure timeval containing the input association pdu timestamp - not used.
"old_pkt" is an unsigned character points that points to the input pdu.
"new_pkt" is an unsigned character points - not used(NULL is passed in).
"max_length" is an unsigned short storage containing the number bytes in the input association layer pdu.
"ALPDU_kind" is an unsigned character storage containing the type of pdu input - not used(NULL is passed in).

OUTPUTS: None.

b. Local data elements.

"pduhdr" is a pointer of DMC_Header_type that points to the input association pdu.
"pducmn" is a pointer of DMC_CommonBlock_type.

c. Interrupts and signals. None

d. Algorithms.

check the PDU to make sure it is a Digital Message else return to caller.
Offset to CommonBlock to get the CEOIs.
send the received datagram to "dms_copy" to copy buffer to DMS Shared Memory.

e. Error handling.
output error message with printf if dms_bcopy failed to copy buffer to DMS Shared Memory.

f. Data conversion. None

g. Use of other elements.
Data elements:
ASSOC_hdr_leng.
Library routines:
sizeof
if
printf
CSUs:
dms_bcopy

h. Logic flow.
set pduhdr points to old_pkt;

if pduhdr->HeaderType.SIM.Header.kind is not a Digital Message
return to caller;

offset to common block with pducmn = (DMC_CommonBlock_type *)
(old_pkt + sizeof(DMC_Header_type);

invoke dms_copy with old_pkt, max_length, pducmn->TargetCEOI, and pduhdr->HeaderType.SIM.Header.exercise as parameters;

if dms_copy returns zero
display error message " failed to copy buffer
to DMS Shared Memory ";

return to caller;

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

4.4.1.8 CSU logger

The logger CSU logs any unrecognized data or error conditions that may occur while processing PDUs when logger is invoked.

4.4.1.8.1 CSU logger design requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.4.1.8.2 CSU logger design.

The information identified below represents the detailed design of the logger CSU.

Syntax of invoking this CSU:

logger(errcode, pkt_add, pkt_length);

a. Input/output data elements.

INPUTS:
"errcode" is an unsigned character storage containing the error value which indicating the type of error being logged.
"pkt_add" is an unsigned character pointer that points to the address of the error pdu.
"pkt_length" is an unsigned short storage containing the length of the error pdu in number of bytes.

OUTPUTS: None.

b. Local data elements.
"now" is of time_t type.
"s[]" is a character array of size 100.

c. Interrupts and signals. None

d. Algorithms. None

e. Error handling.
output error message with printf.

f. Data conversion. None

g. Use of other elements.
Data elements:
"logger_datafile" is an integer storage containing the logger file descriptor,
"FILE" file type.

Library routines:
- fopen
- strftime
- fprintf
- if
- printf
- fflush

CSUs:
- None

h. Logic flow.

```
if logger_datafile is NULL
  invoke fopen to create "logger_file.dat"
  with "append" option;
  if file descriptor returned by fopen equals to
  NULL
    printf error message"could not open
    logger_file.dat";
    return to caller;

  set now = time(NULL);
  get current time by invoking strftime(s, 100,
  "\n%H:%M:%S on %A, %d %B %Y",
  localtime(&now));

  invoke fprintf to write current time in s to
  logger_datafile;

  invoke fprintf to write filename to
  logger_datafile;

  invoke fprintf to write the message/error to
  logger_datafile;

  invoke fflush to flush the file descriptor;
  return to caller;
```

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

4.4.1.9 CSU ether_open

The ether_open CSU returns file descriptor for Ethernet device by name("ie0", "le0", etc...). It invoke nioctl CSU to call the ioctl command for opening the Ethernet device.

4.4.1.9.1 CSU ether_open design requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.4.1.9.2 CSU ether_open design.

The information identified below represents the detailed design of the ether_open CSU.

Syntax of invoking this CSU:
ether_open(name, type, address);

a. Input/output data elements.

INPUTS:
"name" is a character pointer containing the Ethernet interface name.
"type" is an unsigned integer containing Ethernet packet type.
"address" is an ether_addr type pointer containing the receiving Ethernet address.

OUTPUTS:
"fd" is an integer storage containing the file descriptor of the Ethernet to be returned.

b. Local data elements.
"ifr" is of ifreq structure type;
"filter" is of packetfilt type;
"fptr" is an unsigned character pointer;
"inerfaces" is of char**;
"i" is an integer register;
"flag" is a long or 64-bit integer flag;

c. Interrupts and signals. None

d. Algorithms. None

e. Error handling. None

f. Data conversion. None

g. Use of other elements.
Data elements:
None
Library routines:
if
perror
close
ioctl
strncpy
CSUs:
None

h. Logic flow.

```
if name is NULL get default ethernet interface
  interface = ether_interfaces();
  if ether_interface returns zero
    return -1 to caller;

  copy interface name: name = *interface;

invoke open to open NIT_DEV;
if return error
  return -1 to caller;

if NIT_DEV file descriptor >= FD_SETSIZE
  close the file descriptor with close;
  set errno = empty file;
  return -1 to caller;

invoke nioctl(fd, I_SRDOPT, (char *) RMSGD) to get discrete message
      from stream;
if return error
  return -1 to caller;

if type does not equals to ETHER_ALLTYPES
  if type > ETHER_ALLTYPES
    close file descriptor with close;
    set errno = EINVAL;
    return -1 to caller;
```

```
/* push the filter to prevent being flooded with extraneous packets */
  if (nioctl(fd, I_PUSH, "pf") < 0
    return -1 to caller;

  fptr = &fileter.Pf_Filter[0];
  *fptr++ = ENF_PUSHWORD + ETHER_TYPE / sizeof (short);
  *fptr++ = ENF_PUSHLIT | ENF_EQ;
  *fptr++ = htons ((u_short)type;

  filter.Pf_FilterLen = fptr - &filter.Pf_Filter[0];
  filter.Pf_Priority = 1;

  if (nioctl(fd, NIOSETF, (char*)&filter) < 0)
    return -1 to caller;

/* binding */
invoke strncpy (ifr.ifr_name, name, sizeof(ifr.ifr_name));

if (nioctl(fd, NIOCBIND, (char*)&ifr) < 0)
  return -1 to caller;

if (addressx is not 0)
  if address->bytes[0] is not set to PROMISCUOUS_MODE
    if ether_cmp(address, &promiscuous)

    if MULTICAST is not a compilation      define    ether_addr
      local_addr;

    if not ETHER_MCAST (address)
      close the file descriptior;
      set errno = EINVAL;
      return -1 to caller;

if MULTICAST is a compilation define
      exit with -1 for error;
else
    if fptr equals to zero
      if (nioctl(fd, I_PUSH, "pf") < 0
        return -1 to caller;

    fptr = (unsigned short *)      &filter.Pf_Filter[0];
    if (type does not equal to     ETHER_ALLTYPES )
      *fptr++ = ENF_PUSHWORD +          ETHER_TYPE/
      sizeof(short);
      *fptr++ = ENF+PUSHLIT;
      *fptr++ = htons( (u_short) type);
```

```
        *fptr++ = ENF_CAND;

      /* compare the address */
      for i < 3
        *fptr++ = (unsigned short)
          (ENF_PUSHWORD + ETHER_DST) /
          sizeof (short) + i);
        *fptr++ = (unsigned short)
                (ENF_PUSHLIT | ENF_EQ);
        *fptr++ = (unsigned short)
               (address->shorts[i]);
        *fptr++ = (unsigned short)
          (ENF_PUSHWORD + ETHER_DST /
          sizeof (short) + i;

        if SHOW_BCAST is a compilation
       define
        *fptr++ = (unsigned short)
                (ENF_PUSHLIT | ENF_EQ);
        *fptr++ = (unsigned short)
                (ether_bcast_addr.shorts[i]);
        *fptr++ = (unsigned short) ENF_OR;

        if SHOW_TCPIP is a compilation define
        *fptr++ = ENF_PUSHLIT | ENF_EQ;
       else
        *fptr++ = ENF_PUSHLIT | ENF_NEQ;

     filter.Pf_FilterLen = fptr -
     &filter.Pf_Filter[0];
     filter.Pf_Priority = 1;

     if (nioctl (fd, NIOCSETF, (char *) &filter) <
       0)
       return -1 to caller;

if MULTICAST is a compilation define
  else

     flag = NI_PROMISC;
     if (nioctl(fd, NIOSCFLAGS, (char *) &flag
       < 0)
        return -1 to caller;

else if in PROMISCUOUS_MODE
   flag = NI_PROMISC;
```

```
if (nioctl(fd, NIOCSFLAGS, (char*)&flag <
  0)
   return -1 to caller;

ether_type = type;
return file descriptor(fd) to caller;
```

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

4.4.1.10 CSU nioctl

The nioctl CSU invoke the ioctl function to set the input/output control.

4.4.1.10.1 nioctl CSU design requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.4.1.10.2 nioctl CSU design.

The information identified below represents the detailed design of the nioctl CSU.

Syntax of invoking this CSU:
nioctl (fd, cmd, ptr);

a. Input/output data elements.

INPUTS:
"fd" is an integer storage containing the file descriptor of to set I/O control on.
"cmd" is an integer storage containing the I/O control command value.
"ptr" is a character pointer containing the address of a ifconf structure type.

OUTPUTS: None.

b. Local data elements. None

c. Interrupts and signals. None

d. Algorithms. None

e. Error handling.

f. Data conversion. None

g. Use of other elements.
Data elements:
None
Library routines:
ioctl
close
CSUs:
None

h. Logic flow.
invoke ioctl with input fd, cmd, ptr;
if ioctl return < 0
close fd;
return -1 to caller;
return zero to caller;

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

4.4.1.11 CSU ether_interfaces

The ether_interfaces CSU returns an array of strings each entry of which is an ethernet interface name valid for use in ether_open().

4.4.1.11.1 CSU ether_interfaces design requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.4.1.11.2 CSU ether_interfaces design.

The information identified below represents the detailed design of the ether_interfaces CSU.

Syntax of invoking this CSU:
ether_interface();

a. Input/output data elements.

INPUTS: None.

OUTPUTS:
"result" is a static character pointer that points to an array of size 16.

b. Local data elements.
/* local global */
"checked" is a static boolean character flag;
"names" is a static character buffer of size 256.
"validname[]" is a static character array initialized with the first two possible interface name letter in a Sun workstation {"le", "ie", "ec", 0}
"_ether_ifindex[16]" is a short integer that is used by _ether_localif().
"_ether_ifconf" is fo struct ifconf.
"cbuf" is a static char buffer of size[16*sizeof(struct ifreq);

/* local data */
"sfd" is an integer storage containing the file descriptor.
"i,j,k,l" are integer registers.
"numinter" is an integer storage containing the number of interface.

c. Interrupts and signals. None

d. Algorithms. None

e. Error handling. None

f. Data conversion. None

g. Use of other elements.
Data elements:
None
Library routines:
close
if-else
goto
strcmp
strncmp
strncpy
for-loop
ioctl
sizeof
socket
if
CSUs:
None

h. Logic flow.

```
if checked is true
  return result to caller;

open the socket by invoking sfd = socket (AF_INET, sock_DGRAM, 0);
if sdf is less than zero
  return zero to caller;

_ether_ifconf.ifc_len = sizeof (cbuf);
_ether_ifconf.ifc_buf = cbuf;

if (ioctl (sfd, SIOCGIFCONF, (char*)&ether_ifconf < 0)
  invoke close to close the file descriptor;
  return zero to caller;

k = 0;
numinter = _ether_ifconf.ifc_len /sizeof (struct ifreq);
for i < numinter
  for j not equals to zero
    if (strcmp   i
        (_ether_ifconf.ifc_req[i].ifr_name,
       validname[j], 2) == 0)
      for l < k
        if not strcmp(result[l],
            _ether_ifconf. ifc_req[i].ifr_name)
          _ether_ifindex[l] = i;
```

```
        goto duplicate;
      strncpy(name + 16 * k,
            _ether_ifconf.ifc_req[i].ifr_name,
           IFNAMSIZ);
      result[k] = names + 16 * k;
       if _ether_ifconf.ifc_req[i].ifr_addr.
         sa_family equals to AF_INET)
         _ether_ifindex[k] = i;
      else
        _ether_ifindex[k] = -1;
      ++k;

duplicate:

append zero to result;
close sfd;
set checked flag to one;
return interface name result to caller;
```

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

4.4.1.12 CSU shutdown_SIM_rx

The shutdown_SIM_rx CSU displays an error message based on the input shutdown code and close the SIM_rx_etherFd.

4.4.1.12.1 CSU shutdown_SIM_rx design requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.4.1.12.2 CSU shutdown_SIM_rx design.

The information identified below represents the detailed design of the shutdown_SIM_rx CSU.

Syntax of invoking this CSU:

```
shutdown_SIM_rx( shutdown_code);
```

a. Input/output data elements.

INPUTS:
"shutdown_code" is an integer value containing the shutdown code.

OUTPUTS: None.

b. Local data elements.

c. Interrupts and signals. None

d. Algorithms. None

e. Error handling.

f. Data conversion. None

g. Use of other elements.
Data elements:
SIM_rx_etherFd
Library routines:
perror
printf
CSUs:
None

h. Logic flow.

```
switch (shutdown_code)
        case 0 or 1
           displays "Error opening stream:";
        case 2
          displays "can't receive data frame message from
        Ethernet";
        case 3
          displays "Only LLC(SNAP) header is read, no other
        data"
        default
          displays "invalid shutdown code"

close SIM_rx_etherFd;
return to caller;
```

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

4.4.1.13 CSU dmsinit

refer to section 4.3.1

4.4.1.14 CSU dms_bcopy

refer to section 4.3.7

4.4.2 Sub-Level CSC dmcc_sim_tx.c.

The following paragraphs under this section describe the relationship of the sim_tx Sub-Level CSC in terms of data flow between the CSU's of this Sub-level CSC.

The following diagram depicts the overall structure of the dmcc_sim_tx Sublevel CSC.

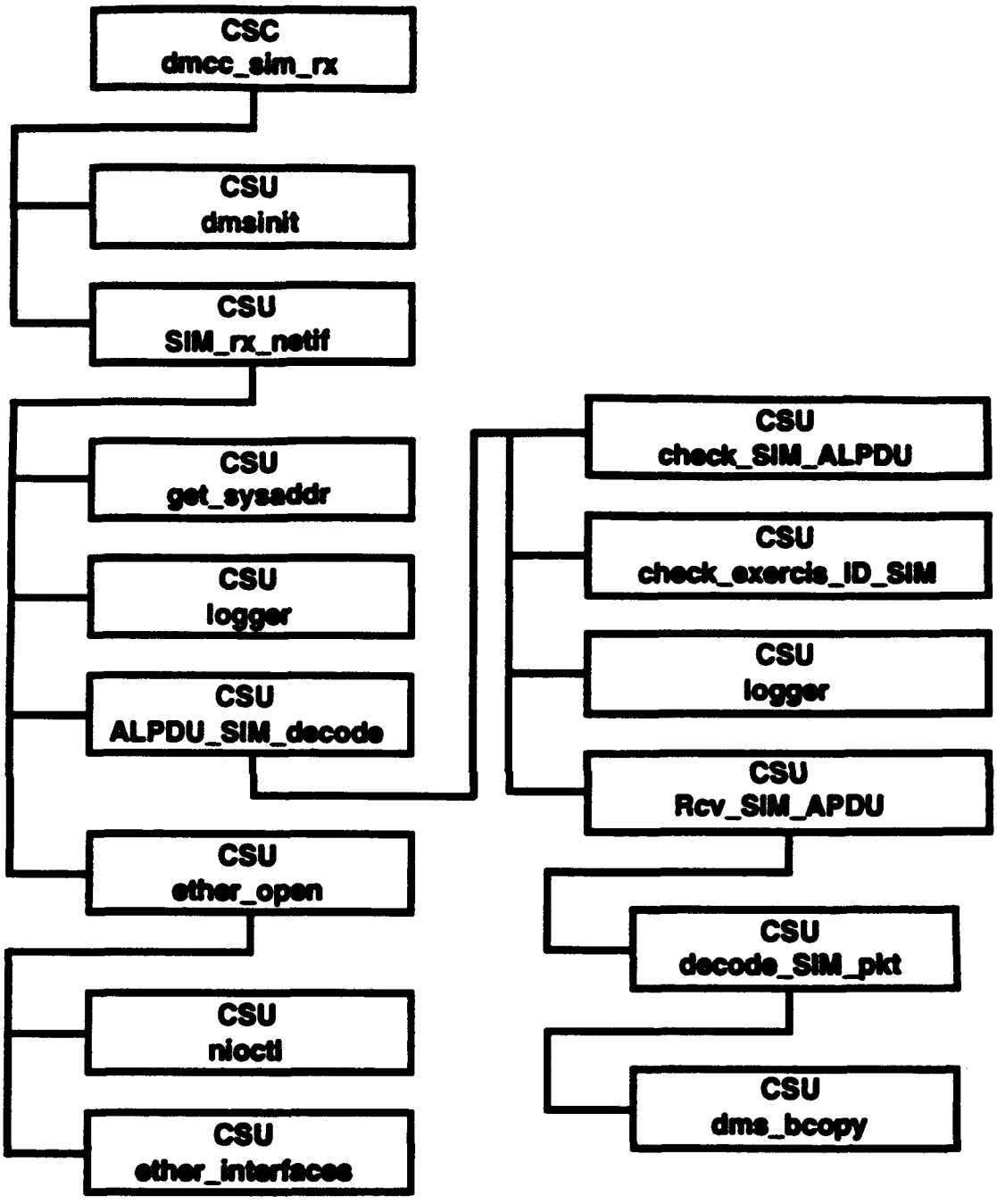


Figure 10 Sublevel CSU dmcc_sim_tx Structure

4.4.2.1 CSU main

The main CSU is a driver for the dmcc_sim_tx function, gets Ethernet interface name from the command line argument, calls routines to initialize message queue, to gets pdu from message queue, and to send pdu to a message encode program.

4.4.2.1.1 CSU main design requirements

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.4.2.1.2 CSU main design.

The information identified below represents the detailed design of the main CSU.

Syntax of invoking this CSU:
dmcc_sim_tx

a. Input/output data elements.

INPUTS:
"argv" is the command line input argument containing the Ethernet interface name.

OUTPUTS:
None.

b. Local data elements.
"i" is an integer for-loop counter.
"buff" is an unsigned character buffer of size 1500 for holding the input pdu.
"nbyte_read" is an unsigned short integer storage containing the number of byte read in from the pdu.

c. Interrupts and signals. None

d. Algorithms. None

e. Error handling.
display message with printf.

f. Data conversion. None

g. Use of other elements.
Data elements:
SIM_interface_name
Library routines:
if - else
printf
bzero
exit
while-loop
CSUs:
dmsinit
dms_recvdm
ALPDU_SIM_encode

h. Logic flow.
if command line argument is less than two
print the proper command usage;
else
get the SIMNET interface name from command line argument;
invoke dmsinit(
DMS_ENET_TRANSMIT) to initialize message queue;
if dmsinit return error
output error message and exit;

while forever
invoke bzero(buff, BUFF_SIZE) to initialize buff;

invoke nbyte_read = dms_recvdm to get input message from message queue;

if nbyte_read equals to zero
output error message and exit with one;

invoke ALPDU_SIM_encode(NEW_ALPDU,
buff,
nbyte_read) to encode the input pdu;

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

4.4.2.2 CSU ALPDU_SIM_encode

The ALPDU_SIM_encode CSU prepares a PDU for transmission by adding the Association Layer and sending the PDU to the network interface routine. The following paragraphs privide design information for this CSU.

4.4.2.2.1 CSU ALPDU_SIM_encode requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.4.2.2.2 CSU ALPDU_SIM_encode design.

The information identified below represents the detailed design of the ALPDU_SIM_encode CSU.

Syntax of invoking this CSU:

```
ALPDU_SIM_encode(
        unsigned char end_ALPDU,
        unsigned char *PDU_pkt,
        unsigned short PDU_length);
```

a. Input/output data elements.

INPUTS:
"end_ALPDU"is an unsigned character containing the sequence(type) of the input pdu_pkt;
"pdu_pkt" is a pointer to the first word of the buffer that is to be encoded and sent onto the network.
"pdu_length" is a count of the number of bytes in the buffer.

OUTPUTS: None.

b. Local data elements.

"i" is a temporary counter used in various simple loops.
"padding" is a temporary counter used to in a loop to fill the transmit buffer with 0.
"offset" is a temporary variable used to position data in the transmit buffer.
"ALPDU_kind" is an unsigned character storage containing the type of association pdu.
" assoc_PDU" is an AssocPDU_type pointer that points to SIM_txbuf[][];

c. Interrupts and signals. None.

d. Algorithms. None.

e. Error handling. None.

f. Data conversion. None.

g. Use of other elements.

Data elements:

SIM_txbuf[][]
SIM_next_frame
SIM_Assoc_Layer

Library routines:

if-else
for - loop
if

CSUs:

SIM_tx_netif

h. Logic flow.

```
if SIM_Assoc_Layer is TRUE
        fill in association layer information

        /*add padding bytes if necessary*/
        if (padding = (PDU_length %    ASSOC_MULTIPLIER)
          padding = ASSOC_MULTIPLIER -
                padding;
        for i < padding
          SIM_txbuf[SIM_next_frame][offset +
                PDU_length + i] = 0x00;
        set assoc_PDU->DataLength =    (unsigned char) ( (padding +
                PDU_length) /
                ASSOC_MULTIPLIER);
else
        ALPDU_kind =
                ASSOC_UNKNOWN_KIND;
        offset = SNAP_LENGTH for                    SIM_tx_netif()   to
        fill in;

for i < PDU_length
        SIM_txbuf[SIM_next_frame][offset+i]         = PDU_pkt[i];

set the total pdu length:  offset = offset +
                PDU_length;
```

invoke SIM_tx_netif to send pdu to Ethernet with &SIM_txbuf[SIM_next_frame[0], offset.

advance the circular input storage index:
SIM_next_frame = (SIM_next_frame + 1) %
MAX_DATAGRAMS;

return to caller;

i. Data structures. None.

j. Local data files or database. None.

k. Limitations. None.

4.4.2.3 CSU SIM_tx_netif

The SIM_tx_netif CSU allows association Protocol Data Units (APDUs) to be transmitted onto the EtherNet as broadcast messages. The calling function specifies the start address and length of buffer to be sent to the Ethernet. The following paragraphs privide design information for this CSU.

4.4.2.3.1 CSU SIM_tx_netif requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.4.2.3.2 CSU SIM_tx_netif design.

The information identified below represents the detailed design of the SIM_tx_netif CSU.

a. Input/output data elements.

INPUTS:
"pdu_in" is an unsigned character pointer to the first word of the buffer that is sent onto the network.

"pdu_len" is an unsigned short containing the number of bytes in the input data.

OUTPUTS: None.

b. Local data elements.
"i,j" are integer loop counter.
"padding" is an integer storage containing the number of bytes to be padded with zero prior to transmission.
"data_len" is an integer storage containing the number of padding bytes when multiply it by eight.
"assoc_data_len" is an integer storage containing the number of padding bytes.
"etherPacket" is a static ether_packet storage.
"sa" is a static sockaddr structure.
"cbuf" is a static strbuf structure containing the pdu control information.
"dbuf" is a static strbuf structure containing the pdu information.

c. Interrupts and signals. None.

d. Algorithms. None.

e. Error handling. On transmission error, log the error and notify the operator.

f. Data conversion.
Data elements:
SIM_tx_etherFd
SIM_interface_name
SIM_Assoc_Layer
SIM_send_frame
Library routines:
if
perror
for-loop
if - else
sizeof
bcopy
CSUs:
ether_open
logger
putmsg

g. Use of other elements. Other elements that are used by the SIM_tx_netif CSU: ether_open, logger.

h. Logic flow. Simple IF ... Then ... Else, Simple for loop.

```
if (init_SIM_tx_netif is FALSE)
    invoke ether_open(SIM_interface_name,
                      ETHER_ALLTYPES,
                      NULL) to initialize the
  Ethernet interface;
  if ether_open returns <= zero
    invoke perror to output error message;
    invoke exit with one;

  set buffer fields for sending Ethernet frame:
  sa.sa_family = AF_UNSPEC:
  cbuf.maxlen = cbuf.len = sizeof(sa);
  cbuf.buf = (char *)&sa;

   set init_SIM_tx_netif to TRUE;

set the IEEE 802.3 LLC sublayer frame control header:
  ALPDU_in[0] = AA;            - DSAP
  ALPDU_in[1] = AA;            - SSAP
  ALPDU_in[2] = 0x03;          - CNTL
  ALPDU_in[3] = 0x080008;      - Protocol ID
  ALPDU_in[6] = 0x5208;        - Ether Type

if (padding = ASSOC_PDU_MIN_LEN -
              ALPDU_len) > 0 ) padding requires
  if  SIM_Assoc_Layer is TRUE
      append padding association layer at     the end of the input pdu:
      1st byte of association layer =
            ASSOC_VER_PADKIND;
      2nd byte of association layer =
            padding/8;
      3rd byte of association layer =
            ASSOC_BROADCAST;
      4th byte of association layer =
            ASSOC_SIMULATION;
      5th to 8th byte of association layer =
            0x00;
  else
      pad the association layer with 0x00;

for j < # of padding bytes
  pad with 0x00;

ALPDU_len = ALPDU_len + association layer's 8 bytes + padding bytes;

etherPacket.LLClen = ALPDU_len;
```

```
etherPacket.pktlen = ALPDU_len;
etherPacket.pktbuf[0] = ALPDU_in;

invoke bcopy((char*)&etherPacket, sa.sa_data, ETHER_HDR_SIZE) to
      copy the Ethernet header onto the control buffer;

set data buffer length to be sent:
dbuf.len = etherPacket.pktlen;
dbuf.buf = (char *)&etherPacket.pktbuf[0][0];

invoke putmsg(SIM_tx_etherFd, &cbuf, &dbuf, 0) to send Ethernet
      frame to STREAM;
if putmsg returns less than zero
   invoke logger to log "PDU_BAD_SENT";
   invoke perror to output error message;

signal ALPDU_SIM_encode that a frame is sent by decrementing
      SIM_send_frame;

return to caller;
```

i. Data structures. None.

j. Local data files or database. None.

k. Limitations.
FUNC must be defined at compilation time.
MODULE_TEST must not be defined at compilation time.

4.4.2.4 CSU dmsinit

refer to section 4.3.1

4.4.2.5 CSU dms_recvdm

refer to section 4.3.3

4.4.2.6 CSU alt_pause

The alt_pause CSU stops the current process until a key is struck from the keyboard.

4.4.2.6.1 CSU pause requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.4.2.6.2 CSU pause design.

The information identified below represents the detailed design of the pause CSU.

Syntax of invoking this CSU:

alt_pause();

a. Input/output data elements.

INPUTS: None

OUTPUTS: None

b. Local data elements. None

c. Interrupts and signals. None

d. Algorithms. None

e. Error handling.

f. Data conversion. None

g. Use of other elements.

Data elements:

stdin

Library routines:

printf
alt_flush
fflush
getchar

CSUs:

None

h. Logic flow.

if #ifdef ALT_FLUSH is defined as a compiler option
invoke alt_flush(stdin) to flush the standard input;

else invoke fflush(stdin) to flush the standard input;

invoke printf to output the "pause" message;
invoke getchar;

if #ifdef ALT_FLUSH is defined as a compiler option
  invoke alt_flush(stdin) to flush the standard input;
else invoke fflush(stdin) to flush the standard input;

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

4.4.2.7 CSU alt_flush

The alt_flush CSU flush all typeheads in the input buffer. alt_flush CSU is used only if #ifdef ALT_FLUSH is defined as a compiler option.

4.4.2.7.1 CSU alt_flush requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.4.2.7.2 CSU alt_flush design.

The information identified below represents the detailed design of the alt_flush CSU.

Syntax of invoking this CSU:
alt_flush(fp);

a. Input/output data elements.

INPUTS:
"fp" FILE pointer;

OUTPUTS:
return 1.

b. Local data elements.

c. Interrupts and signals. None

d. Algorithms. None

e. Error handling. None

f. Data conversion. None

g. Use of other elements.
Data elements:
None
Library routines:
getc
fdopen
fclose
CSUs:
None

h. Logic flow.
while fp->cnt > zero
invoke getc(fp);
return(1);

if the compiler control line
#if constant-expression evaluates to zero
declare fp_tmp as FILE *;

fp_tmp = fdopen(fileno(fp), "r");
invoke fclose(fp) to close the original fp and
loose its buffer;
set the original fp = fp_tmp;

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

4.4.2.8 CSU logger

reference to 4.4.1.8

4.4.2.9 CSU ether_open

reference to 4.4.1.9

4.4.2.10 CSU nioctl

reference to 4.4.1.10

4.4.2.11 CSU ether_interfaces

reference to 4.4.1.11

4.5 CSC client

The DMCC Client CSC is an event driven, X-windows application. It uses the OSF Motif® graphical user interface. It contains hand-coded C language routines as well as routines generated by a graphical user interface design environment (GUIDE) called Builders Xcessory, which is published by Integrated Computer Solutions, Incorporated.

The client CSC manages the X-Windows/OSF Motif® graphical user interface. The following diagram details the top level structure of this CSC.

The client CSC contains the functionalities which satisfy, in large measure, the actual DMCC SYstem Segment Specification requirements 3.2.1.2.2.2, d.2.1.2.2.3, 3.2.1.2.2.3.1, 3.2.1.2.2.3.2, 3.2.1.2.2.3.3, 3.2.1.2.2.3.4 and 3.2. 1.2.2.3.5 and 3.2.1.2.2.3.6.

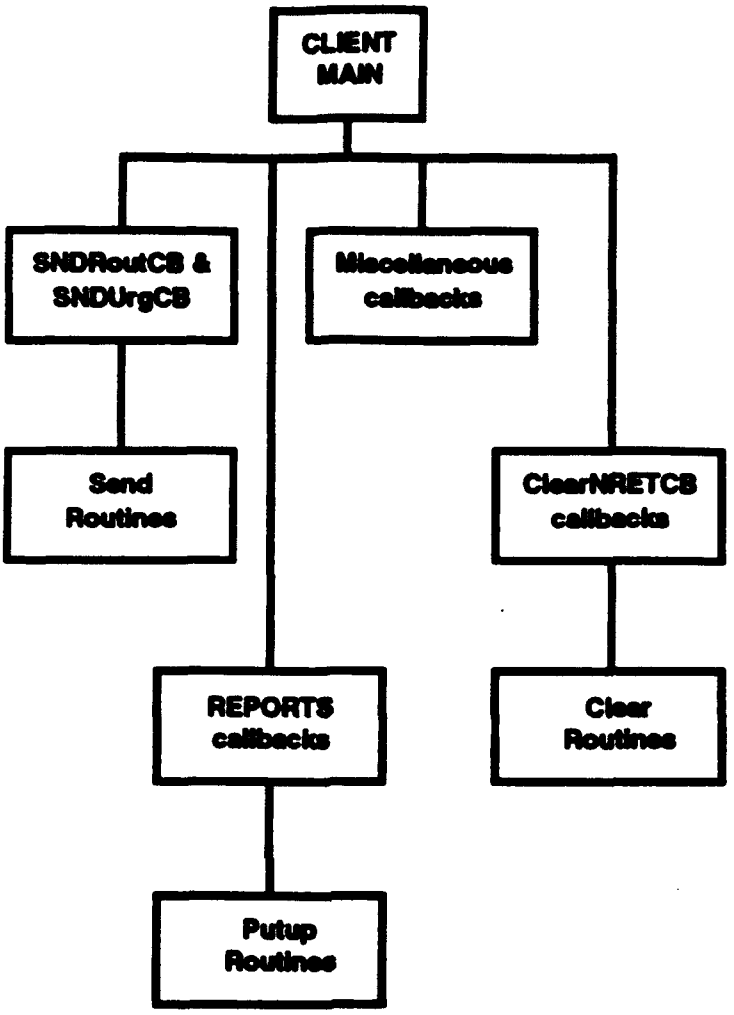


Figure 11 CSC Client Structure Chart

4.5.1 Sub-Level CSC addrListCB.

The following paragraphs under this section describe the relationship of the addrListCB Sub-Level CSC in terms of data flow between the CSU's of this Sub-level CSC.

4.5.1.1 CSU AddrListAddCB

The AddrListAddCB CSU contains the callback for the address list entry screen add button. It takes the entry from the text field and adds it to the address list.

4.5.1.1.1 CSU AddrListAddCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.1.1.2 CSU AddrListAddCB design.

The information identified below represents the detailed design of the AddrListAddCB CSU.

4.5.1.1.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: None

4.5.1.1.2.2 Local data elements.
"text" - contains the text of the highlighted item
"numItems" - contains the number of items in the list

4.5.1.1.2.3 Interrupts and signals. None

4.5.1.1.2.4 Algorithms. None

4.5.1.1.2.5 Error handling. None

4.5.1.1.2.6 Data conversion. None

4.5.1.1.2.7 Use of other elements. Other elements that are used by the AddrListAddCB CSU:
"addrListList" - widget id of the address list
"addrListNewItem" - widget id of the new address text entry field
"addrListAddBtn" - widget id for the add button on the address list screen
"addrListDeleteBtn" - widget id for the delete button on the address list screen
"addressList" - array with all of the entries in the address list

4.5.1.1.2.8 Logic flow. Single pass.

4.5.1.1.2.9 Data structures. None

4.5.1.1.2.10 Local data files or database. None

4.5.1.1.2.11 Limitations. None

4.5.1.2 CSU AddrListDeleteCB

The AddrListDeleteCB CSU contains the callback for the address list entry screen delete button. It takes the entry that is highlighted and deletes it from the address list.

4.5.1.2.1 CSU AddrListDeleteCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.1.2.2 CSU AddrListDeleteCB design.

The information identified below represents the detailed design of the AddrListDeleteCB CSU.

4.5.1.2.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS:

4.5.1.2.2.2 Local data elements.
"i" - counter for loop
"itemNum" - the item number from the list to be deleted
"numItems" - the number of items in the list

4.5.1.2.2.3 Interrupts and signals. None

4.5.1.2.2.4 Algorithms. None

4.5.1.2.2.5 Error handling. None

4.5.1.2.2.6 Data conversion. None

4.5.1.2.2.7 Use of other elements. Other elements that are used by the AddrListDeleteCB CSU:
"addrListList" - widget id of the address list
"addrListAddBtn" - widget id for the add button on the address list screen
"addrListDeleteBtn" - widget id for the delete button on the address list screen
"addressList" - array with all of the entries in the address list

4.5.1.2.2.8 Logic flow. Single pass.

4.5.1.2.2.9 Data structures. None

4.5.1.2.2.10 Local data files or database. None

4.5.1.2.2.11 Limitations. None

4.5.1.3 CSU AddrListSaveExitCB

The AddrListSaveExitCB CSU handles the callback from the Save and Exit pushbutton. It unmanages the Address List screens and remanages the SysMain screens.

4.5.1.3.1 CSU AddrListSaveExitCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.1.3.2 CSU AddrListSaveExitCB design.

The information identified below represents the detailed design of the AddrListSaveExitCB CSU.

4.5.1.3.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS:

4.5.1.3.2.2 Local data elements. None

4.5.1.3.2.3 Interrupts and signals. None

4.5.1.3.2.4 Algorithms. None

4.5.1.3.2.5 Error handling. None

4.5.1.3.2.6 Data conversion. None

4.5.1.3.2.7 Use of other elements. Other elements that are used by the AddrListSaveExitCB CSU:

4.5.1.3.2.8 Logic flow. Single pass.

4.5.1.3.2.9 Data structures. None

4.5.1.3.2.10 Local data files or database. None

4.5.1.3.2.11 Limitations. None

4.5.1.4 CSU AddrListClearExitCB

The AddrListClearExitCB CSU handles the callback from the Clear and Exit pushbutton. It unmanages the Address List screens and remanages the SysMain screens.

4.5.1.4.1 CSU AddrListClearExitCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.1.4.2 CSU AddrListClearExitCB design.

The information identified below represents the detailed design of the AddrListClearExitCB CSU.

4.5.1.4.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS:

4.5.1.4.2.2 Local data elements. None

4.5.1.4.2.3 Interrupts and signals. None

4.5.1.4.2.4 Algorithms. None

4.5.1.4.2.5 Error handling. None

4.5.1.4.2.6 Data conversion. None

4.5.1.4.2.7 Use of other elements. Other elements that are used by the AddrListClearExitCB CSU:

4.5.1.4.2.8 Logic flow. Single pass.

4.5.1.4.2.9 Data structures. None

4.5.1.4.2.10 Local data files or database. None

4.5.1.4.2.11 Limitations. None

4.5.1.5 CSU PutupAddrList

The PutupAddrList CSU creates the Address List entry screen if necessary and manages the objects with the first item of the list highlighted.

4.5.1.5.1 CSU PutupAddrList requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.1.5.2 CSU PutupAddrList design.

The information identified below represents the detailed design of the main CSU.

4.5.1.5.2.1 Input/output data elements.

INPUTS:

OUTPUTS:

4.5.1.5.2.2 Local data elements. None

4.5.1.5.2.3 Interrupts and signals. None

4.5.1.5.2.4 Algorithms. None

4.5.1.5.2.5 Error handling. None

4.5.1.5.2.6 Data conversion.

4.5.1.5.2.7 Use of other elements. Other elements that are used by the PutupAddrList CSU:
"addrListForm" - widget id for the address list entry form
"addrListTmi" - widget id for the address list TMI form
"addrListList" - widget id of the address list

4.5.1.5.2.8 Logic flow. Single pass.

4.5.1.5.2.9 Data structures. None

4.5.1.5.2.10 Local data files or database. None

4.5.1.5.2.11 Limitations. None

4.5.2 Sub-Level CSC consoleCB.

The following paragraphs under this section describe the relationship of the consoleCB Sub-Level CSC in terms of data flow between the CSU's of this Sub-level CSC.

4.5.2.1 CSU ConsoleSysMainBtnCB

The ConsoleSysMainBtnCB CSU handles the callback to from the SysMain pushbutton. It returns dmcc to the SysMain screen at any point during a session.

4.5.2.1.1 CSU ConsoleSysMainBtnCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.2.1.2 CSU ConsoleSysMainBtnCB design.

The information identified below represents the detailed design of the ConsoleSysMainBtnCB CSU.

4.5.2.1.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS:

4.5.2.1.2.2 Local data elements.
"argcnt" keeps track of the number of arguments set for XtSetValues
"args" the arguments set for XtSetValues
"argok" a boolean which is used to check the success of the CONVERT call
"xmstr" local to hold a XmString

4.5.2.1.2.3 Interrupts and signals. None

4.5.2.1.2.4 Algorithms. None

4.5.2.1.2.5 Error handling. None

4.5.2.1.2.6 Data conversion. None

4.5.2.1.2.7 Use of other elements. Other elements that are used by the ConsoleSysMainBtnCB CSU:
"tmiScreen" - widget id which contains the id of the TMI form currently being displayed
"smdScreen" - widget id which contains the id of the entry form currently being displayed
"cikLabel2" - label of the cik2 label

4.5.2.1.2.8 Logic flow. Single pass.

4.5.2.1.2.9 Data structures. None

4.5.2.1.2.10 Local data files or database. None

4.5.2.1.2.11 Limitations. None

4.5.3 Sub-Level CSC freeTxtCB.

The following paragraphs under this section describe the relationship of the freeTxtCB Sub-Level CSC in terms of data flow between the CSU's of this Sub-level CSC.

4.5.3.1 CSU FreeTxtSendCB

The FreeTxtSendCB CSU handles the callback from both the Send Urgent and the Send Routine pushbuttons. It retrieves the data from the text entry area and the annotation area and retrieves the address from the address selection menu. It then sends the data to bldFreeText to be sent out. It also handles the case that the message is a reply by fixing the address to that from the message being replied to. If the message is a Reuse and Include, the included message will also be sent to the address selected.

4.5.3.1.1 CSU FreeTxtSendCB requirements.

The FreeTxtSendCB CSU satisfies section 3.2.1.2.2.2.2, 3.2.1.2.2.3.6.1 and 3.2.1.2.2.3.6.1 of the specific requirments of the DMCC system.

4.5.3.1.2 CSU FreeTxtSendCB design.

The information identified below represents the detailed design of the FreeTxtSendCB CSU.

4.5.3.1.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: .

4.5.3.1.2.2 Local data elements.
"targetCEOI" - the name of the target for the Digital Message
"text" - text from the free text window
"optText" - annotation text
"pri" - priority of the message
"nullChar" null character

4.5.3.1.2.3 Interrupts and signals. None

4.5.3.1.2.4 Algorithms. None

4.5.3.1.2.5 Error handling. None

4.5.3.1.2.6 Data conversion. None

4.5.3.1.2.7 Use of other elements. Other elements that are used by the FreeTxtSendCB CSU:
"FreeTxtAdd" - information on the FreeText address selection menu
"replyFlag" - boolean; is this a reply
"reuseFlag" - boolean; is this a reuse and include
"replyCEOI" - target name for reply call
"reusePDU" - PDU to be forwarded

4.5.3.1.2.8 Logic flow. Single pass.

4.5.3.1.2.9 Data structures. None

4.5.3.1.2.10 Local data files or database. None

4.5.3.1.2.11 Limitations. None

4.5.3.2 CSU FreeTxtSaveReturn

The FreeTxtSaveReturn CSU handles the callback from the Save and Return pushbutton. It will unmanage the Free Text window and remanage the Reports window.

4.5.3.2.1 CSU FreeTxtSaveReturn requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.3.2.2 CSU FreeTxtSaveReturn design.

The information identified below represents the detailed design of the FreeTxtSaveReturn CSU.

4.5.3.2.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: .

4.5.3.2.2.2 Local data elements. None

4.5.3.2.2.3 Interrupts and signals. None

4.5.3.2.2.4 Algorithms. None

4.5.3.2.2.5 Error handling. None

4.5.3.2.2.6 Data conversion. None

4.5.3.2.2.7 Use of other elements. Other elements that are used by the FreeTxtSaveReturn CSU:
"freeTxtTmi" - widget id for Free Text TMI form
"freeTxtForm" - widget id for Free Text entry form

4.5.3.2.2.8 Logic flow. Single pass.

4.5.3.2.2.9 Data structures. None

4.5.3.2.2.10 Local data files or database. None

4.5.3.2.2.11 Limitations. None

4.5.3.3 CSU FreeTxtClearReturn

The FreeTxtClearReturn CSU handles the callback from the Clear and Return pushbutton. It will clear and unmanage the Free Text window and remanage the Reports window.

4.5.3.3.1 CSU FreeTxtClearReturn requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.3.3.2 CSU FreeTxtClearReturn design.

The information identified below represents the detailed design of the FreeTxtClearReturn CSU.

4.5.3.3.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: .

4.5.3.3.2.2 Local data elements. None

4.5.3.3.2.3 Interrupts and signals. None

4.5.3.3.2.4 Algorithms. None

4.5.3.3.2.5 Error handling. None

4.5.3.3.2.6 Data conversion. None

4.5.3.3.2.7 Use of other elements. Other elements that are used by the FreeTxtClearReturn CSU:
"freeTxtTmi" - widget id for Free Text TMI form
"freeTxtForm" - widget id for Free Text entry form

4.5.3.3.2.8 Logic flow. Single pass.

4.5.3.3.2.9 Data structures. None

4.5.3.3.2.10 Local data files or database. None

4.5.3.3.2.11 Limitations. None

4.5.3.4 CSU freetxtad_moreCB

The freetxtad_moreCB CSU handles the callback from the address more button. It will only function if there are more than 4 addresses, in which case it will show a second bank of addresses.

4.5.3.4.1 CSU freetxtad_moreCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.3.4.2 CSU freetxtad_moreCB design.

The information identified below represents the detailed design of the freetxtad_moreCB CSU.

4.5.3.4.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: .

4.5.3.4.2.2 Local data elements. None

4.5.3.4.2.3 Interrupts and signals. None

4.5.3.4.2.4 Algorithms. None

4.5.3.4.2.5 Error handling. None

4.5.3.4.2.6 Data conversion. None

4.5.3.4.2.7 Use of other elements. Other elements that are used by the freetxtad_moreCB CSU:
"FreeTxtAdd" - information on the FreeText address selection menu

4.5.3.4.2.8 Logic flow. Single pass.

4.5.3.4.2.9 Data structures. None

4.5.3.4.2.10 Local data files or database. None

4.5.3.4.2.11 Limitations. None

4.5.3.5 CSU freetxtad_rotateCB

The freetxtad_rotateCB CSU handles the callback from the address rotate button. It will only work if there is more than one address currently being displayed, in which case it will highlight the next address in the list or the first in the list if the last is currently highlighted.

4.5.3.5.1 CSU freetxtad_rotateCB requirements

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.3.5.2 CSU freetxtad_rotateCB design.

The information identified below represents the detailed design of the freetxtad_rotateCB CSU.

4.5.3.5.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: .

4.5.3.5.2.2 Local data elements. None

4.5.3.5.2.3 Interrupts and signals. None

4.5.3.5.2.4 Algorithms. None

4.5.3.5.2.5 Error handling. None

4.5.3.5.2.6 Data conversion. None

4.5.3.5.2.7 Use of other elements. Other elements that are used by the freetxtad_rotateCB CSU:
"FreeTxtAdd" - information on the FreeText address selection menu

4.5.3.5.2.8 Logic flow. Single pass.

4.5.3.5.2.9 Data structures. None

4.5.3.5.2.10 Local data files or database. None

4.5.3.5.2.11 Limitations. None

4.5.3.6 CSU PutupFreeTxt

The PutupFreeTxt CSU will create the Free Text report screen if necessary and will also manage the screen.

4.5.3.6.1 CSU PutupFreeTxt requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.3.6.2 CSU PutupFreeTxt design.

The information identified below represents the detailed design of the PutupFreeTxt CSU.

4.5.3.6.2.1 Input/output data elements.

INPUTS:

OUTPUTS: .

4.5.3.6.2.2 Local data elements.
"argcnt" keeps track of the number of arguments set for XtSetValues
"args" the arguments set for XtSetValues
"argok" a boolean which is used to check the success of the CONVERT call
"xmstr" local to hold a XmString
"i" - counter
"entries" - number of address entries

4.5.3.6.2.3 Interrupts and signals. None

4.5.3.6.2.4 Algorithms. None

4.5.3.6.2.5 Error handling. None

4.5.3.6.2.6 Data conversion. None

4.5.3.6.2.7 Use of other elements. Other elements that are used by the PutupFreeTxt CSU:
"FreeTxtAdd" - information on the FreeText address selection menu
"freeTxtForm" - widget id for the free text entry form
"addressList" - list of addresses

4.5.3.6.2.8 Logic flow. Single pass.

4.5.3.6.2.9 Data structures. None

4.5.3.6.2.10 Local data files or database. None

4.5.3.6.2.11 Limitations. None

4.5.3.7 CSU ClearFreeTxt

The ClearFreeTxt CSU will clear all entries selected in the Free Text report.

4.5.3.7.1 CSU ClearFreeTxt requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.3.7.2 CSU ClearFreeTxt design.

The information identified below represents the detailed design of the ClearFreeTxt CSU.

4.5.3.7.2.1 Input/output data elements.

INPUTS: None

OUTPUTS: None

4.5.3.7.2.2 Local data elements.
"argcnt" keeps track of the number of arguments set for XtSetValues
"args" the arguments set for XtSetValues
"argok" a boolean which is used to check the success of the CONVERT call
"xmstr" local to hold a XmString
"nullChar" null character

4.5.3.7.2.3 Interrupts and signals. None

4.5.3.7.2.4 Algorithms. None

4.5.3.7.2.5 Error handling. None

4.5.3.7.2.6 Data conversion. None

4.5.3.7.2.7 Use of other elements. Other elements that are used by the ClearFreeTxt CSU:
"FreeTxtAdd" - information on the FreeText address selection menu
"textFreeTxt" - widget id of text area of Free Text screen

4.5.3.7.2.8 Logic flow. Single pass.

4.5.3.7.2.9 Data structures. None

4.5.3.7.2.10 Local data files or database. None

4.5.3.7.2.11 Limitations. None

4.5.4 Sub-Level CSC getMessageCB.

The following paragraphs under this section describe the relationship of the getMessageCB Sub-Level CSC in terms of data flow between the CSU's of this Sub-level CSC.

4.5.4.1 CSU getIncomingMsg

The getIncomingMsg CSU handles the receipt of any incoming message pdu. It is called when there is a message waiting to be read. It then reads the message from a pipe and processes it accordingly. If it is an acknowledgment, it will display a popup message with an ack message. If the queue is already full, it will replace the oldest routine message or oldest urgent if no routines exist. Otherwise, it will just place it in the queue and update the Msgs message list. It also updates the numMessages and the numUnread globals.

4.5.4.1.1 CSU getIncomingMsg requirements.

The getIncomingMsg CSU satisfies section 3.2.1.2.2.1.2, 3.2.1.2.2.1.6; 3.2.1.2.2.3.4.1 and 3.2.1.2.2.1.6.3 of the specific requirments of the DMCC system.

4.5.4.1.2 CSU getIncomingMsg design.

The information identified below represents the detailed design of the getIncomingMsg CSU.

4.5.4.1.2.1 Input/output data elements.

INPUTS:
"cli_data" - client data
"sourceFd" - pointer to a pipe file descriptor

"iid" - event handler call info

OUTPUTS: .

4.5.4.1.2.2 Local data elements.
"msgLen" - length of the PDU received
"i" - counter
"readBuf" - buffer read from the pipe sourceFD
"pduhdr"- PDU header taken from readBuf
"string" - temp string to hold the formatted message info string
"messageString" - temp string to hold message to be displayed upon arrival of a message
"index" - index for messages

4.5.4.1.2.3 Interrupts and signals.

4.5.4.1.2.4 Algorithms.

4.5.4.1.2.5 Error handling.

4.5.4.1.2.6 Data conversion.

4.5.4.1.2.7 Use of other elements. Other elements that are used by the getIncomingMsg CSU:
"numMessages" - number of messages currently in the message queue
"smdForm" - widget id of the currently displayed entry form
"messages" - global queue containing messages
"numUnreadMsgs" - number of unread messages in the queue
"msgCount" - count of total messages
"msgListInfo" - array containing the mail box headers for all the messages

4.5.4.1.2.8 Logic flow. Single pass.

4.5.4.1.2.9 Data structures.

4.5.4.1.2.10 Local data files or database.

4.5.4.1.2.11 Limitations.

4.5.5 Sub-Level CSC groupListCB.

The following paragraphs under this section describe the relationship of the groupListCB Sub-Level CSC in terms of data flow between the CSU's of this Sub-level CSC.

4.5.5.1 CSU GrpListAddCB requirements.

The GrpListAddCB CSU contains the callback for the group list entry screen add button. It takes the entry from the text field and adds it to the group list.

4.5.5.1.1 CSU GrpListAddCB design specification/constraints.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.5.1.2 CSU GrpListAddCB design.

The information identified below represents the detailed design of the GrpListAddCB CSU.

4.5.5.1.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: None

4.5.5.1.2.2 Local data elements.
"text" - contains the text of the highlighted item
"numItems" - contains the number of items in the list

4.5.5.1.2.3 Interrupts and signals. None

4.5.5.1.2.4 Algorithms. None

4.5.5.1.2.5 Error handling. None

4.5.5.1.2.6 Data conversion. None

4.5.5.1.2.7 Use of other elements. Other elements that are used by the GrpListAddCB CSU:
"grpListList" - widget id of the grpess list
"grpListNewItem" - widget id of the new grpess text entry field
"grpListAddBtn" - widget id for the add button on the grpess list screen
"grpListDeleteBtn" - widget id for the delete button on the grpess list screen
"grpessList" - array with all of the entries in the grpess list

4.5.5.1.2.8 Logic flow. Single pass.

4.5.5.1.2.9 Data structures. None

4.5.5.1.2.10 Local data files or database. None

4.5.5.1.2.11 Limitations. None

4.5.5.2 CSU GrpListDeleteCB

The GrpListDeleteCB CSU contains the callback for the group list entry screen delete button. It takes the entry that is highlighted and deletes it from the group list.

4.5.5.2.1 CSU GrpListDeleteCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.5.2.2 CSU GrpListDeleteCB design.

The information identified below represents the detailed design of the GrpListDeleteCB CSU.

4.5.5.2.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS:

4.5.5.2.2.2 Local data elements.
"i" - counter for loop
"itemNum" - the item number from the list to be deleted
"numItems" - the number of items in the list

4.5.5.2.2.3 Interrupts and signals. None

4.5.5.2.2.4 Algorithms. None

4.5.5.2.2.5 Error handling. None

4.5.5.2.2.6 Data conversion. None

4.5.5.2.2.7 Use of other elements. Other elements that are used by the GrpListDeleteCB CSU:
"grpListList" - widget id of the grpess list
"grpListAddBtn" - widget id for the add button on the grpess list screen
"grpListDeleteBtn" - widget id for the delete button on the grpess list screen
"grpessList" - array with all of the entries in the grpess list

4.5.5.2.2.8 Logic flow. Single pass.

4.5.5.2.2.9 Data structures. None

4.5.5.2.2.10 Local data files or database. None

4.5.5.2.2.11 Limitations. None

4.5.5.3 CSU GrpListSaveExitCB

The GrpListSaveExitCB CSU handles the callback from the Save and Exit pushbutton. It unmanages the Group List screens and remanages the SysMain screens.

4.5.5.3.1 CSU GrpListSaveExitCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.5.3.2 CSU GrpListSaveExitCB design.

The information identified below represents the detailed design of the GrpListSaveExitCB CSU.

4.5.5.3.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS:

4.5.5.3.2.2 Local data elements. None

4.5.5.3.2.3 Interrupts and signals. None

4.5.5.3.2.4 Algorithms. None

4.5.5.3.2.5 Error handling. None

4.5.5.3.2.6 Data conversion. None

4.5.5.3.2.7 Use of other elements. Other elements that are used by the GrpListSaveExitCB CSU:

4.5.5.3.2.8 Logic flow. Single pass.

4.5.5.3.2.9 Data structures. None

4.5.5.3.2.10 Local data files or database. None

4.5.5.3.2.11 Limitations. None

4.5.5.4 CSU GrpListClearExitCB

The GrpListClearExitCB CSU handles the callback from the Clear and Exit pushbutton. It unmanages the Group List screens and remanages the SysMain screens.

4.5.5.4.1 CSU GrpListClearExitCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.5.4.2 CSU GrpListClearExitCB design.

The information identified below represents the detailed design of the GrpListClearExitCB CSU.

4.5.5.4.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed

"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS:

4.5.5.4.2.2 Local data elements. None

4.5.5.4.2.3 Interrupts and signals. None

4.5.5.4.2.4 Algorithms. None

4.5.5.4.2.5 Error handling. None

4.5.5.4.2.6 Data conversion. None

4.5.5.4.2.7 Use of other elements. Other elements that are used by the GrpListClearExitCB CSU:

4.5.5.4.2.8 Logic flow. Single pass.

4.5.5.4.2.9 Data structures. None

4.5.5.4.2.10 Local data files or database. None

4.5.5.4.2.11 Limitations. None

4.5.5.5 CSU PutupGroupList

The PutupGroupList CSU creates the Group List entry screen if necessary and manages the objects with the first item of the list highlighted.

4.5.5.5.1 CSU PutupGroupList design requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.5.5.2 CSU PutupGroupList design.

The information identified below represents the detailed design of the main CSU.

4.5.5.5.2.1 Input/output data elements.

INPUTS:

OUTPUTS:

4.5.5.5.2.2 Local data elements. None

4.5.5.5.2.3 Interrupts and signals. None

4.5.5.5.2.4 Algorithms. None

4.5.5.5.2.5 Error handling. None

4.5.5.5.2.6 Data conversion.

4.5.5.5.2.7 Use of other elements. Other elements that are used by the PutupGroupList CSU:
"grpListForm" - widget id for the grpess list entry form
"grpListTmi" - widget id for the grpess list TMI form
"grpListList" - widget id of the grpess list

4.5.5.5.2.8 Logic flow. Single pass.

4.5.5.5.2.9 Data structures. None

4.5.5.5.2.10 Local data files or database. None

4.5.5.5.2.11 Limitations. None

4.5.6 Sub-Level CSC listUtilities.

The following paragraphs under this section describe the relationship of the listUtilities Sub-Level CSC in terms of data flow between the CSU's of this Sub-level CSC.

4.5.6.1 CSU ListUpCB

The ListUpCB CSU handles the callback from the Up pushbutton. It changes the highlighted entry to be the previous item on the list or the last item on the list if the first item is currently highlighted.

4.5.6.1.1 CSU ListUpCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.6.1.2 CSU ListUpCB design.

The information identified below represents the detailed design of the ListUpCB CSU.

4.5.6.1.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains the list widget to manipulate
"call" contains X Windows data about the call reason

OUTPUTS: None

4.5.6.1.2.2 Local data elements.
"pos" - the position in the list widget of the higlighted item
"count" - the number of items higlighted
"list" - widget id of the list widget to be rotated

4.5.6.1.2.3 Interrupts and signals. None

4.5.6.1.2.4 Algorithms. None

4.5.6.1.2.5 Error handling. None

4.5.6.1.2.6 Data conversion. None

4.5.6.1.2.7 Use of other elements. Other elements that are used by the ListUpCB CSU:

4.5.6.1.2.8 Logic flow. Single pass.

4.5.6.1.2.9 Data structures. None

4.5.6.1.2.10 Local data files or database. None

4.5.6.1.2.11 Limitations. None

4.5.6.2 CSU ListDownCB

The ListDownCB CSU handles the callback from the Down pushbutton. It changes the highlighted entry to be the next item on the list or the first item on the list if the last item is currently highlighted.

4.5.6.2.1 CSU ListDownCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.6.2.2 CSU ListDownCB design.

The information identified below represents the detailed design of the ListDownCB CSU.

4.5.6.2.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains the list widget to manipulate
"call" contains X Windows data about the call reason

OUTPUTS: None

4.5.6.2.2.2 Local data elements.
"pos" - the position in the list widget of the higlighted item
"count" - the number of items higlighted
"list" - widget id of the list widget to be rotated
"numItems" - the total number of items in the list

4.5.6.2.2.3 Interrupts and signals. None

4.5.6.2.2.4 Algorithms. None

4.5.6.2.2.5 Error handling. None

4.5.6.2.2.6 Data conversion. None

4.5.6.2.2.7 Use of other elements. Other elements that are used by the ListDownCB CSU:

4.5.6.2.2.8 Logic flow. Single pass.

4.5.6.2.2.9 Data structures. None

4.5.6.2.2.10 Local data files or database. None

4.5.6.2.2.11 Limitations. None

4.5.6.3 CSU ListAddItem

The ListAddItem CSU adds a given item into a given list widget. It adds the item to the end of the list. It checks to make sure the string is not empty.

4.5.6.3.1 CSU ListAddItem requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.6.3.2 CSU ListAddItem design.

The information identified below represents the detailed design of the ListAddItem CSU.

4.5.6.3.2.1 Input/output data elements.

INPUTS:
"list" - widget id of the list the item is to be added to
"textItem" - widget id of the text field the string to be added is to be taken from
"addWidget" - widget id of the add entry pushbutton
"delWidget" - widget id of the delete entry pushbutton
"maxItems" - maximum number of items allowed in the list

OUTPUTS:
"text" - the text added to the list

4.5.6.3.2.2 Local data elements.
"text" - text to be added to the list
"tempText" - temporary string holding the text to be added to the list
"nullChar" - empty string
"numItems" - the total number of items in the list

4.5.6.3.2.3 Interrupts and signals. None

4.5.6.3.2.4 Algorithms. None

4.5.6.3.2.5 Error handling. The routine makes sure that the string is not empty.

4.5.6.3.2.6 Data conversion. None

4.5.6.3.2.7 Use of other elements. Other elements that are used by the ListAddItem CSU:

4.5.6.3.2.8 Logic flow. Single pass.

4.5.6.3.2.9 Data structures. None

4.5.6.3.2.10 Local data files or database. None

4.5.6.3.2.11 Limitations. None

4.5.6.4 CSU ListDeleteItem

The ListDeleteItem CSU deletes the highlighted item from a list widget.

4.5.6.4.1 CSU ListDeleteItem requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.6.4.2 CSU ListDeleteItem design.

The information identified below represents the detailed design of the ListDeleteItem CSU.

4.5.6.4.2.1 Input/output data elements.

INPUTS:
"list" - widget id of the list the item is to be added to
"addWidget" - widget id of the add entry pushbutton
"delWidget" - widget id of the delete entry pushbutton

OUTPUTS:
"deletedPos" - position of item deleted from the list

4.5.6.4.2.2 Local data elements.
"pos" - position of item highlighted
"count" - number of items highlighted
"deletedPos" - position of item deleted
"numItems" - the total number of items in the list

4.5.6.4.2.3 Interrupts and signals. None

4.5.6.4.2.4 Algorithms. None

4.5.6.4.2.5 Error handling. None

4.5.6.4.2.6 Data conversion. None

4.5.6.4.2.7 Use of other elements. Other elements that are used by the ListDeleteItem CSU:

4.5.6.4.2.8 Logic flow. Single pass.

4.5.6.4.2.9 Data structures. None

4.5.6.4.2.10 Local data files or database. None

4.5.6.4.2.11 Limitations. None

4.5.7 Sub-Level CSC locListCB.

The following paragraphs under this section describe the relationship of the locListCB Sub-Level CSC in terms of data flow between the CSU's of this Sub-level CSC.

4.5.7.1 CSU LocListAddCB

The LocListAddCB CSU contains the callback for the location list entry screen add button. It takes the entry from the text field and adds it to the location list.

4.5.7.1.1 CSU LocListAddCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.7.1.2 CSU LocListAddCB design.

The information identified below represents the detailed design of the LocListAddCB CSU.

4.5.7.1.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: None

4.5.7.1.2.2 Local data elements.
"text" - contains the text of the highlighted item
"numItems" - contains the number of items in the list

4.5.7.1.2.3 Interrupts and signals. None

4.5.7.1.2.4 Algorithms. None

4.5.7.1.2.5 Error handling. None

4.5.7.1.2.6 Data conversion. None

4.5.7.1.2.7 Use of other elements. Other elements that are used by the LocListAddCB CSU:
"locListList" - widget id of the locess list
"locListNewItem" - widget id of the new locess text entry field
"locListAddBtn" - widget id for the add button on the locess list screen
"locListDeleteBtn" - widget id for the delete button on the locess list screen
"locessList" - array with all of the entries in the locess list

4.5.7.1.2.8 Logic flow. Single pass.

4.5.7.1.2.9 Data structures. None

4.5.7.1.2.10 Local data files or database. None

4.5.7.1.2.11 Limitations. None

4.5.7.2 CSU LocListDeleteCB

The LocListDeleteCB CSU contains the callback for the location list entry screen delete button. It takes the entry that is highlighted and deletes it from the location list.

4.5.7.2.1 CSU LocListDeleteCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.7.2.2 CSU LocListDeleteCB design.

The information identified below represents the detailed design of the LocListDeleteCB CSU.

4.5.7.2.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS:

4.5.7.2.2.2 Local data elements.
"i" - counter for loop
"itemNum" - the item number from the list to be deleted
"numItems" - the number of items in the list

4.5.7.2.2.3 Interrupts and signals. None

4.5.7.2.2.4 Algorithms. None

4.5.7.2.2.5 Error handling. None

4.5.7.2.2.6 Data conversion. None

4.5.7.2.2.7 Use of other elements. Other elements that are used by the LocListDeleteCB CSU:
"locListList" - widget id of the locess list
"locListAddBtn" - widget id for the add button on the locess list screen
"locListDeleteBtn" - widget id for the delete button on the locess list screen

"locessList" - array with all of the entries in the locess list

4.5.7.2.2.8 Logic flow. Single pass.

4.5.7.2.2.9 Data structures. None

4.5.7.2.2.10 Local data files or database. None

4.5.7.2.2.11 Limitations. None

4.5.7.3 CSU LocListSaveExitCB

The LocListSaveExitCB CSU handles the callback from the Save and Exit pushbutton. It unmanages the Location List screens and remanages the SysMain screens.

4.5.7.3.1 CSU LocListSaveExitCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.7.3.2 CSU LocListSaveExitCB design.

The information identified below represents the detailed design of the LocListSaveExitCB CSU.

4.5.7.3.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS:

4.5.7.3.2.2 Local data elements. None

4.5.7.3.2.3 Interrupts and signals. None

4.5.7.3.2.4 Algorithms. None

4.5.7.3.2.5 Error handling. None

4.5.7.3.2.6 Data conversion. None

4.5.7.3.2.7 Use of other elements. Other elements that are used by the LocListSaveExitCB CSU:

4.5.7.3.2.8 Logic flow. Single pass.

4.5.7.3.2.9 Data structures. None

4.5.7.3.2.10 Local data files or database. None

4.5.7.3.2.11 Limitations. None

4.5.7.4 CSU LocListClearExitCB .

The LocListSaveExitCB CSU handles the callback from the Save and Exit pushbutton. It unmanages the Location List screens and remanages the SysMain screens.

4.5.7.4.1 CSU LocListClearExitCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.7.4.2 CSU LocListClearExitCB design.

The information identified below represents the detailed design of the LocListClearExitCB CSU.

4.5.7.4.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: None

4.5.7.4.2.2 Local data elements. None

4.5.7.4.2.3 Interrupts and signals. None

4.5.7.4.2.4 Algorithms. None

4.5.7.4.2.5 Error handling. None

4.5.7.4.2.6 Data conversion. None

4.5.7.4.2.7 Use of other elements. Other elements that are used by the LocListClearExitCB CSU:

4.5.7.4.2.8 Logic flow. Single pass.

4.5.7.4.2.9 Data structures. None

4.5.7.4.2.10 Local data files or database. None

4.5.7.4.2.11 Limitations. None

4.5.7.5 CSU PutupLocationList.

The PutupLocationList CSU creates the Location List entry screen if necessary and manages the objects with the first item of the list highlighted.

4.5.7.5.1 CSU PutupLocationList requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.7.5.2 CSU PutupLocationList design.

The information identified below represents the detailed design of the main CSU.

4.5.7.5.2.1 Input/output data elements.

INPUTS:

OUTPUTS:

4.5.7.5.2.2 Locational data elements. None

4.5.7.5.2.3 Interrupts and signals. None

4.5.7.5.2.4 Algorithms. None

4.5.7.5.2.5 Error handling. None

4.5.7.5.2.6 Data conversion.

4.5.7.5.2.7 Use of other elements. Other elements that are used by the PutupLocationList CSU:
"locListForm" - widget id for the locess list entry form
"locListTmi" - widget id for the locess list TMI form
"locListList" - widget id of the locess list

4.5.7.5.2.8 Logic flow. Single pass.

4.5.7.5.2.9 Data structures. None

4.5.7.5.2.10 Locational data files or database. None

4.5.7.5.2.11 Limitations. None

4.5.8 Sub-Level CSC logonCB.

The following paragraphs under this section describe the relationship of the logonCB Sub-Level CSC in terms of data flow between the CSU's of this Sub-level CSC.

4.5.8.1 CSU LogonNetLogonCB

The LogonNetLogonCB CSU handles the callback for the logon pushbutton. It calls dmslogin to inform dms that it exists and sets up the process for receiving messages via a pipe.

4.5.8.1.1 CSU LogonNetLogonCB requirements

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.8.1.2 CSU LogonNetLogonCB design.

The information identified below represents the detailed design of the LogonNetLogonCB CSU.

4.5.8.1.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: None

4.5.8.1.2.2 Local data elements.
"exerID" - string with the exercise id
"logonName" - string with logon name
"exerIDtemp" - temp with integer version of the exercise id
"offset_file" - file descriptor for the file containing the zulu time offset
"nullChar" - empty string

4.5.8.1.2.3 Interrupts and signals. None

4.5.8.1.2.4 Algorithms. None

4.5.8.1.2.5 Error handling. None

4.5.8.1.2.6 Data conversion. None

4.5.8.1.2.7 Use of other elements. Other elements that are used by the LogonNetLogonCB CSU:
"exerIDnum" - global with the exercise ID number
"logonNameText" - widget id of the text field containing the logon name
"exerciseIDText" - widget id of the text field containing the exercise id
"groupList" - global containing the CEOI/group list
"OFFSET_FILENAME" - define with the name of the offset file
"pipeFD" - file descriptor of the pipe over which messages will come in
"sysMainButton" - widget id of sys main pushbutton
"logonTmi" - widget id of the TMI area of the logon screen
"logonForm" - widget id of the entry area of the logon screen

4.5.8.1.2.8 Logic flow. Single pass.

4.5.8.1.2.9 Data structures. None

4.5.8.1.2.10 Local data files or database. None

4.5.8.1.2.11 Limitations. None

4.5.8.2 CSU PutupLogon

The PutupLogon CSU creates the logon screen if necessary and manages the objects.

4.5.8.2.1 CSU PutupLogon requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.8.2.2 CSU PutupLogon design.

The information identified below represents the detailed design of the PutupLogon CSU.

4.5.8.2.2.1 Input/output data elements.

INPUTS: None

OUTPUTS: None

4.5.8.2.2.2 Local data elements. None

4.5.8.2.2.3 Interrupts and signals. None

4.5.8.2.2.4 Algorithms. None

4.5.8.2.2.5 Error handling. None

4.5.8.2.2.6 Data conversion. None

4.5.8.2.2.7 Use of other elements. Other elements that are used by the PutupLogon CSU:
"logonTmi" - widget id of the TMI area of the logon screen
"logonForm" - widget id of the entry area of the logon screen

4.5.8.2.2.8 Logic flow. Single pass.

4.5.8.2.2.9 Data structures. None

4.5.8.2.2.10 Local data files or database. None

4.5.8.2.2.11 Limitations. None

4.5.9 Sub-Level CSC main.

The following paragraphs under this section describe the relationship of the main Sub-Level CSC in terms of data flow between the CSU's of this Sub-level CSC.

4.5.9.1 CSU main

The main CSU initializes the X window client, all variables to be used by the report menus, creates the toplevel form and goes into a loop waiting for stuff to happen.

4.5.9.1.1 CSU main requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.9.1.2 CSU main design.

The information identified below represents the detailed design of the main CSU.

4.5.9.1.2.1 Input/output data elements.

INPUTS:
"argc" - number of arguments
"argv" - array of argument strings

OUTPUTS: None

4.5.9.1.2.2 Local data elements.
"display" - the X display being used
"argcnt" - keeps track of the number of arguments set for XtSetValues
"args" - the arguments set for XtSetValues
"argok" - a boolean which is used to check the success of the CONVERT call

4.5.9.1.2.3 Interrupts and signals. None

4.5.9.1.2.4 Algorithms. None

4.5.9.1.2.5 Error handling. None

4.5.9.1.2.6 Data conversion. None

4.5.9.1.2.7 Use of other elements. Other elements that are used by the main CSU:
"context" - X windows context of the application
"numMessages" - number of messages in the message queue
"numUnreadMsgs" - number of unread messages in the message queue
"msgCount" - how high the messages can count
"AppShell" - widget id of the application shell
"Shell000" - widget id of the top level popup shell
"Form" - widget id of the top level form

4.5.9.1.2.8 Logic flow. Single pass with infinite loop at the end.

4.5.9.1.2.9 Data structures. None

4.5.9.1.2.10 Local data files or database. None

4.5.9.1.2.11 Limitations. None

4.5.10 Sub-Level CSC movcmdTmi_call

The following paragraphs under this section describe the relationship of the movcmdTmi_call Sub-Level CSC in terms of data flow between the CSU's of this Sub-level CSC.

4.5.10.1 CSU MovcmdClearNRetCB

The MovcmdClearNRetCB CSU handles the callback from the Clear and Return pushbutton. It will clear and unmanage the Movcmd window and remanage the Reports window.

4.5.10.1.1 CSU MovcmdClearNRetCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.10.1.2 CSU MovcmdClearNRetCB design.

Th nformation identified below represents the detailed design of the Mo cmdClearNRetCB CSU.

4.5.10.1.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: None

4.5.10.1.2.2 Local data elements.
"argcnt" keeps track of the number of arguments set for XtSetValues
"args" the arguments set for XtSetValues
"argok" a boolean which is used to check the success of the CONVERT call
"xmstr" local to hold a XmString

4.5.10.1.2.3 Interrupts and signals. None

4.5.10.1.2.4 Algorithms. None

4.5.10.1.2.5 Error handling. None

4.5.10.1.2.6 Data conversion. None

4.5.10.1.2.7 Use of other elements. Other elements that are used by the MovcmdClearNRetCB CSU:
"MovcmdTmiForm" - form widget containing Movcmd TMI
"MovcmdForm" - form widget containing Movcmd Entry area
"cikLabel2" - label widget

4.5.10.1.2.8 Logic flow. Single pass

4.5.10.1.2.9 Data structures. None

4.5.10.1.2.10 Local data files or database. None

4.5.10.1.2.11 Limitations. None

4.5.10.2 CSU MovcmdSaveNRetCB .

The MovcmdSaveNRetCB CSU handles the callback from the Save and Return pushbutton. It will unmanage the Movcmd window and remanage the Reports window.

4.5.10.2.1 CSU MovcmdSaveNRetCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.10.2.2 CSU MovcmdSaveNRetCB design.

The information identified below represents the detailed design of the MovcmdSaveNRetCB CSU.

4.5.10.2.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: None

4.5.10.2.2.2 Local data elements.
"argcnt" keeps track of the number of arguments set for XtSetValues
"args" the arguments set for XtSetValues
"argok" a boolean which is used to check the success of the CONVERT call
"xmstr" local to hold a XmString
"nullChar" null character

4.5.10.2.2.3 Interrupts and signals. None

4.5.10.2.2.4 Algorithms. None

4.5.10.2.2.5 Error handling. None

4.5.10.2.2.6 Data conversion. None

4.5.10.2.2.7 Use of other elements. Other elements that are used by the MovcmdSaveNRetCB CSU:
"MovcmdTmiForm" - form widget containing Movcmd TMI
"MovcmdForm" - form widget containing Movcmd Entry area
"cikLabel2" - label widget
"cikLabel1" - label widget

4.5.10.2.2.8 Logic flow. Single pass

4.5.10.2.2.9 Data structures. None

4.5.10.2.2.10 Local data files or database. None

4.5.10.2.2.11 Limitations. None

4.5.10.3 CSU MovcmdSndRoutCB

The MovcmdSndRoutCB CSU handles the callback from the Send Routine pushbutton. It calls MovcmdSend with priority ROUTINE.

4.5.10.3.1 CSU MovcmdSndRoutCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.10.3.2 CSU MovcmdSndRoutCB design.

The information identified below represents the detailed design of the MovcmdSndRoutCB CSU.

4.5.10.3.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: None

4.5.10.3.2.2 Local data elements. None

4.5.10.3.2.3 Interrupts and signals. None

4.5.10.3.2.4 Algorithms. None

4.5.10.3.2.5 Error handling. None

4.5.10.3.2.6 Data conversion. None

4.5.10.3.2.7 Use of other elements. Other elements that are used by the MovcmdSndRoutCB CSU:

"PRIORITY_ROUTINE" - define for priority message

4.5.10.3.2.8 Logic flow. Single pass

4.5.10.3.2.9 Data structures. None

4.5.10.3.2.10 Local data files or database. None

4.5.10.3.2.11 Limitations. None

4.5.10.4 CSU MovcmdSend

The MovcmdSend CSU sends a message with a given priority. It retrieves the data from the text entry area and the annotation area and retrieves the address from the address selection menu. It then sends the data to bldMovcmd to be sent out. It also handles the case that the message is a reply by fixing the address to that from the message being replied to. If the message is a Reuse and Include, the included message will also be sent to the address selected.

4.5.10.4.1 CSU MovcmdSend requirements.

The MovcmdSend CSU satisfies section 3.2.1.2.2.2.1, 3.2.1.2.2.3.6.1 and 3.2.1.2.2.3.6.1 of the specific requirments of the DMCC system.

4.5.10.4.2 CSU MovcmdSend design.

The information identified below represents the detailed design of the MovcmdSend CSU.

4.5.10.4.2.1 Input/output data elements.

INPUTS:
"prio" - priority of the message being sent

OUTPUTS: None

4.5.10.4.2.2 Local data elements.
"targetCEOI" - name of user message is being sent to
"annotation" - text of annotation string
"ObserverLocation" - location of observer
"TargetID" - ID of the target
"MoTask" - movement task
"Who" - who to move
"MoWhen" - when to move
"dateString" - string with date info

"DTG" - structure with the date info

4.5.10.4.2.3 Interrupts and signals. None

4.5.10.4.2.4 Algorithms. None

4.5.10.4.2.5 Error handling. None

4.5.10.4.2.6 Data conversion. None

4.5.10.4.2.7 Use of other elements. Other elements that are used by the MovcmdSend CSU:
"MovcmdAdd" - information on the Movcmd address selection menu
"reply flag" - boolean; is this a reply
"replyCEOI" - target name for reply call
"When" - information on the When selection menu
"Task" - information on the Task selection menu
"reuseFlag" - boolean; is this a reuse and include

4.5.10.4.2.8 Logic flow. Single pass

4.5.10.4.2.9 Data structures. None

4.5.10.4.2.10 Local data files or database. None

4.5.10.4.2.11 Limitations. None

4.5.10.5 CSU MovcmdSndUrgCB design specification/constraints.

The MovcmdSndUrgCB CSU handles the callback from the Send Urgent pushbutton. It calls MovcmdSend with priority URGENT.

4.5.10.5.1 CSU MovcmdSndUrgCB requirementss.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.10.5.2 CSU MovcmdSndUrgCB design.

The information identified below represents the detailed design of the MovcmdSndUrgCB CSU.

4.5.10.5.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: None

4.5.10.5.2.2 Local data elements. None

4.5.10.5.2.3 Interrupts and signals. None

4.5.10.5.2.4 Algorithms. None

4.5.10.5.2.5 Error handling. None

4.5.10.5.2.6 Data conversion. None

4.5.10.5.2.7 Use of other elements. Other elements that are used by the MovcmdSndUrgCB CSU:
"PRIORITY_URGENT" - define for urgent priority

4.5.10.5.2.8 Logic flow. Single pass

4.5.10.5.2.9 Data structures. None

4.5.10.5.2.10 Local data files or database. None

4.5.10.5.2.11 Limitations. None

4.5.10.6 CSU ClearMovcmd

The ClearMovcmd CSU clears all the entries of the Movcmd report generation screen.

4.5.10.6.1 CSU ClearMovcmd requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.10.6.2 CSU ClearMovcmd design.

The information identified below represents the detailed design of the ClearMovcmd CSU.

4.5.10.6.2.1 Input/output data elements.

INPUTS:

OUTPUTS: None

4.5.10.6.2.2 Local data elements.
"argcnt" keeps track of the number of arguments set for XtSetValues
"args" the arguments set for XtSetValues
"argok" a boolean which is used to check the success of the CONVERT call
"xmstr" local to hold a XmString
"nullChar" null char

4.5.10.6.2.3 Interrupts and signals. None

4.5.10.6.2.4 Algorithms. None

4.5.10.6.2.5 Error handling. None

4.5.10.6.2.6 Data conversion. None

4.5.10.6.2.7 Use of other elements. Other elements that are used by the ClearMovcmd CSU:
"MovcmdAdd" - info on the selection menu for the address
"EnemyType" - info on the selection menu for the enemy type
"EnemyActivity" - info on the selection menu for the enemy activity
"Direc" - info on the selection menu for the direction
"ObsInt" - info on the selection menu for the observer intentions
"SpeedText" - widget id for speed text
"NumText" - widget id for number text
"UnitText" - widget id for unit text
"cikText1"- widget id for cik1 text
"cikText2"- widget id for cik2 text

4.5.10.6.2.8 Logic flow. Single pass

4.5.10.6.2.9 Data structures. None

4.5.10.6.2.10 Local data files or database. None

4.5.10.6.2.11 Limitations. None

4.5.11 Sub-Level CSC movcmd_call.

The following paragraphs under this section describe the relationship of the movcmd_call Sub-Level CSC in terms of data flow between the CSU's of this Sub-level CSC.

4.5.11.1 CSU lctn_moreCB

The lctn_moreCB CSU handles the callback for the lctn More pushbutton. It displays additional selections if they exist.

4.5.11.1.1 CSU lctn_moreCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.11.1.2 CSU lctn_moreCB design.

The information identified below represents the detailed design of the lctn_moreCB CSU.

4.5.11.1.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: None

4.5.11.1.2.2 Local data elements.
"argcnt" keeps track of the number of arguments set for XtSetValues
"args" the arguments set for XtSetValues
"argok" a boolean which is used to check the success of the CONVERT call
"xmstr" local to hold a XmString

4.5.11.1.2.3 Interrupts and signals. None

4.5.11.1.2.4 Algorithms. None

4.5.11.1.2.5 Error handling. None

4.5.11.1.2.6 Data conversion. None

4.5.11.1.2.7 Use of other elements. Other elements that are used by the lctn_moreCB CSU:
"Lctn" - info on Enemy Activity selection menu

4.5.11.1.2.8 Logic flow. Single pass

4.5.11.1.2.9 Data structures. None

4.5.11.1.2.10 Local data files or database. None

4.5.11.1.2.11 Limitations. None

4.5.11.2 CSU lctn_rotateCB

The lctn_rotateCB CSU handles the callback for the lctn Rotate pushbutton. It changes the currently selected entry to the next entry (the first entry if it is at the end already).

4.5.11.2.1 CSU lctn_rotateCB requirements

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.11.2.2 CSU lctn_rotateCB design.

The information identified below represents the detailed design of the lctn_rotateCB CSU.

4.5.11.2.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: .

4.5.11.2.2.2 Local data elements. None

4.5.11.2.2.3 Interrupts and signals. None

4.5.11.2.2.4 Algorithms. None

4.5.11.2.2.5 Error handling. None

4.5.11.2.2.6 Data conversion. None

4.5.11.2.2.7 Use of other elements. Other elements that are used by the lctn_rotateCB CSU:
"Lctn" - info on the enemy activity selection menu

4.5.11.2.2.8 Logic flow. Single pass

4.5.11.2.2.9 Data structures. None

4.5.11.2.2.10 Local data files or database. None

4.5.11.2.2.11 Limitations. None

4.5.11.3 CSU task_moreCB

The task_moreCB CSU handles the callback for the task More pushbutton. It displays additional selections if they exist.

4.5.11.3.1 CSU task_moreCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.11.3.2 CSU task_moreCB design.

The information identified below represents the detailed design of the task_moreCB CSU.

4.5.11.3.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: None

4.5.11.3.2.2 Local data elements.
"argcnt" keeps track of the number of arguments set for XtSetValues
"args" the arguments set for XtSetValues
"argok" a boolean which is used to check the success of the CONVERT call
"xmstr" local to hold a XmString

4.5.11.3.2.3 Interrupts and signals. None

4.5.11.3.2.4 Algorithms. None

4.5.11.3.2.5 Error handling. None

4.5.11.3.2.6 Data conversion. None

4.5.11.3.2.7 Use of other elements. Other elements that are used by the task_moreCB CSU:
"Task" - info on Enemy Activity selection menu

4.5.11.3.2.8 Logic flow. Single pass

4.5.11.3.2.9 Data structures. None

4.5.11.3.2.10 Local data files or database. None

4.5.11.3.2.11 Limitations. None

4.5.11.4 CSU task_rotateCB

The task_rotateCB CSU handles the callback for the task Rotate pushbutton. It changes the currently selected entry to the next entry (the first entry if it is at the end already).

4.5.11.4.1 CSU task_rotateCB requirements

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.11.4.2 CSU task_rotateCB design.

The information identified below represents the detailed design of the task_rotateCB CSU.

4.5.11.4.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: .

4.5.11.4.2.2 Local data elements. None

4.5.11.4.2.3 Interrupts and signals. None

4.5.11.4.2.4 Algorithms. None

4.5.11.4.2.5 Error handling. None

4.5.11.4.2.6 Data conversion. None

4.5.11.4.2.7 Use of other elements. Other elements that are used by the task_rotateCB CSU:
"Task" - info on the enemy activity selection menu

4.5.11.4.2.8 Logic flow. Single pass

4.5.11.4.2.9 Data structures. None

4.5.11.4.2.10 Local data files or database. None

4.5.11.4.2.11 Limitations. None

4.5.11.5 CSU when_rotateCB

The when_rotateCB CSU handles the callback for the when Rotate pushbutton. It changes the currently selected entry to the next entry (the first entry if it is at the end already).

4.5.11.5.1 CSU when_rotateCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.11.5.2 CSU when_rotateCB design.

The information identified below represents the detailed design of the when_rotateCB CSU.

4.5.11.5.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: .

4.5.11.5.2.2 Local data elements. None

4.5.11.5.2.3 Interrupts and signals. None

4.5.11.5.2.4 Algorithms. None

4.5.11.5.2.5 Error handling. None

4.5.11.5.2.6 Data conversion. None

4.5.11.5.2.7 Use of other elements. Other elements that are used by the when_rotateCB CSU:
"When" - info on the enemy activity selection menu

4.5.11.5.2.8 Logic flow. Single pass

4.5.11.5.2.9 Data structures. None

4.5.11.5.2.10 Local data files or database. None

4.5.11.5.2.11 Limitations. None

4.5.11.6 CSU movcmdad_moreCB

The movcmdad_moreCB CSU handles the callback for the movcmdad More pushbutton. It displays additional selections if they exist.

4.5.11.6.1 CSU movcmdad_moreCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.11.6.2 CSU movcmdad_moreCB design.

The information identified below represents the detailed design of the movcmdad_moreCB CSU.

4.5.11.6.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: .

4.5.11.6.2.2 Local data elements. None

4.5.11.6.2.3 Interrupts and signals. None

4.5.11.6.2.4 Algorithms. None

4.5.11.6.2.5 Error handling. None

4.5.11.6.2.6 Data conversion. None

4.5.11.6.2.7 Use of other elements. Other elements that are used by the movcmdad_moreCB CSU:
"MovcmdAdd" - info on the Movcmd address selection menu

4.5.11.6.2.8 Logic flow. Single pass

4.5.11.6.2.9 Data structures. None

4.5.11.6.2.10 Local data files or database. None

4.5.11.6.2.11 Limitations. None

4.5.11.7 CSU movcmdad_rotateCB

The movcmdad_rotateCB CSU handles the callback for the movcmdad Rotate pushbutton. It changes the currently selected entry to the next entry (the first entry if it is at the end already).

4.5.11.7.1 CSU movcmdad_rotateCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.11.7.2 CSU movcmdad_rotateCB design.

The information identified below represents the detailed design of the movcmdad_rotateCB CSU.

4.5.11.7.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: .

4.5.11.7.2.2 Local data elements. None

4.5.11.7.2.3 Interrupts and signals. None

4.5.11.7.2.4 Algorithms. None

4.5.11.7.2.5 Error handling. None

4.5.11.7.2.6 Data conversion. None

4.5.11.7.2.7 Use of other elements. Other elements that are used by the movcmdad_rotateCB CSU:
"MovcmdAdd" - info on Movcmd address selection menu

4.5.11.7.2.8 Logic flow. Single pass

4.5.11.7.2.9 Data structures. None

4.5.11.7.2.10 Local data files or database. None

4.5.11.7.2.11 Limitations. None

4.5.11.8 CSU PutupMovcmd

The PutupMovcmd CSU will create the Movcmd report screen if necessary and will also manage the screen.

4.5.11.8.1 CSU PutupMovcmd requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.11.8.2 CSU PutupMovcmd design.

The information identified below represents the detailed design of the PutupMovcmd CSU.

4.5.11.8.2.1 Input/output data elements.

INPUTS: None

OUTPUTS: None

4.5.11.8.2.2 Local data elements.
"argcnt" keeps track of the number of arguments set for XtSetValues
"args" the arguments set for XtSetValues
"argok" a boolean which is used to check the success of the CONVERT call
"xmstr" local to hold a XmString
"entries" - number of addresses

4.5.11.8.2.3 Interrupts and signals. None

4.5.11.8.2.4 Algorithms. None

4.5.11.8.2.5 Error handling. None

4.5.11.8.2.6 Data conversion. None

4.5.11.8.2.7 Use of other elements. Other elements that are used by the PutupMovcmd CSU:
"MovcmdForm" - widget id of movcmd form
"MovcmdAdd" - info on Movcmd address selection menu
"addressList" - list of addresses
"MovcmdMoreLabel" - widget id of More label
"MovcmdMoreArrow" - widget id of More arrow
"replyFlag" - boolean; is this a reply
"MovcmdTmiForm" widget id of movcmd TMI form

4.5.11.8.2.8 Logic flow. Single pass

4.5.11.8.2.9 Data structures. None

4.5.11.8.2.10 Local data files or database. None

4.5.11.8.2.11 Limitations. None

4.5.12 Sub-Level CSC msg1CB.

The following paragraphs under this section describe the relationship of the msg1CB Sub-Level CSC in terms of data flow between the CSU's of this Sub-level CSC.

4.5.12.1 CSU MsgsPrevCB

The MsgsPrevCB CSU handles the callback from the previous pushbutton. The previous message in the list is highlighted.

4.5.12.1.1 CSU MsgsPrevCB requirements.

The MsgsPrevCB CSU satisfies section 3.2.1.2.2.3.1.1 of the specific requirments of the DMCC system.

4.5.12.1.2 CSU MsgsPrevCB design.

The information identified below represents the detailed design of the MsgsPrevCB CSU.

4.5.12.1.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: None

4.5.12.1.2.2 Local data elements.
"pos" - position in list highlighted
"count" - number of items highlighted

4.5.12.1.2.3 Interrupts and signals. None

4.5.12.1.2.4 Algorithms. None

4.5.12.1.2.5 Error handling. None

4.5.12.1.2.6 Data conversion. None

4.5.12.1.2.7 Use of other elements. Other elements that are used by the MsgsPrevCB CSU:
"msg1List" - widget id of the msg1 list

4.5.12.1.2.8 Logic flow. Single pass.

4.5.12.1.2.9 Data structures. None

4.5.12.1.2.10 Local data files or database. None

4.5.12.1.2.11 Limitations. None

4.5.12.2 CSU MsgsNextCB

The MsgsNextCB CSU handles the callback from the next pushbutton. The next message in the list is highlighted.

4.5.12.2.1 CSU MsgsNextCB requirements

The MsgsNextCB CSU satisfies section 3.2.1.2.2.3.1.1 of the specific requirments of the DMCC system.

4.5.12.2.2 CSU MsgsNextCB design.

The information identified below represents the detailed design of the MsgsNextCB CSU.

4.5.12.2.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed

"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: None

4.5.12.2.2.2 Local data elements.
"pos" - position in list highlighted
"count" - number of items highlighted
"numItems" - number of items in the list

4.5.12.2.2.3 Interrupts and signals. None

4.5.12.2.2.4 Algorithms. None

4.5.12.2.2.5 Error handling. None

4.5.12.2.2.6 Data conversion. None

4.5.12.2.2.7 Use of other elements. Other elements that are used by the MsgsNextCB CSU:
"msg1List" - widget id of the msg1 list

4.5.12.2.2.8 Logic flow. Single pass.

4.5.12.2.2.9 Data structures. None

4.5.12.2.2.10 Local data files or database. None

4.5.12.2.2.11 Limitations. None

4.5.12.3 CSU MsgsReadCB

The MsgsReadCB CSU handles the callback for the Read pushbutton. It brings up a formatted version of the currently higlighted pdu.

4.5.12.3.1 CSU MsgsReadCB requirements

The MsgsReadCB CSU satisfies section 3.2.1.2.2.2.3 of the specific requirments of the DMCC system.

4.5.12.3.2 CSU MsgsReadCB design.

The information identified below represents the detailed design of the MsgsReadCB CSU.

4.5.12.3.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: None

4.5.12.3.2.2 Local data elements.
"pos" - position in list highlighted
"count" - number of items highlighted
"msgBuf" - message buffer

4.5.12.3.2.3 Interrupts and signals. None

4.5.12.3.2.4 Algorithms. None

4.5.12.3.2.5 Error handling. None

4.5.12.3.2.6 Data conversion. None

4.5.12.3.2.7 Use of other elements. Other elements that are used by the MsgsReadCB CSU:
"msg1List" - widget id of the msg1 list
"msg1Tmi" - widget id of the TMI of msg1
"msg1Form" - widget id of the entery aread of msg1
"messages" - queue with messages
"msgListIndex" - array of mappings of msg1 list entries to messages entries
"numUnreadMsgs" - number of unread messages in the queue
"msgReadText" - widget id of read screen text

4.5.12.3.2.8 Logic flow. Single pass.

4.5.12.3.2.9 Data structures. None

4.5.12.3.2.10 Local data files or database. None

4.5.12.3.2.11 Limitations. None

4.5.12.4 CSU MsgsDeleteCB

The MsgsDeleteCB CSU handles the callback for the Delete pushbutton. It deletes the currently higlighted pdu.

4.5.12.4.1 CSU MsgsDeleteCB requirements.

The MsgsDeleteCB CSU satisfies section 3.2.1.2.2.2.3 and 3.2.1.2.2.3.5.1 of the specific requirments of the DMCC system.

4.5.12.4.2 CSU MsgsDeleteCB design.

The information identified below represents the detailed design of the MsgsDeleteCB CSU.

4.5.12.4.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: None

4.5.12.4.2.2 Local data elements.
"pos" - position in list highlighted
"count" - number of items highlighted
"numItems" - number of items in the list
"i" - counter

4.5.12.4.2.3 Interrupts and signals. None

4.5.12.4.2.4 Algorithms. None

4.5.12.4.2.5 Error handling. None

4.5.12.4.2.6 Data conversion. None

4.5.12.4.2.7 Use of other elements. Other elements that are used by the MsgsDeleteCB CSU:
"msg1List" - widget id of the msg1 list
"numUnreadMsgs" - number of unread messages in the queue
"messages" - global queue containing messages
"msgListIndex" - array of mappings of msg1 list entries to messages entries
"numMessages" - number of messages in the queue

4.5.12.4.2.8 Logic flow. Single pass.

4.5.12.4.2.9 Data structures. None

4.5.12.4.2.10 Local data files or database. None

4.5.12.4.2.11 Limitations. None

4.5.12.5 CSU FormatMessageInfo

The FormatMessageInfo CSU takes information about the message and creates the header string for the Msg1 window.

4.5.12.5.1 CSU FormatMessageInfo requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.12.5.2 CSU FormatMessageInfo design.

The information identified below represents the detailed design of the FormatMessageInfo CSU.

4.5.12.5.2.1 Input/output data elements.

INPUTS:
"type" - type of message
"sender" - sender of message
"timeHour" - hour message sent
"timeMin" - minute message sent
"pri" - priority of message

OUTPUTS:
"info" - formatted string of message information

4.5.12.5.2.2 Local data elements.
"info" - formatted string of message information
"priChar" - character containing the priority of the message

4.5.12.5.2.3 Interrupts and signals. None

4.5.12.5.2.4 Algorithms. None

4.5.12.5.2.5 Error handling. None

4.5.12.5.2.6 Data conversion. None

4.5.12.5.2.7 Use of other elements. Other elements that are used by the FormatMessageInfo CSU:
None

4.5.12.5.2.8 Logic flow. Single pass.

4.5.12.5.2.9 Data structures. None

4.5.12.5.2.10 Local data files or database. None

4.5.12.5.2.11 Limitations. None

4.5.12.6 CSU PutupMsg1

The PutupMsg1 CSU will create the Msg1 screen if necessary and will also manage the screen.

4.5.12.6.1 CSU MsgsPrevCB requirements

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.12.6.2 CSU MsgsPrevCB design.

The information identified below represents the detailed design of the MsgsPrevCB CSU.

4.5.12.6.2.1 Input/output data elements.

INPUTS: None

OUTPUTS: None

4.5.12.6.2.2 Local data elements. None

4.5.12.6.2.3 Interrupts and signals. None

4.5.12.6.2.4 Algorithms. None

4.5.12.6.2.5 Error handling. None

4.5.12.6.2.6 Data conversion. None

4.5.12.6.2.7 Use of other elements. Other elements that are used by the MsgsPrevCB CSU:
"msg1List" - widget id of the msg1 list
"msg1Tmi" - widget id of the TMI of msg1
"msg1Form" - widget id of the entery aread of msg1
"smdForm" - widget id of the currently displayed entry form

4.5.12.6.2.8 Logic flow. Single pass.

4.5.12.6.2.9 Data structures. None

4.5.12.6.2.10 Local data files or database None

4.5.12.6.2.11 Limitations. None

4.5.12.7 CSU TestPriority

The TestPriority CSU will take two messages and determine which one has higher priority. Priority is determined by not only the priority field, but also the timestamp of the message.

4.5.12.7.1 CSU TestPriority requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.12.7.2 CSU TestPriority design.

The information identified below represents the detailed design of the TestPriority CSU.

4.5.12.7.2.1 Input/output data elements.

INPUTS:
"index1" - first index to be tested
"index2" - second index to be tested

OUTPUTS:
"result" - result of the test

4.5.12.7.2.2 Local data elements.
"prio1" - priority of index 1

"prio2" - priority of index 2
"ind1" - index of index 1
"ind2" - index of index 2
"result" - result of the test

4.5.12.7.2.3 Interrupts and signals. None

4.5.12.7.2.4 Algorithms. None

4.5.12.7.2.5 Error handling. None

4.5.12.7.2.6 Data conversion. None

4.5.12.7.2.7 Use of other elements. Other elements that are used by the TestPriority CSU:
"messages" - global queue containing messages

4.5.12.7.2.8 Logic flow. Single pass.

4.5.12.7.2.9 Data structures. None

4.5.12.7.2.10 Local data files or database. None

4.5.12.7.2.11 Limitations. None

4.5.12.8 CSU UpdateList

The UpdateList CSU will update the Msg1 list with any new messages and order the messages appropriately.

4.5.12.8.1 CSU UpdateList requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.12.8.2 CSU UpdateList design.

The information identified below represents the detailed design of the UpdateList CSU.

4.5.12.8.2.1 Input/output data elements.

INPUTS: None

OUTPUTS: None

4.5.12.8.2.2 Local data elements.
"argcnt" - keeps track of the number of arguments set for XtSetValues
"args" - the arguments set for XtSetValues
"argok" - a boolean which is used to check the success of the CONVERT call
"xmstr" - local to hold a XmString
"i" - counter
"listEntery" - temp string carrying the msg1 display info
"swap" - boolean for the sort
"tmp" - a temp variable

4.5.12.8.2.3 Interrupts and signals. None

4.5.12.8.2.4 Algorithms. None

4.5.12.8.2.5 Error handling. None

4.5.12.8.2.6 Data conversion. None

4.5.12.8.2.7 Use of other elements. Other elements that are used by the UpdateList CSU:
"numMessages" - number of messages currently in the message queue
"msgListInfo" - array containing the mail box headers for all the messages
"msgListIndex" - array of mappings of msg1 list entries to messages entries

4.5.12.8.2.8 Logic flow. Single pass.

4.5.12.8.2.9 Data structures. None

4.5.12.8.2.10 Local data files or database. None

4.5.12.8.2.11 Limitations. None

4.5.12.9 CSU MsgsReuseCB

The MsgsReuseCB CSU handles the callback for the Reuse pushbutton. It saves the currently highlighted PDU in a global and thing brings up the reuse report screen.

4.5.12.9.1 CSU MsgsReuseCB requirements.

The MsgsReuseCB CSU satisfies section 3.2.1.2.2.2.3 and 3.2.1.2.2.3.2.2 of the specific requirments of the DMCC system.

4.5.12.9.2 CSU MsgsReuseCB design.

The information identified below represents the detailed design of the MsgsReuseCB CSU.

4.5.12.9.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: None

4.5.12.9.2.2 Local data elements.
"pos" - position in list highlighted
"count" - number of items highlighted

4.5.12.9.2.3 Interrupts and signals. None

4.5.12.9.2.4 Algorithms. None

4.5.12.9.2.5 Error handling. None

4.5.12.9.2.6 Data conversion. None

4.5.12.9.2.7 Use of other elements. Other elements that are used by the MsgsReuseCB CSU:
"msg1List" - widget id of the msg1 list
"reusePDU" - pdu to be forwarded
"msg1Tmi" - widget id of TMI area of msg1 form
"msg1Form" - widget id of entry area of msg1 form
"msgListIndex" - array of mappings of msg1 list entries to messages entries

4.5.12.9.2.8 Logic flow. Single pass.

4.5.12.9.2.9 Data structures. None

4.5.12.9.2.10 Local data files or database. None

4.5.12.9.2.11 Limitations. None

4.5.12.10 CSU MsgsReuseNIncludeCB

The MsgsReuseNIncludeCB CSU handles the callback for the Reuse and Include pushbutton. It saves the currently highlighted PDU in a global and thing brings up the report screen so that a new report can be generated for inclusion with the reused message.

4.5.12.10.1 CSU MsgsReuseNIncludeCB requirements.

The MsgsReuseNIncludeCB CSU satisfies section 3.2.1.2.2.2.3 and 3.2.1.2.2.3.2.3 of the specific requirments of the DMCC system.

4.5.12.10.2 CSU MsgsReuseNIncludeCB design.

The information identified below represents the detailed design of the MsgsReuseNIncludeCB CSU.

4.5.12.10.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: None

4.5.12.10.2.2 Local data elements.
"pos" - position in list highlighted
"count" - number of items highlighted

4.5.12.10.2.3 Interrupts and signals. None

4.5.12.10.2.4 Algorithms. None

4.5.12.10.2.5 Error handling. None

4.5.12.10.2.6 Data conversion. None

4.5.12.10.2.7 Use of other elements. Other elements that are used by the MsgsReuseNIncludeCB CSU:
"msg1List" - widget id of the msg1 list
"reusePDU" - pdu to be forwarded
"msg1Tmi" - widget id of TMI area of msg1 form

"msg1Form" - widget id of entry area of msg1 form
"msgListIndex" - array of mappings of msg1 list entries to messages entries
"reuseFlag" - are we including a message

4.5.12.10.2.8 Logic flow. Single pass.

4.5.12.10.2.9 Data structures. None

4.5.12.10.2.10 Local data files or database. None

4.5.12.10.2.11 Limitations. None

4.5.12.11 CSU MsgsSaECB.

The MsgsSaECB CSU handles the callback from the Save and Exit pushbutton. It leaves the Msgs1 screen and returns to the SysMain screen.

4.5.12.11.1 CSU MsgsSaECB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.12.11.2 CSU MsgsSaECB design.

The information identified below represents the detailed design of the MsgsSaECB CSU.

4.5.12.11.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: None

4.5.12.11.2.2 Local data elements.
"pos" - position in list highlighted
"count" - number of items highlighted

4.5.12.11.2.3 Interrupts and signals. None

4.5.12.11.2.4 Algorithms. None

4.5.12.11.2.5 Error handling. None

4.5.12.11.2.6 Data conversion. None

4.5.12.11.2.7 Use of other elements. Other elements that are used by the MsgsSaECB CSU:
"msg1List" - widget id of the msg1 list
"msg1Tmi" - widget id of TMI area of msg1 form
"msg1Form" - widget id of entry area of msg1 form

4.5.12.11.2.8 Logic flow. Single pass.

4.5.12.11.2.9 Data structures. None

4.5.12.11.2.10 Local data files or database. None

4.5.12.11.2.11 Limitations. None

4.5.12.12 CSU MsgsReplyCB

The MsgsReplyCB callback handles the callback for the Reply pushbutton. It stores the CEOI of the sender of the higlighted message and then brings up the report screen for a message to be generated to the sender of said message.

4.5.12.12.1 CSU MsgsReplyCB requirements.

The MsgsReplyCB CSU satisfies section 3.2.1.2.2.2.3 of the specific requirments of the DMCC system.

4.5.12.12.2 CSU MsgsReplyCB design.

The information identified below represents the detailed design of the MsgsReplyCB CSU.

4.5.12.12.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: None

4.5.12.12.2.2 Local data elements.
"pos" - position in list highlighted
"count" - number of items highlighted

4.5.12.12.2.3 Interrupts and signals. None

4.5.12.12.2.4 Algorithms. None

4.5.12.12.2.5 Error handling. None

4.5.12.12.2.6 Data conversion. None

4.5.12.12.2.7 Use of other elements. Other elements that are used by the MsgsReplyCB CSU:
"msg1List" - widget id of the msg1 list
"replyFlag" - is this a reply
"replyCEOI" - reply return address
"msg1Tmi" - widget id of TMI area of msg1 form
"msg1Form" - widget id of entry area of msg1 form

4.5.12.12.2.8 Logic flow. Single pass.

4.5.12.12.2.9 Data structures. None

4.5.12.12.2.10 Local data files or database. None

4.5.12.12.2.11 Limitations. None

4.5.12.13 CSU MsgsReportCB

The MsgsReportCB CSU handles the callback for the Report pushbutton. It unmanages the Msgs1 screen and brings up the Reports screen.

4.5.12.13.1 CSU MsgsReportCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.12.13.2 CSU MsgsReportCB design.

The information identified below represents the detailed design of the MsgsReportCB CSU.

4.5.12.13.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: None

4.5.12.13.2.2 Local data elements. None

4.5.12.13.2.3 Interrupts and signals. None

4.5.12.13.2.4 Algorithms. None

4.5.12.13.2.5 Error handling. None

4.5.12.13.2.6 Data conversion. None

4.5.12.13.2.7 Use of other elements. Other elements that are used by the MsgsReportCB CSU:
"msg1Tmi" - widget id of TMI area of msg1 form
"msg1Form" - widget id of entry area of msg1 form

4.5.12.13.2.8 Logic flow. Single pass.

4.5.12.13.2.9 Data structures. None

4.5.12.13.2.10 Local data files or database. None

4.5.12.13.2.11 Limitations. None

4.5.13 Sub-Level CSC msgReadCB.

The following paragraphs under this section describe the relationship of the msgReadCB Sub-Level CSC in terms of data flow between the CSU's of this Sub-level CSC.

4.5.13.1 CSU MsgReadReadCB

The MsgReadReadCB CSU handles the callback for the Read pushbutton. It brings up the Msg1 screen.

4.5.13.1.1 CSU MsgReadReadCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.13.1.2 CSU MsgReadReadCB design.

The information identified below represents the detailed design of the MsgReadReadCB CSU.

4.5.13.1.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: None

4.5.13.1.2.2 Local data elements.
"nullChar" - empty string

4.5.13.1.2.3 Interrupts and signals. None

4.5.13.1.2.4 Algorithms. None

4.5.13.1.2.5 Error handling. None

4.5.13.1.2.6 Data conversion. None

4.5.13.1.2.7 Use of other elements. Other elements that are used by the MsgReadReadCB CSU:
"msgReadTmi" - widget id of TMI area of msgRead form
"msgReadForm" - widget id of entry area of msgRead form
"cikText1" - widget id of CIK text 1

4.5.13.1.2.8 Logic flow. Single pass.

4.5.13.1.2.9 Data structures. None

4.5.13.1.2.10 Local data files or database. None

4.5.13.1.2.11 Limitations. None

4.5.13.2 CSU MsgReadDeleteCB

The MsgReadDeleteCB CSU handles the callback for the Delete button. It delete the message currently being displayed and returns to the Msgs1 screen.

4.5.13.2.1 CSU MsgReadDeleteCB requirements.

The MsgReadDeleteCB CSU satisfies section 3.2.1.2.2.3.1.2, 3.2.1.2.2.3.5.1 and 3.2.1.2.2.3.5.2 of the specific requirments of the DMCC system.

4.5.13.2.2 CSU MsgReadDeleteCB design.

The information identified below represents the detailed design of the MsgReadDeleteCB CSU.

4.5.13.2.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: None

4.5.13.2.2.2 Local data elements.

4.5.13.2.2.3 Interrupts and signals. None

4.5.13.2.2.4 Algorithms. None

4.5.13.2.2.5 Error handling. None

4.5.13.2.2.6 Data conversion. None

4.5.13.2.2.7 Use of other elements. Other elements that are used by the MsgReadDeleteCB CSU:
None

4.5.13.2.2.8 Logic flow. Single pass.

4.5.13.2.2.9 Data structures. None

4.5.13.2.2.10 Local data files or database. None

4.5.13.2.2.11 Limitations. None

4.5.13.3 CSU PutupMsgRead .

The PutupMsgRead CSU will create the MsgRead screen if necessary and will also manage the screen.

4.5.13.3.1 CSU PutupMsgRead requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.13.3.2 CSU PutupMsgRead design.

The information identified below represents the detailed design of the PutupMsgRead CSU.

4.5.13.3.2.1 Input/output data elements.

INPUTS: None

OUTPUTS: None

4.5.13.3.2.2 Local data elements.
"argcnt" - keeps track of the number of arguments set for XtSetValues
"args" - the arguments set for XtSetValues
"argok" - a boolean which is used to check the success of the CONVERT call
"xmstr" - local to hold a XmString

4.5.13.3.2.3 Interrupts and signals. None

4.5.13.3.2.4 Algorithms. None

4.5.13.3.2.5 Error handling. None

4.5.13.3.2.6 Data conversion. None

4.5.13.3.2.7 Use of other elements. Other elements that are used by the PutupMsgRead CSU:
"msgReadTmi" - widget id of TMI area of msgRead form
"msgReadForm" - widget id of entry area of msgRead form
"smdForm" - widget id of the smd
"tmiScreen" - widget id of current TMI

"smdScreen" - widget id of current entery screen

4.5.13.3.2.8 Logic flow. Single pass.

4.5.13.3.2.9 Data structures. None

4.5.13.3.2.10 Local data files or database. None

4.5.13.3.2.11 Limitations. None

4.5.13.4 CSU MsgReadReuseNIncCB.

The MsgsReuseNIncCB CSU handles the callback for the Reuse and Include pushbutton. It saves the currently displayed PDU in a global and then brings up the report screen so that a new report can be generated for inclusion with the reused message.

4.5.13.4.1 CSU MsgReadReuseNIncCB requirements.

The MsgReadReuseNIncCB CSU satisfies section 3.2.1.2.2.3.1.2 and 3.2.1.2.2.3.2.3 of the specific requirments of the DMCC system.

4.5.13.4.2 CSU MsgReadReuseNIncCB design.

The information identified below represents the detailed design of the MsgReadReuseNIncCB CSU.

4.5.13.4.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: None

4.5.13.4.2.2 Local data elements.
"pos" - position in list highlighted
"count" - number of items highlighted

4.5.13.4.2.3 Interrupts and signals. None

4.5.13.4.2.4 Algorithms. None

4.5.13.4.2.5 Error handling. None

4.5.13.4.2.6 Data conversion. None

4.5.13.4.2.7 Use of other elements. Other elements that are used by the MsgReadReuseNIncCB CSU:
"msgReadTmi" - widget id of TMI area of msgRead form
"msgReadForm" - widget id of entry area of msgRead form
"msg1List" - widget id of msg1 list
"reusePDU" - pdu to be forwarded
"reuseFlag" - is this a forwarded message
"msgListIndex" - array of mappings of msg1 list entries to messages entries

4.5.13.4.2.8 Logic flow. Single pass.

4.5.13.4.2.9 Data structures. None

4.5.13.4.2.10 Local data files or database. None

4.5.13.4.2.11 Limitations. None

4.5.13.5 CSU MsgReadReuseCB

The MsgsReuseCB CSU handles the callback for the Reuse pushbutton. It saves the currently displayed PDU in a global and thing brings up the reuse report screen.

4.5.13.5.1 CSU MsgReadReuseCB requirements.

The MsgReadReuseCB CSU satisfies section 3.2.1.2.2.3.1.2 and 3.2.1.2.2.3.2.2 of the specific requirments of the DMCC system.

4.5.13.5.2 CSU MsgReadReuseCB design.

The information identified below represents the detailed design of the MsgReadReuseCB CSU.

4.5.13.5.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: None

4.5.13.5.2.2 Local data elements.
"pos" - position in list highlighted

"count" - number of items highlighted

4.5.13.5.2.3 Interrupts and signals. None

4.5.13.5.2.4 Algorithms. None

4.5.13.5.2.5 Error handling. None

4.5.13.5.2.6 Data conversion. None

4.5.13.5.2.7 Use of other elements. Other elements that are used by the MsgReadReuseCB CSU:
"msgReadTmi" - widget id of TMI area of msgRead form
"msgReadForm" - widget id of entry area of msgRead form
"msg1List" - widget id of msg1 list
"reusePDU" - pdu to be forwarded
"reuseFlag" - is this a forwarded message
"msgListIndex" - array of mappings of msg1 list entries to messages entries

4.5.13.5.2.8 Logic flow. Single pass.

4.5.13.5.2.9 Data structures. None

4.5.13.5.2.10 Local data files or database. None

4.5.13.5.2.11 Limitations. None

4.5.13.6 CSU MsgReplyCB

The MsgsReplyCB callback handles the callback for the Reply pushbutton. It stores the CEOI of the sender of the displayed message and then brings up the report screen for a message to be generated to the sender of said message.

4.5.13.6.1 CSU MsgReplyCB requirements.

The MsgReplyCB CSU satisfies section 3.2.1.2.2.3.1.2 of the specific requirments of the DMCC system.

4.5.13.6.2 CSU MsgReplyCB design.

The information identified below represents the detailed design of the MsgReplyCB CSU.

4.5.13.6.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: None

4.5.13.6.2.2 Local data elements.
"pos" - position in list highlighted
"count" - number of items highlighted

4.5.13.6.2.3 Interrupts and signals. None

4.5.13.6.2.4 Algorithms. None

4.5.13.6.2.5 Error handling. None

4.5.13.6.2.6 Data conversion. None

4.5.13.6.2.7 Use of other elements. Other elements that are used by the MsgReplyCB CSU:
"msgReadTmi" - widget id of TMI area of msgRead form
"msgReadForm" - widget id of entry area of msgRead form
"msg1List" - widget id of msg1 list
"replyFlag" - is this a reply
"replyCEOI" - reply return address

4.5.13.6.2.8 Logic flow. Single pass.

4.5.13.6.2.9 Data structures. None

4.5.13.6.2.10 Local data files or database. None

4.5.13.6.2.11 Limitations. None

4.5.14 Sub-Level CSC mtoTmi_call.

The following paragraphs under this section describe the relationship of the mtoTmi_call Sub-Level CSC in terms of data flow between the CSU's of this Sub-level CSC.

4.5.14.1 CSU MtoClearNRetCB

The MtoClearNRetCB CSU handles the callback from the Clear and Return pushbutton. It will clear and unmanage the Mto window and remanage the Reports window.

4.5.14.1.1 CSU MtoClearNRetCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.14.1.2 CSU MtoClearNRetCB design.

The information identified below represents the detailed design of the MtoClearNRetCB CSU.

4.5.14.1.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: None

4.5.14.1.2.2 Local data elements.
"argcnt" keeps track of the number of arguments set for XtSetValues
"args" the arguments set for XtSetValues
"argok" a boolean which is used to check the success of the CONVERT call
"xmstr" local to hold a XmString

4.5.14.1.2.3 Interrupts and signals. None

4.5.14.1.2.4 Algorithms. None

4.5.14.1.2.5 Error handling. None

4.5.14.1.2.6 Data conversion. None

4.5.14.1.2.7 Use of other elements. Other elements that are used by the MtoClearNRetCB CSU:
"MtoTmiForm" - form widget containing Mto TMI
"MtoForm" - form widget containing Mto Entry area

"cikLabel2" - label widget

4.5.14.1.2.8 Logic flow. Single pass

4.5.14.1.2.9 Data structures. None

4.5.14.1.2.10 Local data files or database. None

4.5.14.1.2.11 Limitations. None

4.5.14.2 CSU MtoSaveNRetCB

The MtoSaveNRetCB CSU handles the callback from the Save and Return pushbutton. It will unmanage the Mto window and remanage the Reports window.

4.5.14.2.1 CSU MtoSaveNRetCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.14.2.2 CSU MtoSaveNRetCB design.

The information identified below represents the detailed design of the MtoSaveNRetCB CSU.

4.5.14.2.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: None

4.5.14.2.2.2 Local data elements.
"argcnt" keeps track of the number of arguments set for XtSetValues
"args" the arguments set for XtSetValues
"argok" a boolean which is used to check the success of the CONVERT call
"xmstr" local to hold a XmString
"nullChar" null character

4.5.14.2.2.3 Interrupts and signals. None

4.5.14.2.2.4 Algorithms. None

4.5.14.2.2.5 Error handling. None

4.5.14.2.2.6 Data conversion. None

4.5.14.2.2.7 Use of other elements. Other elements that are used by the MtoSaveNRetCB CSU:
"MtoTmiForm" - form widget containing Mto TMI
"MtoForm" - form widget containing Mto Entry area
"cikLabel2" - label widget
"cikLabel1" - label widget

4.5.14.2.2.8 Logic flow. Single pass

4.5.14.2.2.9 Data structures. None

4.5.14.2.2.10 Local data files or database. None

4.5.14.2.2.11 Limitations. None

4.5.14.3 CSU MtoSndRoutCB

The MtoSndRoutCB CSU handles the callback from the Send Routine pushbutton. It calls MtoSend with priority ROUTINE.

4.5.14.3.1 CSU MtoSndRoutCB requirements

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.14.3.2 CSU MtoSndRoutCB design.

The information identified below represents the detailed design of the MtoSndRoutCB CSU.

4.5.14.3.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data

"call" contains X Windows data about the call reason

OUTPUTS: None

4.5.14.3.2.2 Local data elements. None

4.5.14.3.2.3 Interrupts and signals. None

4.5.14.3.2.4 Algorithms. None

4.5.14.3.2.5 Error handling. None

4.5.14.3.2.6 Data conversion. None

4.5.14.3.2.7 Use of other elements. Other elements that are used by the MtoSndRoutCB CSU:
"PRIORITY_ROUTINE" - define for priority message

4.5.14.3.2.8 Logic flow. Single pass

4.5.14.3.2.9 Data structures. None

4.5.14.3.2.10 Local data files or database. None

4.5.14.3.2.11 Limitations. None

4.5.14.4 CSU MtoSend

The MtoSend CSU sends a message with a given priority. It retrieves the data from the text entry area and the annotation area and retrieves the address from the address selection menu. It then sends the data to bldMto to be sent out. It also handles the case that the message is a reply by fixing the address to that from the message being replied to. If the message is a Reuse and Include, the included message will also be sent to the address selected.

4.5.14.4.1 CSU MtoSend requirements.

The MtoSend CSU satisfies section 3.2.1.2.2.2.1, 3.2.1.2.2.3.6.1 and 3.2.1.2.2.3.6.1 of the specific requirments of the DMCC system.

4.5.14.4.2 CSU MtoSend design.

The information identified below represents the detailed design of the MtoSend CSU.

4.5.14.4.2.1 Input/output data elements.

INPUTS:
"prio" - priority of the message being sent

OUTPUTS: None

4.5.14.4.2.2 Local data elements.
"targetCEOI" - name of user message is being sent to
"annotation" - text of annotation string
"MissionStatus" - boolean; is this a Mission Status message
"RequestAdjustment" - boolean; is this a Request Adjustment message
"EnterTarget" - boolean; is this a Enter Target message
"EndMission" - boolean; is this a End Mission message

4.5.14.4.2.3 Interrupts and signals. None

4.5.14.4.2.4 Algorithms. None

4.5.14.4.2.5 Error handling. None

4.5.14.4.2.6 Data conversion. None

4.5.14.4.2.7 Use of other elements. Other elements that are used by the MtoSend CSU:
"MtoAdd" - information on the Mto address selection menu
"reply flag" - boolean; is this a reply
"replyCEOI" - target name for reply call
"reuseFlag" - boolean; is this a reuse and include

4.5.14.4.2.8 Logic flow. Single pass

4.5.14.4.2.9 Data structures. None

4.5.14.4.2.10 Local data files or database. None

4.5.14.4.2.11 Limitations. None

4.5.14.5 CSU MtoSndUrgCB

The MtoSndUrgCB CSU handles the callback from the Send Urgent pushbutton. It calls MtoSend with priority URGENT.

4.5.14.5.1 CSU MtoSndUrgCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.14.5.2 CSU MtoSndUrgCB design.

The information identified below represents the detailed design of the MtoSndUrgCB CSU.

4.5.14.5.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: None

4.5.14.5.2.2 Local data elements. None

4.5.14.5.2.3 Interrupts and signals. None

4.5.14.5.2.4 Algorithms. None

4.5.14.5.2.5 Error handling. None

4.5.14.5.2.6 Data conversion. None

4.5.14.5.2.7 Use of other elements. Other elements that are used by the MtoSndUrgCB CSU:
"PRIORITY_URGENT" - define for urgent priority

4.5.14.5.2.8 Logic flow. Single pass

4.5.14.5.2.9 Data structures. None

4.5.14.5.2.10 Local data files or database. None

4.5.14.5.2.11 Limitations. None

4.5.14.6 CSU ClearMto

The ClearMto CSU clears all the entries of the Mto report generation screen.

4.5.14.6.1 CSU ClearMto requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.14.6.2 CSU ClearMto design.

The information identified below represents the detailed design of the ClearMto CSU.

4.5.14.6.2.1 Input/output data elements.

INPUTS:

OUTPUTS: None

4.5.14.6.2.2 Local data elements.
"argcnt" keeps track of the number of arguments set for XtSetValues
"args" the arguments set for XtSetValues
"argok" a boolean which is used to check the success of the CONVERT call
"xmstr" local to hold a XmString
"nullChar" null char

4.5.14.6.2.3 Interrupts and signals. None

4.5.14.6.2.4 Algorithms. None

4.5.14.6.2.5 Error handling. None

4.5.14.6.2.6 Data conversion. None

4.5.14.6.2.7 Use of other elements. Other elements that are used by the ClearMto CSU:
"MtoAdd" - info on the selection menu for the address
"EnemyType" - info on the selection menu for the enemy type
"EnemyActivity" - info on the selection menu for the enemy activity
"Direc" - info on the selection menu for the direction

"ObsInt" - info on the selection menu for the observer intentions
"SpeedText" - widget id for speed text
"NumText" - widget id for number text
"UnitText" - widget id for unit text
"cikText1"- widget id for cik1 text
"cikText2"- widget id for cik2 text

4.5.14.6.2.8 Logic flow. Single pass

4.5.14.6.2.9 Data structures. None

4.5.14.6.2.10 Local data files or database. None

4.5.14.6.2.11 Limitations. None

4.5.15 Sub-Level CSC mto_call.

The following paragraphs under this section describe the relationship of the mto_call Sub-Level CSC in terms of data flow between the CSU's of this Sub-level CSC.

4.5.15.1 CSU mtoad_moreCB

The mtoad_moreCB CSU handles the callback for the mtoad More pushbutton. It displays additional selectins if they exist.

4.5.15.1.1 CSU mtoad_moreCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.15.1.2 CSU mtoad_moreCB design.

The information identified below represents the detailed design of the mtoad_moreCB CSU.

4.5.15.1.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: .

4.5.15.1.2.2 Local data elements. None

4.5.15.1.2.3 Interrupts and signals. None

4.5.15.1.2.4 Algorithms. None

4.5.15.1.2.5 Error handling. None

4.5.15.1.2.6 Data conversion. None

4.5.15.1.2.7 Use of other elements. Other elements that are used by the mtoad_moreCB CSU:
"MtoAdd" - info on the Mto address selection menu

4.5.15.1.2.8 Logic flow. Single pass

4.5.15.1.2.9 Data structures. None

4.5.15.1.2.10 Local data files or database. None

4.5.15.1.2.11 Limitations. None

4.5.15.2 CSU mtoad_rotateCB

The mtoad_rotateCB CSU handles the callback for the mtoad Rotate pushbutton. It changes the currently selected entry to the next entry (the first entry if it is at the end already).

4.5.15.2.1 CSU mtoad_rotateCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.15.2.2 CSU mtoad_rotateCB design.

The information identified below represents the detailed design of the mtoad_rotateCB CSU.

4.5.15.2.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: .

4.5.15.2.2.2 Local data elements. None

4.5.15.2.2.3 Interrupts and signals. None

4.5.15.2.2.4 Algorithms. None

4.5.15.2.2.5 Error handling. None

4.5.15.2.2.6 Data conversion. None

4.5.15.2.2.7 Use of other elements. Other elements that are used by the mtoad_rotateCB CSU:
"MtoAdd" - info on Mto address selection menu

4.5.15.2.2.8 Logic flow. Single pass

4.5.15.2.2.9 Data structures. None

4.5.15.2.2.10 Local data files or database. None

4.5.15.2.2.11 Limitations. None

4.5.15.3 CSU PutupMto

The PutupMto CSU will create the Mto report screen if necessary and will also manage the screen.

4.5.15.3.1 CSU PutupMto requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.15.3.2 CSU PutupMto design.

The information identified below represents the detailed design of the PutupMto CSU.

4.5.15.3.2.1 Input/output data elements.

INPUTS: None

OUTPUTS: None

4.5.15.3.2.2 Local data elements.
"argcnt" keeps track of the number of arguments set for XtSetValues
"args" the arguments set for XtSetValues
"argok" a boolean which is used to check the success of the CONVERT call
"xmstr" local to hold a XmString
"entries" - number of addresses

4.5.15.3.2.3 Interrupts and signals. None

4.5.15.3.2.4 Algorithms. None

4.5.15.3.2.5 Error handling. None

4.5.15.3.2.6 Data conversion. None

4.5.15.3.2.7 Use of other elements. Other elements that are used by the PutupMto CSU:
"MtoForm" - widget id of mto form
"MtoAdd" - info on Mto address selection menu
"addressList" - list of addresses
"MtoMoreLabel" - widget id of More label
"MtoMoreArrow" - widget id of More arrow
"replyFlag" - boolean; is this a reply
"MtoTmiForm" widget id of mto TMI form

4.5.15.3.2.8 Logic flow. Single pass

4.5.15.3.2.9 Data structures. None

4.5.15.3.2.10 Local data files or database. None

4.5.15.3.2.11 Limitations. None

4.5.16 Sub-Level CSC popupMessage.

The following paragraphs under this section describe the relationship of the popupMessage Sub-Level CSC in terms of data flow between the CSU's of this Sub-level CSC.

4.5.16.1 CSU CreateMessageBox

The CreateMessageBox CSU creates a Popup message box with a given message and a given parent.

4.5.16.1.1 CSU CreateMessageBox requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.16.1.2 CSU CreateMessageBox design.

The information identified below represents the detailed design of the CreateMessageBox CSU.

4.5.16.1.2.1 Input/output data elements.

INPUTS:
"parent" - widget id of parent object
"messageString" - string to be displayed in the message box

OUTPUTS:
"retval" - widget id of the message box

4.5.16.1.2.2 Local data elements.
"argcnt" - keeps track of the number of arguments set for XtSetValues
"args" - the arguments set for XtSetValues
"argok" - a boolean which is used to check the success of the CONVERT call
"xmstr" - local to hold a XmString
"popupMessage" - widget id of message box

4.5.16.1.2.3 Interrupts and signals. None

4.5.16.1.2.4 Algorithms. None

4.5.16.1.2.5 Error handling. None

4.5.16.1.2.6 Data conversion. None

4.5.16.1.2.7 Use of other elements. Other elements that are used by the CreateMessageBox CSU:
None

4.5.16.1.2.8 Logic flow. Single pass.

4.5.16.1.2.9 Data structures. None

4.5.16.1.2.10 Local data files or database. None

4.5.16.1.2.11 Limitations. None

4.5.16.2 CSU createMessage

The createMessage CSU unmanages the Cancel and Help buttons of the Message Box created by CreateMessageBox.

4.5.16.2.1 CSU createMessage requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.16.2.2 CSU createMessage design.

The information identified below represents the detailed design of the createMessage CSU.

4.5.16.2.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: None

4.5.16.2.2.2 Local data elements. None

4.5.16.2.2.3 Interrupts and signals. None

4.5.16.2.2.4 Algorithms. None

4.5.16.2.2.5 Error handling. None

4.5.16.2.2.6 Data conversion. None

4.5.16.2.2.7 Use of other elements. Other elements that are used by the createMessage CSU:
None

4.5.16.2.2.8 Logic flow. Single pass.

4.5.16.2.2.9 Data structures. None

4.5.16.2.2.10 Local data files or database. None

4.5.16.2.2.11 Limitations. None

4.5.16.3 CSU okbutton

The okbutton CSU handles the callback for the OK button. It destroys the Message Box.

4.5.16.3.1 CSU okbutton requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.16.3.2 CSU okbutton design.

The information identified below represents the detailed design of the okbutton CSU.

4.5.16.3.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: None

4.5.16.3.2.2 Local data elements. None

4.5.16.3.2.3 Interrupts and signals. None

4.5.16.3.2.4 Algorithms. None

4.5.16.3.2.5 Error handling. None

4.5.16.3.2.6 Data conversion. None

4.5.16.3.2.7 Use of other elements. Other elements that are used by the okbutton CSU:
None

4.5.16.3.2.8 Logic flow. Single pass.

4.5.16.3.2.9 Data structures. None

4.5.16.3.2.10 Local data files or database. None

4.5.16.3.2.11 Limitations. None

4.5.16.4 CSU unmanageMessage

The unmanageMessage CSU handles the timer event set for three seconds and destorys the Message Box.

4.5.16.4.1 CSU unmanageMessage requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.16.4.2 CSU unmanageMessage design.

The information identified below represents the detailed design of the unmanageMessage CSU.

4.5.16.4.2.1 Input/output data elements.

INPUTS:
"data" - widget id of widget to be destroyed
"id" - calling data

OUTPUTS: None

4.5.16.4.2.2 Local data elements. None

4.5.16.4.2.3 Interrupts and signals. None

4.5.16.4.2.4 Algorithms. None

4.5.16.4.2.5 Error handling. None

4.5.16.4.2.6 Data conversion. None

4.5.16.4.2.7 Use of other elements. Other elements that are used by the unmanageMessage CSU:
None

4.5.16.4.2.8 Logic flow. Single pass.

4.5.16.4.2.9 Data structures. None

4.5.16.4.2.10 Local data files or database. None

4.5.16.4.2.11 Limitations. None

4.5.16.5 CSU popupTransientMessage

The popupTransientMessage CSU creates a Message Box which can either be dismissed or will destroy itself after 5 seconds.

4.5.16.5.1 CSU popupTransientMessage requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.16.5.2 CSU popupTransientMessage design.

The information identified below represents the detailed design of the popupTransientMessage CSU.

4.5.16.5.2.1 Input/output data elements.

INPUTS:
"parent" - widget id of parent object
"messageString" - string to be displayed in the message box

OUTPUTS: None

4.5.16.5.2.2 Local data elements.
"argcnt" - keeps track of the number of arguments set for XtSetValues
"args" - the arguments set for XtSetValues
"argok" - a boolean which is used to check the success of the CONVERT call
"xmstr" - local to hold a XmString

4.5.16.5.2.3 Interrupts and signals. None

4.5.16.5.2.4 Algorithms. None

4.5.16.5.2.5 Error handling. None

4.5.16.5.2.6 Data conversion. None

4.5.16.5.2.7 Use of other elements. Other elements that are used by the popupTransientMessage CSU:
None

4.5.16.5.2.8 Logic flow. Single pass.

4.5.16.5.2.9 Data structures. None

4.5.16.5.2.10 Local data files or database. None

4.5.16.5.2.11 Limitations. None

4.5.16.6 CSU popupMessageBox

The popupMessageBox CSU creates a Message Box which can be dismissed.

4.5.16.6.1 CSU popupMessageBox requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.16.6.2 CSU popupMessageBox design.

The information identified below represents the detailed design of the popupMessageBox CSU.

4.5.16.6.2.1 Input/output data elements.

INPUTS:
"parent" - widget id of parent object
"messageString" - string to be displayed in the message box

OUTPUTS: None

4.5.16.6.2.2 Local data elements.
"argcnt" - keeps track of the number of arguments set for XtSetValues
"args" - the arguments set for XtSetValues
"argok" - a boolean which is used to check the success of the CONVERT call
"xmstr" - local to hold a XmString

4.5.16.6.2.3 Interrupts and signals. None

4.5.16.6.2.4 Algorithms. None

4.5.16.6.2.5 Error handling. None

4.5.16.6.2.6 Data conversion. None

4.5.16.6.2.7 Use of other elements. Other elements that are used by the popupMessageBox CSU:
None

4.5.16.6.2.8 Logic flow. Single pass.

4.5.16.6.2.9 Data structures. None

4.5.16.6.2.10 Local data files or database. None

4.5.16.6.2.11 Limitations. None

4.5.17 Sub-Level CSC reportCB.

The following paragraphs under this section describe the relationship of the reportCB Sub-Level CSC in terms of data flow between the CSU's of this Sub-level CSC.

4.5.17.1 CSU ReportMtoCB

The ReportMtoCB CSU is the callback that brings up the MTO report screen for report entry.

4.5.17.1.1 CSU ReportMtoCB requirements.

The ReportMtoCB CSU satisfies section 3.2.1.2.2.3.2.2 of the specific requirments of the DMCC system.

4.5.17.1.2 CSU ReportMtoCB design.

The information identified below represents the detailed design of the ReportMtoCB CSU.

4.5.17.1.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: None

4.5.17.1.2.2 Local data elements. None

4.5.17.1.2.3 Interrupts and signals. None

4.5.17.1.2.4 Algorithms. None

4.5.17.1.2.5 Error handling. None

4.5.17.1.2.6 Data conversion. None

4.5.17.1.2.7 Use of other elements. Other elements that are used by the ReportMtoCB CSU:
"reportTmi" - widgit id for the report window TMI area
"reportForm" - widgit id for the report window entry area

4.5.17.1.2.8 Logic flow. Single pass.

4.5.17.1.2.9 Data structures. None

4.5.17.1.2.10 Local data files or database. None

4.5.17.1.2.11 Limitations. None

4.5.17.2 CSU ReportSplashCB

The ReportSplashCB CSU is the callback that brings up the Splash report screen for report entry.

4.5.17.2.1 CSU ReportSplashCB requirements.

The ReportSplashCB CSU satisfies section 3.2.1.2.2.3.2.2 of the specific requirments of the DMCC system.

4.5.17.2.2 CSU ReportSplashCB design.

The information identified below represents the detailed design of the ReportSplashCB CSU.

4.5.17.2.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: None

4.5.17.2.2.2 Local data elements. None

4.5.17.2.2.3 Interrupts and signals. None

4.5.17.2.2.4 Algorithms. None

4.5.17.2.2.5 Error handling. None

4.5.17.2.2.6 Data conversion. None

4.5.17.2.2.7 Use of other elements. Other elements that are used by the ReportSplashCB CSU:
"reportTmi" - widgit id for the report window TMI area
"reportForm" - widgit id for the report window entry area

4.5.17.2.2.8 Logic flow. Single pass.

4.5.17.2.2.9 Data structures. None

4.5.17.2.2.10 Local data files or database. None

4.5.17.2.2.11 Limitations. None

4.5.17.3 CSU ReportReqtCB

The ReportReqtCB CSU is the callback that brings up the Request report screen for report entry.

4.5.17.3.1 CSU ReportReqtCB requirements.

The ReportReqtCB CSU satisfies section 3.2.1.2.2.3.2.2 of the specific requirments of the DMCC system.

4.5.17.3.2 CSU ReportReqtCB design.

The information identified below represents the detailed design of the ReportReqtCB CSU.

4.5.17.3.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: None

4.5.17.3.2.2 Local data elements. None

4.5.17.3.2.3 Interrupts and signals. None

4.5.17.3.2.4 Algorithms. None

4.5.17.3.2.5 Error handling. None

4.5.17.3.2.6 Data conversion. None

4.5.17.3.2.7 Use of other elements. Other elements that are used by the ReportReqtCB CSU:
"reportTmi" - widgit id for the report window TMI area
"reportForm" - widgit id for the report window entry area

4.5.17.3.2.8 Logic flow. Single pass.

4.5.17.3.2.9 Data structures. None

4.5.17.3.2.10 Local data files or database. None

4.5.17.3.2.11 Limitations. None

4.5.17.4 CSU ReportMovcmdCB

The ReportMovcmdCB CSU is the callback that brings up the Movcmd report screen for report entry.

4.5.17.4.1 CSU ReportMovcmdCB requirements.

The ReportMovcmdCB CSU satisfies section 3.2.1.2.2.3.2.2 of the specific requirments of the DMCC system.

4.5.17.4.2 CSU ReportMovcmdCB design.

The information identified below represents the detailed design of the ReportMovcmdCB CSU.

4.5.17.4.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: None

4.5.17.4.2.2 Local data elements. None

4.5.17.4.2.3 Interrupts and signals. None

4.5.17.4.2.4 Algorithms. None

4.5.17.4.2.5 Error handling. None

4.5.17.4.2.6 Data conversion. None

4.5.17.4.2.7 Use of other elements. Other elements that are used by the ReportMovcmdCB CSU:
"reportTmi" - widgit id for the report window TMI area
"reportForm" - widgit id for the report window entry area

4.5.17.4.2.8 Logic flow. Single pass.

4.5.17.4.2.9 Data structures. None

4.5.17.4.2.10 Local data files or database. None

4.5.17.4.2.11 Limitations. None

4.5.17.5 CSU ReportShotCB

The ReportShotCB CSU is the callback that brings up the Shot report screen for report entry.

4.5.17.5.1 CSU ReportShotCB requirements.

The ReportShotCB CSU satisfies section 3.2.1.2.2.3.2.2 of the specific requirments of the DMCC system.

4.5.17.5.2 CSU ReportShotCB design.

The information identified below represents the detailed design of the ReportShotCB CSU.

4.5.17.5.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: None

4.5.17.5.2.2 Local data elements. None

4.5.17.5.2.3 Interrupts and signals. None

4.5.17.5.2.4 Algorithms. None

4.5.17.5.2.5 Error handling. None

4.5.17.5.2.6 Data conversion. None

4.5.17.5.2.7 Use of other elements. Other elements that are used by the ReportShotCB CSU:
"reportTmi" - widgit id for the report window TMI area
"reportForm" - widgit id for the report window entry area

4.5.17.5.2.8 Logic flow. Single pass.

4.5.17.5.2.9 Data structures. None

4.5.17.5.2.10 Local data files or database. None

4.5.17.5.2.11 Limitations. None

4.5.17.6 CSU ReportSpotCB

The ReportSpotCB CSU is the callback that brings up the Spot report screen for report entry.

4.5.17.6.1 CSU ReportSpotCB requirements.

The ReportSpotCB CSU satisfies section 3.2.1.2.2.3.2.2 of the specific requirments of the DMCC system.

4.5.17.6.2 CSU ReportSpotCB design.

The information identified below represents the detailed design of the ReportSpotCB CSU.

4.5.17.6.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: None

4.5.17.6.2.2 Local data elements. None

4.5.17.6.2.3 Interrupts and signals. None

4.5.17.6.2.4 Algorithms. None

4.5.17.6.2.5 Error handling. None

4.5.17.6.2.6 Data conversion. None

4.5.17.6.2.7 Use of other elements. Other elements that are used by the ReportSpotCB CSU:
"reportTmi" - widgit id for the report window TMI area
"reportForm" - widgit id for the report window entry area

4.5.17.6.2.8 Logic flow. Single pass.

4.5.17.6.2.9 Data structures. None

4.5.17.6.2.10 Local data files or database. None

4.5.17.6.2.11 Limitations. None

4.5.17.7 CSU ReportFreeTxtCB

The ReportFreeTxtCB CSU is the callback that brings up the Free Text report screen for report entry.

4.5.17.7.1 CSU ReportFreeTxtCB requirements.

The ReportFreeTxtCB CSU satisfies section 3.2.1.2.2.3.2.2 of the specific requirments of the DMCC system.

4.5.17.7.2 CSU ReportFreeTxtCB design.

The information identified below represents the detailed design of the ReportFreeTxtCB CSU.

4.5.17.7.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: None

4.5.17.7.2.2 Local data elements. None

4.5.17.7.2.3 Interrupts and signals. None

4.5.17.7.2.4 Algorithms. None

4.5.17.7.2.5 Error handling. None

4.5.17.7.2.6 Data conversion. None

4.5.17.7.2.7 Use of other elements. Other elements that are used by the ReportFreeTxtCB CSU:
"reportTmi" - widgit id for the report window TMI area
"reportForm" - widgit id for the report window entry area

4.5.17.7.2.8 Logic flow. Single pass.

4.5.17.7.2.9 Data structures. None

4.5.17.7.2.10 Local data files or database. None

4.5.17.7.2.11 Limitations. None

4.5.17.8 CSU ReportSaveExitCB

4.5.17.8.1 CSU ReportSaveExitCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.17.8.2 CSU ReportSaveExitCB design.

The information identified below represents the detailed design of the ReportSaveExitCB CSU.

4.5.17.8.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: None

4.5.17.8.2.2 Local data elements. None

4.5.17.8.2.3 Interrupts and signals. None

4.5.17.8.2.4 Algorithms. None

4.5.17.8.2.5 Error handling. None

4.5.17.8.2.6 Data conversion. None

4.5.17.8.2.7 Use of other elements. Other elements that are used by the ReportSaveExitCB CSU:
"reportTmi" - widgit id for the report window TMI area
"reportForm" - widgit id for the report window entry area
"replyFlag" - is this a reply
"reuseFlag" - is this a forwarded message

4.5.17.8.2.8 Logic flow. Single pass.

4.5.17.8.2.9 Data structures. None

4.5.17.8.2.10 Local data files or database. None

4.5.17.8.2.11 Limitations. None

4.5.17.9 CSU ReportMsgsCB

4.5.17.9.1 CSU ReportMsgsCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.17.9.2 CSU ReportMsgsCB design.

The information identified below represents the detailed design of the ReportMsgsCB CSU.

4.5.17.9.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: None

4.5.17.9.2.2 Local data elements. None

4.5.17.9.2.3 Interrupts and signals. None

4.5.17.9.2.4 Algorithms. None

4.5.17.9.2.5 Error handling. None

4.5.17.9.2.6 Data conversion. None

4.5.17.9.2.7 Use of other elements. Other elements that are used by the ReportMsgsCB CSU:
"reportTmi" - widgit id for the report window TMI area
"reportForm" - widgit id for the report window entry area
"replyFlag" - is this a reply
"reuseFlag" - is this a forwarded message

4.5.17.9.2.8 Logic flow. Single pass.

4.5.17.9.2.9 Data structures. None

4.5.17.9.2.10 Local data files or database. None

4.5.17.9.2.11 Limitations. None

4.5.17.10 CSU PutupReport

The PutupReport CSU is the callback that brings up the MTO report screen for report entry.

4.5.17.10.1 CSU PutupReport requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.17.10.2 CSU PutupReport design.

The information identified below represents the detailed design of the PutupReport CSU.

4.5.17.10.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: None

4.5.17.10.2.2 Local data elements. None

4.5.17.10.2.3 Interrupts and signals. None

4.5.17.10.2.4 Algorithms. None

4.5.17.10.2.5 Error handling. None

4.5.17.10.2.6 Data conversion. None

4.5.17.10.2.7 Use of other elements. Other elements that are used by the PutupReport CSU:

"reportTmi" - widgit id for the report window TMI area
"reportForm" - widgit id for the report window entry area
"tmiScreen" - widget id of current TMI
"smdScreen" - widget id of current entery screen

4.5.17.10.2.8 Logic flow. Single pass.

4.5.17.10.2.9 Data structures. None

4.5.17.10.2.10 Local data files or database. None

4.5.17.10.2.11 Limitations. None

4.5.18 Sub-Level CSC reqtTmi_call.

The following paragraphs under this section describe the relationship of the reqtTmi_call Sub-Level CSC in terms of data flow between the CSU's of this Sub-level CSC.

4.5.18.1 CSU ReqtClearNRetCB

The ReqtClearNRetCB CSU handles the callback from the Clear and Return pushbutton. It will clear and unmanage the Reqt window and remanage the Reports window.

4.5.18.1.1 CSU ReqtClearNRetCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.18.1.2 CSU ReqtClearNRetCB design.

The information identified below represents the detailed design of the ReqtClearNRetCB CSU.

4.5.18.1.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: None

4.5.18.1.2.2 Local data elements.
"argcnt" keeps track of the number of arguments set for XtSetValues
"args" the arguments set for XtSetValues
"argok" a boolean which is used to check the success of the CONVERT call
"xmstr" local to hold a XmString

4.5.18.1.2.3 Interrupts and signals. None

4.5.18.1.2.4 Algorithms. None

4.5.18.1.2.5 Error handling. None

4.5.18.1.2.6 Data conversion. None

4.5.18.1.2.7 Use of other elements. Other elements that are used by the ReqtClearNRetCB CSU:
"ReqtTmiForm" - form widget containing Reqt TMI
"ReqtForm" - form widget containing Reqt Entry area
"cikLabel2" - label widget

4.5.18.1.2.8 Logic flow. Single pass

4.5.18.1.2.9 Data structures. None

4.5.18.1.2.10 Local data files or database. None

4.5.18.1.2.11 Limitations. None

4.5.18.2 CSU ReqtSaveNRetCB

The ReqtSaveNRetCB CSU handles the callback from the Save and Return pushbutton. It will unmanage the Reqt window and remanage the Reports window.

4.5.18.2.1 CSU ReqtSaveNRetCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.18.2.2 CSU ReqtSaveNRetCB design.

The information identified below represents the detailed design of the ReqtSaveNRetCB CSU.

4.5.18.2.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: None

4.5.18.2.2.2 Local data elements.
"argcnt" keeps track of the number of arguments set for XtSetValues
"args" the arguments set for XtSetValues
"argok" a boolean which is used to check the success of the CONVERT call
"xmstr" local to hold a XmString
"nullChar" null character

4.5.18.2.2.3 Interrupts and signals. None

4.5.18.2.2.4 Algorithms. None

4.5.18.2.2.5 Error handling. None

4.5.18.2.2.6 Data conversion. None

4.5.18.2.2.7 Use of other elements. Other elements that are used by the ReqtSaveNRetCB CSU:
"ReqtTmiForm" - form widget containing Reqt TMI
"ReqtForm" - form widget containing Reqt Entry area
"cikLabel2" - label widget
"cikLabel1" - label widget

4.5.18.2.2.8 Logic flow. Single pass

4.5.18.2.2.9 Data structures. None

4.5.18.2.2.10 Local data files or database. None

4.5.18.2.2.11 Limitations. None

4.5.18.3 CSU ReqtSndRoutCB

The ReqtSndRoutCB CSU handles the callback from the Send Routine pushbutton. It calls ReqtSend with priority ROUTINE.

4.5.18.3.1 CSU ReqtSndRoutCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.18.3.2 CSU ReqtSndRoutCB design.

The information identified below represents the detailed design of the ReqtSndRoutCB CSU.

4.5.18.3.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: None

4.5.18.3.2.2 Local data elements. None

4.5.18.3.2.3 Interrupts and signals. None

4.5.18.3.2.4 Algorithms. None

4.5.18.3.2.5 Error handling. None

4.5.18.3.2.6 Data conversion. None

4.5.18.3.2.7 Use of other elements. Other elements that are used by the ReqtSndRoutCB CSU:
"PRIORITY_ROUTINE" - define for priority message

4.5.18.3.2.8 Logic flow. Single pass

4.5.18.3.2.9 Data structures. None

4.5.18.3.2.10 Local data files or database. None

4.5.18.3.2.11 Limitations. None

4.5.18.4 CSU ReqtSend

The ReqtSend CSU sends a message with a given priority. It retrieves the data from the text entry area and the annotation area and retrieves the address from the address selection menu. It then sends the data to bldReqt to be sent out. It also handles the case that the message is a reply by fixing the address to that from the message being replied to. If the message is a Reuse and Include, the included message will also be sent to the address selected.

4.5.18.4.1 CSU ReqtSend requirements

The ReqtSend CSU satisfies section 3.2.1.2.2.2.1, 3.2.1.2.2.3.6.1 and 3.2.1.2.2.3.6.1 of the specific requirments of the DMCC system.

4.5.18.4.2 CSU ReqtSend design.

The information identified below represents the detailed design of the ReqtSend CSU.

4.5.18.4.2.1 Input/output data elements.

INPUTS:
"prio" - priority of the message being sent

OUTPUTS: None

4.5.18.4.2.2 Local data elements.
"targetCEOI" - name of user message is being sent to
"annotation" - text of annotation string
"reportType" - type of report requested
"reconType" - type of recon

4.5.18.4.2.3 Interrupts and signals. None

4.5.18.4.2.4 Algorithms. None

4.5.18.4.2.5 Error handling. None

4.5.18.4.2.6 Data conversion. None

4.5.18.4.2.7 Use of other elements. Other elements that are used by the ReqtSend CSU:
"ReqtAdd" - information on the Reqt address selection menu
"reply flag" - boolean; is this a reply

"replyCEOI" - target name for reply call
"Type" - information on the Type selection menu
"Recon" - information on the Recon selection menu
"reuseFlag" - boolean; is this a reuse and include

4.5.18.4.2.8 Logic flow. Single pass

4.5.18.4.2.9 Data structures. None

4.5.18.4.2.10 Local data files or database. None

4.5.18.4.2.11 Limitations. None

4.5.18.5 CSU ReqtSndUrgCB

The ReqtSndUrgCB CSU handles the callback from the Send Urgent pushbutton. It calls ReqtSend with priority URGENT.

4.5.18.5.1 CSU ReqtSndUrgCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.18.5.2 CSU ReqtSndUrgCB design.

The information identified below represents the detailed design of the ReqtSndUrgCB CSU.

4.5.18.5.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: None

4.5.18.5.2.2 Local data elements. None

4.5.18.5.2.3 Interrupts and signals. None

4.5.18.5.2.4 Algorithms. None

4.5.18.5.2.5 Error handling. None

4.5.18.5.2.6 Data conversion. None

4.5.18.5.2.7 Use of other elements. Other elements that are used by the ReqtSndUrgCB CSU:
"PRIORITY_URGENT" - define for urgent priority

4.5.18.5.2.8 Logic flow. Single pass

4.5.18.5.2.9 Data structures. None

4.5.18.5.2.10 Local data files or database. None

4.5.18.5.2.11 Limitations. None

4.5.18.6 CSU ClearReqt

The ClearReqt CSU clears all the entries of the Reqt report generation screen.

4.5.18.6.1 CSU ClearReqt requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.18.6.2 CSU ClearReqt design.

The information identified below represents the detailed design of the ClearReqt CSU.

4.5.18.6.2.1 Input/output data elements.

INPUTS:

OUTPUTS: None

4.5.18.6.2.2 Local data elements.
"argcnt" keeps track of the number of arguments set for XtSetValues
"args" the arguments set for XtSetValues
"argok" a boolean which is used to check the success of the CONVERT call
"xmstr" local to hold a XmString
"nullChar" null char

4.5.18.6.2.3 Interrupts and signals. None

4.5.18.6.2.4 Algorithms. None

4.5.18.6.2.5 Error handling. None

4.5.18.6.2.6 Data conversion. None

4.5.18.6.2.7 Use of other elements. Other elements that are used by the ClearReqt CSU:
"ReqtAdd" - info on the selection menu for the address
"EnemyType" - info on the selection menu for the enemy type
"EnemyActivity" - info on the selection menu for the enemy activity
"Direc" - info on the selection menu for the direction
"ObsInt" - info on the selection menu for the observer intentions
"SpeedText" - widget id for speed text
"NumText" - widget id for number text
"UnitText" - widget id for unit text
"cikText1"- widget id for cik1 text
"cikText2"- widget id for cik2 text

4.5.18.6.2.8 Logic flow. Single pass

4.5.18.6.2.9 Data structures. None

4.5.18.6.2.10 Local data files or database. None

4.5.18.6.2.11 Limitations. None

4.5.19 Sub-Level CSC reqt_call.

The following paragraphs under this section describe the relationship of the reqt_call Sub-Level CSC in terms of data flow between the CSU's of this Sub-level CSC.

4.5.19.1 CSU type_moreCB

The typead_moreCB CSU handles the callback for the typead More pushbutton. It displays additional selectins if they exist.

4.5.19.1.1 CSU type_moreCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.19.1.2 CSU type_moreCB design.

The information identified below represents the detailed design of the type_moreCB CSU.

4.5.19.1.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: None

4.5.19.1.2.2 Local data elements.
"argcnt" keeps track of the number of arguments set for XtSetValues
"args" the arguments set for XtSetValues
"argok" a boolean which is used to check the success of the CONVERT call
"xmstr" local to hold a XmString

4.5.19.1.2.3 Interrupts and signals. None

4.5.19.1.2.4 Algorithms. None

4.5.19.1.2.5 Error handling. None

4.5.19.1.2.6 Data conversion. None

4.5.19.1.2.7 Use of other elements. Other elements that are used by the type_moreCB CSU:
"Type" - info on Enemy Activity selection menu

4.5.19.1.2.8 Logic flow. Single pass

4.5.19.1.2.9 Data structures. None

4.5.19.1.2.10 Local data files or database. None

4.5.19.1.2.11 Limitations. None

4.5.19.2 CSU type_rotateCB

The typead_rotateCB CSU handles the callback for the typead Rotate pushbutton. It changes the currently selected entry to the next entry (the first entry if it is at the end already).

4.5.19.2.1 CSU type_rotateCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.19.2.2 CSU type_rotateCB design.

The information identified below represents the detailed design of the type_rotateCB CSU.

4.5.19.2.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: .

4.5.19.2.2.2 Local data elements. None

4.5.19.2.2.3 Interrupts and signals. None

4.5.19.2.2.4 Algorithms. None

4.5.19.2.2.5 Error handling. None

4.5.19.2.2.6 Data conversion. None

4.5.19.2.2.7 Use of other elements. Other elements that are used by the type_rotateCB CSU:
"Type" - info on the enemy activity selection menu

4.5.19.2.2.8 Logic flow. Single pass

4.5.19.2.2.9 Data structures. None

4.5.19.2.2.10 Local data files or database. None

4.5.19.2.2.11 Limitations. None

4.5.19.3 CSU recon_rotateCB

The reconad_rotateCB CSU handles the callback for the reconad Rotate pushbutton. It changes the currently selected entry to the next entry (the first entry if it is at the end already).

4.5.19.3.1 CSU recon_rotateCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.19.3.2 CSU recon_rotateCB design.

The information identified below represents the detailed design of the recon_rotateCB CSU.

4.5.19.3.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: .

4.5.19.3.2.2 Local data elements. None

4.5.19.3.2.3 Interrupts and signals. None

4.5.19.3.2.4 Algorithms. None

4.5.19.3.2.5 Error handling. None

4.5.19.3.2.6 Data conversion. None

4.5.19.3.2.7 Use of other elements. Other elements that are used by the recon_rotateCB CSU:
"Recon" - info on the enemy activity selection menu

4.5.19.3.2.8 Logic flow. Single pass

4.5.19.3.2.9 Data structures. None

4.5.19.3.2.10 Local data files or database. None

4.5.19.3.2.11 Limitations. None

4.5.19.4 CSU reqtad_moreCB

The reqtad_moreCB CSU handles the callback for the reqtad More pushbutton. It displays additional selectins if they exist.

4.5.19.4.1 CSU reqtad_moreCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.19.4.2 CSU reqtad_moreCB design.

The information identified below represents the detailed design of the reqtad_moreCB CSU.

4.5.19.4.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: .

4.5.19.4.2.2 Local data elements. None

4.5.19.4.2.3 Interrupts and signals. None

4.5.19.4.2.4 Algorithms. None

4.5.19.4.2.5 Error handling. None

4.5.19.4.2.6 Data conversion. None

4.5.19.4.2.7 Use of other elements. Other elements that are used by the reqtad_moreCB CSU:
"ReqtAdd" - info on the Reqt address selection menu

4.5.19.4.2.8 Logic flow. Single pass

4.5.19.4.2.9 Data structures. None

4.5.19.4.2.10 Local data files or database. None

4.5.19.4.2.11 Limitations. None

4.5.19.5 CSU reqtad_rotateCB

The reqtad_rotateCB CSU handles the callback for the reqtad Rotate pushbutton. It changes the currently selected entry to the next entry (the first entry if it is at the end already).

4.5.19.5.1 CSU reqtad_rotateCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.19.5.2 CSU reqtad_rotateCB design.

The information identified below represents the detailed design of the reqtad_rotateCB CSU.

4.5.19.5.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: .

4.5.19.5.2.2 Local data elements. None

4.5.19.5.2.3 Interrupts and signals. None

4.5.19.5.2.4 Algorithms. None

4.5.19.5.2.5 Error handling. None

4.5.19.5.2.6 Data conversion. None

4.5.19.5.2.7 Use of other elements. Other elements that are used by the reqtad_rotateCB CSU:
"ReqtAdd" - info on Reqt address selection menu

4.5.19.5.2.8 Logic flow. Single pass

4.5.19.5.2.9 Data structures. None

4.5.19.5.2.10 Local data files or database. None

4.5.19.5.2.11 Limitations. None

4.5.19.6 CSU PutupReqt

The PutupReqt CSU will create the Reqt report screen if necessary and will also manage the screen.

4.5.19.6.1 CSU PutupReqt requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.19.6.2 CSU PutupReqt design.

The information identified below represents the detailed design of the PutupReqt CSU.

4.5.19.6.2.1 Input/output data elements.

INPUTS: None

OUTPUTS: None

4.5.19.6.2.2 Local data elements.
"argcnt" keeps track of the number of arguments set for XtSetValues
"args" the arguments set for XtSetValues

"argok" a boolean which is used to check the success of the CONVERT call
"xmstr" local to hold a XmString
"entries" - number of addresses

4.5.19.6.2.3 Interrupts and signals. None

4.5.19.6.2.4 Algorithms. None

4.5.19.6.2.5 Error handling. None

4.5.19.6.2.6 Data conversion. None

4.5.19.6.2.7 Use of other elements. Other elements that are used by the PutupReqt CSU:
"ReqtForm" - widget id of reqt form
"ReqtAdd" - info on Reqt address selection menu
"addressList" - list of addresses
"ReqtMoreLabel" - widget id of More label
"ReqtMoreArrow" - widget id of More arrow
"replyFlag" - boolean; is this a reply
"ReqtTmiForm" widget id of reqt TMI form

4.5.19.6.2.8 Logic flow. Single pass

4.5.19.6.2.9 Data structures. None

4.5.19.6.2.10 Local data files or database. None

4.5.19.6.2.11 Limitations. None

4.5.20 Sub-Level CSC reuseTmi_call.

The following paragraphs under this section describe the relationship of the reuseTmi_call Sub-Level CSC in terms of data flow between the CSU's of this Sub-level CSC.

4.5.20.1 CSU ReuseClearNRetCB

The ReuseClearNRetCB CSU handles the callback from the Clear and Return pushbutton. It will clear and unmanage the Reuse window and remanage the Reports window.

4.5.20.1.1 CSU ReuseClearNRetCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.20.1.2 CSU ReuseClearNRetCB design.

The information identified below represents the detailed design of the ReuseClearNRetCB CSU.

4.5.20.1.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: None

4.5.20.1.2.2 Local data elements.
"argcnt" keeps track of the number of arguments set for XtSetValues
"args" the arguments set for XtSetValues
"argok" a boolean which is used to check the success of the CONVERT call
"xmstr" local to hold a XmString

4.5.20.1.2.3 Interrupts and signals. None

4.5.20.1.2.4 Algorithms. None

4.5.20.1.2.5 Error handling. None

4.5.20.1.2.6 Data conversion. None

4.5.20.1.2.7 Use of other elements. Other elements that are used by the ReuseClearNRetCB CSU:
"ReuseTmiForm" - form widget containing Reuse TMI
"ReuseForm" - form widget containing Reuse Entry area
"cikLabel2" - label widget

4.5.20.1.2.8 Logic flow. Single pass

4.5.20.1.2.9 Data structures. None

4.5.20.1.2.10 Local data files or database. None

4.5.20.1.2.11 Limitations. None

4.5.20.2 CSU ReuseSendCB

The ReuseSendCB CSU handles the callback from the Send pushbutton. It forwards the selected message to the selected address.

4.5.20.2.1 CSU ReuseSend requirements.

The ReuseSend CSU satisfies section 3.2.1.2.2.3.3.1 of the specific requirments of the DMCC system.

4.5.20.2.2 CSU ReuseSend design.

The information identified below represents the detailed design of the ReuseSend CSU.

4.5.20.2.2.1 Input/output data elements.

INPUTS:
"prio" - priority of the message being sent

OUTPUTS: None

4.5.20.2.2.2 Local data elements.
"targetCEOI" - name of user message is being sent to

4.5.20.2.2.3 Interrupts and signals. None

4.5.20.2.2.4 Algorithms. None

4.5.20.2.2.5 Error handling. None

4.5.20.2.2.6 Data conversion. None

4.5.20.2.2.7 Use of other elements. Other elements that are used by the ReuseSend CSU:
"ReuseAdd" - information on the Reuse address selection menu
"reusePDU" - PDU to be forwarded

4.5.20.2.2.8 Logic flow. Single pass

4.5.20.2.2.9 Data structures. None

4.5.20.2.2.10 Local data files or database. None

4.5.20.2.2.11 Limitations. None

4.5.20.3 CSU ClearReuse

The ClearReuse CSU clears all the entries of the Reuse report generation screen.

4.5.20.3.1 CSU ClearReuse requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.20.3.2 CSU ClearReuse design.

The information identified below represents the detailed design of the ClearReuse CSU.

4.5.20.3.2.1 Input/output data elements.

INPUTS:

OUTPUTS: None

4.5.20.3.2.2 Local data elements.
"argcnt" keeps track of the number of arguments set for XtSetValues
"args" the arguments set for XtSetValues
"argok" a boolean which is used to check the success of the CONVERT call
"xmstr" local to hold a XmString
"nullChar" null char

4.5.20.3.2.3 Interrupts and signals. None

4.5.20.3.2.4 Algorithms. None

4.5.20.3.2.5 Error handling. None

4.5.20.3.2.6 Data conversion. None

4.5.20.3.2.7 Use of other elements. Other elements that are used by the ClearReuse CSU:
"ReuseAdd" - info on the selection menu for the address
"EnemyType" - info on the selection menu for the enemy type
"EnemyActivity" - info on the selection menu for the enemy activity
"Direc" - info on the selection menu for the direction
"ObsInt" - info on the selection menu for the observer intentions
"SpeedText" - widget id for speed text
"NumText" - widget id for number text
"UnitText" - widget id for unit text
"cikText1"- widget id for cik1 text
"cikText2"- widget id for cik2 text

4.5.20.3.2.8 Logic flow. Single pass

4.5.20.3.2.9 Data structures. None

4.5.20.3.2.10 Local data files or database. None

4.5.20.3.2.11 Limitations. None

4.5.21 Sub-Level CSC reuse_call.

The following paragraphs under this section describe the relationship of the reuse_call Sub-Level CSC in terms of data flow between the CSU's of this Sub-level CSC.

4.5.21.1 CSU reusead_moreCB

The reusead_moreCB CSU handles the callback for the reusead More pushbutton. It displays additional selectins if they exist.

4.5.21.1.1 CSU reusead_moreCB requirements

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.21.1.2 CSU reusead_moreCB design.

The information identified below represents the detailed design of the reusead_moreCB CSU.

4.5.21.1.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: .

4.5.21.1.2.2 Local data elements. None

4.5.21.1.2.3 Interrupts and signals. None

4.5.21.1.2.4 Algorithms. None

4.5.21.1.2.5 Error handling. None

4.5.21.1.2.6 Data conversion. None

4.5.21.1.2.7 Use of other elements. Other elements that are used by the reusead_moreCB CSU:
"ReuseAdd" - info on the Reuse address selection menu

4.5.21.1.2.8 Logic flow. Single pass

4.5.21.1.2.9 Data structures. None

4.5.21.1.2.10 Local data files or database. None

4.5.21.1.2.11 Limitations. None

4.5.21.2 CSU reusead_rotateCB

The reusead_rotateCB CSU handles the callback for the reusead Rotate pushbutton. It changes the currently selected entry to the next entry (the first entry if it is at the end already).

4.5.21.2.1 CSU reusead_rotateCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.21.2.2 CSU reusead_rotateCB design.

The information identified below represents the detailed design of the reusead_rotateCB CSU.

4.5.21.2.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: .

4.5.21.2.2.2 Local data elements. None

4.5.21.2.2.3 Interrupts and signals. None

4.5.21.2.2.4 Algorithms. None

4.5.21.2.2.5 Error handling. None

4.5.21.2.2.6 Data conversion. None

4.5.21.2.2.7 Use of other elements. Other elements that are used by the reusead_rotateCB CSU:
"ReuseAdd" - info on Reuse address selection menu

4.5.21.2.2.8 Logic flow. Single pass

4.5.21.2.2.9 Data structures. None

4.5.21.2.2.10 Local data files or database. None

4.5.21.2.2.11 Limitations. None

4.5.21.3 CSU PutupReuse

The PutupReuse CSU will create the Reuse report screen if necessary and will also manage the screen.

4.5.21.3.1 CSU PutupReuse requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.21.3.2 CSU PutupReuse design.

The information identified below represents the detailed design of the PutupReuse CSU.

4.5.21.3.2.1 Input/output data elements.

INPUTS: None

OUTPUTS: None

4.5.21.3.2.2 Local data elements.
"argcnt" keeps track of the number of arguments set for XtSetValues
"args" the arguments set for XtSetValues
"argok" a boolean which is used to check the success of the CONVERT call
"xmstr" local to hold a XmString
"entries" - number of addresses

4.5.21.3.2.3 Interrupts and signals. None

4.5.21.3.2.4 Algorithms. None

4.5.21.3.2.5 Error handling. None

4.5.21.3.2.6 Data conversion. None

4.5.21.3.2.7 Use of other elements. Other elements that are used by the PutupReuse CSU:
"ReuseForm" - widget id of reuse form
"ReuseAdd" - info on Reuse address selection menu
"addressList" - list of addresses
"ReuseMoreLabel" - widget id of More label
"ReuseMoreArrow" - widget id of More arrow
"replyFlag" - boolean; is this a reply
"ReuseTmiForm" widget id of reuse TMI form

4.5.21.3.2.8 Logic flow. Single pass

4.5.21.3.2.9 Data structures. None

4.5.21.3.2.10 Local data files or database. None

4.5.21.3.2.11 Limitations. None

4.5.22 Sub-Level CSC shotTmi_call.

The following paragraphs under this section describe the relationship of the shotTmi_call Sub-Level CSC in terms of data flow between the CSU's of this Sub-level CSC.

4.5.22.1 CSU ShotClearNRetCB

The ShotClearNRetCB CSU handles the callback from the Clear and Return pushbutton. It will clear and unmanage the Shot window and remanage the Reports window.

4.5.22.1.1 CSU ShotClearNRetCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.22.1.2 CSU ShotClearNRetCB design.

The information identified below represents the detailed design of the ShotClearNRetCB CSU.

4.5.22.1.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: None

4.5.22.1.2.2 Local data elements.
"argcnt" keeps track of the number of arguments set for XtSetValues
"args" the arguments set for XtSetValues
"argok" a boolean which is used to check the success of the CONVERT call

"xmstr" local to hold a XmString

4.5.22.1.2.3 Interrupts and signals. None

4.5.22.1.2.4 Algorithms. None

4.5.22.1.2.5 Error handling. None

4.5.22.1.2.6 Data conversion. None

4.5.22.1.2.7 Use of other elements. Other elements that are used by the ShotClearNRetCB CSU:
"ShotTmiForm" - form widget containing Shot TMI
"ShotForm" - form widget containing Shot Entry area
"cikLabel2" - label widget

4.5.22.1.2.8 Logic flow. Single pass

4.5.22.1.2.9 Data structures. None

4.5.22.1.2.10 Local data files or database. None

4.5.22.1.2.11 Limitations. None

4.5.22.2 CSU ShotSaveNRetCB

The ShotSaveNRetCB CSU handles the callback from the Save and Return pushbutton. It will unmanage the Shot window and remanage the Reports window.

4.5.22.2.1 CSU ShotSaveNRetCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.22.2.2 CSU ShotSaveNRetCB design.

The information identified below represents the detailed design of the ShotSaveNRetCB CSU.

4.5.22.2.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: None

4.5.22.2.2.2 Local data elements.
"argcnt" keeps track of the number of arguments set for XtSetValues
"args" the arguments set for XtSetValues
"argok" a boolean which is used to check the success of the CONVERT call
"xmstr" local to hold a XmString
"nullChar" null character

4.5.22.2.2.3 Interrupts and signals. None

4.5.22.2.2.4 Algorithms. None

4.5.22.2.2.5 Error handling. None

4.5.22.2.2.6 Data conversion. None

4.5.22.2.2.7 Use of other elements. Other elements that are used by the ShotSaveNRetCB CSU:
"ShotTmiForm" - form widget containing Shot TMI
"ShotForm" - form widget containing Shot Entry area
"cikLabel2" - label widget
"cikLabel1" - label widget

4.5.22.2.2.8 Logic flow. Single pass

4.5.22.2.2.9 Data structures. None

4.5.22.2.2.10 Local data files or database. None

4.5.22.2.2.11 Limitations. None

4.5.22.3 CSU ShotSndRoutCB

The ShotSndRoutCB CSU handles the callback from the Send Routine pushbutton. It calls ShotSend with priority ROUTINE.

4.5.22.3.1 CSU ShotSndRoutCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.22.3.2 CSU ShotSndRoutCB design.

The information identified below represents the detailed design of the ShotSndRoutCB CSU.

4.5.22.3.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: None

4.5.22.3.2.2 Local data elements. None

4.5.22.3.2.3 Interrupts and signals. None

4.5.22.3.2.4 Algorithms. None

4.5.22.3.2.5 Error handling. None

4.5.22.3.2.6 Data conversion. None

4.5.22.3.2.7 Use of other elements. Other elements that are used by the ShotSndRoutCB CSU:
"PRIORITY_ROUTINE" - define for priority message

4.5.22.3.2.8 Logic flow. Single pass

4.5.22.3.2.9 Data structures. None

4.5.22.3.2.10 Local data files or database. None

4.5.22.3.2.11 Limitations. None

4.5.22.4 CSU ShotSend

The ShotSend CSU sends a message with a given priority. It retrieves the data from the text entry area and the annotation area and retrieves the address from the address selection menu. It then sends the data to bldShot to be sent out. It also handles the case that the message is a reply by fixing the address to that from the message being replied to. If the message is a Reuse and Include, the included message will also be sent to the address selected.

4.5.22.4.1 CSU ShotSend requirements.

The ShotSend CSU satisfies section 3.2.1.2.2.2.1, 3.2.1.2.2.3.6.1 and 3.2.1.2.2.3.6.1 of the specific requirments of the DMCC system.

4.5.22.4.2 CSU ShotSend design.

The information identified below represents the detailed design of the ShotSend CSU.

4.5.22.4.2.1 Input/output data elements.

INPUTS:
"prio" - priority of the message being sent

OUTPUTS: None

4.5.22.4.2.2 Local data elements.
"targetCEOI" - name of user message is being sent to
"annotation" - text of annotation string

4.5.22.4.2.3 Interrupts and signals. None

4.5.22.4.2.4 Algorithms. None

4.5.22.4.2.5 Error handling. None

4.5.22.4.2.6 Data conversion. None

4.5.22.4.2.7 Use of other elements. Other elements that are used by the ShotSend CSU:
"ShotAdd" - information on the Shot address selection menu
"reply flag" - boolean; is this a reply
"replyCEOI" - target name for reply call
"reuseFlag" - boolean; is this a reuse and include

4.5.22.4.2.8 Logic flow. Single pass

4.5.22.4.2.9 Data structures. None

4.5.22.4.2.10 Local data files or database. None

4.5.22.4.2.11 Limitations. None

4.5.22.5 CSU ShotSndUrgCB

The ShotSndUrgCB CSU handles the callback from the Send Urgent pushbutton. It calls ShotSend with priority URGENT.

4.5.22.5.1 CSU ShotSndUrgCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.22.5.2 CSU ShotSndUrgCB design.

The information identified below represents the detailed design of the ShotSndUrgCB CSU.

4.5.22.5.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: None

4.5.22.5.2.2 Local data elements. None

4.5.22.5.2.3 Interrupts and signals. None

4.5.22.5.2.4 Algorithms. None

4.5.22.5.2.5 Error handling. None

4.5.22.5.2.6 Data conversion. None

4.5.22.5.2.7 Use of other elements. Other elements that are used by the ShotSndUrgCB CSU:
"PRIORITY_URGENT" - define for urgent priority

4.5.22.5.2.8 Logic flow. Single pass

4.5.22.5.2.9 Data structures. None

4.5.22.5.2.10 Local data files or database. None

4.5.22.5.2.11 Limitations. None

4.5.22.6 CSU ClearShot

The ClearShot CSU clears all the entries of the Shot report generation screen.

4.5.22.6.1 CSU ClearShot requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.22.6.2 CSU ClearShot design.

The information identified below represents the detailed design of the ClearShot CSU.

4.5.22.6.2.1 Input/output data elements.

INPUTS:

OUTPUTS: None

4.5.22.6.2.2 Local data elements.
"argcnt" keeps track of the number of arguments set for XtSetValues
"args" the arguments set for XtSetValues
"argok" a boolean which is used to check the success of the CONVERT call
"xmstr" local to hold a XmString
"nullChar" null char

4.5.22.6.2.3 Interrupts and signals. None

4.5.22.6.2.4 Algorithms. None

4.5.22.6.2.5 Error handling. None

4.5.22.6.2.6 Data conversion. None

4.5.22.6.2.7 Use of other elements. Other elements that are used by the ClearShot CSU:
"ShotAdd" - info on the selection menu for the address
"EnemyType" - info on the selection menu for the enemy type
"EnemyActivity" - info on the selection menu for the enemy activity
"Direc" - info on the selection menu for the direction
"ObsInt" - info on the selection menu for the observer intentions
"SpeedText" - widget id for speed text
"NumText" - widget id for number text
"UnitText" - widget id for unit text
"cikText1"- widget id for cik1 text
"cikText2"- widget id for cik2 text

4.5.22.6.2.8 Logic flow. Single pass

4.5.22.6.2.9 Data structures. None

4.5.22.6.2.10 Local data files or database. None

4.5.22.6.2.11 Limitations. None

4.5.23 Sub-Level CSC shot_call.

The following paragraphs under this section describe the relationship of the shot_call Sub-Level CSC in terms of data flow between the CSU's of this Sub-level CSC.

4.5.23.1 CSU shotad_moreCB

The shotad_moreCB CSU handles the callback for the shotad More pushbutton. It displays additional selectins if they exist.

4.5.23.1.1 CSU shotad_moreCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.23.1.2 CSU shotad_moreCB design.

The information identified below represents the detailed design of the shotad_moreCB CSU.

4.5.23.1.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: .

4.5.23.1.2.2 Local data elements. None

4.5.23.1.2.3 Interrupts and signals. None

4.5.23.1.2.4 Algorithms. None

4.5.23.1.2.5 Error handling. None

4.5.23.1.2.6 Data conversion. None

4.5.23.1.2.7 Use of other elements. Other elements that are used by the shotad_moreCB CSU:
"ShotAdd" - info on the Shot address selection menu

4.5.23.1.2.8 Logic flow. Single pass

4.5.23.1.2.9 Data structures. None

4.5.23.1.2.10 Local data files or database. None

4.5.23.1.2.11 Limitations. None

4.5.23.2 CSU shotad_rotateCB

The shotad_rotateCB CSU handles the callback for the shotad Rotate pushbutton. It changes the currently selected entry to the next entry (the first entry if it is at the end already).

4.5.23.2.1 CSU shotad_rotateCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.23.2.2 CSU shotad_rotateCB design.

The information identified below represents the detailed design of the shotad_rotateCB CSU.

4.5.23.2.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: .

4.5.23.2.2.2 Local data elements. None

4.5.23.2.2.3 Interrupts and signals. None

4.5.23.2.2.4 Algorithms. None

4.5.23.2.2.5 Error handling. None

4.5.23.2.2.6 Data conversion. None

4.5.23.2.2.7 Use of other elements. Other elements that are used by the shotad_rotateCB CSU:
"ShotAdd" - info on Shot address selection menu

4.5.23.2.2.8 Logic flow. Single pass

4.5.23.2.2.9 Data structures. None

4.5.23.2.2.10 Local data files or database. None

4.5.23.2.2.11 Limitations. None

4.5.23.3 CSU PutupShot

The PutupShot CSU will create the Shot report screen if necessary and will also manage the screen.

4.5.23.3.1 CSU PutupShot requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.23.3.2 CSU PutupShot design.

The information identified below represents the detailed design of the PutupShot CSU.

4.5.23.3.2.1 Input/output data elements.

INPUTS: None

OUTPUTS: None

4.5.23.3.2.2 Local data elements.
"argcnt" keeps track of the number of arguments set for XtSetValues
"args" the arguments set for XtSetValues
"argok" a boolean which is used to check the success of the CONVERT call
"xmstr" local to hold a XmString
"entries" - number of addresses

4.5.23.3.2.3 Interrupts and signals. None

4.5.23.3.2.4 Algorithms. None

4.5.23.3.2.5 Error handling. None

4.5.23.3.2.6 Data conversion. None

4.5.23.3.2.7 Use of other elements. Other elements that are used by the PutupShot CSU:
"ShotForm" - widget id of shot form
"ShotAdd" - info on Shot address selection menu
"addressList" - list of addresses
"ShotMoreLabel" - widget id of More label
"ShotMoreArrow" - widget id of More arrow

"replyFlag" - boolean; is this a reply
"ShotTmiForm" widget id of shot TMI form

4.5.23.3.2.8 Logic flow. Single pass

4.5.23.3.2.9 Data structures. None

4.5.23.3.2.10 Local data files or database. None

4.5.23.3.2.11 Limitations. None

4.5.24 Sub-Level CSC splashTmi_call.

The following paragraphs under this section describe the relationship of the splashTmi_call Sub-Level CSC in terms of data flow between the CSU's of this Sub-level CSC.

4.5.24.1 CSU SplashClearNRetCB

The SplashClearNRetCB CSU handles the callback from the Clear and Return pushbutton. It will clear and unmanage the Splash window and remanage the Reports window.

4.5.24.1.1 CSU SplashClearNRetCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.24.1.2 CSU SplashClearNRetCB design.

The information identified below represents the detailed design of the SplashClearNRetCB CSU.

4.5.24.1.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: None

4.5.24.1.2.2 Local data elements.
"argcnt" keeps track of the number of arguments set for XtSetValues
"args" the arguments set for XtSetValues
"argok" a boolean which is used to check the success of the CONVERT call
"xmstr" local to hold a XmString

4.5.24.1.2.3 Interrupts and signals. None

4.5.24.1.2.4 Algorithms. None

4.5.24.1.2.5 Error handling. None

4.5.24.1.2.6 Data conversion. None

4.5.24.1.2.7 Use of other elements. Other elements that are used by the SplashClearNRetCB CSU:
"SplashTmiForm" - form widget containing Splash TMI
"SplashForm" - form widget containing Splash Entry area
"cikLabel2" - label widget

4.5.24.1.2.8 Logic flow. Single pass

4.5.24.1.2.9 Data structures. None

4.5.24.1.2.10 Local data files or database. None

4.5.24.1.2.11 Limitations. None

4.5.24.2 CSU SplashSaveNRetCB

The SplashSaveNRetCB CSU handles the callback from the Save and Return pushbutton. It will unmanage the Splash window and remanage the Reports window.

4.5.24.2.1 CSU SplashSaveNRetCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.24.2.2 CSU SplashSaveNRetCB design.

The information identified below represents the detailed design of the SplashSaveNRetCB CSU.

4.5.24.2.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: None

4.5.24.2.2.2 Local data elements.
"argcnt" keeps track of the number of arguments set for XtSetValues
"args" the arguments set for XtSetValues
"argok" a boolean which is used to check the success of the CONVERT call
"xmstr" local to hold a XmString
"nullChar" null character

4.5.24.2.2.3 Interrupts and signals. None

4.5.24.2.2.4 Algorithms. None

4.5.24.2.2.5 Error handling. None

4.5.24.2.2.6 Data conversion. None

4.5.24.2.2.7 Use of other elements. Other elements that are used by the SplashSaveNRetCB CSU:
"SplashTmiForm" - form widget containing Splash TMI
"SplashForm" - form widget containing Splash Entry area
"cikLabel2" - label widget
"cikLabel1" - label widget

4.5.24.2.2.8 Logic flow. Single pass

4.5.24.2.2.9 Data structures. None

4.5.24.2.2.10 Local data files or database. None

4.5.24.2.2.11 Limitations. None

4.5.24.3 CSU SplashSndRoutCB

The SplashSndRoutCB CSU handles the callback from the Send Routine pushbutton. It calls SplashSend with priority ROUTINE.

4.5.24.3.1 CSU SplashSndRoutCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.24.3.2 CSU SplashSndRoutCB design.

The information identified below represents the detailed design of the SplashSndRoutCB CSU.

4.5.24.3.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: None

4.5.24.3.2.2 Local data elements. None

4.5.24.3.2.3 Interrupts and signals. None

4.5.24.3.2.4 Algorithms. None

4.5.24.3.2.5 Error handling. None

4.5.24.3.2.6 Data conversion. None

4.5.24.3.2.7 Use of other elements. Other elements that are used by the SplashSndRoutCB CSU:
"PRIORITY_ROUTINE" - define for priority message

4.5.24.3.2.8 Logic flow. Single pass

4.5.24.3.2.9 Data structures. None

4.5.24.3.2.10 Local data files or database. None

4.5.24.3.2.11 Limitations. None

4.5.24.4 CSU SplashSend

The SplashSend CSU sends a message with a given priority. It retrieves the data from the text entry area and the annotation area and retrieves the address from the address selection menu. It then sends the data to bldSplash to be sent out. It also handles the case that the message is a reply by fixing the address to that from the message being replied to. If the message is a Reuse and Include, the included message will also be sent to the address selected.

4.5.24.4.1 CSU SplashSend requirements.

The SplashSend CSU satisfies section 3.2.1.2.2.2.1, 3.2.1.2.2.3.6.1 and 3.2.1.2.2.3.6.1 of the specific requirments of the DMCC system.

4.5.24.4.2 CSU SplashSend design.

The information identified below represents the detailed design of the SplashSend CSU.

4.5.24.4.2.1 Input/output data elements.

INPUTS:
"prio" - priority of the message being sent

OUTPUTS: None

4.5.24.4.2.2 Local data elements.
"targetCEOI" - name of user message is being sent to
"annotation" - text of annotation string

4.5.24.4.2.3 Interrupts and signals. None

4.5.24.4.2.4 Algorithms. None

4.5.24.4.2.5 Error handling. None

4.5.24.4.2.6 Data conversion. None

4.5.24.4.2.7 Use of other elements. Other elements that are used by the SplashSend CSU:
"SplashAdd" - information on the Splash address selection menu
"reply flag" - boolean; is this a reply
"replyCEOI" - target name for reply call
"reuseFlag" - boolean; is this a reuse and include

4.5.24.4.2.8 Logic flow. Single pass

4.5.24.4.2.9 Data structures. None

4.5.24.4.2.10 Local data files or database. None

4.5.24.4.2.11 Limitations. None

4.5.24.5 CSU SplashSndUrgCB

The SplashSndUrgCB CSU handles the callback from the Send Urgent pushbutton. It calls SplashSend with priority URGENT.

4.5.24.5.1 CSU SplashSndUrgCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.24.5.2 CSU SplashSndUrgCB design.

The information identified below represents the detailed design of the SplashSndUrgCB CSU.

4.5.24.5.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: None

4.5.24.5.2.2 Local data elements. None

4.5.24.5.2.3 Interrupts and signals. None

4.5.24.5.2.4 Algorithms. None

4.5.24.5.2.5 Error handling. None

4.5.24.5.2.6 Data conversion. None

4.5.24.5.2.7 Use of other elements. Other elements that are used by the SplashSndUrgCB CSU:
"PRIORITY_URGENT" - define for urgent priority

4.5.24.5.2.8 Logic flow. Single pass

4.5.24.5.2.9 Data structures. None

4.5.24.5.2.10 Local data files or database. None

4.5.24.5.2.11 Limitations. None

4.5.24.6 CSU ClearSplash

The ClearSplash CSU clears all the entries of the Splash report generation screen.

4.5.24.6.1 CSU ClearSplash requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.24.6.2 CSU ClearSplash design.

The information identified below represents the detailed design of the ClearSplash CSU.

4.5.24.6.2.1 Input/output data elements.

INPUTS:

OUTPUTS: None

4.5.24.6.2.2 Local data elements.
"argcnt" keeps track of the number of arguments set for XtSetValues
"args" the arguments set for XtSetValues
"argok" a boolean which is used to check the success of the CONVERT call
"xmstr" local to hold a XmString
"nullChar" null char

4.5.24.6.2.3 Interrupts and signals. None

4.5.24.6.2.4 Algorithms. None

4.5.24.6.2.5 Error handling. None

4.5.24.6.2.6 Data conversion. None

4.5.24.6.2.7 Use of other elements. Other elements that are used by the ClearSplash CSU:
"SplashAdd" - info on the selection menu for the address
"EnemyType" - info on the selection menu for the enemy type
"EnemyActivity" - info on the selection menu for the enemy activity
"Direc" - info on the selection menu for the direction
"ObsInt" - info on the selection menu for the observer intentions
"SpeedText" - widget id for speed text
"NumText" - widget id for number text
"UnitText" - widget id for unit text
"cikText1"- widget id for cik1 text
"cikText2"- widget id for cik2 text

4.5.24.6.2.8 Logic flow. Single pass

4.5.24.6.2.9 Data structures. None

4.5.24.6.2.10 Local data files or database. None

4.5.24.6.2.11 Limitations None

4.5.25 Sub-Level CSC splash_call.

The following paragraphs under this section describe the relationship of the splash_call Sub-Level CSC in terms of data flow between the CSU's of this Sub-level CSC.

4.5.25.1 CSU splashad_moreCB

The splashad_moreCB CSU handles the callback for the splashad More pushbutton. It displays additional selectins if they exist.

4.5.25.1.1 CSU splashad_moreCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.25.1.2 CSU splashad_moreCB design.

The information identified below represents the detailed design of the splashad_moreCB CSU.

4.5.25.1.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: .

4.5.25.1.2.2 Local data elements. None

4.5.25.1.2.3 Interrupts and signals. None

4.5.25.1.2.4 Algorithms. None

4.5.25.1.2.5 Error handling. None

4.5.25.1.2.6 Data conversion. None

4.5.25.1.2.7 Use of other elements. Other elements that are used by the splashad_moreCB CSU:
"SplashAdd" - info on the Splash address selection menu

4.5.25.1.2.8 Logic flow. Single pass

4.5.25.1.2.9 Data structures. None

4.5.25.1.2.10 Local data files or database. None

4.5.25.1.2.11 Limitations. None

4.5.25.2 CSU splashad_rotateCB

The splashad_rotateCB CSU handles the callback for the splashad Rotate pushbutton. It changes the currently selected entry to the next entry (the first entry if it is at the end already).

4.5.25.2.1 CSU splashad_rotateCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.25.2.2 CSU splashad_rotateCB design.

The information identified below represents the detailed design of the splashad_rotateCB CSU.

4.5.25.2.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: .

4.5.25.2.2.2 Local data elements. None

4.5.25.2.2.3 Interrupts and signals. None

4.5.25.2.2.4 Algorithms. None

4.5.25.2.2.5 Error handling. None

4.5.25.2.2.6 Data conversion. None

4.5.25.2.2.7 Use of other elements. Other elements that are used by the splashad_rotateCB CSU:
"SplashAdd" - info on Splash address selection menu

4.5.25.2.2.8 Logic flow. Single pass

4.5.25.2.2.9 Data structures. None

4.5.25.2.2.10 Local data files or database. None

4.5.25.2.2.11 Limitations. None

4.5.25.3 CSU PutupSplash

The PutupSplash CSU will create the Splash report screen if necessary and will also manage the screen.

4.5.25.3.1 CSU PutupSplash requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.25.3.2 CSU PutupSplash design.

The information identified below represents the detailed design of the PutupSplash CSU.

4.5.25.3.2.1 Input/output data elements.

INPUTS: None

OUTPUTS: None

4.5.25.3.2.2 Local data elements.
"argcnt" keeps track of the number of arguments set for XtSetValues
"args" the arguments set for XtSetValues
"argok" a boolean which is used to check the success of the CONVERT call
"xmstr" local to hold a XmString
"entries" - number of addresses

4.5.25.3.2.3 Interrupts and signals. None

4.5.25.3.2.4 Algorithms. None

4.5.25.3.2.5 Error handling. None

4.5.25.3.2.6 Data conversion. None

4.5.25.3.2.7 Use of other elements. Other elements that are used by the PutupSplash CSU:
"SplashForm" - widget id of splash form

"SplashAdd" - info on Splash address selection menu
"addressList" - list of addresses
"SplashMoreLabel" - widget id of More label
"SplashMoreArrow" - widget id of More arrow
"replyFlag" - boolean; is this a reply
"SplashTmiForm" widget id of splash TMI form

4.5.25.3.2.8 Logic flow. Single pass

4.5.25.3.2.9 Data structures. None

4.5.25.3.2.10 Local data files or database. None

4.5.25.3.2.11 Limitations. None

4.5.26 Sub-Level CSC spotTmi_call.

The following paragraphs under this section describe the relationship of the spotTmi_call Sub-Level CSC in terms of data flow between the CSU's of this Sub-level CSC.

4.5.26.1 CSU SpotClearNRetCB

The SpotClearNRetCB CSU handles the callback from the Clear and Return pushbutton. It will clear and unmanage the Spot window and remanage the Reports window.

4.5.26.1.1 CSU SpotClearNRetCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.26.1.2 CSU SpotClearNRetCB design.

The information identified below represents the detailed design of the SpotClearNRetCB CSU.

4.5.26.1.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: None

4.5.26.1.2.2 Local data elements.
"argcnt" keeps track of the number of arguments set for XtSetValues
"args" the arguments set for XtSetValues
"argok" a boolean which is used to check the success of the CONVERT call
"xmstr" local to hold a XmString

4.5.26.1.2.3 Interrupts and signals. None

4.5.26.1.2.4 Algorithms. None

4.5.26.1.2.5 Error handling. None

4.5.26.1.2.6 Data conversion. None

4.5.26.1.2.7 Use of other elements. Other elements that are used by the SpotClearNRetCB CSU:
"SpotTmiForm" - form widget containing Spot TMI
"SpotForm" - form widget containing Spot Entry area
"cikLabel2" - label widget

4.5.26.1.2.8 Logic flow. Single pass

4.5.26.1.2.9 Data structures. None

4.5.26.1.2.10 Local data files or database. None

4.5.26.1.2.11 Limitations. None

4.5.26.2 CSU SpotSaveNRetCB

The SpotSaveNRetCB CSU handles the callback from the Save and Return pushbutton. It will unmanage the Spot window and remanage the Reports window.

4.5.26.2.1 CSU SpotSaveNRetCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.26.2.2 CSU SpotSaveNRetCB design.

The information identified below represents the detailed design of the SpotSaveNRetCB CSU.

4.5.26.2.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: None

4.5.26.2.2.2 Local data elements.
"argcnt" keeps track of the number of arguments set for XtSetValues
"args" the arguments set for XtSetValues
"argok" a boolean which is used to check the success of the CONVERT call
"xmstr" local to hold a XmString
"nullChar" null character

4.5.26.2.2.3 Interrupts and signals. None

4.5.26.2.2.4 Algorithms. None

4.5.26.2.2.5 Error handling. None

4.5.26.2.2.6 Data conversion. None

4.5.26.2.2.7 Use of other elements. Other elements that are used by the SpotSaveNRetCB CSU:
"SpotTmiForm" - form widget containing Spot TMI
"SpotForm" - form widget containing Spot Entry area
"cikLabel2" - label widget
"cikLabel1" - label widget

4.5.26.2.2.8 Logic flow. Single pass

4.5.26.2.2.9 Data structures. None

4.5.26.2.2.10 Local data files or database. None

4.5.26.2.2.11 Limitations. None

4.5.26.3 CSU SpotSndRoutCB

The SpotSndRoutCB CSU handles the callback from the Send Routine pushbutton. It calls SpotSend with priority ROUTINE.

4.5.26.3.1 CSU SpotSndRoutCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.26.3.2 CSU SpotSndRoutCB design.

The information identified below represents the detailed design of the SpotSndRoutCB CSU.

4.5.26.3.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: None

4.5.26.3.2.2 Local data elements. None

4.5.26.3.2.3 Interrupts and signals. None

4.5.26.3.2.4 Algorithms. None

4.5.26.3.2.5 Error handling. None

4.5.26.3.2.6 Data conversion. None

4.5.26.3.2.7 Use of other elements. Other elements that are used by the SpotSndRoutCB CSU:
"PRIORITY_ROUTINE" - define for priority message

4.5.26.3.2.8 Logic flow. Single pass

4.5.26.3.2.9 Data structures. None

4.5.26.3.2.10 Local data files or database. None

4.5.26.3.2.11 Limitations. None

4.5.26.4 CSU SpotSend

The SpotSend CSU sends a message with a given priority. It retrieves the data from the text entry area and the annotation area and retrieves the address from the address selection menu. It then sends the data to bldSpot to be sent out. It also handles the case that the message is a reply by fixing the address to that from the message being replied to. If the message is a Reuse and Include, the included message will also be sent to the address selected.

4.5.26.4.1 CSU SpotSend requirements.

The SpotSend CSU satisfies section 3.2.1.2.2.2.1, 3.2.1.2.2.3.6.1 and 3.2.1.2.2.3.6.1 of the specific requirments of the DMCC system.

4.5.26.4.2 CSU SpotSend design.

The information identified below represents the detailed design of the SpotSend CSU.

4.5.26.4.2.1 Input/output data elements.

INPUTS:
"prio" - priority of the message being sent

OUTPUTS: None

4.5.26.4.2.2 Local data elements.
"targetCEOI" - name of user message is being sent to
"annotation" - text of annotation string
"ObserverLocation" - location of observer
"TargetQuantity" - quantity of targets
"ObserverIntentions" - intentions of observers
"TargetType" - type of target
"Target Activity" - type of activity
"locn_string" - string containing location information
"num_string" - string containing number information
"num" - integer containing number information
"speed_string" - string containing speed information
"speed" - integer containing the speed
"TargetSpeed" - speed of the target
"TargetLocation" - location of the target
"TargetUnit" - pointer to the string of the target unit

4.5.26.4.2.3 Interrupts and signals. None

4.5.26.4.2.4 Algorithms. None

4.5.26.4.2.5 Error handling. None

4.5.26.4.2.6 Data conversion. None

4.5.26.4.2.7 Use of other elements. Other elements that are used by the SpotSend CSU:
"SpotAdd" - information on the Spot address selection menu
"reply flag" - boolean; is this a reply
"replyCEOI" - target name for reply call
"EnemyActivity" - information on the Enemy Activity selection menu
"ObsInt" - information on the Observer Intentions selection menu
"reuseFlag" - boolean; is this a reuse and include

4.5.26.4.2.8 Logic flow. Single pass

4.5.26.4.2.9 Data structures. None

4.5.26.4.2.10 Local data files or database. None

4.5.26.4.2.11 Limitations. None

4.5.26.5. CSU SpotSndUrgCB

The SpotSndUrgCB CSU handles the callback from the Send Urgent pushbutton. It calls SpotSend with priority URGENT.

4.5.26.5.1 CSU SpotSndUrgCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.26.5.2 CSU SpotSndUrgCB design.

The information identified below represents the detailed design of the SpotSndUrgCB CSU.

4.5.26.5.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed

"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: None

4.5.26.5.2.2 Local data elements. None

4.5.26.5.2.3 Interrupts and signals. None

4.5.26.5.2.4 Algorithms. None

4.5.26.5.2.5 Error handling. None

4.5.26.5.2.6 Data conversion. None

4.5.26.5.2.7 Use of other elements. Other elements that are used by the SpotSndUrgCB CSU:
"PRIORITY_URGENT" - define for urgent priority

4.5.26.5.2.8 Logic flow. Single pass

4.5.26.5.2.9 Data structures. None

4.5.26.5.2.10 Local data files or database. None

4.5.26.5.2.11 Limitations. None

4.5.26.6 CSU ClearSpot

The ClearSpot CSU clears all the entries of the Spot report generation screen.

4.5.26.6.1 CSU ClearSpot requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.26.6.2 CSU ClearSpot design.

The information identified below represents the detailed design of the ClearSpot CSU.

4.5.26.6.2.1 Input/output data elements.

INPUTS:

OUTPUTS: None

4.5.26.6.2.2 Local data elements.
"argcnt" keeps track of the number of arguments set for XtSetValues
"args" the arguments set for XtSetValues
"argok" a boolean which is used to check the success of the CONVERT call
"xmstr" local to hold a XmString
"nullChar" null char

4.5.26.6.2.3 Interrupts and signals. None

4.5.26.6.2.4 Algorithms. None

4.5.26.6.2.5 Error handling. None

4.5.26.6.2.6 Data conversion. None

4.5.26.6.2.7 Use of other elements. Other elements that are used by the ClearSpot CSU:
"SpotAdd" - info on the selection menu for the address
"EnemyType" - info on the selection menu for the enemy type
"EnemyActivity" - info on the selection menu for the enemy activity
"Direc" - info on the selection menu for the direction
"ObsInt" - info on the selection menu for the observer intentions
"SpeedText" - widget id for speed text
"NumText" - widget id for number text
"UnitText" - widget id for unit text
"cikText1"- widget id for cik1 text
"cikText2"- widget id for cik2 text

4.5.26.6.2.8 Logic flow. Single pass

4.5.26.6.2.9 Data structures. None

4.5.26.6.2.10 Local data files or database. None

4.5.26.6.2.11 Limitations. None

4.5.26.7 CSU KPH_MPHCB

The KPH_MPHCB CSU handles the callback from the KPH/MPH pushbutton. It toggles the units the speed is used as. In addition, it changes the label after the entry area to be the currently chosen units and the label of the pushbutton to the units currently not chosen.

4.5.26.7.1 CSU KPH_MPHCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.26.7.2 CSU KPH_MPHCB design.

The information identified below represents the detailed design of the KPH_MPHCB CSU.

4.5.26.7.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: None

4.5.26.7.2.2 Local data elements.
"argcnt" keeps track of the number of arguments set for XtSetValues
"args" the arguments set for XtSetValues
"argok" a boolean which is used to check the success of the CONVERT call
"xmstr" local to hold a XmString

4.5.26.7.2.3 Interrupts and signals. None

4.5.26.7.2.4 Algorithms. None

4.5.26.7.2.5 Error handling. None

4.5.26.7.2.6 Data conversion. None

4.5.26.7.2.7 Use of other elements. Other elements that are used by the KPH_MPHCB CSU:
"SpotTmiKPH" - widget id for KPH button

"SpeedLabel" - widget id for KPH label

4.5.26.7.2.8 Logic flow. Single pass

4.5.26.7.2.9 Data structures. None

4.5.26.7.2.10 Local data files or database. None

4.5.26.7.2.11 Limitations. None

4.5.27 Sub-Level CSC spot_call.

The following paragraphs under this section describe the relationship of the spot_call Sub-Level CSC in terms of data flow between the CSU's of this Sub-level CSC.

4.5.27.1 CSU enemyActivity_moreCB

The enemyActivityad_moreCB CSU handles the callback for the enemyActivityad More pushbutton. It displays additional selectins if they exist.

4.5.27.1.1 CSU enemyActivity_moreCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.27.1.2 CSU enemyActivity_moreCB design.

The information identified below represents the detailed design of the enemyActivity_moreCB CSU.

4.5.27.1.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: None

4.5.27.1.2.2 Local data elements.
"argcnt" keeps track of the number of arguments set for XtSetValues

"args" the arguments set for XtSetValues
"argok" a boolean which is used to check the success of the CONVERT call
"xmstr" local to hold a XmString

4.5.27.1.2.3 Interrupts and signals. None

4.5.27.1.2.4 Algorithms. None

4.5.27.1.2.5 Error handling. None

4.5.27.1.2.6 Data conversion. None

4.5.27.1.2.7 Use of other elements. Other elements that are used by the enemyActivity_moreCB CSU:
"EnemyActivity" - info on Enemy Activity selection menu

4.5.27.1.2.8 Logic flow. Single pass

4.5.27.1.2.9 Data structures. None

4.5.27.1.2.10 Local data files or database. None

4.5.27.1.2.11 Limitations. None

4.5.27.2 CSU enemyActivity_rotateCB

The enemyActivityad_rotateCB CSU handles the callback for the enemyActivityad Rotate pushbutton. It changes the currently selected entry to the next entry (the first entry if it is at the end already).

4.5.27.2.1 CSU enemyActivity_rotateCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.27.2.2 CSU enemyActivity_rotateCB design.

The information identified below represents the detailed design of the enemyActivity_rotateCB CSU.

4.5.27.2.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: .

4.5.27.2.2.2 Local data elements. None

4.5.27.2.2.3 Interrupts and signals. None

4.5.27.2.2.4 Algorithms. None

4.5.27.2.2.5 Error handling. None

4.5.27.2.2.6 Data conversion. None

4.5.27.2.2.7 Use of other elements. Other elements that are used by the enemyActivity_rotateCB CSU:
"EnemyActivity" - info on the enemy activity selection menu

4.5.27.2.2.8 Logic flow. Single pass

4.5.27.2.2.9 Data structures. None

4.5.27.2.2.10 Local data files or database. None

4.5.27.2.2.11 Limitations. None

4.5.27.3 CSU enemyType_rotateCB

The enemyTypead_rotateCB CSU handles the callback for the enemyTypead Rotate pushbutton. It changes the currently selected entry to the next entry (the first entry if it is at the end already).

4.5.27.3.1 CSU enemyType_rotateCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.27.3.2 CSU enemyType_rotateCB design.

The information identified below represents the detailed design of the enemyType_rotateCB CSU.

4.5.27.3.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: .

4.5.27.3.2.2 Local data elements. None

4.5.27.3.2.3 Interrupts and signals. None

4.5.27.3.2.4 Algorithms. None

4.5.27.3.2.5 Error handling. None

4.5.27.3.2.6 Data conversion. None

4.5.27.3.2.7 Use of other elements. Other elements that are used by the enemyType_rotateCB CSU:
"EnemyType" - info on the Enemy Type selection menu
"spotTime" - time of last selection

4.5.27.3.2.8 Logic flow. Single pass

4.5.27.3.2.9 Data structures. None

4.5.27.3.2.10 Local data files or database. None

4.5.27.3.2.11 Limitations. None

4.5.27.4 CSU obsInt_rotateCB

The obsIntad_rotateCB CSU handles the callback for the obsIntad Rotate pushbutton. It changes the currently selected entry to the next entry (the first entry if it is at the end already).

4.5.27.4.1 CSU obsInt_rotateCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.27.4.2 CSU obsInt_rotateCB design.

The information identified below represents the detailed design of the obsInt_rotateCB CSU.

4.5.27.4.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: .

4.5.27.4.2.2 Local data elements. None

4.5.27.4.2.3 Interrupts and signals. None

4.5.27.4.2.4 Algorithms. None

4.5.27.4.2.5 Error handling. None

4.5.27.4.2.6 Data conversion. None

4.5.27.4.2.7 Use of other elements. Other elements that are used by the obsInt_rotateCB CSU:

4.5.27.4.2.8 Logic flow. Single pass

4.5.27.4.2.9 Data structures. None

4.5.27.4.2.10 Local data files or database. None

4.5.27.4.2.11 Limitations. None

4.5.27.5 CSU spotad_moreCB

The spotad_moreCB CSU handles the callback for the spotad More pushbutton. It displays additional selectins if they exist.

4.5.27.5.1 CSU spotad_moreCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.27.5.2 CSU spotad_moreCB design.

The information identified below represents the detailed design of the spotad_moreCB CSU.

4.5.27.5.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: .

4.5.27.5.2.2 Local data elements. None

4.5.27.5.2.3 Interrupts and signals. None

4.5.27.5.2.4 Algorithms. None

4.5.27.5.2.5 Error handling. None

4.5.27.5.2.6 Data conversion. None

4.5.27.5.2.7 Use of other elements. Other elements that are used by the spotad_moreCB CSU:
"SpotAdd" - info on the Spot address selection menu

4.5.27.5.2.8 Logic flow. Single pass

4.5.27.5.2.9 Data structures. None

4.5.27.5.2.10 Local data files or database. None

4.5.27.5.2.11 Limitations. None

4.5.27.6 CSU spotad_rotateCB

The spotad_rotateCB CSU handles the callback for the spotad Rotate pushbutton. It changes the currently selected entry to the next entry (the first entry if it is at the end already).

4.5.27.6.1 CSU spotad_rotateCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.27.6.2 CSU spotad_rotateCB design.

The information identified below represents the detailed design of the spotad_rotateCB CSU.

4.5.27.6.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: .

4.5.27.6.2.2 Local data elements. None

4.5.27.6.2.3 Interrupts and signals. None

4.5.27.6.2.4 Algorithms. None

4.5.27.6.2.5 Error handling. None

4.5.27.6.2.6 Data conversion. None

4.5.27.6.2.7 Use of other elements. Other elements that are used by the spotad_rotateCB CSU:
"SpotAdd" - info on Spot address selection menu

4.5.27.6.2.8 Logic flow. Single pass

4.5.27.6.2.9 Data structures. None

4.5.27.6.2.10 Local data files or database. None

4.5.27.6.2.11 Limitations. None

4.5.27.7 CSU direc_rotateCB

The direcad_rotateCB CSU handles the callback for the direcad Rotate pushbutton. It changes the currently selected entry to the next entry (the first entry if it is at the end already).

4.5.27.7.1 CSU direc_rotateCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.27.7.2 CSU direc_rotateCB design.

The information identified below represents the detailed design of the direc_rotateCB CSU.

4.5.27.7.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: .

4.5.27.7.2.2 Local data elements. None

4.5.27.7.2.3 Interrupts and signals. None

4.5.27.7.2.4 Algorithms. None

4.5.27.7.2.5 Error handling. None

4.5.27.7.2.6 Data conversion. None

4.5.27.7.2.7 Use of other elements. Other elements that are used by the direc_rotateCB CSU:

"Direc" - info on the Direction selection menu

4.5.27.7.2.8 Logic flow. Single pass

4.5.27.7.2.9 Data structures. None

4.5.27.7.2.10 Local data files or database. None

4.5.27.7.2.11 Limitations. None

4.5.27.8 CSU PutupSpot

The PutupSpot CSU will create the Spot report screen if necessary and will also manage the screen.

4.5.27.8.1 CSU PutupSpot requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.27.8.2 CSU PutupSpot design.

The information identified below represents the detailed design of the PutupSpot CSU.

4.5.27.8.2.1 Input/output data elements.

INPUTS: None

OUTPUTS: None

4.5.27.8.2.2 Local data elements.
"argcnt" keeps track of the number of arguments set for XtSetValues
"args" the arguments set for XtSetValues
"argok" a boolean which is used to check the success of the CONVERT call
"xmstr" local to hold a XmString
"entries" - number of addresses

4.5.27.8.2.3 Interrupts and signals. None

4.5.27.8.2.4 Algorithms. None

4.5.27.8.2.5 Error handling. None

4.5.27.8.2.6 Data conversion. None

4.5.27.8.2.7 Use of other elements. Other elements that are used by the PutupSpot CSU:
"SpotForm" - widget id of spot form
"SpotAdd" - info on Spot address selection menu
"addressList" - list of addresses
"SpotMoreLabel" - widget id of More label
"SpotMoreArrow" - widget id of More arrow
"replyFlag" - boolean; is this a reply
"SpotTmiForm" widget id of spot TMI form

4.5.27.8.2.8 Logic flow. Single pass

4.5.27.8.2.9 Data structures. None

4.5.27.8.2.10 Local data files or database. None

4.5.27.8.2.11 Limitations. None

4.5.28 Sub-Level CSC sysMainCB.

The following paragraphs under this section describe the relationship of the sysMainCB Sub-Level CSC in terms of data flow between the CSU's of this Sub-level CSC.

4.5.28.1 CSU SysMainMsgsCB

The SysMainMsgsCB CSU handles the callback for bringing up the Msgs screen.

4.5.28.1.1 CSU SysMainMsgsCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.28.1.2 CSU SysMainMsgsCB design.

The information identified below represents the detailed design of the SysMainMsgsCB CSU.

4.5.28.1.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: None

4.5.28.1.2.2 Local data elements. None

4.5.28.1.2.3 Interrupts and signals. None

4.5.28.1.2.4 Algorithms. None

4.5.28.1.2.5 Error handling. None

4.5.28.1.2.6 Data conversion. None

4.5.28.1.2.7 Use of other elements. Other elements that are used by the SysMainMsgsCB CSU:
"sysMainTmi" - widget id of TMI area of sysMain form
"sysMainForm" - widget id of entry area of sysMain form

4.5.28.1.2.8 Logic flow. Single pass.

4.5.28.1.2.9 Data structures. None

4.5.28.1.2.10 Local data files or database. None

4.5.28.1.2.11 Limitations. None

4.5.28.2 CSU SysMainRprtCB

The SysMainRprtCB CSU handles the callback for bringing up the Rprt screen.

4.5.28.2.1 CSU SysMainRprtCB requirements

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.28.2.2 CSU SysMainRprtCB design.

The information identified below represents the detailed design of the SysMainRprtCB CSU.

4.5.28.2.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: None

4.5.28.2.2.2 Local data elements. None

4.5.28.2.2.3 Interrupts and signals. None

4.5.28.2.2.4 Algorithms. None

4.5.28.2.2.5 Error handling. None

4.5.28.2.2.6 Data conversion. None

4.5.28.2.2.7 Use of other elements. Other elements that are used by the SysMainRprtCB CSU:
"sysMainTmi" - widget id of TMI area of sysMain form
"sysMainForm" - widget id of entry area of sysMain form

4.5.28.2.2.8 Logic flow. Single pass.

4.5.28.2.2.9 Data structures. None

4.5.28.2.2.10 Local data files or database. None

4.5.28.2.2.11 Limitations. None

4.5.28.3 CSU SysMainAddListCB

The SysMainAddListCB CSU handles the callback for bringing up the AddList screen.

4.5.28.3.1 CSU SysMainAddListCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.28.3.2 CSU SysMainAddListCB design.

The information identified below represents the detailed design of the SysMainAddListCB CSU.

4.5.28.3.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: None

4.5.28.3.2.2 Local data elements. None

4.5.28.3.2.3 Interrupts and signals. None

4.5.28.3.2.4 Algorithms. None

4.5.28.3.2.5 Error handling. None

4.5.28.3.2.6 Data conversion. None

4.5.28.3.2.7 Use of other elements. Other elements that are used by the SysMainAddListCB CSU:
"sysMainTmi" - widget id of TMI area of sysMain form
"sysMainForm" - widget id of entry area of sysMain form

4.5.28.3.2.8 Logic flow. Single pass.

4.5.28.3.2.9 Data structures. None

4.5.28.3.2.10 Local data files or database. None

4.5.28.3.2.11 Limitations. None

4.5.28.4 CSU SysMainGrpListCB

The SysMainGrpListCB CSU handles the callback for bringing up the GrpList screen.

4.5.28.4.1 CSU SysMainGrpListCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.28.4.2 CSU SysMainGrpListCB design.

The information identified below represents the detailed design of the SysMainGrpListCB CSU.

4.5.28.4.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: None

4.5.28.4.2.2 Local data elements. None

4.5.28.4.2.3 Interrupts and signals. None

4.5.28.4.2.4 Algorithms. None

4.5.28.4.2.5 Error handling. None

4.5.28.4.2.6 Data conversion. None

4.5.28.4.2.7 Use of other elements. Other elements that are used by the SysMainGrpListCB CSU:
"sysMainTmi" - widget id of TMI area of sysMain form
"sysMainForm" - widget id of entry area of sysMain form

4.5.28.4.2.8 Logic flow. Single pass.

4.5.28.4.2.9 Data structures. None

4.5.28.4.2.10 Local data files or database. None

4.5.28.4.2.11 Limitations. None

4.5.28.5 CSU SysMainLocListCB

The SysMainLocListCB CSU handles the callback for bringing up the LocList screen.

4.5.28.5.1 CSU SysMainLocListCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.28.5.2 CSU SysMainLocListCB design.

The information identified below represents the detailed design of the SysMainLocListCB CSU.

4.5.28.5.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: None

4.5.28.5.2.2 Local data elements. None

4.5.28.5.2.3 Interrupts and signals. None

4.5.28.5.2.4 Algorithms. None

4.5.28.5.2.5 Error handling. None

4.5.28.5.2.6 Data conversion. None

4.5.28.5.2.7 Use of other elements. Other elements that are used by the SysMainLocListCB CSU:
"sysMainTmi" - widget id of TMI area of sysMain form
"sysMainForm" - widget id of entry area of sysMain form

4.5.28.5.2.8 Logic flow. Single pass.

4.5.28.5.2.9 Data structures. None

4.5.28.5.2.10 Local data files or database. None

4.5.28.5.2.11 Limitations. None

4.5.28.6 CSU SysMainLogoutCB

4.5.28.6.1 CSU SysMainLogoutCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.28.6.2 CSU SysMainLogoutCB design.

The information identified below represents the detailed design of the SysMainLogoutCB CSU.

4.5.28.6.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: None

4.5.28.6.2.2 Local data elements. None

4.5.28.6.2.3 Interrupts and signals. None

4.5.28.6.2.4 Algorithms. None

4.5.28.6.2.5 Error handling. None

4.5.28.6.2.6 Data conversion. None

4.5.28.6.2.7 Use of other elements. Other elements that are used by the SysMainLogoutCB CSU:
None

4.5.28.6.2.8 Logic flow. Single pass.

4.5.28.6.2.9 Data structures. None

4.5.28.6.2.10 Local data files or database. None

4.5.28.6.2.11 Limitations. None

4.5.28.7 CSU PutupSysMain

4.5.28.7.1 CSU PutupSysMain requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.28.7.2 CSU PutupSysMain design.

The information identified below represents the detailed design of the PutupSysMain CSU.

4.5.28.7.2.1 Input/output data elements.

INPUTS:
"w" contains widget id of button pressed
"client" contains client data
"call" contains X Windows data about the call reason

OUTPUTS: None

4.5.28.7.2.2 Local data elements. None

4.5.28.7.2.3 Interrupts and signals. None

4.5.28.7.2.4 Algorithms. None

4.5.28.7.2.5 Error handling. None

4.5.28.7.2.6 Data conversion. None

4.5.28.7.2.7 Use of other elements. Other elements that are used by the PutupSysMain CSU:
"sysMainTmi" - widget id of TMI area of sysMain form

"sysMainForm" - widget id of entry area of sysMain form
"tmiScreen" - widget id of current TMI
"smdScreen" - widget id of current entery screen

4.5.28.7.2.8 Logic flow. Single pass.

4.5.28.7.2.9 Data structures. None

4.5.28.7.2.10 Local data files or database. None

4.5.28.7.2.11 Limitations. None

4.5.29 Sub-Level CSC util.

The following paragraphs under this section describe the relationship of the util Sub-Level CSC in terms of data flow between the CSU's of this Sub-level CSC.

4.5.29.1 CSU init_selections

The init_selections CSU initializes selection menus.

4.5.29.1.1 CSU init_selections requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.29.1.2 CSU init_selections design.

The information identified below represents the detailed design of the init_selections CSU.

4.5.29.1.2.1 Input/output data elements.

INPUTS: None

OUTPUTS: None

4.5.29.1.2.2 Local data elements. None

4.5.29.1.2.3 Interrupts and signals. None

4.5.29.1.2.4 Algorithms. None

4.5.29.1.2.5 Error handling. None

4.5.29.1.2.6 Data conversion. None

4.5.29.1.2.7 Use of other elements. Other elements that are used by the init_selections CSU:

4.5.29.1.2.8 Logic flow. Single pass.

4.5.29.1.2.9 Data structures. None

4.5.29.1.2.10 Local data files or database. None

4.5.29.1.2.11 Limitations. None

4.5.29.2 CSU rotate

The rotate CSU handles the rotations for all of the selection menus.

4.5.29.2.1 CSU rotate requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.29.2.2 CSU rotate design.

The information identified below represents the detailed design of the rotate CSU.

4.5.29.2.2.1 Input/output data elements.

INPUTS:
"panel" - panel menu to be acted upon

OUTPUTS: None

4.5.29.2.2.2 Local data elements.
"argcnt" - keeps track of the number of arguments set for XtSetValues
"args" - the arguments set for XtSetValues
"argok" - a boolean which is used to check the success of the CONVERT call
"xmstr" - local to hold a XmString

4.5.29.2.2.3 Interrupts and signals. None

4.5.29.2.2.4 Algorithms. None

4.5.29.2.2.5 Error handling. None

4.5.29.2.2.6 Data conversion. None

4.5.29.2.2.7 Use of other elements. Other elements that are used by the rotate CSU:

4.5.29.2.2.8 Logic flow. Single pass.

4.5.29.2.2.9 Data structures. None

4.5.29.2.2.10 Local data files or database. None

4.5.29.2.2.11 Limitations. None

4.5.29.3 CSU moreAddress

The moreAddress CSU handles the more button calls for the address selection menus from all the reports.

4.5.29.3.1 CSU moreAddress requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.29.3.2 CSU moreAddress design.

The information identified below represents the detailed design of the moreAddress CSU.

4.5.29.3.2.1 Input/output data elements.

INPUTS:
"panel" - panel menu to be acted upon

OUTPUTS: None

4.5.29.3.2.2 Local data elements.
"argcnt" - keeps track of the number of arguments set for XtSetValues
"args" - the arguments set for XtSetValues
"argok" - a boolean which is used to check the success of the CONVERT call
"xmstr" - local to hold a XmString

4.5.29.3.2.3 Interrupts and signals. None

4.5.29.3.2.4 Algorithms. None

4.5.29.3.2.5 Error handling. None

4.5.29.3.2.6 Data conversion. None

4.5.29.3.2.7 Use of other elements. Other elements that are used by the moreAddress CSU:

4.5.29.3.2.8 Logic flow. Single pass.

4.5.29.3.2.9 Data structures. None

4.5.29.3.2.10 Local data files or database. None

4.5.29.3.2.11 Limitations. None

4.5.29.4 CSU moreLctn

The moreLctn CSU handles the more button calls for the location selection menus from all the reports.

4.5.29.4.1 CSU moreLctn requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.29.4.2 CSU moreLctn design.

The information identified below represents the detailed design of the moreLctn CSU.

4.5.29.4.2.1 Input/output data elements.

INPUTS:
"panel" - panel menu to be acted upon

OUTPUTS: None

4.5.29.4.2.2 Local data elements.
"argcnt" - keeps track of the number of arguments set for XtSetValues
"args" - the arguments set for XtSetValues
"argok" - a boolean which is used to check the success of the CONVERT call
"xmstr" - local to hold a XmString

4.5.29.4.2.3 Interrupts and signals. None

4.5.29.4.2.4 Algorithms. None

4.5.29.4.2.5 Error handling. None

4.5.29.4.2.6 Data conversion. None

4.5.29.4.2.7 Use of other elements. Other elements that are used by the moreLctn CSU:

4.5.29.4.2.8 Logic flow. Single pass.

4.5.29.4.2.9 Data structures. None

4.5.29.4.2.10 Local data files or database. None

4.5.29.4.2.11 Limitations. None

4.5.29.5 CSU string2utm

The string2utm CSU converts string versions of UTM locations into the UTM structure type.

4.5.29.5.1 CSU string2utm requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.5.29.5.2 CSU string2utm design.

The information identified below represents the detailed design of the string2utm CSU.

4.5.29.5.2.1 Input/output data elements.

INPUTS:
"string" - string version of location

OUTPUTS:
"UTM" - UTM structure with the location

4.5.29.5.2.2 Local data elements.
"length" - length of the string
"index" - counter
"valid" - boolean; is this a valid UTM

4.5.29.5.2.3 Interrupts and signals. None

4.5.29.5.2.4 Algorithms. None

4.5.29.5.2.5 Error handling. None

4.5.29.5.2.6 Data conversion. None

4.5.29.5.2.7 Use of other elements. Other elements that are used by the string2utm CSU:

4.5.29.5.2.8 Logic flow. Single pass.

4.5.29.5.2.9 Data structures. None

4.5.29.5.2.10 Local data files or database. None

4.5.29.5.2.11 Limitations. None

4.5.29.6 CSU UpdateEnvelope

The UpdateEnvelope CSU updates both envelope icons to reflect the current number of messages that are unread.

4.5.29.6.1 CSU UpdateEnvelope requirements.

The UpdateEnvelope CSU satisfies section 3.2.1.2.2.1.4 of the specific requirments of the DMCC system.

4.5.29.6.2 CSU UpdateEnvelope design.

The information identified below represents the detailed design of the UpdateEnvelope CSU.

4.5.29.6.2.1 Input/output data elements.

INPUTS: None

OUTPUTS: None

4.5.29.6.2.2 Local data elements.
"argcnt" - keeps track of the number of arguments set for XtSetValues
"args" - the arguments set for XtSetValues
"argok" - a boolean which is used to check the success of the CONVERT call
"xmstr" - local to hold a XmString

4.5.29.6.2.3 Interrupts and signals. None

4.5.29.6.2.4 Algorithms. None

4.5.29.6.2.5 Error handling. None

4.5.29.6.2.6 Data conversion. None

4.5.29.6.2.7 Use of other elements. Other elements that are used by the UpdateEnvelope CSU:
"numMsgLabel" - widget id of the Msgs screen envelope
"envLabel2" - widget id of the SysMain screen envelope

4.5.29.6.2.8 Logic flow. Single pass.

4.5.29.6.2.9 Data structures. None

4.5.29.6.2.10 Local data files or database. None

4.5.29.6.2.11 Limitations. None

4.5.30 Sub-Level CSC Builder Xcessory Functions.

The following paragraphs under this section describe the relationship of the Builder Xcessory Functions Sub-Level CSC in terms of data flow between the CSU's of this Sub-level CSC. The functions contained herein are generated by the tool Builder Xcessory and are not modified by the user, therefore there is no information on the design of the code.

4.5.30.1. CSU CreateAddrList

The CreateAddrList CSU creates the Address List entry screen.

4.5.30.2. CSU Createform requirements.

The Createform CSU creates the toplevel form screen.

4.5.30.3. CSU formTrim

The formTrim CSU creates the decorations on most of the screens.

4.5.30.4. CSU CreateFreeTxtForm

The CreateFreeTxtForm CSU creates the Free Text entry screen.

4.5.30.5. CSU CreateGrpListForm

The CreateGrpListForm CSU creates the CEOI/Group List entry screen.

4.5.30.6. CSU CreateLocListForm

The CreateLocListForm CSU creates the Location List entry screen.

4.5.30.7 CSU CreateLogonForm

The CreateLogonForm CSU creates the Logon screen.

4.5.30.8 CSU CreatemovcmdTmi

The CreatemovcmdTmi CSU creates the TMI portion of the Movcmd report generation screen.

4.5.30.9 CSU CreatemovcmdForm

The CreatemovcmdForm CSU creates the entry portion of the Movcmd report generation screen.

4.5.30.10 CSU CreateMsg1Form

The CreateMsg1Form CSU creates the Msg1screen.

4.5.30.11 CSU CreateMsgReadForm

The CreateMsgReadForm CSU creates the Message Read screen.

4.5.30.12 CSU CreatemtoTmi

The CreatemtoTmi CSU creates the TMI portion of the MTO report generation screen.

4.5.30.13 CSU CreatemtoForm

The CreatemtoForm CSU creates the entry portion of the MTO report generation screen.

4.5.30.14 CSU CreateReportForm

The CreateReportForm CSU creates the Reports screen.

4.5.30.15 CSU CreatereqtTmi

The CreatereqtTmi CSU creates the TMI portion of the Request report generation screen.

4.5.30.16 CSU CreatereqtForm

The CreatereqtForm CSU creates the entry portion of the Request report generation screen.

4.5.30.17 CSU CreatereuseTmi

The CreatereuseTmi CSU creates the TMI portion of the Reuse report generation screen.

4.5.30.18 CSU CreatereuseForm

The CreatereuseForm CSU creates the entry portion of the Reuse report generation screen.

4.5.30.19 CSU CreateshotTmi

The CreateshotTmi CSU creates the TMI portion of the Shot report generation screen.

4.5.30.20 CSU CreateshotForm

The CreateshotForm CSU creates the entry portion of the Shot report generation screen.

4.5.30.21 CSU CreatesplashTmi

The CreatesplashTmi CSU creates the TMI portion of the Splash report generation screen.

4.5.30.22 CSU CreatesplashForm

The CreatesplashForm CSU creates the entry portion of the Splash report generation screen.

4.5.30.23. CSU CreatespotTmi

The CreatespotTmi CSU creates the TMI portion of the Spot report generation screen.

4.5.30.24 CSU CreatespotForm

The CreatespotForm CSU creates the entry portion of the Spot report generation screen.

4.5.30.25 CSU CreateSysMainForm

The CreateSysMainForm CSU creates the Sys Main screen.

4.6 CSC pdu decode

The following paragraphs under this section describe the relationship of the pdu decode CSC in terms of data flow between the CSU's of this CSC and identifies all CSU interfaces that are external to the pdu decode CSC.

CSC pdu decode partially satisfies System Segment Specification requirements 3.2.1.2.1.3 and 3.2.1.2.2.1.

The following diagram depicts the structure of the pdu_decode CSC:

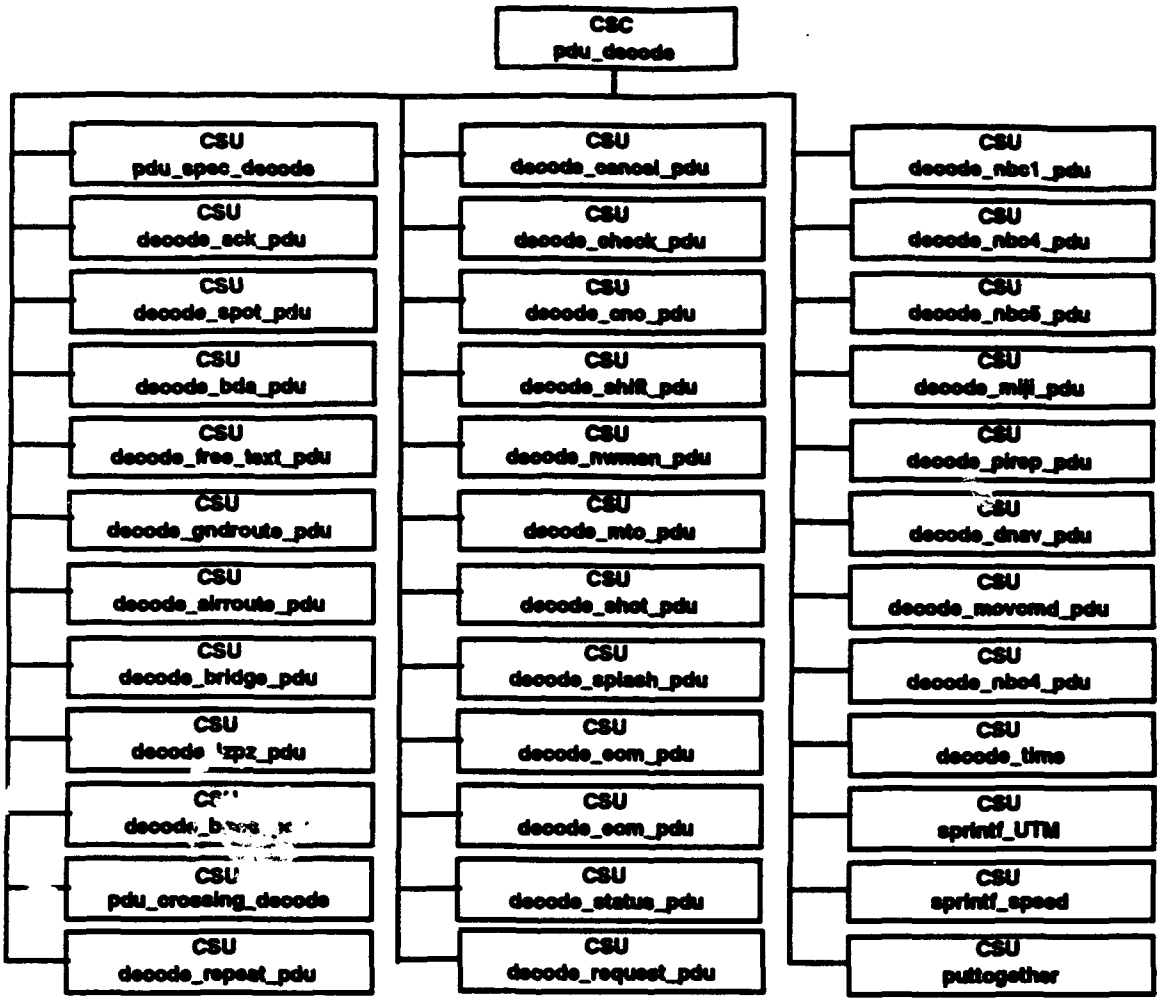


Figure 12 CSU pdu_decode Structure

4.6.1 CSU pdu_decode.

The pdu_decode CSU calls the appropriate function to decode the input message. The following paragraphs provide design information for this CSU.

4.6.1.1 CSU pdu_decode requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.6.1.2 CSU pdu_decode design.

The information identified below represents the detailed design of the pdu_decode CSU.

output = pdu_decode (input message pointer of DMCPDU_type)

a. Input/output data elements.

INPUTS:
"pdu" is a DMCPDU_type pointer containing the input message to be decoded.

OUTPUTS:
"dmc_buffer" is a static character buffer of size (18+15)x36 containing the decoded message to be returned.

b. Local data elements.
"hdr_str" is a character pointer to a buffer that contains the message header information.
"msg_str" is a character pointer to a buffer that contains the message information.
"header_common" is a Non_specific pointer that points to the common header structure.
"subtype" is an integer storage that contains the pdu subtype id.
"variation" is an integer storage that contains the pdu variation id.
"good_pdu" is a boolean flag of integer type.

c. Interrupts and signals. None

d. Algorithms.
decode and append the message variation type to ret_str based upon its subtype.
decode and append the following fields to ret_str in order:

Based upon the input message subtype and
variation field, invoke pdu_spec_decode() and one of the following :
1. decode_ack_pdu() with an Ack_type input message pointer.
2. decode_spot_pdu( ) with a Spot_type input message pointer.
3. decode_bda_pdu() with an BDA_type input message pointer
4. decode_free_text() wiht a FreeText_type input message pointer
5. decode_gndroute_pdu() with a Recon_type input message pointer
6. decode_airroute_pdu() with a Recon_type input message pointer
7. decode_bridge_pdu() with a Recon_type input message pointer
8. decode_lzpz_pdu() with a Recon_type input message pointer

9. decode_bpop_pdu() with a Recon_type input message pointer
10. decode_crossing_pdu() with a Recon_type input message pointer
11. decode_repeat_pdu() with an Artillery_type input message pointer
12. decode_cancel_pdu() with an Artillery_type input message pointer
13. decode_check_pdu() with an Artillery_type input message pointer
14. decode_cno_pdu() with an Artillery_type input message pointer
15. decode_shift_pdu() with an Artillery_type input message pointer
16. decode_nwmsn_pdu() an Artillery_type input message pointer
17. decode_mto_pdu() with an Artillery_type input message pointer
18. decode_shot_pdu() with an Artillery_type input message pointer
19. decode_splash_pdu() with an Artillery_type input message pointer
20. decode_eom_pdu() with an Artillery_type input message pointer

e. Error handling. Use of system 'printf' statement to output error message to console screen.

f. Data conversion. None

g. Use of other elements.

The following provide the string values of "Enumerated" types - character pointer:
decode_spot_label,
arget_enum_str,
activity_enum_str,
direction_enum_str,
obs_int_enum_str.

Library routines:
switch - case,
strcat.

CSUs:
pdu_spec_decode,
decode_ack_pdu,
decode_spot_pdu,
decode_bda_pdu,
decode_free_text_pdu,
decode_gndroute_pdu,
decode_airroute_pdu,
decode_bridge_pdu,
decode_lzpz_pdu,
decode_bpop_pdu,
decode_crossing_pdu,
decode_repeat_pdu,
decode_cancel_pdu,
decode_check_pdu,
decode_cno_pdu,
decode_shift_pdu,

decode_nwmsn_pdu,
decode_mto_pdu,
decode_shot_pdu,
decode_splash_pdu,
decode_eom_pdu,
decode_status_pdu,
decode_request_pdu,
decode_nbc1_pdu,
decode_nbc4_pdu,
decode_nbc5_pdu,
decode_miji_pdu,
decode_pirep_pdu,
decode_dnav_pdu,
decode_movcmd_pdu.

h. Logic flow.

```
map header_common template to Non_specific input message
    pointer;
retrieve subtype and variation value from input
    message(header_common template);
set good_pdu to true;
Switch on input message subtye
case Acknowlegement invoke
  pdu_spec_decode;
  decode_ack_pdu;
case Spot invoke
  pdu_spec_decode;
  decode_spot_pdu;
case Battle Damage Assessment invoke
  pdu_spec_decode;
  decode_bda_pdu;
case Free Text invoke
  pdu_spec_decode;
  decode_free-text_pdu;
case Reconaissance invoke
  switch on message variation
  case ground route  reconaissance report invoke
    pdu_spec_decode;
    decode_gndroute_pdu;
  case air route reconaissance report invoke
    pdu_spec_decode;
    decode_airroute_pdu;
  case bridge reconaissance report invoke
    pdu_spec_decode;
    decode_bridge_pdu;
  case landing zone/pickup zone invoke
```

```
        pdu_spec_decode;
        decode_lzpz_pdu;
      case BP/OP invoke
        pdu_spec_decode;
        decode_bpop_pdu;
      case crossing invoke
        pdu_spec_decode;
        decode_crossing_pdu;
    case Artillery invoke
      switch on header_common->Common.Variation
      case repeat invoke
        pdu_spec_decode;
        decode_repeat_pdu;
      case cancel invoke
        pdu_spec_decode;
        decode_cancel_pdu;
      case check invoke
        pdu_spec_decode;
        decode_check_pdu;
      case cno invoke
        pdu_spec_decode;
        decode_cno_pdu;
      case shift invoke
        pdu_spec_decode;
        decode_shift_pdu;
      case new mission invoke
        pdu_spec_decode;
        decode_nwmsn_pdu;
      case message to observer invoke
        pdu_spec_decode;
        decode_mto_pdu;
      case shot invoke
        pdu_spec_decode;
        decode_shot_pdu;
      case splash invoke
        pdu_spec_decode;
        decode_splash_pdu;
      case end of mission invoke
        pdu_spec_decode;
        decode_eom_pdu;
    case status invoke
      pdu_spec_decode;
      decode_status_pdu;
    case request invoke
      pdu_spec_decode;
      decode_request_pdu;
```

```
case Nuclear, Biological, Chemical invoke
  switch on variation
  case Nuclear, Biological, Chemical Type 1 invoke
    pdu_spec_decode;
    decode_nbc1_pdu;
  case Nuclear, Biological, Chemical Type 4 invoke
    pdu_spec_decode;
    decode_nbc4_pdu;
  case Nuclear, Biological, Chemical Type 5 invoke
    pdu_spec_decode;
    decode_nbc5_pdu;
case Meaconing, Intrusion, Jamming, Interface
  pdu_spec_decode;
  decode_miji_pdu;
case Pilot Report invoke
  pdu_spec_decode;
  decode_pirep_pdu;
case Down Air Vehicle invoke
  pdu_spec_decode;
  decode_dnav_pdu;
case Move Comand invoke
  pdu_spec_decode;
  decode_movcmd_pdu;
return pdu_spec_decode to hdr_str;
retunr decode message to msg_str;
display error message for invalid subtype;
append the decoded message hdr_str and msg_str strings to
      dmc_buffer;
return dmc_buffer to caller;
```

i. Data structures. None.

j. Local data files or database. None

k. Limitations. None

4.6.2 CSU pdu_spec_decode

The pdu_spec_decode CSU decodes the input message's header information. The following paragraphs provide design information for this CSU.

4.6.2.1 CSU pdu_spec_decode requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.6.2.2 CSU pdu_spec_decode design.

The information identified below represents the detailed design of the pdu_spec_decode CSU.

output = pdu_spec_decode(input message)

a. Input/output data elements.

INPUTS:
"pdu" is a DMCPDU_type pointer containing the input message to be decoded.

OUTPUTS:
"ret_str" is a character buffer of size18x36 containing the decoded message string to be returned.

b. Local data elements.
"str_ptr" is a static character pointer that points to a temporary decoded UTM message.
"temp_str" is a static unsigned char buffer of size 36 that contains the temporary decoded message.
"header_common" is a Non_specific pointer that points to the common header structure.
"temp" is a temporary storage that contains the integer value of a message field.
"subtype" is an integer storage that contains the pdu subtype id.
"variation" is an integer storage that contains the pdu variation id.

c. Interrupts and signals. None

d. Algorithms.
store a null in the first byte of ret_str.
decode and append the message variation type to ret_str based upon its subtype.
decode and append the following fields to ret_str in order:
message priority,

message sender CEOI,
message receiver,
message forward by,
message sent time/date,
message transmit location,
message transmit altitude in feet.

* append a carriage return to ret_str at the end of each field.

e. Error handling. None

f. Data conversion. None

g. Use of other elements.
The following provide the string values of "Enumerated" types - character pointer:
decode_recon_variation, decode_artillery_variation,
decode_nbc_variation,
decode_subtype,
decode_header_label.
Library routines:
sprintf,
strcat,
strncat.
CSUs:
sprintf_UTM,
decode_time.

h. Logic flow.
set header_common template to Non_specific * pdu;
retrieve message subtype and variation value from input message;
place a null in the first byte of ret_str;
switch on subtype
case Reconaissance pdu
append its variation to ret_str;
case Artillery pdu
append its variation to ret_str;
case Nuclear, Biological, Chemical type pdu
append its variation to ret_str;
others, invoke decode_subtype[subtype] to the
return buffer ret_str;

if message Priority is true
append "urgency" to ret_str;
else append "routine" to ret_str;
append a carriage return to ret_str;

append the Sender CEOI title, the Sender CEOI string, and a carriage return to ret_str;

append the Target CEOI title, the Target CEOI string, and a carriage return to ret_str;

append the Last Forward CEOI title, the Last Foward CEOI string, and a carriage return to ret_str;

invoke decode_time() to decode the Send time;
append the Send time title, the decoded Send time, and a carriage return to ret_str;

invoke sprintf_UTM() to decode the transmit location;
append the [transmit location title, the sprintf_UTM result, and a carraige return to ret_str.

convert Altitude into a string with "feet" as the unit;
append the Altitude title, the converted Altitude string, and a carriage return to ret_str;
return ret_str to caller.

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

4.6.3 CSU decode_ack_pdu

The decode_ack_pdu CSU decodes the Acknowledgement input message. The following paragraphs provide design information for this CSU.

4.6.3.1 CSU decode_ack_pdu requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.6.3.2 CSU decode_ack_pdu design.

The information identified below represents the detailed design of the decode_ack_pdu CSU.

Syntax for invoking this function:
output = decode_ack_pdu(pdu);

a. Input/output data elements.

INPUTS:
"pdu" is a Ack_type pointer containing the input message to be decoded.

OUTPUTS:
"ret_str" is a static character buffer of size18x36 containing the decoded message string to be returned.

b. Local data elements.
"temp_str" is a static unsigned char buffer of size 36 that contains the temporary decoded message.

c. Interrupts and signals. None

d. Algorithms.
store a null in the first byte of ret_str.
append a carriage return to ret_str.
decode and append the following fields to ret_str in order:
Acknowledge CEOI;
Original DTG;
Free Text Annotation String.

* append a carriage return to ret_str at the end of each field.

e. Error handling. None

f. Data conversion. None

g. Use of other elements.
The following provide the string values of "Enumerated" types - constant char:
decode_ack_label - two dimensional array of size 15x16,
Library routines:
sprintf,
strcat,
strncat.
CSUs:
decode_time.

h. Logic flow.
set ret_str to null;
append a carriage return to ret_str;

append the Acknowledge CEOI title, the Acknowledge CEOI string, and a carriage return to ret_str;

invoke decode_time to decode the Original DTG;
append the Original DTG title, the decode Original DTG string, and a carriage return to ret_str;

append nine carriage returns to ret_str(nine blank lines);
append the first 32 Free Text characters and a carriage to ret_str;
append the next 32 Free Text characters) and a carriage to ret_str;
return ret_str to caller.

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

4.6.4 CSU decode_spot_pdu

The decode_spot_pdu CSU decodes the Spot input message. The following paragraphs provide design information for this CSU.

4.6.4.1 CSU decode_spot_pdu requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.6.4.2 CSU decode_spot_pdu design.

The information identified below represents the detailed design of the decode_spot_pdu CSU.

Syntax for invoking this function:
output = decode_spot_pdu(pdu);

a. Input/output data elements.

INPUTS:
" pdu" is a Spot_type pointer containing the input message to be decoded.

OUTPUTS:
"ret_str" is a character buffer of size18x36 containing the decoded message string to be returned.

b. Local data elements.
"str_ptr" is a static character pointer that points to the temporary decoded UTM message.
"temp_str" is a static unsigned char buffer of size 36 that contains the temporary decoded message.
"temp" is a temporary storage that contains the integer value of a message field.

c. Interrupts and signals. None

d. Algorithms.
store a null in the first byte of ret_str.
append a carriage return to ret_str.
decode and append the following fields to ret_str in order:
Target Quantitiy,
Target Type,
Target Activity Type,
Target Speed,
Target Unit,
Target Location,
Target Sighted Time,
Observer Location,
Observer Intention,
Free Text Annotation String.

* append a carriage return to ret_str at the end of each field.

e. Error handling. None

f. Data conversion. None

g. Use of other elements.
The following provide the string values of "Enumerated" types - character pointer:
decode_spot_label,
arget_enum_str,

activity_enum_str,
direction_enum_str,
obs_int_enum_str.

Library routines:

sprintf,
strcat,
strncat.

CSUs:

puttogether,
sprintf_speed,
sprintf_UTM,
decode_time.

h. Logic flow.

```
set ret_str to null;
append a carriage return to ret_str;

convert Target Quantity into a string;
append the Target Quantity title and the Target Quantity string to
      ret_str with a carriage return at the end;

if TargetType is greater than or equal to zero and it is less than the
      number of possible Target Type.
   decode Target Type;
else
    decode Target Type with an invalid_entry string;
invoke puttogether() to append the Target Type title, the decoded
      Target Type string and a carriage return to ret_str;

if Target Activity is greater than or equal to zero and it is less than the
      number of possible Target Activity Type.
   decode Target Activity;
else
    decode Target Type with an invalid_entry string;
invoke puttogether() to append the Target Activity title, the decoded
      Target Activity string and a carriage return to ret_str;

invoke sprintf_speed() to decode the Target Speed;
append the Target Speed title, the decoded Target Speed string and a
      carriage return to ret_str;

if Target Direction is greater than or equal to zero and it is less than the
      number of possible Target Direction.
   decode Target Direction;
else
    decode Target Direction with an invalid_entry string;
```

invoke puttogether() to append the Target Direction title, the decoded Target Direction string and a carriage return to ret_str;

append the Target Unit title, the decode Target Unit and a carriage return to ret_str;

invoke sprintf_UTM() to decode Target Location;
append the Target Location title, the decoded Target Location string and a carriage return to ret_str;

invoke decode_time() to decode Time Sighted;
append the Target Time Sighted title, the decoded Time Sighted string and a carriage return to ret_str;

invoke sprintf_UTM() to decode Observer Location;
append the Observer Location title, the decoded Observer Location string and a carriage return to ret_str;

if Observer Intention is greater than or equal to zero and it is less than the number of possible Observer Intentions.
decode Observer Intention;
else
decode Observer Intention with an invalid_entry string;
invoke puttogether() to append the Observer Intention title, the decoded Observer Intention string and a carriage return to ret_str;

append a carriage retun for a blank line;
append the first 32 Free Text characters and a carriage to ret_str;
append the next 32 Free Text characters) and a carriage to ret_str;

return ret_str.

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

4.6.5 CSU decode_bda_pdu

The decode_bda_pdu CSU decodes the Battle Damage Assessment input message. The following paragraphs provide design information for this CSU.

4.6.5.1 CSU decode_bda_pdu requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.6.5.2 CSU decode_bda_pdu design.

The information identified below represents the detailed design of the decode_bda_pdu CSU.

Syntax for invoking this function:
output = decode_bda_pdu(pdu);

a. Input/output data elements.

INPUTS:
" pdu" is a Battle Damage Assessment (BDA_type) type pointer containing the input message to be decoded.

OUTPUTS:
"ret_str" is a static character buffer of size18x36 containing the decoded message string to be returned.

b. Local data elements.
"temp_str" is a static unsigned char buffer of size 36 that contains the tempo· y decoded message.
"temp" is a tempo· storage that contains the integer value of a message field.

c. Interrupts and signals. None

d. Algorithms.
store a null in the first byte of ret_str.
append a carriage return to ret_str.
decode and append the following fields to ret_str in order:
Strike Start Time,
Strike End Time,
Target Category,
Target Type,
# of Target Destroyed,
# of Percent Coverage,
Free Text Annotation String.

*append a carriage return at the end of each field.

e. Error handling. None

f. Data conversion. None

g. Use of other elements.

The following provide the string values of "Enumerated" types -
character pointer:
decode_bda_str,
target_enum_str,
percent_cover_enum_str.

Library routines:
sprintf,
strcat,
strncat.

CSUs:
puttogether,
decode_time.

h. Logic flow.

set ret_str to null;
append a carriage return to ret_str;

invoke decode_time() to decode the start of Strike Time;
append the Strike Time Start title, the decode time and a crriage return to ret_str;

invoke decode_time() to decode the end of Strike Time;
append the Strike Time End title, the decode time and a crriage return to ret_str;

append the Target Category title and a carriage return to ret_str.

if TargetType is greater than or equal to zero and it is less than the number of possible Target Type.
decode Target Type;
else
decode Target Type with an invalid_entry string;
invoke puttogether() to append the Target Type title, the decoded Target Type string and a carriage return to ret_str;

convert the Target Destroyed into a string;
append the Target Destroyed title, the Target Destroyed string and a carriage reture to ret_str;

if Percent Coverage type is greater than or equal to zero and it is less than the number of possible Percent Coverage type.
decode Percent Coverage type;
else
decode Percent Coverate with an invalid_entry string;
invoke puttogether() to append the Percent Coverage title, the decoded Percent Coverage string and a carriage return to ret_str;

append five carriage return to ret_str(for five blank lines);
append the first 32 Free Text characters and a carriage to ret_str;
append the next 32 Free Text characters) and a carriage to ret_str;
return ret_str to caller.

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

4.6.6 CSU decode_free_text_pdu

The decode_free_text_pdu CSU decodes the Free Text input message. The following paragraphs provide design information for this CSU.

4.6.6.1 CSU decode_free_text_pdu requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.6.6.2 CSU decode_free_text_pdu design.

The information identified below represents the detailed design of the decode_free_text_pdu CSU.

Syntax for invoking this function:
output = decode_free_text_pdu(pdu);

a. Input/output data elements.

INPUTS:
" pdu" is a Free Text (FreeText_type) type pointer containing the input message to be decoded.

OUTPUTS:
"ret_str" is a static character buffer of size18x36 containing the decoded message string to be returned.

b. Local data elements. None

c. Interrupts and signals. None

d. Algorithms.
store a null in the first byte of ret_str.
decode and append the Free Text string to ret_str;

e. Error handling. None

f. Data conversion. None

g. Use of other elements.
Data elements: None
Library routines:
sprintf,
strcat,
strncat.
CSUs: None

h. Logic flow.
set ret_str to null;
append the Free Text string and a carriage return to ret_str;
append the first 32 Free Text characters and a carriage to ret_str;
append the next 32 Free Text characters) and a carriage to ret_str;
return ret_str to caller.

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

4.6.7 CSU decode_gndroute_pdu

The decode_gndroute_pdu CSU decodes the Ground Route input message. The following paragraphs provide design information for this CSU.

4.6.7.1 CSU decode_gndroute_pdu requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.6.7.2 CSU decode_gndroute_pdu design.

The information identified below represents the detailed design of the decode_gndroute_pdu CSU.

Syntax for invoking this function:
output = decode_bda_pdu(pdu);

a. Input/output data elements.

INPUTS:
" pdu" is a ground route reconaissance(Recon _type) type pointer containing the input message to be decoded.

OUTPUTS:
"ret_str" is a static character buffer of size18x36 containing the decoded message string to be returned.

b. Local data elements.
"temp_str" is a static unsigned char buffer of size 36 that contains the temporary decoded message.
"temp" is a temporary storage that contains the integer value of a message field.
"temp_char" is a temporary unsigned character storage.

c. Interrupts and signals. None

d. Algorithms.
store a null in the first byte of ret_str.
append a carriage return to ret_str.
decode and append the following fields to ret_str in order:
Enemy Activity,
Clssification Formula,
Classification Formula Information,
Route ID,
Free Text Annotation String.

*append a carriage return at the end of each field.

e. Error handling. None

f. Data conversion. None

g. Use of other elements.
The following provide the string values of "Enumerated" types - character pointer:
decode_gndroute,
classif_enum_str,
activity_enum_str,
Library routines:
sprintf,
strcat,
strncat.
CSUs:
puttogether().

h. Logic flow.
set ret_str to null;
append a carriage return to ret_str;

if Enemy Activity is greater than or equal to zero and it is less than the number of possible Enemy Activity Type.
decode Enemy Activity Type;
else
decode Enemy Activity Type with an invalid_entry string;
invoke puttogether() to append the Enemy Activity title, the decoded Enemy Activity Type string and a carriage return to ret_str;

if Classification Formula is greater than or equal to zero and it is less than the number of possible Classification Formula Type.
decode Classification Formula Type;
else
decode Classification Formula Type with an invalid_entry string;
invoke puttogether() to append the Classification Formula title, the decoded Classification Formula Type string and a carriage return to ret_str;

append the Classification Information title, the Classification Formular Information string, and a carriage return to ret_str;

append the Route ID title, the Route ID string, and a carraige return to ret_str;

append seven carriage return to ret_str(seven blank lines);
concatenate the first line(32 characters) of the

pdu->Common.FreeText and a carriage into ret_str;
concatenate the next line(32 characters) of the
pdu->Common.FreeText + 32 and a carriage into ret_str;

return ret_str to caller.

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

4.6.8 CSU decode_airroute_pdu

The decode_airroute_pdu CSU decodes the Air Route input message. The following paragraphs provide design information for this CSU.

4.6.8.1 CSU decode_airroute_pdu requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.6.8.2 CSU decode_airroute_pdu design.

The information identified below represents the detailed design of the decode_airroute_pdu CSU.

Syntax for invoking this function:
output = decode_airroute_pdu(pdu);

a. Input/output data elements.

INPUTS:
"pdu" is a airroute_type pointer containing the input message to be decoded.

OUTPUTS:
"ret_str" is a character buffer of size18x36 containing the decoded message string to be returned.

b. Local data elements.
"temp_str" is a static unsigned char buffer of size 36 that contains the temporary decoded message.

"temp" is a temporary storage that contains the integer value of a message field.

c. Interrupts and signals. None

d. Algorithms.
place a null in the first byte of ret_str.
append a carriage return to ret_str.
decode and append the following fields to ret_str in order:
Enemy Activity,
Obstacles,
Route_ID,
Free Text Annotation String.

* append a carriage return to ret_str at the end of each field.

e. Error handling. None

f. Data conversion. None

g. Use of other elements.
The following provide the string values of "Enumerated" types - character pointer:
decode_airroute,
activity_enum_str,
obstacle_enum_str.
Library routines:
sprintf,
strcat,
strncat.
CSUs:
puttogether.

h. Logic flow.
set ret_str to null;
append a carriage return to ret_str(one blank line);

if Enemy Activity is greater than or equal to zero and it is less than the number of possible Enemy Activity Type.
decode Enemy Activity Type;
else
decode Enemy Activity Type with an invalid_entry string;
invoke puttogether() to append the Enemy Activity title, the decoded Enemy Activity Type string and a carriage return to ret_str;

if Obstacles is greater than or equal to zero and it is less than the number of possible Obstacle Type.
decode Obstacle Type;
else
decode Obstacle Type with an invalid_entry string;
invoke puttogether() to append the Obstacle title, the decoded Obstacle Type string and a carriage return to ret_str;

append the Route ID title, the Route ID string to ret_str, and a carriage return to ret_str;

append six carriage retun to ret_str(for six blank lines);
append the first 32 Free Text characters and a carriage to ret_str;
append the next 32 Free Text characters) and a carriage to ret_str;
return ret_str to caller.

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

4.6.9 CSU decode_bridge_pdu

The decode_bridge_pdu CSU decodes the Bridge Reconaissance input message. The following paragraphs provide design information for this CSU.

4.6.9.1 CSU decode_bridge_pdu requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.6.9.2 CSU decode_bridge_pdu design.

The information identified below represents the detailed design of the decode_bridge_pdu CSU.

Syntax for invoking this function:
output = decode_bridge_pdu(pdu);

a. Input/output data elements.

INPUTS:
"pdu" is a Recon_type pointer containing the input message to be decoded.

OUTPUTS:
"ret_str" is a character buffer of size18x36 containing the decoded message string to be returned.

b. Local data elements.
"temp_str" is a static unsigned char buffer of size 36 that contains the temporary decoded message.
"temp" is a temporary storage that contains the integer value of a message field.

c. Interrupts and signals. None

d. Algorithms.
store a null in the first byte of ret_str.
decode and append the following fields to ret_str in order:
Bridge Type,
Damage,
Spans,
Construction Material,
Bridge Length,
Bridge Width,
Bridge Height,
Bridge Under,
Bridge Construction Description,
Bridge ID,
Bridge Span Length,
Bridge Load Class,
Free Text Annotation String.

* append a carriage return to ret_str at the end of each field.

e. Error handling. None

f. Data conversion. None

g. Use of other elements.
The following provide the string values of "Enumerated" types - character pointer:
decode_bridge,
bridge_enum_str,
bridge_damage_enum_str,
bridge_material_enum_str,

Library routines:
sprintf,
strcat,
strncat.
CSUs:
puttogether.

h. Logic flow.
set ret_str to null;
if Bridge Type is greater than or equal to zero and it is less than the number of possible Bridge Type.
decode Bridge Type;
else
decode Bridge Type with an invalid_entry string;
invoke puttogether() to append the "BRIDGE TYPE" title, the decoded Bridge Type string and a carriage return to ret_str;

if Bridge Damage Type is greater than or equal to zero and it is less than the number of possible Bridge Damage Type.
decode Bridge Damage Type;
else
decode Bridge Damage Type with an invalid_entry string;
invoke puttogether() to append the Damage title, the decoded Bridge Damage Type string and a carriage return to ret_str;

convert the Bridge Spans into a string;
append the Spans title, the Bridge Spans string, and a carriage return to ret_str;

if Bridge Construction Material Type is greater than or equal to zero and it is less than the number of possible Bridge Construction Material Type.
decode Bridge Construction Material Type;
else
decode Bridge Construction Material Type with an invalid_entry string;
invoke puttogether() to append the Construction Material title, the decoded Bridge Construction Material Type string and a carriage return to ret_str;

append the Length title, the Bridge Length string, and a carriage return into ret_str;

append the Width title, the Bridge Width string and a carriage return into ret_str;

append the Height title, the Bridge Height string and a carriage return into ret_str;

append the Under title, the Bridge Under string and a carriage return into ret_str;

append the Construction Description title, the Bridge Construction Description string and a carriage return into ret_str;

append the Bridge ID title, the Bridge ID string and a carriage return into ret_str;

append the Span Length title, the Bridge Span Length string and a carriage return into ret_str;

append the Load Class title, the Bridge Load Class string and a carriage return into ret_str;

append the first 32 Free Text characters and a carriage to ret_str;
append the next 32 Free Text characters) and a carriage to ret_str;
return ret_str to caller.

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

4.6.10 CSU decode_lzpz_pdu

The decode_lzpz_pdu CSU decodes the Landing/Pickup Zone Reconaissance input message. The following paragraphs provide design information for this CSU.

4.6.10.1 CSU decode_lzpz_pdu requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.6.10.2 CSU decode_lzpz_pdu design.

The information identified below represents the detailed design of the decode_lzpz_pdu CSU.

Syntax for invoking this function:
output = decode_lzpz_pdu(pdu);

a. Input/output data elements.

INPUTS:
" pdu" is a Recon_type pointer containing the input message to be decoded.

OUTPUTS:
"ret_str" is a character buffer of size18x36 containing the decoded message string to be returned.

b. Local data elements.
"temp_str" is a static unsigned char buffer of size 36 that contains the temporary decoded message.
"temp" is a temporary storage that contains the integer value of a message field.

c. Interrupts and signals. None

d. Algorithms.
store a null in the first byte of ret_str.
append a carriage return to ret_str.
decode and append the following fields to ret_str in order:
Activity Likely Type,
Obstacles Type,
Obstacles Description,
Landing/Pickup Zone ID,
Landing/Pickup Zone Size,
Free Text Annotation String.

* append a carriage return to ret_str at the end of each field.

e. Error handling. None

f. Data conversion. None

g. Use of other elements.
The following provide the string values of "Enumerated" types - character pointer:
decode_lzpz,
activity_enum_str,
obstacle_enum_str,
Library routines:

sprintf,
strcat,
strncat.

CSUs:
puttogether.

h. Logic flow.
set ret_str to null;
append a carriage return to ret_str(a blank line);

if Activity Likely Type is greater than or equal to zero and it is less than the number of possible Activity Likely Type.
decode Activity Likely Type;
else
decode Activity Likely Type with an invalid_entry string;
invoke puttogether() to append the Activity Likely title, the decoded Activity Likely Type string and a carriage return to ret_str;

if Obstacle is greater than or equal to zero and is less than the number of possible ObstacleType.
convert Obstacle into a string;
else
decode Obstacle Type with an invalid_entry string;
invoke puttogether() to append the Obstacles title, the decoded Obstacle Type string and a carriage return to ret_str;

append the Obstacle Description title, the Obstacle Description string, and a carriage return to ret_str;

append the LZ/PZ ID title, the Loading/Pickup Zone ID string, and a carriage return to ret_str;

append the LZ/PZ Size title, the Loading/Pickup Zone Size string, and a carriage return to ret_str;

append six carriage returns to ret_str(six blank lines);
append the first 32 Free Text characters and a carriage to ret_str;
append the next 32 Free Text characters) and a carriage to ret_str;
return ret_str to caller.

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

4.6.11 CSU decode_bpop_pdu

The decode_bpop_pdu CSU decodes the bpop Reconaissance input message. The following paragraphs provide design information for this CSU.

4.6.11.1 CSU decode_bpop_pdu requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.6.11.2 CSU decode_bpop_pdu design.

The information identified below represents the detailed design of the decode_bpop_pdu CSU.

Syntax for invoking this function:
output = decode_bpop_pdu(pdu);

a. Input/output data elements.

INPUTS:
" pdu" is a Recon_type pointer containing the input message to be decoded.

OUTPUTS:
"ret_str" is a character buffer of size18x36 containing the decoded message string to be returned.

b. Local data elements.
"temp_str" is a static unsigned char buffer of size 36 that contains the temporary decoded message.
"temp" is a temporary storage that contains the integer value of a message field.

c. Interrupts and signals. None

d. Algorithms.
store a null in the first byte of ret_str.
append a carriage return to ret_str.
decode and append the following fields to ret_str in order:
Enemy Activity Type,
Obstacles Type,
Obstacles Description,
BP/OP ID,

BP/OP Size,
BP/OP Axis,
Free Text Annotation String.

* append a carriage return to ret_str at the end of each field.

e. Error handling. None

f. Data conversion. None

g. Use of other elements.
The following provide the string values of "Enumerated" types - character pointer:
decode_bpop,
activity_enum_str,
obstacle_enum_str,
Library routines:
sprintf,
strcat,
strncat.
CSUs:
puttogether.

h. Logic flow.
set ret_str to null;
append a carriage return to ret_str(a blank line);

if Enemy Activity Type is greater than or equal to zero and it is less than the number of possible Enemy Activity Type.
decode Enemy Activity Type;
else
decode Enemy Activity Type with an invalid_entry string;
invoke puttogether() to append the Activity Likely title, the decoded Enemy Activity Type string and a carriage return to ret_str;

if Obstacle is greater than or equal to zero and is less than the number of possible ObstacleType.
convert Obstacle into a string;
else
decode Obstacle Type with an invalid_entry string;
invoke puttogether() to append the Obstacle title, the decoded Obstacle Type string and a carriage return to ret_str;

append the Obstacle Description title, the Obstacle Description string, and a carriage return to ret_str;

append the BP/OP ID title, the BP/OP ID string, and a carriage return to ret_str;

append the BP/OP Size title, the BP/OP Size string, and a carriage return to ret_str;

append the Axis title, the BP/OP Axis string, and a carriage return to ret_str;

append five carriage returns to ret_str(five blank lines);
append the first 32 Free Text characters and a carriage to ret_str;
append the next 32 Free Text characters) and a carriage to ret_str;
return ret_str to caller.

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

4.6.12 CSU decode_crossing_pdu

The decode_crossing_pdu CSU decodes the Crossing Reconaissance input messag.. The following paragraphs provide design information for this CSU.

4.6.12.1 CSU decode_crossing_pdu requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.6.12.2 CSU decode_crossing_pdu design.

The information identified below represents the detailed design of the decode_crossing_pdu CSU.

Syntax for invoking this function:
output = decode_crossing_pdu(pdu);

a. Input/output data elements.

INPUTS:
" pdu" is a Recon_type pointer containing the input message to be decoded.

OUTPUTS:
"ret_str" is a character buffer of size18x36 containing the decoded message string to be returned.

b. Local data elements. None

c. Interrupts and signals. None

d. Algorithms.
store a null in the first byte of ret_str.
append a carriage return to ret_str.
decode and append the following fields to ret_str in order:
Bank Slop Entry,
Bank Slop Exit,
Crossing Length,
Crossing Width,
Crossing Depth,
Current Flow,
Crossing ID,
Free Text Annotation String.

* append a carriage return to ret_str at the end of each field.

e. Error handling. None

f. Data conversion. None

g. Use of other elements.
The following provide the string values of "Enumerated" types - character pointer:
decode_crossing.
Library routines:
sprintf,
strcat,
strncat.
CSUs:
None

h. Logic flow.
set ret_str to null;
append a carriage return to ret_str(a blank line);

append the Bank Slope Entry title, the Bank Slope Entry string, and a carriage return to ret_str;

append the Bank Slope Exit title, the Bank Slope Exit string, and a carriage return to ret_str;

append the Crossing Length title, the Crossing Length string, and a carriage return to ret_str;

append the Crossing Width title, the Crossing Width string, and a carriage return to ret_str;

append the Crossing Depth title, the Crossing Depth string, and a carriage return to ret_str;

append the Current Flow title, the Current Flow string, and a carriage return to ret_str;

append the Current ID title, the Crossing ID string, and a carriage return to ret_str;

append four carriage returns to ret_str(four blank lines);
append the first 32 Free Text characters and a carriage to ret_str;
append the next 32 Free Text characters) and a carriage to ret_str;
return ret_str to caller.

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

4.6.14 CSU decode_repeat_pdu

The decode_repeat_pdu CSU decodes the Repeat Artillery input message. The following paragraphs provide design information for this CSU.

4.6.14.1 CSU decode_repeat_pdu requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.6.14.2 CSU decode_repeat_pdu design.

The information identified below represents the detailed design of the decode_repeat_pdu CSU.

Syntax for invoking this function:
output = decode_repeat_pdu(pdu);

a. Input/output data elements.

INPUTS:
" pdu" is a Artillery_type pointer containing the input message to be decoded.

OUTPUTS:
"ret_str" is a character buffer of size18x36 containing the decoded message string to be returned.

b. Local data elements.
"temp_str" is a static unsigned char buffer of size 36 that contains the temporary decoded message.
"temp" is a temporary storage that contains the integer value of a message field.

c. Interrupts and signals. None

d. Algorithms.
store a null in the first byte of ret_str.
append five carriage return to ret_str.
decode and append the following fields to ret_str in order:
repeat message,
Free Text Annotation String.

* append a carriage return to ret_str at the end of each field.

e. Error handling. None

f. Data conversion. None

g. Use of other elements.
The following provide the string values of "Enumerated" types -
character pointer:
decode_repeat.
Library routines:
sprintf,
strcat,
strncat.
CSUs:
None.

h. Logic flow.
set ret_str to null;
append five carriage returns to ret_str;

append the "REPEAT" message string at the center of a line and a carriage return to ret_str;

append six carriage returns to ret_str(six blank lines);
append the first 32 Free Text characters and a carriage to ret_str;
append the next 32 Free Text characters) and a carriage to ret_str;
return ret_str to caller.

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

4.6.15 CSU decode_pirep_pdu

The decode_pirep_pdu CSU decodes the Pilot Report input message. The following paragraphs provide design information for this CSU.

4.6.15.1 CSU decode_pirep_pdu requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.6.15.2 CSU decode_pirep_pdu design.

The information identified below represents the detailed design of the decode_pirep_pdu CSU.

Syntax for invoking this function:
output = decode_pirep_pdu(pdu);

a. Input/output data elements.

INPUTS:
"pdu" is a PIREP_type pointer containing the input message to be decoded.

OUTPUTS:
"ret_str" is a character buffer of size18x36 containing the decoded message string to be returned.

b. Local data elements.
"temp_str" is a static unsigned char buffer of size 36 that contains the temporary decoded message.
"temp" is a temporary storage that contains the integer value of a message field.

c. Interrupts and signals. None

d. Algorithms.
store a null in the first byte of ret_str.
append a carriage return to ret_str.
decode and append the following fields to ret_str in order:
Visibility,
Restrictions,
Cloud Cover,
Cloud Base,
Cloud Top,
Outside Air Temperature,
Barometer,
Wind Speed,
Wind Direction,
Turbulence Frequency,
Turbulence Intensity,
Icing Intensity,
Icing Type,
Free Text Annotation String.

* append a carriage return to ret_str at the end of each field.

e. Error handling. None

f. Data conversion. None

g. Use of other elements.
The following provide the string values of "Enumerated" types - character pointer:
decode_pirep,
visibility_enum_str,
restriction_enum_str,
cloud_enum_str,
direction_enum_str,
turbfreq_enum_str,

turbint_enum_str,
iceint_enum_str,
icing_enum_str.

Library routines:
sprintf,
strcat,
strncat.

CSUs:
None.

h. Logic flow.

set ret_str to null;
append a carriage return to ret_str(a blank line);

if Visibility Type is greater than or equal to zero and is less than the number of possible Visibility Type.
decode Visibility Type;
else
decode Visibility Type with an invalid_entry string;
append the Visibility title, the decoded Visibiltity Type, and a space to ret_str;

if Restriction Type is greater than or equal to zero and is less than the number of possible RestrictionType.
decode Restriction Type into a string;
else
decode Restriction Type with an invalid_entry string;
append the decoded RestrictionType string and a carriage return to ret_str;

if Cloud Cover Type is greater than or equal to zero and is less than the number of possible Cloud Cover Type.
decode Cloud Cover Type into a string;
else
decode Cloud Cover Type with an invalid_entry string;
appends the Clouds title, the decoded Cloud Cover Type string and a comma with a space to ret_str;

convert Cloud Base into a string;
append the Cloud Base title, the Cloud Base string, and a comma with a space to ret_str;

convert Cloud Top into a string;
append the Cloud Top title, the Cloud Top string, and two carriage returns to ret_str;

```
convert Outside Air Temperature into a string with unit "C";
append the Outside Air Temperature title and the Outside Air
      Temperature string to ret_str;

convert Barometer into a string with two spaces after the decimal and
      unit """;
append the Barometer title, the Barometer string, and two carriage
      returns to ret_str;

convert Wind Speed into a string with unit "KNOTS";
append the Wind title, the converted Wind Speed string, and a comma
      with a space to ret_str;

if Wind Direction is greater than or equal to zero and is less than the
      number of possible Wind Speed.
   decode Wind Speed into a string;
else
   decode Wind Speed with an invalid_entry string;
append the decoded Wind Speed string and two carriage returns to
      ret_str;

if Turbulence Frequency is greater than or equal to zero and is less than
      the number of possible Turbulence Frequency.
   decode Turbulence Frequency into a string;
else
   decode Turbulence Frequency with an invalid_entry string;
append the decoded Turbulence Frequency string and a comma to
      ret_str;

if Turbulence Intensity Type is greater than or equal to zero and is less
      than the number of possible Turbulence Intensity Type.
   decode Turbulence Intensity Type into a string;
else
   decode Turbulence Intensity Type with an invalid_entry string;
append the decoded Turbulence Intensity Type string, the Turbulence
      title, and two carriage returns to ret_str;

if Icing Intensity is greater than or equal to zero and is less than the
      number of possible Icing Intensity.
   decode Icing Intensity into a string;
else
   decode Icing Intensity with an invalid_entry string;
append the decoded Icing Intensity string and a space to ret_str;

if Icing Type is greater than or equal to zero and is less than the number
      of possible Icing Type.
```

decode Icing Type into a string;
else
decode Icing Type with an invalid_entry string;
append the decoded IcingType string, the Icing title, and two carriage returns to ret_str;

append the first 32 Free Text characters and a carriage to ret_str;
append the next 32 Free Text characters) and a carriage to ret_str;
return ret_str to caller.

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

4.6.16 CSU decode_cancel_pdu

The decode_cancel_pdu CSU decodes the Cancel Artillery input message. The following paragraphs provide design information for this CSU.

4.6.16.1 CSU decode_cancel_pdu requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.6.16.2 CSU decode_cancel_pdu design.

The information identified below represents the detailed design of the decode_cancel_pdu CSU.

Syntax for invoking this function:
output = decode_cancel_pdu(pdu);

a. Input/output data elements.

INPUTS:
" pdu" is a Artillery_type pointer containing the input message to be decoded.

OUTPUTS:
"ret_str" is a character buffer of size18x36 containing the decoded message string to be returned.

b. Local data elements. None

c. Interrupts and signals. None

d. Algorithms.
store a null in the first byte of ret_str.
append five carriage returns to ret_str.
decode and append the following fields to ret_str in order:
cancel message string;
Free Text Annotation String.

* append a carriage return to ret_str at the end of each field.

e. Error handling. None

f. Data conversion. None

g. Use of other elements.
The following provide the string values of "Enumerated" types - character pointer:
decode_cancel.
Library routines:
sprintf,
strcat,
strncat.
CSUs:
None.

h. Logic flow.
set ret_str to null;
append five carriage return to ret_str(five blank lines);

append "CANCEL" message string to the center of a line and a carriage return to ret_str;

append six carriage returns to ret_str(six blank lines);
append the first 32 Free Text characters and a carriage to ret_str;
append the next 32 Free Text characters) and a carriage to ret_str;
return ret_str to caller.

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

4.6.17 CSU decode_check_pdu

The decode_check_pdu CSU decodes the Check Artillery input message. The following paragraphs provide design information for this CSU.

4.6.17.1 CSU decode_check_pdu requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.6.17.2 CSU decode_check_pdu design.

The information identified below represents the detailed design of the decode_check_pdu CSU.

Syntax for invoking this function:
output = decode_check_pdu(pdu);

a. Input/output data elements.

INPUTS:
" pdu" is a Artillery_type pointer containing the input message to be decoded.

OUTPUTS:
"ret_str" is a character buffer of size18x36 containing the decoded message string to be returned.

b. Local data elements. None

c. Interrupts and signals. None

d. Algorithms.
store a null in the first byte of ret_str.
append five carriage returns to ret_str.
decode and append the following fields to ret_str in order:
check message string,
Free Text Annotation String.

* append a carriage return to ret_str at the end of each field.

e. Error handling. None

f. Data conversion. None

g. Use of other elements.
The following provide the string values of "Enumerated" types - character pointer:
decode_check.
Library routines:
sprintf,
strcat,
strncat.
CSUs:
None.

h. Logic flow.
set ret_str to null;
append five carriage returns to ret_str(five blank lines);

append "CHECK" message string at the center of a line and a carriage return to ret_str;

append six carriage returns to ret_str(six blank lines);
append the first 32 Free Text characters and a carriage to ret_str;
append the next 32 Free Text characters) and a carriage to ret_str;
return ret_str to caller.

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

4.6.18 CSU decode_cno_pdu

The decode_cno_pdu CSU decodes the Can Not Observe Artillery input message. The following paragraphs provide design information for this CSU.

4.6.18.1 CSU decode_cno_pdu requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.6.18.2 CSU decode_cno_pdu design.

The information identified below represents the detailed design of the decode_cno_pdu CSU.

Syntax for invoking this function:
output = decode_cno_pdu(pdu);

a. Input/output data elements.

INPUTS:
" pdu" is a Artillery_type pointer containing the input message to be decoded.

OUTPUTS:
"ret_str" is a character buffer of size18x36 containing the decoded message string to be returned.

b. Local data elements.
"temp_str" is a static unsigned char buffer of size 36 that contains the temporary decoded message.
"temp" is a temporary storage that contains the integer value of a message field.

c. Interrupts and signals. None

d. Algorithms.
store a null in the first byte of ret_str.
append a carriage return to ret_str.
decode and append the following fields to ret_str in order:
Mission ID,
Target ID,
Mission Status,
Can Not Observe Message String,
Free Text Annotation String.

* append a carriage return to ret_str at the end of each field.

e. Error handling. None

f. Data conversion. None

g. Use of other elements.
The following provide the string values of "Enumerated" types - character pointer:
decode_cno,

mission_status_enum_str.
Library routines:
sprintf,
strcat,
strncat.
CSUs:
puttogether.

h. Logic flow.
set ret_str to null;
append a carriage return to ret_str(a blank line);

append the Mission ID title, the Mission ID string, and a carriage return to ret_str;

append the Target ID title, the Target ID string, and a carriage return to ret_str;

if Mission Status Type is greater than or equal to zero and it is less than the number of possible Mission Status Type.
decode Mission Status Type;
else
decode Mission Status Type with an invalid_entry string;
invoke puttogether() to append the Mission Status title, the decoded Mission Status Type string and a carriage return to ret_str;

append three carriage returns to ret_str(three blank lines);

append the "CAN NOT OBSERVE" message string to ret_str;

append four carriage returns to ret_str(four blank lines);
append the first 32 Free Text characters and a carriage to ret_str;
append the next 32 Free Text characters) and a carriage to ret_str;
return ret_str to caller.

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

4.6.19 CSU decode_shift_pdu

The decode_shift_pdu CSU decodes the Shift Artillery input message. The following paragraphs provide design information for this CSU.

4.6.19.1 CSU decode_shift_pdu requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.6.19.2 CSU decode_shift_pdu design.

The information identified below represents the detailed design of the decode_shift_pdu CSU.

Syntax for invoking this function:
output = decode_shift_pdu(pdu);

a. Input/output data elements.

INPUTS:
" pdu" is a Artillery_type pointer containing the input message to be decoded.

OUTPUTS:
"ret_str" is a character buffer of size18x36 containing the decoded message string to be returned.

b. Local data elements.
"temp_str" is a static unsigned char buffer of size 36 that contains the temporary decoded message.
"temp" is a temporary storage that contains the integer value of a message field.

c. Interrupts and signals. None

d. Algorithms.
store a null in the first byte of ret_str.
append a carriage return to ret_str.
decode and append the following fields to ret_str in order:
Mission ID,
Target ID,
Mission Status,
Fire for Effect logical Type,
Shift Message String,
Shift Instruction Message String,
Free Text Annotation String.

* append a carriage return to ret_str at the end of each field.

e. Error handling. None

f. Data conversion. None

g. Use of other elements.
The following provide the string values of "Enumerated" types - character pointer:
decode_shift,
mission_status_enum_str,
logic_enum_str,
Library routines:
sprintf,
strcat,
strncat.
CSUs:
puttogether.

h. Logic flow.
set ret_str to null;
append a carriage return to ret_str(a blank line);

append the Mission ID title, the Mission ID string, and a carriage return to ret_str;

append the Target ID title, the Target ID string, and a carriage return to ret_str;

if Mission Status Type is greater than or equal to zero and it is less than the number of possible Mission Status Type.
decode Mission Status Type;
else
decode Mission Status Type with an invalid_entry string;
invoke puttogether() to append the Mission Status title, the decoded Mission Status Type string and a carriage return to ret_str;

if Fire for Effect Type is greater than or equal to zero and is less than the number of possible Fire for Effect Type.
decode Fire for Effect Type into a string;
else
decode Fire for Effect Type with an invalid_entry string;
invoke puttogether() to append the Fire For Effect title, the decoded Fire for Effect Type string and a carriage return to ret_str;

append two carriage returns to ret_str;

append the "SHIFT" message string at the center of a line and a carriage return to ret_str;

append nine character spaces to ret_str;
append the the Shift Instruction Message string and a carriage return to ret_str;

append two carriage returns to ret_str(two blank lines);
append the first 32 Free Text characters and a carriage to ret_str;
append the next 32 Free Text characters) and a carriage to ret_str;
return ret_str to caller.

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

4.6.20 CSU decode_nwmsn_pdu

The decode_nwmsn_pdu CSU decodes the New Artillery input message. The following paragraphs provide design information for this CSU.

4.6.20.1 CSU decode_nwmsn_pdu requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.6.20.2 CSU decode_nwmsn_pdu design.

The information identified below represents the detailed design of the decode_nwmsn_pdu CSU.

Syntax for invoking this function:
output = decode_nwmsn_pdu(pdu);

a. Input/output data elements.

INPUTS:
" pdu" is a Artillery_type pointer containing the input message to be decoded.

OUTPUTS:
"ret_str" is a character buffer of size18x36 containing the decoded message string to be returned.

b. Local data elements.
"temp_str" is a static unsigned char buffer of size 36 that contains the temporary decoded message.
"temp" is a temporary storage that contains the integer value of a message field.

c. Interrupts and signals. None

d. Algorithms.
store a null in the first byte of ret_str.
append a carriage return to ret_str.
decode and append the following fields to ret_str in order:
New Artillerty Mission title String,
Mission ID,
Target ID,
Mission Status,
Mission Type,
Shell Type,
Control Type,
Fuze Type,
Trajectory Type,
Fire for Effect logical Type,
Free Text Annotation String.

* append a carriage return to ret_str at the end of each field.

e. Error handling. None

f. Data conversion. None

g. Use of other elements.
The following provide the string values of "Enumerated" types - character pointer:
decode_nwmsn,
mission_status_enum_str,
mission_enum_str,
shell_enum_str,
control_enum_str,
fuze_enum_str,
trajectory_enum_str,
logic_enum_str.
Library routines:

sprintf,
strcat,
strncat.

CSUs:

puttogether.

h. Logic flow.

set ret_str to null;
append a carriage return to ret_str;

append the "* * NEW ARTILLERY MISSION * *" message string and a carriage return to ret_str;

append the Mission ID title, the Mission ID string, and a carriage return to ret_str;

append the Target ID title, the Target ID string, and a carriage return to ret_str;

if Mission Status Type is greater than or equal to zero and it is less than the number of possible Mission Status Type.
  decode Mission Status Type;
else
  decode Mission Status Type with an invalid_entry string;
invoke puttogether() to append the Mission Status title, the decoded Mission Status Type string and a carriage return to ret_str;

if Mission Type is greater than or equal to zero and it is less than the number of possible Mission Type.
  decode Mission Type;
else
  decode Mission Type with an invalid_entry string;
invoke puttogether() to append the Mission Type title, the decoded Mission Type string and a carriage return to ret_str;

if Shell Type is greater than or equal to zero and it is less than the number of possible Shell Type.
  decode Shell Type;
else
  decode Shell Type with an invalid_entry string;
invoke puttogether() to append the Shell Type title, the decoded Shell Type string and a carriage return to ret_str;

if Control Type is greater than or equal to zero and it is less than the number of possible Control Type.
  decode Control Type;

else
decode Control Type with an invalid_entry string;
invoke puttogether() to append the Control Type title, the decoded Control Type string and a carriage return to ret_str;

if Fuze Type is greater than or equal to zero and it is less than the number of possible Fuze Type.
decode Fuze Type;
else
decode Fuze Type with an invalid_entry string;
invoke puttogether() to append the Fuze Type title, the decoded Fuze Type string and a carriage return to ret_str;

if Trajectory Type is greater than or equal to zero and it is less than the number of possible Trajectory Type.
decode Trajectory Type;
else
decode Trajectory Type with an invalid_entry string;
invoke puttogether() to append the Trajectory Type title, the decoded Trajectory Type string and a carriage return to ret_str;

if Fire for Effect Type is greater than or equal to zero and it is less than the number of possible Fire for Effect Type.
decode Fire for Effect Type;
else
decode Fire for Effect Type with an invalid_entry string;
invoke puttogether() to append the Fire For Effect title, the decoded Fire for Effect Type string and a carriage return to ret_str;

append the first 32 Free Text characters and a carriage to ret_str;
append the next 32 Free Text characters) and a carriage to ret_str;
return ret_str to caller.

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

4.6.21 CSU decode_mto_pdu

The decode_mto_pdu CSU decodes the Message to Observe input message. The following paragraphs provide design information for this CSU.

4.6.21.1 CSU decode_mto_pdu requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.6.21.2 CSU decode_mto_pdu design.

The information identified below represents the detailed design of the decode_mto_pdu CSU.

Syntax for invoking this function:
output = decode_mto_pdu(pdu);

a. Input/output data elements.

INPUTS:
" pdu" is a Artillery_type pointer containing the input message to be decoded.

OUTPUTS:
"ret_str" is a character buffer of size18x36 containing the decoded message string to be returned.

b. Local data elements.
"temp_str" is a static unsigned char buffer of size 36 that contains the temporary decoded message.
"temp" is a temporary storage that contains the integer value of a message field.

c. Interrupts and signals. None

d. Algorithms.
store a null in the first byte of ret_str.
append a carriage return to ret_str.
decode and append the following fields to ret_str in order:
Mission ID,
Target ID,
Mission Status,
Message To Observe message String,
Request Adjustment,
Enter As Target Logic Type,
End Mission Logic Type,
Free Text Annotation String.

* append a carriage return to ret_str at the end of each field.

e. Error handling. None

f. Data conversion. None

g. Use of other elements.
The following provide the string values of "Enumerated" types - character pointer:
decode_mto,
mission_status_enum_str,
logic_enum_str,
Library routines:
sprintf,
strcat,
strncat.
CSUs:
puttogether.

h. Logic flow.
set ret_str to null;
append a carriage return to ret_str(a blank line);

append the Mission ID title, the Mission ID string, and a carriage return to ret_str;

append the Target ID title, the Target ID string, and a carriage return to ret_str;

if Mission Status Type is greater than or equal to zero and it is less than the number of possible Mission Status Type.
decode Mission Status Type;
else
decode Mission Status Type with an invalid_entry string;
invoke puttogether() to append the Mission Status title, the decoded Mission Status Type string and a carriage return to ret_str;

append the "MESSAGE TO OBSERVER" string at the center of a line and a carriage return to ret_str;

if Request Adjustment Type is greater than or equal to zero and it is less than the number of possible Logical Type.
decode Request Adjustment Logical Type;
else

decode Request Adjustment Logical Type with an invalid_entry string;
invoke puttogether() to append the Request Adjustment title, the decoded Request Adjustment Logical Type string and a carriage return to ret_str;

if Enter As Target Logical Type is greater than or equal to zero and it is less than the number of possible Logical Type.
decode Enter As Target Logical Type;
else
decode Enter As Target Logical Type with an invalid_entry string;
invoke puttogether() to append the Enter As Target title, the decoded Enter As Target Logical Type string and a carriage return to ret_str;

if End Mission Logical Type is greater than or equal to zero and it is less than the number of possible Logical Type.
decode End Mission Logical Type;
else
decode End Mission Logical Type with an invalid_entry string;
invoke puttogether() to append the End Mission title , the decoded End Mission Logical Type string and a carriage return to ret_str;

append two carriage returns to ret_str(two blank lines);
append the first 32 Free Text characters and a carriage to ret_str;
append the next 32 Free Text characters) and a carriage to ret_str;
return ret_str to caller.

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

4.6.22 CSU decode_shot_pdu

The decode_shot_pdu CSU decodes the Shot Artillery input message. The following paragraphs provide design information for this CSU.

4.6.22.1 CSU decode_shot_pdu requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.6.22.2 CSU decode_shot_pdu design.

The information identified below represents the detailed design of the decode_shot_pdu CSU.

Syntax for invoking this function:
output = decode_shot_pdu(pdu);

a. Input/output data elements.

INPUTS:
" pdu" is a Artillery_type pointer containing the input message to be decoded.

OUTPUTS:
"ret_str" is a character buffer of size18x36 containing the decoded message string to be returned.

b. Local data elements.
"temp_str" is a static unsigned char buffer of size 36 that contains the temporary decoded message.
"temp" is a temporary storage that contains the integer value of a message field.
"i" is a loop counter.

c. Interrupts and signals. None

d. Algorithms.
store a null in the first byte of ret_str.
append a carriage return to ret_str.
decode and append the following fields to ret_str in order:
Shot Message String,
Mission ID,
Target ID,
Mission Status Type,
Shot Fire Message String,
Free Text Annotation String.

* append a carriage return to ret_str at the end of each field.

e. Error handling. None

f. Data conversion. None

g. Use of other elements.
The following provide the string values of "Enumerated" types -
character pointer:
decode_shot,
mission_status_enum_str.
Library routines:
sprintf,
strcat,
strncat.
CSUs:
puttogether.

h. Logic flow.
set ret_str to null;
append a carriage return to ret_str(a blank line);

append the Mission ID title, the Mission ID string, and a carriage return to ret_str;

append the Target ID title, the Target ID string, and a carriage return to ret_str;

if Mission Status Type is greater than or equal to zero and it is less than the number of possible Mission Status Type.
decode Mission Status Type;
else
decode Mission Status Type with an invalid_entry string;
invoke puttogether() to append the Mission Status title, the decoded Mission Status Type string and a carriage return to ret_str;

append three carriage returns to ret_str;
append the "* * SHOT FIRED * *" message at the center of a line to ret_str;

append four carriage returns to ret_str(four blank lines);
append the first 32 Free Text characters and a carriage to ret_str;
append the next 32 Free Text characters) and a carriage to ret_str;
return ret_str to caller.

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

4.6.23 CSU decode_splash_pdu

The decode_splash_pdu CSU decodes the Splash Artillery input message. The following paragraphs provide design information for this CSU.

4.6.23.1 CSU decode_splash_pdu requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.6.23.2 CSU decode_splash_pdu design.

The information identified below represents the detailed design of the decode_splash_pdu CSU.

Syntax for invoking this function:
output = decode_splash_pdu(pdu);

a. Input/output data elements.

INPUTS:
" pdu" is a Artillery_type pointer containing the input message to be decoded.

OUTPUTS:
"ret_str" is a character buffer of size18x36 containing the decoded message string to be returned.

b. Local data elements.
"temp_str" is a static unsigned char buffer of size 36 that contains the temporary decoded message.
"temp" is a temporary storage that contains the integer value of a message field.

c. Interrupts and signals. None

d. Algorithms.
store a null in the first byte of ret_str.
append a carriage return to ret_str.
decode and append the following fields to ret_str in order:
Mission ID,
Target ID,
Mission Status Type,
Splash message string,
Round Fired,
Second to Impact time,
Free Text Annotation String.

* append a carriage return to ret_str at the end of each field.

e. Error handling. None

f. Data conversion. None

g. Use of other elements.
The following provide the string values of "Enumerated" types - character pointer:
decode_splash,
mission_status_enum_str.
Library routines:
sprintf,
strcat,
strncat.
CSUs:
puttogether.

h. Logic flow.
set ret_str to null;
append a carriage return to ret_str(a blank line);

append the Mission ID title, the Mission ID string, and a carriage return to ret_str;

append the Target ID title, the Target ID string, and a carriage return to ret_str;

if Mission Status Type is greater than or equal to zero and it is less than the number of possible Mission Status Type.
decode Mission Status Type;
else
decode Mission Status Type with an invalid_entry string;

invoke puttogether() to append the Mission Status title, the decoded Mission Status Type string and a carriage return to ret_str;

append two carriage returns to ret_str;
append the "* * SPLASH * *" message at the center of a line and a carriage return to ret_str;

convert Rounds Fired into a string;
invoke puttogether() to append the Rounds title, the converted Rounds Fired string, and a carriage return to ret_str;

convert Impact Time into a string;
invoke puttogether() to append the Second to Impact title, the converted Impact Time string, and a carriage return to ret_str;

append two carriage returns to ret_str(two blank lines);
append the first 32 Free Text characters and a carriage to ret_str;
append the next 32 Free Text characters) and a carriage to ret_str;
return ret_str to caller.

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

4.6.24 CSU decode_eom_pdu

The decode_eom_pdu CSU decodes the End Of Mission Artillery input message. The following paragraphs provide design information for this CSU.

4.6.24.1 CSU decode_eom_pdu requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.6.24.2 CSU decode_eom_pdu design.

The information identified below represents the detailed design of the decode_eom_pdu CSU.

Syntax for invoking this function:
output = decode_eom_pdu(pdu);

a. Input/output data elements.

INPUTS:
" pdu" is a Recon_type pointer containing the input message to be decoded.

OUTPUTS:
"ret_str" is a character buffer of size18x36 containing the decoded message string to be returned.

b. Local data elements.
"temp_str" is a static unsigned char buffer of size 36 that contains the temporary decoded message.
"temp" is a temporary storage that contains the integer value of a message field.

c. Interrupts and signals. None

d. Algorithms.
store a null in the first byte of ret_str.
append a carriage return to ret_str.
decode and append the following fields to ret_str in order:
Mission ID,
Target ID,
Mission Status Type,
End Of Mission message string,
Disposition Type,
Record As Target Logical Type,
Casualties,
Point Number,
Free Text Annotation String.

* append a carriage return to ret_str at the end of each field.

e. Error handling. None

f. Data conversion. None

g. Use of other elements.
The following provide the string values of "Enumerated" types - character pointer:
decode_eom,
mission_status_enum_str,
disposition_enum_str,
logic_enum_str.

Library routines:
sprintf,
strcat,
strncat.
CSUs:
puttogether.

h. Logic flow.
set ret_str to null;
append a carriage return to ret_str(a blank line);

append the Mission ID title, the Mission ID string, and a carriage return to ret_str;

append the Target ID title, the Target ID string, and a carriage return to ret_str;

if Mission Status Type is greater than or equal to zero and it is less than the number of possible Mission Status Type.
decode Mission Status Type;
else
decode Mission Status Type with an invalid_entry string;
invoke puttogether() to append the Mission Status title, the decoded Mission Status Type string and a carriage return to ret_str;

append a carriage return to ret_str;
append the "* * END OF MISSION * *" message at the center of a line and two carriage returns to ret_str;

if Disposition Type is greater than or equal to zero and it is less than the number of possible Disposition Type.
decode Disposition Type;
else
decode Disposition Type with an invalid_entry string;
invoke puttogether() to append the Disposition title, the decoded Disposition Type string and a carriage return to ret_str;

if Record As Target Type is greater than or equal to zero and it is less than the number of possible Logical Type.
decode Record As Target Type;
else
decode Record As Target Type with an invalid_entry string;
invoke puttogether() to append the Record As Target title, the decoded Record As Target logical Type string and a carriage return to ret_str;

convert Casualties Number into a string;
invoke puttogether() to append the Casualties title, the converted Casualties Number string, and a carriage return to ret_str;

append the Point Number title, the Point Number string, and two carriage returns to ret_str;

append the first 32 Free Text characters and a carriage to ret_str;
append the next 32 Free Text characters) and a carriage to ret_str;
return ret_str to caller.

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

4.6.25 CSU decode_status_pdu

The decode_status_pdu CSU decodes the Status input message. The following paragraphs provide design information for this CSU.

4.6.25.1 CSU decode_status_pdu requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.6.25.2 CSU decode_status_pdu design.

The information identified below represents the detailed design of the decode_status_pdu CSU.

Syntax for invoking this function:
output = decode_status_pdu(pdu);

a. Input/output data elements.

INPUTS:
" pdu" is a Status_type pointer containing the input message to be decoded.

OUTPUTS:
"ret_str" is a character buffer of size18x36 containing the decoded message string to be returned.

b. Local data elements.
"temp_str" is a static unsigned char buffer of size 36 that contains the temporary decoded message.
"temp" is a temporary storage that contains the integer value of a message field.
"i, j" are loop counters.

c. Interrupts and signals. None

d. Algorithms.
store a null in the first byte of ret_str.
append a carriage return to ret_str.
decode and append the following fields to ret_str in order:
Fuel Status,
Failed Equipments Type,
Hell Fires,
Stingers,
Rockets,
Rounds,
Request type,
Free Text Annotation String.

* append a carriage return to ret_str at the end of each field.

e. Error handling. None

f. Data conversion. None

g. Use of other elements.
The following provide the string values of "Enumerated" types -
character pointer:
decode_status,
fail_enum_str.
Library routines:
sprintf,
strcat,
strncat.
CSUs:
None.

h. Logic flow.

```
set ret_str to null;
append a carriage return to ret_str(a blank line);

convert Fuel into a string with unit "LBS" and a carriage return to
      ret_str;

repeat for three failed equipments
      if Failed Equipment Type is greater
       than or equal to zero and is less than the
       number of possible Failed Equipment.
        decode Failed Equipment Type;
      else
        decode Failed Equipment with an
         invalid_entry string;
      append the Failed Equipment title, the
      decoded Failed Equipment Type string, and a
      carriage return to ret_str;

convert Hell Fires into a string;
append the Hell Fires title, the converted Hell Fires string, and a
      carriage return to ret_str;

convert Stingers into a string;
append the Stingers title, the converted Stingers string, and a carriage
      return to ret_str;

convert Rockets into a string;
append the Rockets title, the converted Rockets string, and a carriage
      return to ret_str;

convert Rounds into a string;
append the Rounds title, the converted Rounds string, and a carriage
      return to ret_str;

if Request Type is greater than or equal to zero and is less than the
      number of possible Request Type
      decode Request Type;
else
      decode Request Type with an invalid_entry string;
append the Request Type title, the decoded Request Type string, and a
      carriage return to ret_str;

append two carriage returns to ret_str( two blank
lines);
append the first 32 Free Text characters and a carriage to ret_str;
```

append the next 32 Free Text characters) and a carriage to ret_str; return ret_str to caller.

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

4.6.26 CSU decode_request_pdu

The decode_request_pdu CSU decodes the Request input message. The following paragraphs provide design information for this CSU.

4.6.26.1 CSU decode_request_pdu requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.6.26.2 CSU decode_request_pdu design.

The information identified below represents the detailed design of the decode_request_pdu CSU.

Syntax for invoking this function:
output = decode_request_pdu(pdu);

a. Input/output data elements.

INPUTS:
"pdu" is a Request_type pointer containing the input message to be decoded.

OUTPUTS:
"ret_str" is a character buffer of size18x36 containing the decoded message string to be returned.

b. Local data elements.
"temp_str" is a static unsigned char buffer of size 36 that contains the temporary decoded message.
"temp" is a temporary storage that contains the integer value of a message field.

c. Interrupts and signals. None

d. Algorithms.
store a null in the first byte of ret_str.
append a carriage return to ret_str.
decode and append the following fields to ret_str in order:
Report Type,
Recon Type,
Free Text Annotation String.

* append a carriage return to ret_str at the end of each field.

e. Error handling. None

f. Data conversion. None

g. Use of other elements.
The following provide the string values of "Enumerated" types -
character pointer:
decode_request,
report_enum_str.
Library routines:
sprintf,
strcat,
strncat.
CSUs:
puttogether.

h. Logic flow.
set ret_str to null;
append a carriage return to ret_str(a blank line);

if Report Type is greater than or equal to zero and is less than the number of possible Report Type
decode Report Type;
else
decode Report Type with an invalid_entry string;
invoke puttogether to append the Report Type title, the decoded Report Type string, and a carriage return to ret_str;

if Report type is equal to Reconaissance
if Recon Type is greater than or equal to zero and is less than the number of possible Recon Type
decode Recon Type;
else
decode Recon Type with an

```
            invalid_entry string;
        invoke puttogether to append the Recon    Type  title,  the
        decoded Recon Type string,      and a carriage return to ret_str;
else
        append a carriage to ret_str;

append nine carriage returns to ret_str(nine blank
lines);
append the first 32 Free Text characters and a carriage to ret_str;
append the next 32 Free Text characters) and a carriage to ret_str;
return ret_str to caller.
```

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

4.6.27 CSU decode_nbc1_pdu

The decode_nbc1_pdu CSU decodes the Nuclear, Biological, Chemical of Type one input message. The following paragraphs provide design information for this CSU.

4.6.27.1 CSU decode_nbc1_pdu requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.6.27.2 CSU decode_nbc1_pdu design.

The information identified below represents the detailed design of the decode_nbc1_pdu CSU.

Syntax for invoking this function:
output = decode_nbc1_pdu(pdu);

a. Input/output data elements.

INPUTS:
" pdu" is a NBC_type pointer containing the input message to be decoded.

OUTPUTS:
"ret_str" is a character buffer of size18x36 containing the decoded message string to be returned.

b. Local data elements.
"temp_str" is a static unsigned char buffer of size 36 that contains the temporary decoded message.
"temp" is a temporary storage that contains the integer value of a message field.

c. Interrupts and signals. None

d. Algorithms.
store a null in the first byte of ret_str.
append a carriage return to ret_str.
decode and append the following fields to ret_str in order:
Description Type,
Burst Type,
Delivered By,
Cloud Height Unit,
Cloud Height,
Cloud Description,
Flash Bang Time,
Start Time,
Stop Time,
Free Text Annotation String.

* append a carriage return to ret_str at the end of each field.

e. Error handling. None

f. Data conversion. None

g. Use of other elements.
The following provide the string values of "Enumerated" types - character pointer:
decode_nbc1,
nbc_desc_enum_str,
burst_enum_str,
delivery_enum_str,
cloud_hgt_enum_str.
Library routines:
sprintf,
strcat,
strncat.
CSUs:

decode_time,
puttogether.

h. Logic flow.
set ret_str to null;
append a carriage return to ret_str;

append the ">>>> NUCLEAR BLAST <<<<" message string at the center of a line and a carriage return to ret_str;

if Description is greater than or equal to zero and is less than the number of possible Description Type.
decode Description Type into a string;
else
decode Description Type with an invalid_entry string;
invoke puttogether() to append the Description Type title, the decoded Description Type string and a carriage return to ret_str;

if Burst Type is greater than or equal to zero and is less than the number of possible Burst Type.
decode Burst Type into a string;
else
decode Burst Type with an invalid_entry string;
invoke puttogether() to append the Burst Type title, the decoded Burst Type string and a carriage return to ret_str;

if Delivered By is greater than or equal to zero and is less than the number of possible Delivered By Type.
decode Delivered By Type into a string;
else
decode Delivered By Type with an invalid_entry string;
invoke puttogether() to append the Delivered By Type title, the decoded Delivered By Type string and a carriage return to ret_str;

if Cloud Height Unit Type is greater than or equal to zero and is less than the number of possible Cloud Height Unit Type.
decode Cloud Height Unit Type into a string;
else
decode Cloud Height Unit Type with an invalid_entry string;
invoke puttogether() to append the Cloud Height Unit Type title, the decoded Cloud Height Unit Type string and a carriage return to ret_str;

convert the Cloud Height into a string;
append the Cloud Height title, the converted Cloud Height string, and a carriage return to ret_str;

append the Cloud Description title, the Cloud Description string, and a carriage return to ret_str;

convert the Flash Bang Time into a string with unit "SECONDS";
append the Flash Bang Time title, the converted Flash Bang Time string, and a carriage return to ret_str;

invoke decode_time to decode the Start Time;
append the Start Time title, the decoded Start Time, and a carriage return to ret_str;

invoke decode_time to decode the Stop Time;
append the Stop Time title, the decoded Stop Time, and a carriage return to ret_str;

append the first 32 Free Text characters and a carriage to ret_str;
append the next 32 Free Text characters) and a carriage to ret_str;
return ret_str to caller.

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

4.6.28 CSU decode_nbc4_pdu

The decode_nbc4_pdu CSU decodes the Nuclear, Biological, Chemical of Type four input message. The following paragraphs provide design information for this CSU.

4.6.28.1 CSU decode_nbc4_pdu requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.6.28.2 CSU decode_nbc4_pdu design.

The information identified below represents the detailed design of the decode_nbc4_pdu CSU.

Syntax for invoking this function:
output = decode_nbc4_pdu(pdu);

a. Input/output data elements.

INPUTS:
"pdu" is a NBC_type pointer containing the input message to be decoded.

OUTPUTS:
"ret_str" is a character buffer of size18x36 containing the decoded message string to be returned.

b. Local data elements.
"temp_str" is a static unsigned char buffer of size 36 that contains the temporary decoded message.
"temp" is a temporary storage that contains the integer value of a message field.

c. Interrupts and signals. None

d. Algorithms.
store a null in the first byte of ret_str.
append a carriage return to ret_str.
decode and append the following fields to ret_str in order:
Description Type,
Burst Type,
Delivered By,
Cloud Height Unit,
Cloud Height,
Cloud Description,
Dose Rate,
Free Text Annotation String.

* append a carriage return to ret_str at the end of each field.

e. Error handling. None

f. Data conversion. None

g. Use of other elements.
The following provide the string values of "Enumerated" types - character pointer:
decode_nbc4,
nbc_desc_enum_str,
burst_enum_str,

delivery_enum_str,
cloud_hgt_enum_str.
Library routines:
sprintf,
strcat,
strncat.
CSUs:
puttogether.

h. Logic flow.
set ret_str to a null string;
append a carriage return to ret_str;

append the "* CHEMICAL OR BIOLOGICAL * " message string at the center of a line and two carriage returns to ret_str;

if Description is greater than or equal to zero and is less than the number of possible Description Type.
decode Description Type into a string;
else
decode Description Type with an invalid_entry string;
invoke puttogether() to append the Description Type title, the decoded Description Type string and a carriage return to ret_str;

if Burst Type is greater than or equal to zero and is less than the number of possible Burst Type.
decode Burst Type into a string;
else
decode Burst Type with an invalid_entry string;
invoke puttogether() to append the Burst Type title, the decoded Burst Type string and a carriage return to ret_str;

if Delivered By is greater than or equal to zero and is less than the number of possible Delivered By Type.
decode Delivered By Type into a string;
else
decode Delivered By Type with an invalid_entry string;
invoke puttogether() to append the Delivered By Type title, the decoded Delivered By Type string and a carriage return to ret_str;

if Cloud Height Unit Type is greater than or equal to zero and is less than the number of possible Cloud Height Unit Type.
decode Cloud Height Unit Type into a string;
else
decode Cloud Height Unit Type with an invalid_entry string;

invoke puttogether() to append the Cloud Height Unit Type title, the decoded Cloud Height Unit Type string and a carriage return to ret_str;

convert the Cloud Height into a string;
append the Cloud Height title, the converted Cloud Height string, and a carriage return to ret_str;

append the Cloud Description title, the Cloud Description string, and a carriage return to ret_str;

convert the Dose Rate into a string;
append the Dose Rate title, the converted Dose Rate string, and a carriage return to ret_str;

append two carriage returns to ret_str(two blank lines);
append the first 32 Free Text characters and a carriage to ret_str;
append the next 32 Free Text characters) and a carriage to ret_str;
return ret_str to caller.

i. Data structures. None

j. Local data files or database. None

4.6.29 CSU decode_nbc5_pdu

The decode_nbc5_pdu CSU decodes the Nuclear, Biological, Chemical of Type five input message. The following paragraphs provide design information for this CSU.

4.6.29.1 CSU decode_nbc5_pdu requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.6.29.2 CSU decode_nbc5_pdu design.

The information identified below represents the detailed design of the decode_nbc5_pdu CSU.

Syntax for invoking this function:
output = decode_nbc5_pdu(pdu);

a. Input/output data elements.

INPUTS:
" pdu" is a NBC_type pointer containing the input message to be decoded.

OUTPUTS:
"ret_str" is a character buffer of size18x36 containing the decoded message string to be returned.

b. Local data elements. None.

c. Interrupts and signals. None

d. Algorithms.
store a null in the first byte of ret_str.
append five carriage returns to ret_str.
append the following fields to ret_str in order:
message string;
Free Text Annotation String.

* append a carriage return to ret_str at the end of each field.

e. Error handling. None

f. Data conversion. None

g. Use of other elements.
The following provide the string values of "Enumerated" types - character pointer:
decode_nbc5.
Library routines:
sprintf,
strcat,
strncat.
CSUs:
None.

h. Logic flow.
place a null to ret_str;
append five carriage returns to ret_str(five blank lines);

append the "NBC NEGATIVE" message string at the center of a line and a carriage return to ret_str;

append six carriage returns to ret_str(six blank lines);

append the first 32 Free Text characters and a carriage to ret_str;
append the next 32 Free Text characters) and a carriage to ret_str;
return ret_str to caller.

i. Data structures. None

j. Local data files or database. None

4.6.30 CSU decode_miji_pdu

The decode_miji_pdu CSU decodes the Meaconing, Intrusion, Jamming Interface input message. The following paragraphs provide design information for this CSU.

4.6.30.1 CSU decode_miji_pdu requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.6.30.2 CSU decode_miji_pdu design.

The information identified below represents the detailed design of the decode_miji_pdu CSU.

Syntax for invoking this function:
output = decode_miji_pdu(pdu);

a. Input/output data elements.

INPUTS:
"pdu" is a MIJI_type pointer containing the input message to be decoded.

OUTPUTS:
"ret_str" is a character buffer of size18x36 containing the decoded message string to be returned.

b. Local data elements.
"temp_str" is a static unsigned char buffer of size 36 that contains the temporary decoded message.
"temp" is a temporary storage that contains the integer value of a message field.

"i, j" are integer loop counters.
"temp1" is a 64 bit float storage.

c. Interrupts and signals. None

d. Algorithms.
store a null in the first byte of ret_str.
append a carriage return to ret_str.
decode and append the following fields to ret_str in order:
Affected Radio Frequency,
Programmed Frequency1,
Start Time,
Stop Time,
Description,
Percent Lost,
Type,
Free Text Annotation String.

* append a carriage return to ret_str at the end of each field.

e. Error handling. None

f. Data conversion. None

g. Use of other elements.
The following provide the string values of "Enumerated" types - character pointer:
decode_miji,
description_enum_str.
Library routines:
sprintf,
strcat,
strncat,
for-loop.
CSUs:
freq_hertz,
decode_time,
puttogether.

h. Logic flow.
place a null to ret_str;
append a carriage return to ret_str(a blank line);

invoke freq_hertz to decode the Affected Radio Frequency to its appriorate unit "kilo, Mega, or Giga" Hertz.

append the Affected Radio Frequency title, the decoded Affected Radion Frequency, and a carriage return to ret_str;

repeat four times
invoke freq_hertz to decode the Programmed Frequency to its appriorate unit "kilo, Mega, or Giga" Hertz.
append the Programmed Frequency title, the decoded Programmed Frequency, and a carriage return to ret_str;

invoke decode_time to decode the Start Time;
append the Start Time title, the decode Start Time, and a carriage return to ret_str;

invoke decode_time to decode the Stop Time;
append the Stop Time title, the decode Stop Time, and a carriage return to ret_str;

if Description is greater than or equal to zero and is less than the number of possible Description Type.
decode Description Type into a string;
else
decode Description Type with an invalid_entry string;
append the Description title, the decoded Description Type, and a carriage return to ret_str;

convert Percent Lost into a string with unit"PERCENT";
append the Percent Lost title, the converted Percent Lost string, and a carriage return to ret_str;

if Type is greater than or equal to zero and is less than the number of possible Type.
decode Type into a string;
else
decode Type with an invalid_entry string;
append the Type title, the decoded Type, and two carriage returns to ret_str;

append the first 32 Free Text characters and a carriage to ret_str;
append the next 32 Free Text characters) and a carriage to ret_str;
return ret_str to caller.

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

4.6.31 CSU decode_dnav_pdu

The decode_dnav_pdu CSU decodes the Downed Air Vehicle Report input message. The following paragraphs provide design information for this CSU.

4.6.31.1 CSU decode_dnav_pdu requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.6.31.2 CSU decode_dnav_pdu design.

The information identified below represents the detailed design of the decode_dnav_pdu CSU.

Syntax for invoking this function:
output = decode_dnav_pdu(pdu);

a. Input/output data elements.

INPUTS:
" pdu" is a DNAV_type pointer containing the input message to be decoded.

OUTPUTS:
"ret_str" is a character buffer of size18x36 containing the decoded message string to be returned.

b. Local data elements.
"temp_str" is a static unsigned char buffer of size 36 that contains the temporary decoded message.
"temp" is a temporary storage that contains the integer value of a message field.

c. Interrupts and signals. None

d. Algorithms.
store a null in the first byte of ret_str.
append a carriage return to ret_str.
decode and append the following fields to ret_str in order:
Aircraft Type,
Aircraft Status,

Pilot Status,
Free Text Annotation String.

* append a carriage return to ret_str at the end of each field.

e. Error_handling. None

f. Data_conversion. None

g. Use of other elements.
The following provide the string values of "Enumerated" types - character pointer:
decode_dnav,
aircraft_enum_str,
aircraft_status_enum_str,
pilot_status_enum_str.
Library routines:
sprintf,
strcat,
strncat.
CSUs:
puttogether.

h. Logic flow.
initialize ret_str to a null string;
append a carriage return to ret_str(a blank line);

if Aircraft Type is greater than or equal to zero and it is less than the number of possible Aircraft Type.
decode Aircraft Type;
else
decode Aircraft Type with an invalid_entry string;
invoke puttogether() to append the Aircraft Type title, the decoded Aircraft Type string and a carriage return to ret_str;

if Aircraft Status is greater than or equal to zero and is less than the number of possible Aircraft Status.
decode Aircraft Status;
else
decode Aircraft Status with an invalid_entry string;
invoke puttogether() to append the Aircraft Status title, the decoded Aircraft Status string and a carriage return to ret_str;

if Pilot Status is greater than or equal to zero and is less than the number of possible Pilot Status.
decode Pilot Status;

else
decode Pilot Status with an invalid_entry string;
invoke puttogether() to append the Pilot Status title, the decoded Pilot Status string and a carriage return to ret_str;

append eight carriage returns to ret_str(eight blank lines);
append the first 32 Free Text characters and a carriage to ret_str;
append the next 32 Free Text characters) and a carriage to ret_str;
return ret_str to caller.

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

4.6.32 CSU decode_movcmd_pdu

The decode_movcmd_pdu CSU decodes the Move Command input message. The following paragraphs provide design information for this CSU.

4.6.32.1 CSU decode_movcmd_pdu requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.6.32.2 CSU decode_movcmd_pdu design.

The information identified below represents the detailed design of the decode_movcmd_pdu CSU.

Syntax for invoking this function:
output = decode_movcmd_pdu(input message pointer of Move_type);

a. Input/output data elements.

INPUTS:
"pdu" is a Move_type pointer containing the input message to be decoded.

OUTPUTS:
"ret_str" is a character buffer of size18x36 containing the decoded message string to be returned.

b. Local data elements.
"temp_str" is a static unsigned char buffer of size 36 that contains the temporary decoded message.
"temp" is a temporary storage that contains the integer value of a message field.

c. Interrupts and signals. None

d. Algorithms.
initialize ret_str to null.
append a carriage return to ret_str.
decode and append the following fields to ret_str in order:
Target ID,
Task,
Who,
When,
Where,
Free Text Annotation String.

* append a carriage return to ret_str at the end of each field.

e. Error handling. None

f. Data conversion. None

g. Use of other elements.
The following provide the string values of "Enumerated" types - character pointer:
decode_movcmd,
task_enum_str,
who_enum_str,
when_enum_str.
Library routines:
sprintf,
strcat,
strncat.
CSUs:
puttogether.

h. Logic flow.
initialize ret_str to a null string;
append a carriage return to ret_str;

append the Targe ID title, the Targe ID string, and a carriage return to ret_str;

if Task is greater than or equal to zero and it is less than the number of possible Task.
decode Task;
else
decode Task with an invalid_entry string;
invoke puttogether() to append the Task title, the decoded Task string and a carriage return to ret_str;

if Who is greater than or equal to zero and is less than the number of possible Who list.
decode Who;
else
decode Who with an invalid_entry string;
invoke puttogether() to append the Who title, the decoded Who string and a carriage return to ret_str;

if When is greater than or equal to zero and is less than the number of possible When list.
decode When;
else
decode When with an invalid_entry string;
invoke puttogether() to append the When title, the decoded When string and a carriage return to ret_str;

append the Where title, the Where string, and a carriage return to ret_str;

append six carriage returns to ret_str(six blank lines);
append the first 32 Free Text characters and a carriage to ret_str;
append the next 32 Free Text characters) and a carriage to ret_str;
return ret_str to caller.

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

4.6.33 CSU puttogether

The puttogether CSU appends message title and decoded message into a temporary character string buffer. The following paragraphs provide design information for this CSU.

4.6.33.1 CSU puttogether requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.6.33.2 CSU puttogether design.

The information identified below represents the detailed design of the puttogether CSU.

Syntax for invoking this function:
puttogether(decoded message, return buffer, message header);

a. Input/output data elements.

INPUTS:
"temp_str" is an unsigned character pointer containing the decoded message.
"msg" is an unsigned character pointer containing the message title.

OUTPUTS:
"ret_str" is an unsigned character pointer containing the return buffer.

b. Local data elements. None

c. Interrupts and signals. None

d. Algorithms. None

e. Error handling. None

f. Data conversion. None

g. Use of other elements.
data elements:
None
Library routines:
strcat.
CSUs:
None

h. Logic flow.
append message title, decoded message, and a carriage return to ret_str; return ret_str to caller.

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

4.6.34 CSU decode_time

The decode_time CSU decodes DTG time. The following paragraphs provide design information for this CSU.

4.6.34.1 CSU decode_time requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.6.34.2 CSU decode_time design.

The information identified below represents the detailed design of the decode_time CSU.

Syntax for invoking this function:
decode_time(return buffer, DTG time);

a. Input/output data elements.

INPUTS:
"dtg" is a DTG_type pointer containing the input DTG time o be decoded.

OUTPUTS:
"temp_str" is an unsigned character buffer of containing the decoded time string to be returned.

b. Local data elements.
"month" is an integer storage.
"temp_str2" is a static unsigned character buffer of size 36 containing temporary Month string.

c. Interrupts and signals. None

d. Algorithms. None

e. Error handling. None

f. Data conversion. None

g. Use of other elements.
Data elements:
None
Library routines:
sprintf,
strcat,
strncat.
CSUs:
None.

h. Logic flow.
initialize temp_str to a null string and temp _str2 to initialize the string buffer;

append Day and a space to temp_str;
append Hour, Minute, and a space to temp_str;

decode Month into its appropriate interpretation "JAN, FEB, MAR, ... DEC".
append the decoded Month string and a space to temp_str;

append the last two characters of Year string to temp_str;
return temp_str to caller.

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

4.6.35 CSU freq_hertz

The freq_hertz CSU decodes the the input frequency into its proper unit. The following paragraphs provide design information for this CSU.

4.6.35.1 CSU freq_hertz requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.6.35.2 CSU freq_hertz design.

The information identified below represents the detailed design of the freq_hertz CSU.

Syntax for invoking this function:
freq_hertz(input frequency, return buffer);

a. Input/output data elements.

INPUTS:
"temp1" is a 64 bit float integer containing the input frequency to be decoded.

OUTPUTS:
"temp_str" is a character buffer of size 36 containing the decoded frequency string to be returned.

b. Local data elements.
"temp, temp2" are temporary 64 bit float storage;

c. Interrupts and signals. None

d. Algorithms. None

e. Error handling. None

f. Data conversion. None

g. Use of other elements.
Data elements:
None.
Library routines:
if - else,
sp .tf.
CSUs:
None.

h. Logic flow.
set temp to zero;
if input frequency >= a kilo and is < a million, convert it to kiloHertz;
else if input frequency >= a million and is < a billion, convert it to MegaHertz;
else if input frequency >= a billion, convert it to GigaHertz;
else append the Hertz unit to input frequency;
place the decoded input frequency to temp_str;
return temp_str to caller.

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

4.6.36 CSU sprintf_UTM

The sprintf_UTM CSU decodes the input DTG time. The following paragraphs provide design information for this CSU.

4.6.36.1 CSU sprintf_UTM requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.6.36.2 CSU sprintf_UTM design.

The information identified below represents the detailed design of the sprintf_UTM CSU.

Syntax for invoking this function:
output = sprintf_UTM(address of the input location);

a. Input/output data elements.

INPUTS:
"utm" is a UTM_type pointer containing the input location to be decoded.

OUTPUTS:
"ret_str" is a static character buffer of size 80 containing the decoded UTM string to be returned.

b. Local data elements. None

c. Interrupts and signals. None

d. Algorithms. None

e. Error handling. None

f. Data conversion. None

g. Use of other elements.
Data elements:
None.
Library routines:
strncpy.
CSUs:
None.

h. Logic flow.
place the two Sector Letters and a space to ret_str;
append the first four Easting characters and a space to ret_str;
append the first four Northing characters and a null character to ret_str;
return ret_str to caller.

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

4.6.37 CSU sprintf_speed

The sprintf_speed CSU decodes the input Speed into its string representation. The following paragraphs provide design information for this CSU.

4.6.37.1 CSU sprintf_speed requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

4.6.37.2 CSU sprintf_speed design.

The information identified below represents the detailed design of the sprintf_speed CSU.

Syntax for invoking this function:
output = sprintf_speed(address of the input speed);

a. Input/output data elements.

INPUTS:
"speed" is a Speed_type pointer containing the input Speed to be decoded.

OUTPUTS:
"speed_buffer" is a static character buffer of size 20 containing the decoded Speed string to be returned.

b. Local data elements. None

c. Interrupts and signals. None

d. Algorithms. None

e. Error handling. None

f. Data conversion. None

g. Use of other elements.
Data elements:
None.
Library routines:
if - else
sprintf.
CSUs:
None.

h. Logic flow.
decode the unit of Speed type;
append Speed value, a space, and the decoded Speed unit string to speed_buffer;
return speed_buffer to caller.

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

5. DMCC Global Data Elements

In writing C programs, great use is made of 'header' files. These are include files that contain the information necessary for multiple modules to treat data in the same way. For this CSCI, the header files contain structure definitions that allow different functions to fill selected data entities within the structures, modify or use all or parts of those structures, or pass data the data structures between functions to allow for the ordered manipulation of that data. Also within the header files are case sensitive statements that control how data is accessed or processed. They are delineated through the use of 'define' statements. Most of the time, these items are flags used for comparison within the functions. They can also be used as constants for array sizing. At compile time, the compiler exchanges the statement for the value that it is equated with. For example, if the header file "assoc_PDU.h" is included into module X, then at compile time if anywhere within the source code of module X the fragment "ASSOC_DATAGRAM" is found, the compiler will replace the fragment with the value "1" because in the header assoc_PDU.h is found the statement '#define ASSOC_DATAGRAM 1'. Each header file may be described in 3 parts. Part 1 of the following tables will contain the *structure* definitions, part 2 of the associated tables containing the 'define' statements, and part 3 (where applicable) will dimension variables, initialize variables, or prototype functions.

For this CSCI there are 2 uses for the header files. Those listed in section 5.1.x are for system use, while those listed in section 5.2.x are for the General User Interface (GUI) use.

5.1 System Header files

5.1.1 assoc_PDU.h

This header file provides the data structures and definitions needed for the decode_pkt modules used to decode the incoming message PDU's.

5.1.1.1.1 assoc_PDU structure assocPDU_type

Table 4 Structure assocPDU_Type

Item Name	Type	bytes	Description
Version, PDUKind	Integer	2	Version - bits 0:3, PDUKind bits 4:7
DataLength	Char	1	
MulticastGroup	Char	1	
Protocol	Char	1	
OriginatorSite	Integer	2	
OriginatorSimulator	Integer	2	
ResponderSite	Integer	2	
ResponderSimulator	Integer	2	
TransactionID	Integer	2	
Unused	Integer	2	

5.1.1.1.2 assoc_PDU structure assocPDUWithLLCHeaderType

Table 5 assocPDUWithLLCHeaderType

Item Name	Type	bytes	Description
LLCSubLayerHeader	Char array	8	
AssocPDU	Structure	17	AssocPDU_type structure
ExtraPadding	Char array	32	

5.1.1.1.3 assoc_PDU structure ActivateRequestEntry

Table 6 assocPDU structure ActivateRequestEntry

Item Name	Type	bytes	Description
AssocPDUWithLLCHeader	Structure	57	AssocPDUWithLLCHeaderType
Valid	Char	1	
MorePadding	Char array	7	

5.1.1.1.4 assoc_PDU structure ActivateRequestBufferStructure

Table 7 assocPDU structure ActivateRequestBufferStructure

Item Name	Type	bytes	Description
ActivateRequest	Struc array	1300	array of ActivateRequestEntry types

5.1.1.2 assoc_PDU.h define statements

Table 8 assoc_PDU.h define statements

Name	Value	Description
ASSOC_DATA_LENGTH	1	offset pointer within PDU
ASSOC_MULTICAST_GROUP	2	offset pointer within PDU
ASSOC_PROTOCOL	3	offset pointer within PDU
ASSOC_VERSION_AUG_89	1	values for insertion/comparison in PDU
ASSOC_VERSION_JAN_90	2	values for insertion/comparison in PDU
ASSOC_UNKNOWN_KIND	0	values for insertion/comparison in PDU
ASSOC_DATAGRAM	1	values for insertion/comparison in PDU
ASSOC_REQUEST	2	values for insertion/comparison in PDU
ASSOC_RESPONSE	3	values for insertion/comparison in PDU
ASSOC_PADDING	4	values for insertion/comparison in PDU
ASSOC_LAST_KIND	4	values for insertion/comparison in PDU
ASSOC_MULTIPLIER	8	values for insertion/comparison in PDU
ASSOC_BROADCAST	0	values for insertion/comparison in PDU
ASSOC_SIMULATION	1	values for insertion/comparison in PDU
FORT_RUCKER	4	values for insertion/comparison in PDU
ASSOC_DIGITAL_MSG	0xD0	values for insertion/comparison in PDU
OUR_SIMULATION_NUM	96	values for insertion/comparison in PDU
ASSOC_PDUKINDMASK	0xf0	values for insertion/comparison in PDU
ASSOC_PDU_MIN_LEN	46	values for insertion/comparison in PDU
ASSOC_VER_PADKIND	0x24	values for insertion/comparison in PDU
DIS_NET	0	PDU processing codes
SIM_NET	1	PDU processing codes
NEW_ALPDU	0	PDU processing codes
CONT_ALPDU	1	PDU processing codes
END_ALPDU	2	PDU processing codes
NO_CONFIRM	0	PDU processing codes
CONFIRM	1	PDU processing codes
NO_ASSOC_LAYER	0	PDU processing codes
UDP_PORT	0x4128	Network Interface codes
HOST_NAME	adst5	Network Interface codes
SNAP_LENGTH	8	Network Interface codes
MAX_ACTIVATE_REQUESTS	20	max activate requests that can be stored
ACTIVATE_REQUEST_SHMEM_KEY	96161	
PADDING_FOR_ALPDU	32	number of bytes to complete PDU

5.1.2 databases.h

This header file provides the data structures needed by the database storage functions to maintain operating system data for later use.

5.1.2.1.1 databases structure IndirectFireDataBase_type

Table 9 structure InderectFireDataBase_type

Item Name	Type	bytes	Description
valid	Char	1	
vehicleID	Structure	6	VehicleID_type structure - 5.1.26
velocity	Structure	12	VelocityVector_type - 5.1.26
muzzle	Structure	24	WorldCoordinates_type - 5.1.26
projectileID	Structure	6	VehicleID_type structure - 5.1.26

5.1.2.1.2 databases structure DetonationDataBase_type

Table 10 structure DetonationDataBase_type

Item Name	Type	bytes	Description
valid	Char	1	
Location	Structure	24	WorldCoordinates_type - 5.1.26

5.1.2.1.3 databases structure VehicleAppearanceDataBase_type

Table 11 structure VehicleAppearanceDataBase_type

Item Name	Type	bytes	Description
valid	Char	1	
Change	Char	1	

5.1.2.1.4 databases structure StateDataBase_type

Table 12 structure StateDataBase_type

Item Name	Type	bytes	Description
state	structure		VehicleAppearanceVariant_type-5.1.26
DRA	Structure	40	DeadReckoning_type
valid	Char	1	new information flag
do_dr	Char	1	Dead Reckoning ON flag
time	Structure	4	TimeStamp_type
inlist	Char	1	in linked list flag
send_time	Double	8	time of rebroadcast
interval_time	Double	8	rebroadcast period
state_time	Double	8	Time of last state
X0	Double	8	Initial X position
Y0	Double	8	Initial Y position
Z0	Double	8	Initial Z position
VX0	Float	4	Initial X velocity
VY0	Float	4	Initial Y velocity
VZ0	Float	4	Initial Z velocity
AX0	Float	4	Initial X acceleration
AY0	Float	4	Initial Y acceleration
AZ0	Float	4	Initial Z acceleration
yaw0	Float	4	Initial X yaw
pitch0	Float	4	Initial Y pitch
roll0	Float	4	Initial Z roll
yaw0dot	Float	4	Initial X yaw rate
pitch0dot	Float	4	Initial Y pitch rate
roll0dot	Float	4	Initial Z roll rate
previous	Integer	4	last element in list
next	Integer	4	next element in list

5.1.2.1.5 databases structure SIMActivateRequestDataBase_type

Table 13 structure SIMActivateRequestDataBase_type

Item Name	Type	bytes	Description
valid	Char	1	
unit	Structure	20	OrganizationalUnit_type - 5.1.26

5.1.2.2 databases.h define statements

Table 14 databases.h define statements

Name	Value
MAX_DATAGRAMS	20
BUFF_SIZE	1500
PDU_NAME_NUM_OF_CHARS	40
MAX_ENTITIES	256
MAX_EVENTS	16
ERROR	-1
GE_Fulda_Gap	0
GE_Hunter_Liggett	1

5.1.3 declare_netif.h

This header file declares and dimensions variables to be used by the network interface software modules. Sizes are specified by define statements in databases.h (5.1.2.2) and netif.h (5.1.24.2). Tables 5.1.3.1 and 5.1.3.2 are not applicable.

5.1.3.3 declare_netif dimension statements

Table 15 declare_netif dimension statements

Name	Type	Bytes
selection	Char	4
frame_buff[MAX_DATAGRAMS+1][BUFF_SIZE]	Char array	31500
discard_buff[BUFF_SIZE]	Char array	1500
interface_na[MAX_IFNAME_SIZE]	Char array	10
SIM_interface_name[MAX_IFNAME_SIZE]	Char array	10
DIS_interface_name[MAX_IFNAME_SIZE]	Char array	10
init_DIS_tx_netif (initialized to FALSE)	Integer	4
init_SIM_tx_netif (initialized to FALSE)	Integer	4
DIS_rx_index (initialized to 0)	Integer	4
DIS_rx_etherFd	Integer	4
DIS_rx_sockFd	Integer	4
DIS_tx_etherFd	Integer	4
DIS_tx_sockFd	Integer	4
DIS_currentPDUNumber (initialized to 0)	Integer	4
SIM_rx_index (initialized to 0)	Integer	4
SIM_rx_etherFd	Integer	4
SIM_tx_etherFd	Integer	4
SIM_currentPDUNumber (initialized to 0)	Integer	4
SIM_rx_sockFd	Integer	4
SIM_tx_sockFd	Integer	4

5.1.4 declare_vars.h

This header file declares and dimensions variables to create the actual global shared memory variables. Table 5.1.4.2 is not applicable.

5.1.4.1 declare_vars structures

Table 16 declar_vars structures

Structure Name	bytes	Structure Type
dummy_DRA	40	DeadReckoning_type
DetDB [MAX_ENTITIES]		DetonationDataBase_type
StateDB [MAX_ENTITIES]		StateDataBase_type
IndFireDB [MAX_ENTITIES][MAX_EVENTS]		IndirectFireDataBase_type
VehAppDB [MAX_ENTITIES]		VehicleAppearanceDataBase_type
SActReqDB [MAS_ENTITIES]		SIMActivateRequestDataBase_type
dummy_TimeStamp		TimeStamp_type
Startup_Time		timeval
Time_Zone		timezone
Startup_TimeStamp		TimeStamp_type

5.1.4.3 declare_vars dimension statements

Table 17 declar_vars dimension statements

Name	Type	Bytes
My_site	Integer	2
DIS_netif_pid	Integer	4
DIS_sigrec (initialized to 0)	Integer	4
SIM_sigxmt (initialized to 0)	Integer	4
SIM_next_frame (initialized to 0)	Integer	4
SIM_send_frame (initialized to 0)	Integer	4
SIM_rm_left	Integer	4
DIS_remainder	Integer	4
DIS_consumed (initialized to 0)	Integer	2
SIM_out_ALPDU_kind	Char	1
DIS_PDU_in	Char pointer	4
SIM_PDU_out	Char pointer	4
DIS_rxbuf [BUFF_SIZE]	Char array	1500
SIM_txbuf [MAX_DATAGRAMS][BUFF_SIZE]	Char array	30000
SIM_netif_pid	Integer	4
SIM_sigrec (initialized to 0)	Integer	4
DIS_sigxmt (initialized to 0)	Integer	4
DIS_next_frame (initialized to 0)	Integer	4
DIS_send_frame (initialized to 0)	Integer	4
DIS_rm_left	Integer	2
SIM_remainder	Integer	4
SIM_consumed (initialized to 0)	Integer	2

DIS_out_ALPDU_kind	Char	1
SIM_PDU_in	Char pointer	4
DIS_PDU_out	Char pointer	4
SIM_rxbuff [BUFF_SIZE]	Char	1500
DIS_remainder	Integer	4
DIS_consumed (initialized to 0)	Integer	2
SIM_out_ALPDU_kind	Char	1
DIS_PDU_in	Char pointer	4
SIM_PDU_out	Char pointer	4
SIM_rxbuf [BUFF_SIZE]	Char array	1500
DIS_txbuf [MAX_DATAGRAMS][BUFF_SIZE]	Char array	30000
DIS_Assoc_Layer (initialized to TRUE)	Char	1
SIM_Assoc_Layer (initialized to TRUE)	Char	1
pass_all (initialized to TRUE)	Char	1
Xlat_DMC (initialized to TRUE)	Char	1
Exercise_ID	Char	1
Protocol_ID	Char	1
Battle_Scheme	Char	1
logger_datafile	File pointer	4
send_infile	Char array	80
ASSOC_hdr_leng	Const Char array	5
SIMPDU_name [NUMBER_SIM_PDUS][PDU_NAME_NUM_OF_CHARS]	Const String array	800
DISPDU_name [NUMBER_DIS_PDUS][PDU_NAME_NUM_OF_CHARS]	Const String array	760
DMCPDU_name [NUMBER_DMC_PDUS][PDU_NAME_NUM_OF_CHARS]	Const String array	600
SIMPDU_max_delta [NUMBER_SIM_PDUS]	Integer	40
SIMPDU_base_leng [NUMBER_SIM_PDUS]	Integer	40
DISPDU_max_delta [NUMBER_DIS_PDUS]	Integer	38
DISPDU_base_leng [NUMBER_DIS_PDUS]	Integer	38
DMCPDU_base_leng [NUMBER_DMC_PDUS]	Integer	30
SIMPDU_xlates_to [NUMBER_SIM_PDUS]	Integer	40
DISPDU_xlates_to [NUMBER_DIS_PDUS]	Integer	38
PDU_error [NUMBER_PDU_ERRORS][PDU_NAME_NUM_OF_CHARS]	Const String array	1000
terrain_database	Integer	4
ft_per_asec	Double	8
deg_to_rad	Double	8
meters_to_feet	Double	8
degrees_to_asec	Double	8
feet_to_meters	Double	8
pnumber	Integer	4
GE_DB_origin_lat	Double	8
GE_DB_origin_lon	Double	8
GE_DB_SW_corner_lat	Double	8

GE_DB_SW_corner_lon	Double	8
GE_DB_NE_corner_lat	Double	8
GE_DB_NE_corner_lon	Double	8
lat_ref	Double	8
lon_ref	Double	8
sw_latitude_asec	Double	8
sw_longitude_asec	Double	8
coslat	Double	8
xoffset	Double	8
yoffset	Double	8
Fulda_DB_origin_lat	Double	8
Fulda_DB_origin_lon	Double	8
Fulda_DB_SW_corner_lat	Double	8
Fulda_DB_SW_corner_lon	Double	8
Fulda_DB_NE_corner_lat	Double	8
Fulda_DB_NE_corner_lon	Double	8
lat_ref_fulda	Double	8
lon_ref_fulda	Double	8
sw_latitude_asec_fulda	Double	8
sw_longitude_asec_fulda	Double	8
coslat_fulda	Double	8
xoffset_fulda	Double	8
yoffset_fulda	Double	8
FGEGCX	Double array	80
FGEGCY	Double array	80
FGEGCZ	Double array	80
fgegcX	Double	8
fgegcY	Double	8
fgegcZ	Double	8
fgegcXO	Double	8
fgegcYO	Double	8
fgegcZO	Double	8
FGCGEX	Double array	80
FGCGEY	Double array	80
FGCGEZ	Double array	80
fgcgeX	Double	8
fgcgeY	Double	8
fgcgeZ	Double	8
fgcgeXO	Double	8
fgcgeYO	Double	8
fgcgeZO	Double	8
HGEGCX	Double array	80
HGEGCY	Double array	80
HGEGCZ	Double array	80
hgegcX	Double	8
hgegcY	Double	8
hgegcZ	Double	8
hgegcXO	Double	8
hgegcYO	Double	8

hgegcZO	Double	8
HGCGEX	Double array	80
HGCGEY	Double array	80
HGCGEZ	Double array	80
hgcgeX	Double	8
hgcgeY	Double	8
hgcgeZ	Double	8
hgcgeXO	Double	8
hgcgeYO	Double	8
hgcgeZO	Double	8
HUTMGCX	Double array	80
HUTMGCY	Double array	80
HUTMGCZ	Double array	80
hutmgcX	Double	8
hutmgcY	Double	8
hutmgcZ	Double	8
hutmgcXO	Double	8
hutmgcYO	Double	8
hutmgcZO	Double	8
HGCUTMX	Double array	80
HGCUTMY	Double array	80
HGCUTMZ	Double array	80
hgcutmX	Double	8
hgcutmY	Double	8
hgcutmZ	Double	8
hgcutmXO	Double	8
hgcutmYO	Double	8
hgcutmZO	Double	8
HUTM84GCX	Double array	80
HUTM84GCY	Double array	80
HUTM84GCZ	Double array	80
hutm84gcX	Double	8
hutm84gcY	Double	8
hutm84gcZ	Double	8
hutm84gcXO	Double	8
hutm84gcYO	Double	8
hutm84gcZO	Double	8
HGCUTM84X	Double array	80
HGCUTM84Y	Double array	80
HGCUTM84Z	Double array	80
hgcutm84X	Double	8
hgcutm84Y	Double	8
hgcutm84Z	Double	8
hgcutm84XO	Double	8
hgcutm84YO	Double	8
hgcutm84ZO	Double	8
angular_tolerance	Float	4
linear_tolerance	Float	4
minimum_timer_value	Float	4

cease_time	Float	4
rebroadcast_time	Float	4
head	Integer	4
current	Integer	4
last	Integer	4
ID_index	Integer	4
history_index	Integer	4
shmaddr	Char pointer	4
shmemptr	shmem_structure pointer	4
shmemid	Integer	4
childpid	Integer	4
extension	Integer	4
out_fp	File descriptor pointer	4
start_time	Double	8
end_time	Double	8
old_time1	Double	8
old_time2	Double	8
old_time3	Double	8
old_time4	Double	8
Netif [MAX_IFNAME_SIZE]	Char array	10
s1	Char array	100
s2	Char array	100

5.1.5 global_vars.h

This header file provides the data structures and definitions needed for each module that makes use of global storage. This header includes the following header files to maintain commonality among the various functions; DMC_PDU.h, decode_defines.h, decode_strs.h, decode_list.h, and decode_fprotos.h. These header files, included in this order, dimension and define the globally used structures and data entities.

5.1.6 decode_defines.h

This header file defines the sizes of the data structures used by the Digital Message Communications Console.

5.1.6.1.1 decode_defines structure Non_specific

Table 18 decode_defines structure Non_specific

Item Name	Type	bytes	Description
Header	Struct		DMC_Header_type
Common	Struct		DMC_CommonBlock_type

5.1.6.1.2 decode_defines define statements

Table 19 decode_defines define statements

Name	Value	Description
FREE_TEXT_WIDTH	26	width of Free Text line in Free Text message
NUM_HEADER_LINES	8	
NUM_LINES	15	
MAX_DMC_LINES	--	NUM_HEADER_LINES + NUM_LINES
STRING_SIZE	36	
NUM_DMC_LOGIC_TYPES	2	logic types
NUM_DMC_SPEEDS	2	speed types
NUM_DMC_TERM_TYPES	5	terminal types
NUM_DMC_PERSON_TYPES	10	person types
NUM_DMC_PRIORITY_TYPES	3	priority types
NUM_DMC_ACTIVITY_TYPES	11	activity types
NUM_DMC_TARGET_TYPES	10	target types
NUM_DMC_DIRECTION_TYPES	10	direction types
NUM_DMC_OBS_INTENTIONS	7	observer intention types
NUM_DMC_CLASSIF_TYPES	7	classification types
NUM_DMC_OBSTACLE_TYPES	10	obstacle types
NUM_DMC_BRIDGE_TYPES	11	bridge types
NUM_DMC_BRIDGE_DAMAGE_TYPES	6	bridge damage types
NUM_DMC_BRIDGE_MATERIAL_TYPES	6	bridge construction material types
NUM_DMC_ACTIVITY_LIKELY_TYPES	5	activity likely probability types
NUM_DMC_MISSION_STATUS_TYPES	5	mission status types
NUM_DMC_MISSION_TYPES	5	mission types
NUM_DMC_SHELL_TYPES	5	shell types
NUM_DMC_CONTROL_TYPES	4	control types
NUM_DMC_FUZE_TYPES	4	fuse types
NUM_DMC_TRAJECTORY_TYPES	4	trajectory types
NUM_DMC_DISPOSITION_TYPES	3	disposition types
NUM_DMC_CASUALTY_TYPES	2	casualty types
NUM_DMC_COVERAGE_TYPES	6	percent coverage types
NUM_DMC_TASK_TYPES	11	task types
NUM_DMC_WHO_TYPES	6	
NUM_DMC_WHEN_TYPES	5	
NUM_DMC_FAIL_TYPES	10	equipment failure types
NUM_DMC_REQ_TYPES	2	request types
NUM_DMC_REPORT_TYPES	7	report types
NUM_DMC_RECON_TYPES	7	recon types
NUM_DMC_NBC_DESC_TYPES	10	NBC message enumerations
NUM_DMC_BURST_TYPES	6	burst types
NUM_DMC_DELIVERY_TYPES	7	delivery types
NUM_DMC_CLOUD_HGT_TYPES	4	cloud height unit types

NUM_DMC_DESC_TYPES	8	MIJI descripiton types
NUM_DMC_TYPE_TYPES	6	
NUM_DMC_VISIBITY_TYPES	8	visibility types
NUM_DMC_RESTRICT_TYPES	10	restriction types
NUM_DMC_CLOUD_TYPES	5	cloud cover types
NUM_DMC_TURBINT_TYPES	6	turbulence intensity types
NUM_DMC_TURBFREQ_TYPES	5	turbulence frequency types
NUM_DMC_ICING_INTEN_TYPES	6	icing intensity types
NUM_DMC_ICING_TYPES	4	icing types
NUM_DMC_AIRCRAFT_TYPES	13	aircraft types
NUM_DMC_AIRCRAFT_STATUS_TYPES	5	aircraft status types
NUM_DMC_PILOT_STATUS_TYPES	6	pilot status types
MSG_LINE0	0	
MSG_LINE1	1	
MSG_LINE2	2	
MSG_LINE3	3	
MSG_LINE4	4	
MSG_LINE5	5	
MSG_LINE6	6	
MSG_LINE7	7	
MSG_LINE8	8	
MSG_LINE9	9	
MSG_LINE10	10	
MSG_LINE11	11	
MSG_LINE12	12	
MSG_LINE13	13	
MSG_LINE14	14	
CHAR_STR	4	
CHAR_STR1	8	
CHAR_STR2	16	
CHAR_PRINT	32	
FREE_TEXT_PER_LINE	26	

5.1.7 decode_fprotos.h

In C, function prototyping (the defining the input arguments and resultant output from the function) is required. The compiler will check the prototyping of a function against the actual arguments used within the source code of the calling function and flag as a compiler error any incongruities. This header file prototypes all of the functions used in the decode portion of the DMC.

5.1.8 decode_list.h

This header file provides the message strings that are needed by the decode modules. All of these arrays are filled within this header file to be indexed by the decode modules with pointers previously defined. Tables 5.1.8.1 and 5.1.8.2 are not defined for this header file.

5.1.8.3 decode_list dimension statements

Table 20 decode_list dimension statements

Name	Type	Bytes
decode_header_label	Const String array	288
decode_ack_label	Const String array	540
decode_freetext_value	Char array	540
decode_freetext_label	Const String array	540
decode_spot_label	Const String array	540
decode_gndroute	Const String array	540
decode_airroute	Const String array	540
decode_bridge	Const String array	540
decode_lzpz	Const String array	540
decode_bpop	Const String array	540
decode_crossing	Const String array	540
decode_repeat	Const String array	540
decode_cancel	Const String array	540
decode_check	Const String array	540
decode_cno	Const String array	540
decode_shift	Const String array	540
decode_nwmsn	Const String array	540
decode_mto	Const String array	540
decode_shot	Const String array	540
decode_splash	Const String array	540
decode_eom	Const String array	540
decode_request	Const String array	540
decode_movcmd	Const String array	540
decode_status	Const String array	540
decode_nbc1	Const String array	540
decode_nbc4	Const String array	540
decode_nbc5	Const String array	540
decode_miji	Const String array	540
decode_pirep	Const String array	540
decode_dnav	Const String array	540
decode_bda	Const String array	540
decode_month	Const String array	540
decode_subtype	Const String array	540
decode_recon_variation	Const String array	540
decode_artillery_variation	Const String array	540
decode_nbc_variation	Const String array	540

5.1.9 decode_list_proto.h

This header file prototypes the string arrays defined in **decode_list.h**, section 5.1.8, for use by the decode modules.

5.1.10 decode_strs.h

This header file provides the string values for "Enumerated" character types through the use of pointers, based on sizing provided by **define** statements in **decode_defines.h**, section 5.1.6. As these are strings, they are variable in length. The table will indicate the number of elements in each array. Tables 5.1.10.1 and 5.1.10.2 are not applicable.

5.1.10.3 decode_strs dimension statements

Table 21 decode_strs dimension statements

Name	Type	Number
message_enum_str	Enumerated String	15
invalid_entry	Enumerated String	1
logic_enum_str	Enumerated String	2
speed_enum_str	Enumerated String	2
term_enum_str	Enumerated String	5
person_enum_str	Enumerated String	10
priority_enum_str	Enumerated String	3
target_enum_str	Enumerated String	10
activity_enum_str	Enumerated String	11
direction_enum_str	Enumerated String	10
obs_int_enum_str	Enumerated String	7
classif_enum_str	Enumerated String	7
obstacle_enum_str	Enumerated String	10
bridge_enum_str	Enumerated String	11
bridge_damage_enum_str	Enumerated String	6
bridge_material_enum_str	Enumerated String	6
activity_likely_enum_str	Enumerated String	5
mission_status_enum_str	Enumerated String	5
mission_enum_str	Enumerated String	5
shell_enum_str	Enumerated String	7
control_enum_str	Enumerated String	4
fuze_enum_str	Enumerated String	4
trajectory_enum_str	Enumerated String	4
disposition_enum_str	Enumerated String	3
casualty_enum_str	Enumerated String	2
percent_cover_enum_str	Enumerated String	6
task_enum_str	Enumerated String	11
who_enum_str	Enumerated String	6
when_enum_str	Enumerated String	5
fail_enum_str	Enumerated String	10
request_enum_str	Enumerated String	2

report_enum_str	Enumerated String	7
recon_enum_str	Enumerated String	7
nbc_desc_enum_str	Enumerated String	10
burst_enum_str	Enumerated String	6
delivery_enum_str	Enumerated String	7
cloud_hgt_enum_str	Enumerated String	4
description_enum_str	Enumerated String	8
type_enum_str	Enumerated String	6
visibility_enum_str	Enumerated String	8
restriction_enum_str	Enumerated String	10
cloud_enum_str	Enumerated String	5
turbint_enum_str	Enumerated String	6
turbfreq_enum_str	Enumerated String	5
iceint_enum_str	Enumerated String	6
icing_hgt_enum_str	Enumerated String	4
aircraft_enum_str	Enumerated String	13
aircraft_status_enum_str	Enumerated String	5
pilot_status_enum_str	Enumerated String	6

5.1.11 decode_list_proto.h

This header file prototypes the string arrays defined in **decode_defines.h,** section 5.1.6, and enumerated in **decode_strs.h,** section 5.1.10, for use by the decode modules.

5.1.12 decode_vars.h

This header file provides the data structures and definitions needed for global commonality among the modules that make up the decode CSC of the DMCC. This header includes the following header files; DMC_PDU.h, decode_defines.h, decode_strs_proto.h, decode_list_proto.h, and decode_fprotos.h. These header files, included in this order, dimension and define the globally used structures and data entities.

5.1.13 DIS_PDU.h

This header contains the definitions of the PDU's which define the DIS 1.0 protocol, including the PDU's which are part of the protocol extensions. This header file implements the requirement for entity information and entity interaction protocol data units (PDU's) exchanged between simulators participating in a distributed interactive simulation.

5.1.13.1.1 DIS_PDU structure Header_type

Table 22 DIS_PDU structure Header_type

Item Name	Type	bytes	Description
version	Char	1	
exercise	Char	1	
type	Char	1	
unused	Char	1	

5.1.13.1.2 DIS_PDU structure ActivateRequest_type

Table 23 DIS_PDU structure ActivateRequest_type

Item Name	Type	bytes	Description
Header	Struct	4	Header_type - 5.1.13.1.1
ActivateHostID	Struct		ID_type
NewEntityID	Struct		ID_type
DateTime	Long	4	
Reason	Char	1	
Padding_3	Char array	3	
TerrainDBaseID	Struct		TerrainDBaseID_type
EntityType	Struct		Entity_type
Guise	Struct		Entity_type
Unit	Struct		Unit_type
Marking	Struct		EntityMarking_type
EntityCapabilities	Struct		EntityCapabilities_type
SubsystemParameters	Double	8	
EntityLocation	Struct		WorldCoord_type
Orientation	Struct		Orientation_type
DefualtUpdateThresholds	Struct		DefaultThresholds_type
NumberOfRadarSystems	Char	1	
NumberOfStores	Char	1	
NumberOfArticulationParameters	Char	1	
Padding	Char	1	
RadarSystems	Array of Struct		RadarSystems_type
Stores	Array of Struct		Supply_type
ArticulationParameters	Array of Struct		Articulation_type

5.1.13.1.3 DIS_PDU structure ActivateResponse_type

Table 24 DIS_PDU structure ActivateResponse_type

Item Name	Type	bytes	Description
Header	Struct	4	Header_type - 5.1.13.1.1
EntityID	Struct		ID_type
Timelimit	Integer	2	
ActivateResult	Char	1	
Padding_3	Char array	3	

5.1.13.1.4 DIS_PDU structure DeactivateRequest_type

Table 25 DIS_PDU structure DectivateRequest_type

Item Name	Type	bytes	Description
Header	Struct	4	Header_type - 5.1.13.1.1
EntityID	Struct		ID_type
DeactivateReason	Char	1	
Padding	Char	1	

5.1.13.1.4 DIS_PDU structure DeactivateResponse_type

Table 26 DIS_PDU structure DeactivateResposnse_type

Item Name	Type	bytes	Description
Header	Struct	4	Header_type - 5.1.13.1.1
EntityID	Struct		ID_type
DeactivateResult	Char	1	
Padding	Char	1	

5.1.13.1.5 DIS_PDU structure Collision_type

Table 27 DIS_PDU structure Collision_type

Item Name	Type	bytes	Description
Header	Struct	4	Header_type - 5.1.13.1.1
IssuingEntityID	Struct		ID_type
CollidingEntityID	Struct		ID_type
EventID	Struct		Event_ID_type
Padding_2	Integer	2	
TimeStamp	Struct		TimeStamp_type
Velocity	Struct		LinearVelocity_type
Mass	Float	4	
Location	Struct		EntityCoord_type

5.1.13.1.6 DIS_PDU structure Detonation_type

Table 28 DIS_PDU structure Collision_type

Item Name	Type	bytes	Description
Header	Struct	4	Header_type - 5.1.13.1.1
FiringEntityID	Struct		ID_type
TargetEntityID	Struct		ID_type
MunitionID	Struct		ID_type
EventID	Struct		Event_ID_type
TimeStamp	Struct		TimeStamp_type
Location	Struct		WorldCoord_type
BurstLocation	Struct		BurstLocation_type
Velocity	Struct		LinearVelocity_type
Impact	Struct		EntityCoord_type
DetonationResult	Char	1	
NumberOfArticulationParameters	Char	1	
Padding	Char array	6	
ArticulationParameters	Array of Struct		Articulation_type

5.1.13.1.7 DIS_PDU structure Emitter_type

Table 29 DIS_PDU structure Emitter_Type

Item Name	Type	bytes	Description
Header	Struct	4	Header_type - 5.1.13.1.1
EmittingEntityID	Struct		ID_type
Padding_2	Integer	2	
EntityType	Struct		Entity_type
TimeStamp	Struct		TimeStamp_type
EntityLocation	Struct		WorldCoord_type
EntityLinearVelocity	Struct		LinearVelocity_type
EntityOrientation	Struct		Orientation_type
DeadReckoningParameters	Struct		DeadReckoning_type
NumberOfEmitters	Integer	2	
DatabaseNumber	Integer	2	
Emitter	Array of Struct		Emitters_type

5.1.13.1.8 DIS_PDU structure EntityState_type

Table 30 DIS_PDU structure EntityState_type

Item Name	Type	bytes	Description
Header	Struct	4	Header_type - 5.1.13.1.1
EntityID	Struct		ID_type
Padding	Char	1	
ForceID	Char	1	
EntityType	Struct		Entity_type
Guise	Struct		Entity_type
TimeStamp	Struct		TimeStamp_type
EntityLocation	Struct		WorldCoord_type
EntityLinearVelocity	Struct		LinearVelocity_type
EntityOrientation	Struct		Orientation_type
DeadReckoningParameters	Struct		DeadReckoning_type
EntityAppearance	Struct		EntityAppearance_type
EntityMarking	Struct		EntityMarking_type
Capabilities	Struct		EntityCapabilities_type
Padding_3	Char array	3	
NumberOfArticulationParameters	Char	1	
ArticulationParameters	Array of Struct		Articulation_type

5.1.13.1.9 DIS_PDU structure Fire_type

Table 31 DIS_PDU structure Fire_type

Item Name	Type	bytes	Description
Header	Struct	4	Header_type - 5.1.13.1.1
FiringEntityID	Struct		ID_type
TargetEntityID	Struct		ID_type
MunitionID	Struct		ID_type
EventID	Struct		Event_ID_type
TimeStamp	Struct		TimeStamp_type
Location	Struct		WorldCoord_type
BurstDescription	Struct		BurstDescription_type
Velocity	Struct		LinearVelocity_type
Range	Float	4	

5.1.13.1.10 DIS_PDU structure Radar_type

Table 32 DIS_PDU structure Radar_type

Item Name	Type	bytes	Description
Header	Struct	4	Header_type - 5.1.13.1.1
EmittingEntityID	Struct		ID_type
EventID	Struct		Event_ID_type
TimeStamp	Struct		TimeStamp_type
Padding_3	Char array	3	
NumberOfRadarSystems	Char	1	
RadarSystems	Array of Struct		RadarArray_type

5.1.13.1.11 DIS_PDU structure RepairComplete_type

Table 33 DIS_PDU RepairComplete_type

Item Name	Type	bytes	Description
Header	Struct	4	Header_type - 5.1.13.1.1
RequestingEntityID	Struct		ID_type
ServicingEntityID	Struct		ID_type
Repair	Integer	2	
Padding_2	Integer	2	

5.1.13.1.12 DIS_PDU **structure** RepairResponse_type

Table 34 DIS_PDU structure ReparirResponse_type

Item Name	Type	bytes	Description
Header	Struct	4	Header_type - 5.1.13.1.1
RequestingEntityID	Struct		ID_type
ServicingEntityID	Struct		ID_type
RepairResult	Char	1	
Padding_3	Char array	3	

5.1.13.1.13 DIS_PDU **structure** Resupply_type

Table 35 DIS_PDU structure Resupply_type

Item Name	Type	bytes	Description
Header	Struct	4	Header_type - 5.1.13.1.1
RequestingEntityID	Struct		ID_type
ServicingEntityID	Struct		ID_type
NumberOfSupplyTypes	Char	1	
Padding_3	Char array	3	
Supply	Array of Struct		Supply_type

5.1.13.1.14 DIS_PDU **structure** ResupplyCancel_type

Table 36 DIS_PDU structure ResupplyCancel_type

Item Name	Type	bytes	Description
Header	Struct	4	Header_type - 5.1.13.1.1
RequestingEntityID	Struct		ID_type
ServicingEntityID	Struct		ID_type

5.1.13.1.15 DIS_PDU structure ServiceRequest_type

Table 37 DIS_PDU structure ServiceRequest_type

Item Name	Type	bytes	Description
Header	Struct	4	Header_type - 5.1.13.1.1
RequestingEntityID	Struct		ID_type
ServicingEntityID	Struct		ID_type
ServiceType	Char	1	
NumberOfSupplyTypes	Char	1	
Padding_2	Integer	2	
Supply	Array of Struct		Supply_type

5.1.13.1.16 DIS_PDU structure UpdateThresholdRequest_type

Table 38 DIS_PDU structure UpdateThresholdRequest_type

Item Name	Type	bytes	Description
Header	Struct	4	Header_type - 5.1.13.1.1
RequestorEntityID	Struct		ID_type
ResponderEntityID	Struct		ID_type
LinearThreshold	Struct		EntityCoord_type
RotationalThreshold	Struct		Orientation_type
DurationOfChange	Integer	4	

5.1.13.1.17 DIS_PDU structure UpdateThresholdResponse_type

Table 39 DIS_PDU structure UpdateThresholdResponse_type

Item Name	Type	bytes	Description
Header	Struct	4	Header_type - 5.1.13.1.1
RequestorEntityID	Struct		ID_type
ResponderEntityID	Struct		ID_type
Result	Char	1	
RemainingTime	Char	1	
Padding_2	Integer	2	

5.1.14 DIS_types.h

This header file contains definitions of various basic data elements, including how these elements are represented as communicated bits. The elements compose the fields which compose each PDU.

5.1.14.1.1 DIS_types structure Land

Table 40 DIS_types structure Land

Item Name	Type	bytes	Description
Unused	Long	4	
Hatch	Long	4	
Engine_smoke	Long	4	
Launcher	Long	4	
Paint_scheme	Long	4	
Dust_cloud	Long	4	
Flaming	Long	4	
Smoke_plume	Long	4	
Destroyed	Long	4	

5.1.14.1.2 DIS_types structure Air

Table 41 DIS_types structure Land

Item Name	Type	bytes	Description
Unused	Long	4	
Formation_lights	Long	4	
Navigation_lights	Long	4	
Running_lights	Long	4	
After_burner	Long	4	
Flaming	Long	4	
Destroyed	Long	4	

5.1.14.1.3 DIS_types structure Surface

Table 42 DIS_types structure Surface

Item Name	Type	bytes	Description
Unused	Long	4	
Running_lights	Long	4	
Wake	Long	4	
Flaming	Long	4	
Destroyed	Long	4	

5.1.14.1.4 DIS_types structure Subsurface

Table 43 DIS_types structure Subsurface

Item Name	Type	bytes	Description
Unused	Long	4	
Hatch	Long	4	
Running_lights	Long	4	
Wake	Long	4	
Destroyed	Long	4	

5.1.14.1.5 DIS_types structure Space

Table 44 DIS_types structure Space

Item Name	Type	bytes	Description
Unused	Long	4	
Destroyed	Long	4	

5.1.14.1.6 DIS_types structure GuidedMunitions

Table 45 DIS_types structure GuidedMunitions

Item Name	Type	bytes	Description
Unused	Long	4	
Rocket_flame	Long	4	
Launch_flash	Long	4	

5.1.14.1.7 DIS_types structure Lifeforms

Table 46 DIS_types structure Lifeforms

Item Name	Type	bytes	Description
Unused	Long	4	
Weapon_1	Long	4	
Weapon_2	Long	4	
Lifeform_state	Long	4	

5.1.14.1.8 DIS_types structure Environmentals

Table 47 DIS_types structure Environmentals

Item Name	Type	bytes	Description
Unused	Long	4	
Damage	Long	4	

5.1.14.1.9 DIS_types structure CulturalFeatures

Table 48 DIS_types structure CulturalFeatures

Item Name	Type	bytes	Description
Unused	Long	4	
Flaming	Long	4	
Smoke_plume	Long	4	
Damage	Long	4	

5.1.14.1.10 DIS_types structure ID_type

Table 49 DIS_types structure ID_type

Item Name	Type	bytes	Description
Site	Integer	2	
Host	Integer	2	
Entity	Integer	2	

5.1.14.1.11 DIS_types structure Event_ID_type

Table 50 DIS_types structure Event_ID_type

Item Name	Type	bytes	Description
Site	Integer	2	
Host	Integer	2	
Event	Integer	2	

5.1.14.1.12 DIS_types structure Entity_type

Table 51 DIS_types structure Event_ID_type

Item Name	Type	bytes	Description
EntityKind	Char	1	
Domain	Char	1	
Country	Integer	2	
Category	Char	1	
Subcategory	Char	1	
specific	Char	1	
Extra	Char	1	

5.1.14.1.13 DIS_types structure Supply_type

Table 52 DIS_types structure Supply_type

Item Name	Type	bytes	Description
Entity	Struct		Entity_type - 5.1.13.1.18
Quanity	Float	4	

5.1.14.1.15 DIS_types structure DefaultThresholds_type

Table 53 DIS_types structure DefaultThresholds_type

Item Name	Type	bytes	Description
X	Float	4	
Y	Float	4	
Z	Float	4	
Psi	Long	4	
Theta	Long	4	
Phi	Long	4	

5.1.14.1.16 DIS_types structure WorldCoord_type

Table 54 DIS_types structure WorldCoord_type

Item Name	Type	bytes	Description
X	Double	8	
Y	Double	8	
Z	Double	8	

5.1.14.1.17 DIS_types structure EntityCoord_type

Table 55 DIS_types structure EntityCoord_type

Item Name	Type	bytes	Description
X	Float	4	
Y	Float	4	
Z	Float	4	

5.1.14.1.18 DIS_types structure AngularVelocity_type

Table 56 DIS_types structure AngularVelocity_type

Item Name	Type	bytes	Description
Psi	Integer	4	
Theta	Integer	4	
Phi	Integer	4	

5.1.14.1.19 DIS_types structure FloatAngles_type

Table 57 DIS_types structure FloatAngles_type

Item Name	Type	bytes	Description
Psi	Float	4	
Theta	Float	4	
Phi	Float	4	

5.1.14.1.20 DIS_types structure LinearVelocity_type

Table 58 DIS_types structure LinearVelocity_type

Item Name	Type	bytes	Description
X	Float	4	
Y	Float	4	
Z	Float	4	

5.1.14.1.21 DIS_types structure Orientation_type

Table 59 DIS_types structure Orientation_type

Item Name	Type	bytes	Description
Psi	Long	4	
Theta	Long	4	
Phi	Long	4	

5.1.14.1.22 DIS_types structure DeadReckoning_type

Table 60 DIS_types structure DeadReconing_type

Item Name	Type	bytes	Description
DeadReckoningAlgorithm	Char	1	
OtherParameters	Char array	15	
LinearAcceleration	Struct	12	LinearVelocity_type - 5.1.14.1.20
AngularVelocity	Struct	12	AngularVelocity_type - 5.1.14.1.18

5.1.14.1.23 DIS_types structure EntityMarking_type

Table 61 DIS_types structure EntityMarking_type

Item Name	Type	bytes	Description
CharacterSet	Char	1	
CharacterString	Char array	11	

5.1.14.1.24 DIS_types structure TerrainDBaseID_type

Table 62 DIS_types structure TerrainDBaseID_type

Item Name	Type	bytes	Description
CharacterString	Char array	11	
VersionNumber	Char	1	

5.1.14.1.25 DIS_types structure BurstDescription_type

Table 63 DIS_types structure BurstDescription_type

Item Name	Type	bytes	Description
Munition	Struct	6	Entity_type
Warhead	Integer	2	
Fuze	Integer	2	
Quanity	Integer	2	
Rate	Integer	2	

5.1.14.1.26 DIS_types structure Army

Table 64 DIS_types structure Army

Item Name	Type	bytes	Description
Army_Number	Char	1	
Corps_Number	Char	1	
Division_Number	Char	1	
Regiment_Number	Char	1	
Battalion_Number	Char	1	
Company_Number	Char	1	
Platoon_Number	Char	1	
Squad_Number	Char	1	

5.1.14.1.27 DIS_types structure AirForce

Table 65 DIS_types structure AirForce

Item Name	Type	bytes	Description
AirForce_Number	Char	1	
Wing_Number	Char	1	
Squadron_Number	Char	1	
Flight_Number	Char	1	
Division_Number	Char	1	
Section_Number	Char	1	
Spare	Char array	2	

5.1.14.1.28 DIS_types structure CoastGuard

Table 66 DIS_types structure CoastGuard

Item Name	Type	bytes	Description
Spare	Char array	8	

5.1.14.1.29 DIS_types structure Marines

Table 67 DIS_types structure Marines

Item Name	Type	bytes	Description
Fleet_Number	Char	1	
Force_Number	Char	1	
Brigade_Number	Char	1	
Unit_Number	Char	1	
Spare	Char array	4	

5.1.14.1.30 DIS_types structure Navy

Table 68 DIS_types structure Navy

Item Name	Type	bytes	Description
Fleet_Number	Char	1	
Force_Number	Char	1	
Group_Number	Char	1	
Unit_Number	Char	1	
Element_Number	Char	1	
Spare	Char array	3	

5.1.14.1.31 DIS_types structure Unit_type

Table 69 DIS_types structure Unit_type

Item Name	Type	bytes	Description
Force	Char	1	
Country	Char array	2	
Service	Char	1	
Hierarchy	Union of Struct	8	Either Army, AirForce, CoastGuard, Marines, Navy - 5.1.14.1.26 to 5.1.14.1.30

5.1.14.1.32 DIS_types structure EntityCapabilities_type

Table 70 DIS_types structure EntityCapabilities_type

Item Name	Type	bytes	Description
Unused	Long	4	
Repair	Long	4	
MiscellaneousSupply	Long	4	
FuelSupply	Long	4	
AmmunitionSupply	Long	4	

5.1.14.1.33 DIS_types structure RadarSystems_type

Table 71 DIS_types structure RadarSystems_type

Item Name	Type	bytes	Description
Category	Char	1	
Subcategory	Char	1	
RadarSystemID	Integer	2	

5.1.14.1.34 DIS_types structure Illumined_type

Table 72 DIS_types structure Illumined_type

Item Name	Type	bytes	Description
TargetID	Struct		ID_type
Padding_2	Integer	2	
RadarData	Long	4	

5.1.14.1.35 DIS_types structure Sweep_type

Table 73 DIS_types structure Sweep_type

Item Name	Type	bytes	Description
AzimuthCenter	Long	4	
AzimuthWidth	Long	4	
ElevationCenter	Long	4	
ElevationWidth	Long	4	

5.1.14.1.36 DIS_types structure Emitters_Type_type

Table 74 DIS_types structure Emitters_Type_type

Item Name	Type	bytes	Description
Class	Char	1	
ModeNumber	Char	1	
DBaseEntryNumber	Integer	2	

5.1.14.1.37 DIS_types structure Emitters_Data_type

Table 75 DIS_types structure Emitters_Data_type

Item Name	Type	bytes	Description
Power	Integer	2	
ParameterNumber1	Integer	2	
ParameterNumber2	Integer	2	
ParameterNumber3	Integer	2	

5.1.14.1.38 DIS_types structure Emitters_type

Table 76 DIS_types structure Emitters_type

Item Name	Type	bytes	Description
Type	Struct	4	Emitters_Type_type - 5.1.14.1.36
Parameters	Struct	8	Emitters_Data_type - 5.1.14.1.37
Sweep	Struct	16	Sweep_type - 5.1.14.1.35

5.1.14.1.39 DIS_types structure RadarBaseDate_type

Table 77 DIS_types structure RadarBaseDate_type

Item Name	Type	bytes	Description
Location	Struct		EntityCoord_type
RadarSystem	Struct	8	RadarSystems_type
Power	Integer	2	
RadarMode	Char	1	
NumberOfIlluminated	Char	1	
SpecificData	Long array	8	
Sweep	Struct	16	Sweep_type - 5.1.14.1.35

5.1.14.1.40 DIS_types structure RadarArray_type

Table 78 DIS_types structure RadarArray_type

Item Name	Type	bytes	Description
RadarSystemData	Struct	4	RadarBaseData_type
Illumined	Array of Struct		Illumined_type

5.1.14.1.41 DIS_types structure TimeStamp_type

Table 79 DIS_types structure TimeStamp_type

Item Name	Type	bytes	Description
value	Long	4	
reference	Long	4	

5.1.14.2 DIS_types define statements

Table 80 DIS_types define statements

Name	Value	Description
LIFEFORM_STATE_DESTROYED	0	
LIFEFORM_STATE_UPRIGHT	1	
LIFEFORM_STATE_KNEELING	2	
LIFEFORM_STATE_PRONE	3	
PASS_ALL_ID_CODE	0xFFFF	
IGONRE_ALL_ID_CODE	0	
ENTITY_ID_SIZE	6	
OTHER_KIND	0	
PLATFORM_KIND	1	
MUNITION_KIND	2	
LIFE_FORM_KIND	3	
ENVIRONMENTAL_KIND	4	
CULTURAL_FEATURE_KIND	5	
SUPPLIES_KIND	6	
LAND_DOMAIN	1	
AIR_DOMAIN	2	
SURFACE_DOMAIN	3	
SUBSURFACE_DOMAIN	4	
SPACE_DOMAIN	5	
TANK	1	
ARMORED_FIGHTING_VEH	2	
ARMORED_UTILITY_VEH	3	
SELF_PROPELLED_ARTILLERY	4	
TOWED_ARTILLERY	5	
SMALL_WHEELED_UTILITY_VEH	6	
LARGE_WHEELED_UTILITY_VEH	7	
SMALL_TRACKED_UTILITY_VEH	8	
LARGE_TRACKED_UTILITY_VEH	9	
DIS_MORTAR	10	
AMPHIBIOUS_ASSAULT_VEH	50	
MISC_WEAPONS_SYSTEMS	51	
DIS_M1_TANK	1	

DIS_M60_TANK	2	
DIS_M551_TANK	3	
DIS_M48_TANK	4	
DIS_IFV_AFV	1	
DIS_M2_AFV	3	
DIS_M3_AFV	4	
DIS_M113_AFV	5	
DIS_HUMMV_SWUV	1	
DIS_M577_SWUV	2	
DIS_FAV_SWUV	3	
DIS_M352_LWUV	1	
DIS_M977_LWUV	2	
DIS_M978_LWUV	3	
DIS_M548_LTUV	1	
DIS_MLRS_SPA	1	
DIS_M110_SPA	2	
DIS_M109_SPA	3	
DIS_GLCM_LAUNCHER_SPA	4	
DIS_M198_155MM_TOWED_ART	2	
DIS_M57_TOWED_ART	11	
DIS_M128_GEMSS_TOWED_ART	12	
DIS_M58A1_TOWED_ART	14	
DIS_M88_AUV	1	
DIS_M578_AUV	2	
DIS_M29_MORTAR	1	
DIS_M224_MORTAR	2	
DIS_MK19_GRENADE_LAUNCHER	2	
DIS_VULCAN_ADS_SAM	3	
DIS_M113A2_MODEL	1	
DIS_M88A1_MODEL	1	
DIS_M352_MODEL	1	
DIS_M352A2_MODEL	2	
DIS_USSR_T80_TANK	1	
DIS_USSR_T72M_TANK	2	
DIS_USSR_T64_TANK	3	
DIS_USSR_T62_TANK	4	
DIS_USSR_T54_TANK	5	
DIS_USSR_T55_TANK	6	
DIS_USSR_T10_TANK	7	
DIS_USSR_PT76_TANK	8	
DIS_USSR_D20_TOWED_ART	3	
DIS_USSR_D30_122MM_TOWED_ART	4	
DIS_USSR_D30FIELD_TOWED_ART	6	
DIS_USSR_M1965_TOWED_ART	10	
DIS_USSR_M1943_MORTAR	2	
DIS_USSR_M1952_MORTAR	4	
DIS_USSR_SS12_SPA	11	
DIS_USSR_BM21_MRLS_SPA	13	
DIS_USSR_BM24_MRLS_SPA	15	

DIS_USSR_BMP1_AFV	1	
DIS_USSR_BMP2_AFV	2	
DIS_USSR_BRDM1_AFV	3	
DIS_USSR_BRDM2_AFV	4	
DIS_USSR_BTR40_AFV	5	
DIS_USSR_BTR153_AFV	6	
DIS_USSR_BTR60_70_AFV	8	
DIS_USSR_ACRV_AFV	10	
DIS_USSR_BTS2_AUV	1	
DIS_USSR_MTP_AUV	2	
DIS_USSR_IMR2_AUV	3	
DIS_USSR_IPR_AUV	4	
DIS_USSR_MAV_SWUV	3	
DIS_USSR_GAZ63_LWUV	7	
DIS_USSR_GAZ66_LWUV	8	
DIS_USSR_URAL375_LWUV	9	
DIS_USSR_FROG5_SSML	8	
DIS_USSR_FROG7_SSML	9	
DIS_USSR_SCUDB_SSML	10	
DIS_USSR_PTS_XPORT_LTUV	1	
USSR_BMP1K_AFV_MODEL	1	
USSR_T72_TANK_MODEL	1	
USSR_T72M_TANK_MODEL	2	
FIGHTER	1	
ATTACK	2	
DIS_BOMBER	3	
CARGO	4	
ANTI_SUBMARINE	5	
PATOL	5	
ATTACK_HELICOPTER	6	
UTILITY_HELICOPTER	7	
CARGO_HELICOPTER	50	
OBSERVATION_HELICOPTER	51	
SEA_GOING_HELICOPTER	52	
ELECTRONIC_WARFARE	53	
DIS_A4_ATTACK	1	
DIS_A7_ATTACK	3	
DIS_A10_ATTACK	4	
DIS_F117_STEALTH	1	
DIS_F14_FIGHTER	2	
DIS_F16_FIGHTER	3	
DIS_FB_111_FIGHTER	4	
DIS_F4S_FIGHTER	5	
DIS_F15_FIGHTER	7	
DIS_FA_18_FIGHTER	14	
DIS_F5F_FIGHTER	16	
DIS_B52_BOMBER	3	
DIS_C130_CARGO	1	
DIS_C5_CARGO	2	

CIS_C141_CARGO	3	
DIS_P3_PATROL	1	
DIS_PIONEER_RPV_PATROL	8	
DIS_E2_ELEC_WARFARE	1	
DIS_MODEL_F14A	1	
DIS_MODEL_F14D	4	
DIS_AH64_COPTER	1	
DIS_AH1_COPTER	2	
DIS_OH58_COPTER	1	
DIS_OH58C_COPTER	2	
DIS_OH58D_COPTER	3	
DIS_UH1_COPTER	1	
DIS_UH60_COPTER	2	
DIS_CH47_COPTER	1	
DIS_CH53_COPTER	2	
DIS_MIG31_FIGHTER	1	
DIS_MIG29_FIGHTER	2	
DIS_SU27_FIGHTER	3	
DIS_MIG25_FIGHTER	4	
DIS_MIG23_FIGHTER	5	
DIS_MIG21_FIGHTER	6	
DIS_MIG27_ATTACK	1	
DIS_SU25_ATTACK	8	
DIS_YAK36_ATTACK	10	
DIS_TU16_BOMBER	2	
DIS_TU26_BOMBER	3	
DIS_TU22_BOMBER	4	
DIS_TU142_BOMBER	5	
DIS_TU126_PATROL	3	
DIS_AN124_CARGO	1	
DIS_AN22_CARGO	2	
DIS_II76_CARGO	3	
DIS_MI28_COPTER	1	
DIS_MI24_COPTER	2	
DIS_KA136_COPTER	3	
DIS_MI8_COPTER	1	
DIS_MI14_COPTER	2	
DIS_MI26_COPTER	3	
DIS_MI6_COPTER	4	
DIS_MI2_COPTER	5	
DOS_MIRAGE_F1_FIGHTER	1	
DIS_SA342_GAZELLE	5	
DIS_UK_CHIEFTAIN_TANK	1	
CARRIER	1	
COMMAND_SHIP	2	
GUIDED_MISSLE_CRUISER	3	
GUIDED_MISSLE_DESTROYER	4	
DESTROYER	5	
GUIDED_MISSLE_FRIGATE	6	

PATROL_CRAFT	7	
MINE_COUNTERMEASURE	8	
DOCK_LANDING_SHIP	9	
TANK_LANDING_SHIP	10	
LANDING_CRAFT	11	
FRIGATE	50	
BATTLESHIP	51	
HEAVY_CRUISER	52	
DESTROYER_TENDER	53	
AMPHIBIOUS_ASSAULT_SHIP	54	
AMPHIBIOUS_CARGO_SHIP	55	
AMPHIBIOUS_TRANSPORT_DOCK	56	
AMMUNITION_SHIP	57	
COMBAT_STORES_SHIP	58	
DIS_USS_TAWARA_CLASS	1	
DIS_USS_IWO_JIMA_CLASS	2	
DIS_USS_WASP_CLASS	3	
DIS_USS_CHARLESTON_CLASS	1	
DIS_USS_AUSTIN_CLASS	1	
DIS_USS_RALIEGH_CLASS	2	
DIS_USS_WHIDBEY_ISLANE_CLASS	1	
DIS_USS_ANCHORAGE_CLASS	2	
DIS_USS_THOMASTON_CLASS	3	
DIS_USS_NEWPORT_CLASS	1	
DIS_USS_DESOTO_COUNTY_CLASS	2	
DIS_USS_LCAC_CLASS	1	
DIS_USS_NIMITZ_CLASS	1	
DIS_USS_ENTERPRISE_CLASS	2	
DIS_USS_KITTYHAWK_CLASS	3	
DIS_USS_FORRESTAL_CLASS	4	
DIS_USS_MIDWAY_CLASS	5	
DIS_USS_BLUERIDGE_CLASS	1	
DIS_MISC_CONVERT_CMD_SHIPS	2	
DIS_USS_TICONDEROGA_CLASS	1	
DIS_USS_VIRGINIA_CLASS	2	
DIS_USS_CALIFORNIA_CLASS	3	
DIS_USS_TRUXTUN_CLASS	4	
DIS_USS_BELKNAP_CLASS	5	
DIS_USS_BAINBRIDGE_CLASS	6	
DIS_USS_LEAHY_CLASS	7	
DIS_USS_ALBANY_CLASS	8	
DIS_USS_LONGBEACH_CLASS	9	
DIS_USS_ARLEIGHBURKE_CLASS	1	
DIS_USS_KIDD_CLASS	2	
DIS_USS_FARRAGUT_CLASS	3	
DIS_USS_CHARLESFADAMS_CLASS	4	
DIS_USS_SPRUANCE_CLASS	1	
DIS_USS_OHPERRY_CLASS	1	
DIS_USS_BROOKE_CLASS	2	

DIS_USS_PEGASUS_CLASS	1	
DIS_USS_HIGHPOINT_CLASS	2	
DIS_USS_MM_SPEC_WARFARE_CLASS	3	
DIS_USS_KNOX_CLASS	1	
DIS_USS_GARCIA_CLASS	2	
DIS_USS_BFONSTEIN_CLASS	3	
DIS_USS_IOWA_CLASS	1	
DIS_USS_DESMOINES_CLASS	1	
DIS_USS_SAM_GOMPERS_CLASS	1	
DIS_USS_DIXIE_CLASS	2	
DIS_USS_KILAUEA_CLASS	1	
DIS_USS_MARS_CLASS	2	
DIS_DD_964_PAULFFOSTER	1	
DIS_DD_963_SPRUANCE	2	
DIS_DD_976_MERRILL	14	
DIS_DD_978_STUMP	16	
DIS_DDG_37_FARRAGUT	1	
DIS_DDG_39_MACDONOUGH	3	
DIS_DDG_40_COONTZ	4	
DIS_CG47_TICONDEROGA	1	
DIS_CG48_YORKTOWN	2	
DIS_CG10_ALBANY	1	
DIS_PCH_1	1	
DIS_PBM_1	1	
DIS_CA134_DESMOINES	1	
DIS_FF1037_BRONSTEIN	1	
DIS_LCAC	1	
DIS_TAE26_KILAUEA	1	
DIS_TAFS8_SIRIUS	1	
DIS_AFS1_MARS	1	
DIS_AD41_YELLOWSTONE	3	
DIS_AD42_ACADIA	4	
DIS_AD15_PRAIRIE	1	
MCM_MSH_TYPE	1	
MCM_AVENGER_CLASS	2	
MCM_ACME_CLASS	3	
MCM_AGILE_AGRESS_CLASS	4	
MSM_1_AVENGER	1	
MSM_509_ADROIT	1	
MSM_427_CONSTANT	1	
STRATEGIC_MISSLE_SUB	1	
ATTACK_SUB	2	
RESEARCH_SUB	50	
DIS_USS_OHIO_CLASS	1	
DIS_USS_BEN_FRANKLIN_CLASS	2	
DIS_USS_LOS_ANGELES_CLASS	1	
DIS_USS_STURGEON_CLASS	4	
DIS_USS_SKIPJACK_CLASS	7	
DIS_USS_BARBEL_CLASS	8	

DIS_USS_TRITON_CLASS	10	
FRANKLIN_CLASS_OFFSET	19	
DIS_USS_DOLPHIN_CLASS	1	
DIS_SSBN_616_LAFAYETTE	1	
DIS_SSN_585_SKIPJACK	1	
DIS_SSN_590_SCULPIN	3	
DIS_SSN_590_SNOOK	5	
DIS_SSN_586_TRITON	1	
DIS_AGSS_555_DOLPHIN	1	
MANNED_SPACE_PLATFORM	1	
UNMANNED_SPACE_PLATFORM	2	
SATELLITE_USP	1	
SPACE_SHUTTLE_MSP	1	
SPACE_STATION_MSP	2	
GUIDED_MUNITIONS	1	
BALLISTIC_MUNITIONS	2	
FIXED_MUNITIONS	3	
BOMB_MUNITIONS	4	
FUZE_MUNITIONS	5	
DIS_TMD_B	1	
DIS_YAM_5	2	
DIS_TM_46	5	
DIS_TMN_46	6	
DIS_TM_57	7	
DIS_TM_62M	8	
DIS_TMK_2	9	
DIS_POMZ_2	10	
DIS_OMZ_3	11	
DIS_MON	12	
DIS_PMN	13	
DIS_PMK_40	14	
ANTI_AIR_DOMAIN	1	
ANTI_ARMOR_DOMAIN	2	
ANTI_GUIDED_MUNITION_DOMAIN	3	
ANTI_RADAR_DOMAIN	4	
ANTI_SATELLITE_DOMAIN	5	
ANTI_SHIP_DOMAIN	6	
ANTI_SUBMARINE_DOMAIN	7	
BATTLEFIELD_SUPPORT_DOMAIN	8	
STRATEGIC_DOMAIN	9	
MISCELLANEOUS_MUNITIONS_DOMAIN	10	
ANTI_PERSONNEL	1	
ANTI_ARMOR	2	
BATTLEFIELD_SUPPORT	3	
DIS_SIDEWINDER	1	
DIS_ADATS	2	
DIS_STINGER	3	
TOMAHAWK_AGM_109	5	
TOMAHAWK_BGM_109G	6	

DIS_GREMLIN	1	
TOW_MISSLE	1	
M47_DRAGON	2	
DIS_SWATTER	4	
DIS_SPANDREL	7	
DIS_SPIRAL	8	
DIS_SONGSTER	10	
DIS_STYX	8	
LAND_LIFEFORM	1	
DIS_OTHER_INFANTRY	0	
DIS_DISMOUNTED_INFANTRY	1	
NUM_TEAM	101	
NUM_SQUAD	102	
NUM_PLATOON	103	
NUM_COMPANY	104	
NUM_BATTALION	105	
NUM_REGIMENT	106	
NUM_DIVISION	107	
NUM_CORPS	108	
NUM_ARMY	109	
DIS_M72_LAW	4	
DIS_M60_GPMG	6	
DIS_RPG7_VAT_ROCKET_LAUNCHER	4	
DIS_AA52_MG	6	
BRIDGE_CONCRETE_TWO_LANE	1	
BRIDGE_CONCRETE_FOUR_LANE	2	
BRIDGE_TRUSS_TWO_LANE	3	
BRIDGE_TRUSS_FOUR_LANE	4	
BRIDGE_SUSPENSION_TWO_LANE	5	
BRIDGE_SUSPENSION_FOUR_LANE	6	
BUILDING_ONE_STORY	7	
BUILDING_TWO_STORY	8	
BUILDING_THREE_STORY	9	
BUILDING_FOUR_STORY	10	
MINEFIELD_MARKERS	17	
BREACHED_LANE_FLAGS	18	
SIZE_VERY_SMALL	20	
SIZE_SMALL	40	
SIZE_MEDIUM	60	
SIZE_LARGE	80	
SIZE_VERY_LARGE	100	
FUELS	1	
OILS	2	
LUBRICANTS	3	
FOOD	4	
SPARE_PARTS	5	
PERSONNEL	6	
DIS_PROPELLANT	7	
DIS_CONSTRUCTION	8	

DIS_AMMO	9	
DIS_PERSONAL_DEMAND	9	
DIS_MAJOR_END_ITEMS	10	
DIS_MEDICAL_MATERIAL	11	
GASOLINE	1	
DIESEL_FUEL	2	
JP4_FULE	3	
FUEL_OIL	4	
DIS_BAGGED_PROPELLANT	1	
DIS_CANNISTERED_PROPELLANT	2	
DIS_AIR_DEFENSE_AMMO	1	
DIS_AIR_FORCE_AMMO	2	
DIS_ARMY_AVIATION_AMMO	3	
DIS_ARTILLERY_AMMO	4	
DIS_CHEMICAL_AMMO	5	
DIS_MINES_AND_EXPLOSIVES	6	
DIS_NUCLEAR_AMMO	7	
DIS_ROCKET_ARTILLERY_AMMO	8	
DIS_SMALL_ARMS_AMMO	9	
DIS_TANK_AMMO	10	
DIS_COUNTRY_OTHER	0	
DIS_COUNTRY_FRANCE	55	
DIS_COUNTRY_GERMANY	57	
DIS_COUNTRY_USSR	164	
DIS_COUNTRY_UK	166	
DIS_COUNTRY_USA	168	
GUISE_SIZE	8	
SUPPLY_SIZE	12	
DRM_OTHER	0	
DRM_STATIC	1	
DRM_FPW	2	
DRM_RPW	3	
DRM_RVW	4	
DRM_FVW	5	
DRM_FPB	6	
DRM_RPB	7	
DRM_RVB	8	
DRM_FVB	9	
UNUSED_DR_BYTES	15	
ARTICULATION_SIZE	16	
NO_ART_CHANGE	0	
DIRECTLY_ATTACHED	0	
INDIRECTLY_ATTACHED	1	
PARAM_TYPE_EMPTY	0	
PARAM_TYPE_POSITION	1	
PARAM_TYPE_POSITION_RATE	2	
PARAM_TYPE_EXTENSION	3	
PARAM_TYPE_EXTENSION_RATE	4	
PARAM_TYPE_LOCATION_X	5	

PARAM_TYPE_LOCATION_X_RATE	6	
PARAM_TYPE_LOCATION_Y	7	
PARAM_TYPE_LOCATION_Y_RATE	8	
PARAM_TYPE_LOCATION_Z	9	
PARAM_TYPE_LOCATION_Z_RATE	10	
PARAM_TYPE_AZIMUTH	11	
PARAM_TYPE_AZIMUTH_RATE	12	
PARAM_TYPE_ELEVATION	13	
PARAM_TYPE_ELEVATION_RATE	14	
PARAM_TYPE_ROTATION	15	
PARAM_TYPE_ROTATION_RATE	16	
PARAM_TYPE_ACCELERATION_X	17	
PARAM_TYPE_ACCELERATION_Y	18	
PARAM_TYPE_ACCELERATION_Z	19	
NUM_PARAM_TYPES	20	
STATION_RUDDER	1000	
STATION_LEFT_WING	1020	
STATION_RIGHT_WING	1040	
STATION_LEFT_AILERON	1060	
STATION_RIGHT_AILERON	1080	
STATION_PERISCOPE	2000	
STATION_ANTENNA	2020	
STATION_SNORKEL	2040	
STATION_LANDING_GEAR	3000	
STATION_TAIL_HOOK	3020	
STATION_SPEED_BRAKE	3040	
STATION_LEFT_WEAPON_BAY_DOOR	3060	
STATION_RIGHT_WEAPON_BAY_DOOR	3080	
STATION_HATCH	3100	
STATION_WINGSWEEP	3120	
STATION_PRIMARY_TURRET	4000	
STATION_PRIMARY_GUN	4200	
STATION_PRIMARY_LAUNCHER	4400	
STATION_PRIMARY_DEFENSE	4600	
STATION_PRIMARY_RADAR	4800	
STATION_SECONDARY_TURRET	5000	
STATION_SECONDARY_GUN	5200	
STATION_SECONDARY_LAUNCHER	5400	
STATION_SECONDARY_DEFENSE	5600	
STATION_SECONDARY_RADAR	5800	
MAX_ENTITY_MARKING_LENGTH	11	
CHARACTER_SET_UNUSED	0	
CHARACTER_SET_ASCII	1	
MAX_DBASE_ID_LENGTH	11	
WARHD_OTHER	0	
WARHD_HE	1000	
WARHD_HE_PLASTIC	1100	
WARHD_HE_INCENDIARY	1200	
WARHD_HE_FRAGMENTAION	1300	

WARHD_HE_ANTI_TANK	1400	
WARHD_HE_BOMBLETS	1500	
WARHD_HE_SHAPED_CHARGE	1600	
WARHD_SMOKE	2000	
WARHD_ILLUMINATION	3000	
WARHD_PRACTICE	4000	
WARHD_KINETIC	5000	
WARHD_UNUSED	6000	
WARHD_NUCLEAR	7000	
WARHD_CHEMICAL	8000	
WARHD_CHEMICAL_BLISTER	8100	
WARHD_CHEMICAL_BLOOD	8200	
WARHD_CHEMICAL_NERVE	8300	
WARHD_BIOLOGICAL	9000	
FUZE_OTHER	0	
FUZE_CONTACT	1000	
FUZE_CONTACT_INSTANT	1100	
FUZE_CONTACT_DELAYED	1200	
FUZE_TIMED	2000	
FUZE_PROXIMITY	3000	
FUZE_COMMAND	4000	
FUZE_ALTITUDE	5000	
FUZE_DEPTH	6000	
FUZE_ACOUSTIC	7000	
OTHER_FORCES	0	
BLUE_FORCES	1	
RED_FORCES	2	
WHITE_FORCES	3	
OTHER_SERVICE	0	
ARMY_SERVICE	1	
AIR_FORCE_SERVICE	2	
COAST_GUARD_SERVICE	3	
MARINES_SERVICE	4	
NAVY_SERVICE	5	
RADAR_SYSTEM_SIZE	4	
category_shift	4	
RESERVED_CATEGORY	1	
AIR_BASED_FIRE_CONTROL	1	
AIR_BASED_SEARCH	1	
GROUND_BASED_FIRE_CONTROL	1	
GROUND_BASED_SEARCH	0	
SEA_BASED_FIRE_CONTROL	0	
SEA_BASED_SEARCH	0	
RADAR_MODE_OFF	0	
SEARCH	1	
DOPLER_HPRF	2	
DOPPLER_MPRF	3	
DOPPLER_LPRF	4	
MONOPULSE	5	

ACQUISITION	6	
TRACKING	7	
TRACK_WHILE_SCAN	8	
TERRAIN_FOLLOW	9	
DATA_LINK	10	
MAX_ILLUMINED	10	
RELATIVE_TIME	0	
ABSOLUTE_TIME	1	

5.1.15 DMC_PDU.h

This header file contains definitions of the PDUs which define an extension of the DIS 1.0 protocol, called the Digital Message Communication.

5.1.15.1.1 DIS_types structure DMC_CommonBlock_type

Table 81 DIS_types structure DMC_CommonBlock_type

Item_Name	Type	bytes	Description
SenderID	Struct		ID_type
RadioID	Integer	2	
ExerciseTime	Integer	4	
Unused_15	Char array	15	SIM_UNUSED_ARRAY_SIZE
TimesForwarded	Char	1	
SenderTerminal	Char	1	
SnederPerson	Char	1	
Altitude	Integer	4	
TargetID	Struct		ID_type
TargetTerminal	Char	1	
TargetPerson	Char	1	
SenderCEOI	Char array	8	based on CEOI_LENG
LastFwdCEOI	Char array	8	based on CEOI_LENG
TargetCEOI	Char array	64	based on CEOI_LENG and MAX_CEOI_RCVR
TransmitDTG	Struct		DTG_type
TransmitFrequency	Double	8	
TransmitLocation	Struct		UTM_type
FreeText	Char array	64	based on ANNOTATION
ZuluTime	Integer	2	
SerialNumber	Integer	2	
Priority	Char	1	
Overlay	Char	1	
Subtype	Char	1	
Variation	Char	1	

5.1.15.1.2 DIS_types structure Header_and_CommonBlock_type

Table 82 DIS_types structure Header_and_commonBlock_type

Item Name	Type	bytes	Description
version	Char	1	
exercise	Char	1	
type	Char	1	
unused	Char	1	
SenderID	Struct		ID_type
RadioID	Integer	2	
TimeStamp	Struct		TimeStamp_type
EncodingScheme	Struct		EncodingScheme_type
Padding	Integer	2	
SampleRate	Integer	4	
Length	Integer	2	
Samples	Integer	2	
ProtocolID	Integer	4	
Unused_15	Char array	1	
TimesForwarded	Char	1	
SenderTerminal	Char	1	
SenderPerson	Char	1	
Altitude	Integer	4	
TargetID	Struct		ID_type
TargetTerminal	Char	1	
TargetPerson	Char	1	
SenderCEOI	Char array	8	based on CEOI_LENG
LastFwdCEOI	Char array	8	based on CEOI_LENG
TargetCEOI	Char array	64	based on CEOI_LENG and MAX_CEOI_RCVR
TransmitDTG	Struct		DTG_type
TransmitFrequency	Double	8	
TransmitLocation	Struct		UTM_type
FreeText	Char array	64	based on ANNOTATION
ZuluTime	Integer	2	
SerialNumber	Integer	2	
Priority	Char	1	
Overlay	Char	1	
Subtype	Char	1	
Variation	Char	1	

5.1.15.1.3 DIS_types **structure** ACK_Specific_type

Table 83 DIS_types **structure** ACK_Specific_type

Item Name	Type	bytes	Description
OriginalSenderID	Struct		ID_type
OriginalSenderTerminal	Char	1	
OriginalSenderPerson	Char	1	
AcknowledgeID	Struct		ID_type
AcknowledgeTerminal	Char	1	
AcknowledgePerson	Char	1	
OriginalSenderCEOI	Char array	8	based on CEOI_LENG
AcknowledgeCEOI	Char array	8	based on CEOI_LENG
OriginalDTG	Struct		DTG_type

5.1.15.1.4 DIS_types **structure** Ack_type

Table 84 DIS_types **structure** ACK_type

Item Name	Type	bytes	Description
Header	Struct		DMC_Header_type
Common	Struct		DMC_CommonBlock_type
Specific	Struct		Ack_Specific_type
AcknowledgeID	Struct		ID_type

5.1.15.1.5 DIS_types **structure** DISAck_type

Table 85 DIS_types **structure** DISAck_type

Item Name	Type	bytes	Description
Header_And_Common	Struct		Header_and_CommonBlock_type
Specific	Struct		Ack_Specific_type

5.1.15.1.6 DIS_types **structure** Spot_Specific_type

Table 86 DIS_types **structure** Spot_Specific_type

Item Name	Type	bytes	Description
TimeSighted	Struct		DTG_type
ObserverLocation	Struct		UTM_type
TargetQuanity	Integer	2	
ObserverIntentions	Char	1	
TargetType	Char	1	
TargetActivity	Char	1	
TargetDirection	Char	1	
TargetSpeed	Struct		Speed_type
TargetLocation	Struct		UTM_type
TargetUnit	Char array	16	based on UNIT_LENG
Unused_16	Char array	16	

5.1.15.1.7 DIS_types **structure** Spot_type

Table 87 DIS_types **structure** Spot_type

Item Name	Type	bytes	Description
Header	Struct		DMC_Header_type
Common	Struct		DMC_CommonBlock_type
Specific	Struct		Spot_Specific

5.1.15.1.8 DIS_types **structure** DISSpot_type

Table 88 DIS_types **structure** DISSpot_type

Item Name	Type	bytes	Description
Header_And_Common	Struct		Header_and_CommonBlock_type
Specific	Struct		Spot_Specific_type

5.1.15.1.9 DIS_types structure BDA_Specific_type

Table 89 DIS_types structure BDA_Specific_type

Item Name	Type	bytes	Description
StrikeTimeStart	Struct		DTG_type
StrikeTimeEnd	Struct		DTG_type
TargetCategory	Char	1	
TargetType	Char	1	
TargetsDestroyed	Integer	2	
PercentCoverage	Char	1	
Unused_3	Char array	3	

5.1.15.1.10 DIS_types structure BDA_type

Table 90 DIS_types structure BDA_type

Item Name	Type	bytes	Description
Header	Struct		DMC_Header_type
Common	Struct		DMC_CommonBlock_type
Specific	Sturct		BDA_Specific_type

5.1.15.1.11 DIS_types structure DISBDA_type

Table 91 DIS_types structure DISBDA_type

Item Name	Type	bytes	Description
Header_and_Common	Struct		Header_and_CommonBlock_type
Specific	Struct		BDA_Specific

5.1.15.1.12 DIS_types structure GndRoute

Table 92 DIS_types structure GndRoute

Item Name	Type	bytes	Description
EnemyActivity	Char	1	
ClassificationFormula	Char	1	
Unused_6	Char array	6	
ClassFormInfo	Char array	32	based on RTE_ID_LENG
RouteID	Char array	32	based on RTE_ID_LENG
Leftover	Char array	8	

5.1.15.1.13 DIS_types structure AirRoute

Table 93 DIS_types structure AirRoute

Item Name	Type	bytes	Description
EnemyActivity	Char	1	
Obstacles	Char	1	
Unused_6	Char array	6	
ClassFormInfo	Char array	32	based on RTE_ID_LENG
RouteID	Char array	32	based on RTE_ID_LENG
Leftover	Char array	8	

5.1.15.1.14 DIS_types structure Bridge

Table 94 DIS_types structure Bridge

Item Name	Type	bytes	Description
Type	Char	1	
Damage	Char	1	
Spans	Char	1	
ConstrMaterial	Char	1	
Unused_4	Char array	4	
Length	Char array	4	based on KEYPAD_4
Width	Char array	4	based on KEYPAD_4
Height	Char array	4	based on KEYPAD_4
Under	Char array	4	based on KEYPAD_4
ConstrDescr	Char array	16	based on KEYPAD_16
ID	Char array	32	based on RTE_ID_LENG
SpanLength	Char array	4	based on KEYPAD_4
LoadClass	Char array	4	based on KEYPAD_4

5.1.15.1.15 DIS_types structure LZ_PZ

Table 95 DIS_types structure LZ_PZ

Item Name	Type	bytes	Description
ActivityLikely	Char	1	
Obstacles	Char	1	
Unused_6	Char array	6	
ObstaclesDescr	Char array	16	based on KEYPAD_16
ID	Char array	16	based on KEYPAD_16
LZ_Size	Char array	16	based on KEYPAD_16
Axis	Char array	16	based on KEYPAD_16
Leftover	Char array	8	

5.1.15.1.16 DIS_types structure BP_OP

Table 96 DIS_types structure BP_OP

Item Name	Type	bytes	Description
EnemyActivity	Char	1	
Obstacles	Char	1	
Unused_6	Char array	6	
ObstaclesDescr	Char array	16	based on KEYPAD_16
ID	Char array	16	based on KEYPAD_16
BP_Size	Char array	16	based on KEYPAD_16
Axis	Char array	16	based on KEYPAD_16
Leftover	Char array	8	

5.1.15.1.17 DIS_types structure Crossing

Table 97 DIS_types structure Crossing

Item Name	Type	bytes	Description
BankSlopeEntry	Char array	4	based on KEYPAD_4
BankSlopeExit	Char array	4	based on KEYPAD_4
CrossingLength	Char array	4	based on KEYPAD_4
CrossingWidth	Char array	4	based on KEYPAD_4
CrossingDepth	Char array	4	based on KEYPAD_4
CurrentFlow	Char array	4	based on KEYPAD_4
ID	Char array	8	based on KEYPAD_8
Leftover	Char array	48	

5.1.15.1.18 DIS_types structure Recon_Specific_type

Table 98 DIS_types structure Recon_Specific_type

Item Name	Type	bytes	Description
Subtype	Union		GndRoute, AirRoute, Bridge, LZ_PZ, BP_OP, Crossing

5.1.15.1.19 DIS_types structure Recon_type

Table 99 DIS_types structure Recon_type

Item Name	Type	bytes	Description
Header	Struct		DMC_Header_type
Common	Struct		DMC_CommonBlock_type
Specific	Struct		Recon_Specific_type

5.1.15.1.20 DIS_types structure DISRecon_type

Table 100 DIS_types structure DisRecon_type

Item Name	Type	bytes	Description
Header_And_Common	Struct		Header_And_CommonBlock_type
Specific	Struct		Recon_Specific_type

5.1.15.1.21 DIS_types structure Repeat

Table 101 DIS_types structure Repeat

Item Name	Type	bytes	Description
MissionStatus	Char	1	
Unused_7	Char array	7	
Leftover	Char array	16	

5.1.15.1.22 DIS_types structure Cancel

Table 102 DIS_types structure Cancel

Item Name	Type	bytes	Description
MissionStatus	Char	1	
Unused_7	Char array	7	
Leftover	Char array	16	

5.1.15.1.23 DIS_types structure CheckFire

Table 103 DIS_types structure CheckFire

Item Name	Type	bytes	Description
MissionStatus	Char	1	
Unused_7	Char array	7	
Leftover	Char array	16	

5.1.15.1.24 DIS_types structure CNO

Table 104 DIS_types structure CNO

Item Name	Type	bytes	Description
MissionStatus	Char	1	
Unused_7	Char array	7	
Leftover	Char array	16	

5.1.15.1.25 DIS_types structure Shift

Table 105 DIS_types structure Shift

Item Name	Type	bytes	Description
MissionStatus	Char	1	
FireForEffect	Char	1	
Unused_6	Char array	6	
Message	Char array	16	based on SHIFT_MSG

5.1.15.1.26 DIS_types structure NewMission

Table 106 DIS_types structure NewMission

Item Name	Type	bytes	Description
MissionStatus	Char	1	
MissionType	Char	1	
Shell	Char	1	
Control	Char	1	
Fuze	Char	1	
Trajectory	Char	1	
RoundFFE	Char	1	
Unused	Char	1	
Leftover	Char array	16	

5.1.15.1.27 DIS_types structure MTQ

Table 107 DIS_types structure MTQ

Item Name	Type	bytes	Description
MissionStatus	Char	1	
RequestAdjustment	Char	1	
EnterTarget	Char	1	
EndMission	Char	1	
Unused_4	Char array	4	
Leftover	Char array	16	

5.1.15.1.28 DIS_types structure Shot

Table 108 DIS_types structure Shot

Item Name	Type	bytes	Description
MissionStatus	Char	1	
RoundsFired	Char	1	
ImpactTime	Char	1	
Unused_5	Char array	5	
Leftover	Char array	16	

5.1.15.1.29 DIS_types structure Splash

Table 109 DIS_types structure Splash

Item Name	Type	bytes	Description
MissionStatus	Char	1	
RoundsFired	Char	1	
ImpactTime	Char	1	
Unused_5	Char array	5	
Leftover	Char array	16	

5.1.15.1.30 DIS_types structure EndOfMission

Table 110 DIS_types structure EndOfMission

Item Name	Type	bytes	Description
MissionStatus	Char	1	
Dispositon	Char	1	
Casualties	Char	1	
RecordTarget	Char	1	
CasualtyNumber	Integer	2	
Unused_2	Char array	2	
PointNumber	Char array	16	based on PT_NUM

5.1.15.1.31 DIS_types structure Artillery_Specific_type

Table 111 DIS_types structure Artillery_Specific_type

Item Name	Type	bytes	Description
MissionID	Char array	16	based on ID_NUM_LENG
TargetID	Char array	16	based on ID_NUM_LENG
Subtype	Union	24	one of the previous structures 5.1.15.1.21 through 5.1.15.1.30

5.1.15.1.32 DIS_types structure Artillery_type

Table 112 DIS_types structure Artillery_type

Item Name	Type	bytes	Description
Header	Struct		DMC_Header_type
Common	Struct		DMC_CommonBlock_type
Specific	Struct		Artillery_Specific_type

5.1.15.1.33 DIS_types structure DISArtillery_type

Table 113 DIS_types structure DISArtillery_type

Item Name	Type	bytes	Description
Header_And_Common	Struct		Header_And_CommonBlock_type
Specific	Struct		Artillery_Specific_type

5.1.15.1.34 DIS_types structure Move_Specific_type

Table 114 DIS_types structure Move_Specific_type

Item Name	Type	bytes	Description
TargetID	Char array	16	based on ID_NUM_LENG
Task	Char	1	
Who	Char	1	
When	Char	1	
Unused_3	Char array	3	
ZuluTime	Integer	2	
Where	Char array	8	based on WHERE_LENG
DTG	Struct		DTG_type

5.1.15.1.35 DIS_types structure Move_type

Table 115 DIS_types structure Move_type

Item Name	Type	bytes	Description
Header	Struct		DMC_Header_type
Common	Struct		DMC_CommonBlock_type
Specific	Struct		Move_Specific_type

5.1.15.1.36 DIS_types structure DISMove_type

Table 116 DIS_types structure DISMove_type

Item Name	Type	bytes	Description
Header_And_Common	Struct		Header_And_CommonBlock_type
Specific	Struct		Move_Specific_type

5.1.15.1.37 DIS_types structure Status_Specific_type

Table 117 DIS_types structure Status_Specific_type

Item Name	Type	bytes	Description
Fuel	Integer	2	
Elements	Char	1	
FailedEquip	Char array	3	based on FAILED_EQUIP
Hellfires	Char	1	
Stingers	Char	1	
Rockets	Char	1	
Rounds	Char	1	
RequestType	Char	1	
Unused_5	Char array	5	

5.1.15.1.38 DIS_types structure Status_type

Table 118 DIS_types structure Status_type

Item Name	Type	bytes	Description
Header	Struct		DMC_Header_type
Common	Struct		DMC_CommonBlock_type
Specific	Struct		Status_Specific_type

5.1.15.1.39 DIS_types structure DISStatus_type

Table 119 DIS_types structure DISStatus_type

Item Name	Type	bytes	Description
Header_And_Common	Struct		Header_And_CommonBlock_type
Specific	Struct		Status_Specific_type

5.1.15.1.40 DIS_types structure Request_Specific_type

Table 120 DIS_types structure Request_Specific_type

Item Name	Type	bytes	Description
ReportType	Char	1	
ReconType	Char	1	
Unused_6	Char array	6	

5.1.15.1.41 DIS_types structure Request_type

Table 121 DIS_types structure Request_type

Item Name	Type	bytes	Description
Header	Struct		DMC_Header_type
Common	Struct		DMC_CommonBlock_type
Specific	Struct		Request_Specific_type

5.1.15.1.42 DIS_types structure DISRequest_type

Table 122 DIS_types structure DISRequest_type

Item Name	Type	bytes	Description
Header_And_Common	Struct		Header_And_CommonBlock_type
Specific	Struct		Request_Specific_type

5.1.15.1.43 DIS_types structure NBC_1

Table 123 DIS_types structure DMC_CommonBlock_type

Item Name	Type	bytes	Description
Description	Char	1	
BurstType	Char	1	
DeliveredBy	Char	1	
CloudHtUnits	Char	1	
CloudWidth	Float	4	
CloudDescr	Char array	16	based on CLOUD_DESCR
CloudHeight	Integer	2	
FlashBangtime	Integer	2	
Unused_4	Char array	4	
StartDTG	Struct	16	DTG_type
EndDTG	Struct	16	DTG_type

5.1.15.1.44 DIS_types structure NBC_4

Table 124 DIS_types structure NBC_4

Item Name	Type	bytes	Description
Description	Char	1	
BurstType	Char	1	
DeliveredBy	Char	1	
CloudHtUnits	Char	1	
CloudWidth	Float	4	
CloudDescr	Char array	16	based on CLOUD_DESCR
CloudHeight	Integer	2	
DoseRate	Integer	2	
Unused_4	Char array	4	
StartDTG	Struct	16	DTG_type
EndDTG	Struct	16	DTG_type

5.1.15.1.45 DIS_types structure NBC_Negative

Table 125 DIS_types structure DMC_CommonBlock_type

Item Name	Type	bytes	Description
Description	Char	1	
BurstType	Char	1	
DeliveredBy	Char	1	
CloudHtUnits	Char	1	
CloudWidth	Float	4	
CloudDescr	Char array	16	based on CLOUD_DESCR
CloudHeight	Integer	2	
Leftover	Char array	38	

5.1.15.1.46 DIS_types structure NBC_Specific_type

Table 126 DIS_types structure NBC_Specific_type

Item Name	Type	bytes	Description
Subtype	Union	64	one of the previous structures 5.1.15.1.43 through 5.1.15.1.45

5.1.15.1.47 DIS_types structure NBC_type

Table 127 DIS_types structure NBC_type

Item Name	Type	bytes	Description
Header	Struct		DMC_Header_type
Common	Struct		DMC_CommonBlock_type
Specific	Struct		NBC_Specific_type

5.1.15.1.48 DIS_types structure DISNBC_type

Table 128 DIS_types structure DISNBC_type

Item Name	Type	bytes	Description
Header_And_Common	Struct		Header_And_CommonBlock_type
Specific	Struct		NBC_Specific_type

5.1.15.1.49 DIS_types structure MIJI_Specific_type

Table 129 DIS_types structure MIJI_Specific_type

Item Name	Type	bytes	Description
AffectedRadioFreq	Double	8	
ProgrammedFreq	Double array	32	
KeypadEntry	Char array	16	based on KEYPAD_16
JamStartDTG	Struct	16	DTG_type
JamEndDTG	Struct	16	DTG_type
Description	Char	1	
PercentLost	Char	1	
Type	Char	1	
Unused_5	Char array	5	

5.1.15.1.50 DIS_types structure MIJI_type

Table 130 DIS_types structure MIJI_type

Item Name	Type	bytes	Description
Header	Struct		DMC_Header_type
Common	Struct		DMC_CommonBlock_type
Specific	Struct	96	MIJI_Specific_type

5.1.15.1.51 DIS_types structure DISMIJI_type

Table 131 DIS_types structure DISMIJI_type

Item Name	Type	bytes	Description
Header_And_Common	Struct		Header_And_CommonBlock_type
Specific	Struct	96	MIJI_Specific_type

5.1.15.1.52 DIS_types structure PIREP_Specific_type

Table 132 DIS_types structure PIREP_Specific_type

Item Name	Type	bytes	Description
Visibility	Char	1	
Restrictions	Char	1	
CloudCover	Char	1	
TurbIntensity	Char	1	
TurbFrequency	Char	1	
IcingIntensity	Char	1	
IcingType	Char	1	
WindDirection	Char	1	
WindSpeed	Integer	2	
OutsideAirTemp	Integer	2	
Barometer	Float	4	
CloudBase	Integer	4	
CloudTop	Integer	4	

5.1.15.1.53 DIS_types structure PIREP_type

Table 133 DIS_types structure PIREP_type

Item Name	Type	bytes	Description
Header	Struct		DMC_Header_type
Common	Struct		DMC_CommonBlock_type
Specific	Struct	24	PIREP_Specific_type

5.1.15.1.54 DIS_types structure DISPIREP_type

Table 134 DIS_types structure DISPIREP_type

Item Name	Type	bytes	Description
Header_And_Common	Struct		Header_And_CommonBlock_type
Specific	Struct	24	PIREP_Specific_type

5.1.15.1.55 DIS_types structure DNAV_Specific_type

Table 135 DIS_types structure DNAV_Specific_type

Item Name	Type	bytes	Description
AircraftType	Char	1	
AircraftStatus	Char	1	
PilotStatus	Char	1	
Unused_5	Char array	5	

5.1.15.1.56 DIS_types structure DNAV_type

Table 136 DIS_types structure DNAV_type

Item Name	Type	bytes	Description
Header	Struct		DMC_Header_type
Common	Struct		DMC_CommonBlock_type
Specific	Struct	8	DNAV_Specific_type

5.1.15.1.57 DIS_types structure DISDNAV_type

Table 137 DIS_types structure DISDNAV_type

Item Name	Type	bytes	Description
Header_And_Common	Struct		Header_And_CommonBlock_type
Specific	Struct	8	DNAV_Specific_type

5.1.15.1.58 DIS_types structure FreeText_Specific_type

Table 138 DIS_types structure FreeText_Specific_type

Item Name	Type	bytes	Description
FreeText	Char array	256	based on FREE_TEXT_LENG

5.1.15.1.59 DIS_types structure FreeText_type

Table 139 DIS_types structure FreeText_type

Item Name	Type	bytes	Description
Header	Struct		DMC_Header_type
Common	Struct		DMC_CommonBlock_type
Specific	Struct	256	FreeText_Specific_type

5.1.15.1.60 DIS_types structure DISFreeText_type

Table 140 DIS_types structure DISFreeText_type

Item Name	Type	bytes	Description
Header_And_Common	Struct		Header_And_CommonBlock_type
Specific	Struct	256	FreeText_Specific_type

5.1.15.1.61 DIS_types structure DMCPDU_type

Table 141 DIS_types structure DMCPDU_type

Item Name	Type	bytes	Description
Variant	Union		one of the following structures :
Ack			Ack_type
Spot			Spot_type
BDA			BDA_type
Recon			Recon_type
Artillery			Artillery_type
Move			Move_type
Status			Status_type
Request			Request_type
NBC			NBC_type
MIJI			MIJI_type
PIREP			PIREP_type
DNAV			DNAV_type
FreeText			FreeText_type

5.1.15.1.62 DIS_types structure DMCDISPDU_type

Table 142 DIS_types structure DMCDISPDU_type

Item Name	Type	bytes	Description
Variant	Union		one of the following structures :
DISAck			DISAck_type
DISSpot			DISSpot_type
DISBDA			DISBDA_type
DISRecon			DISRecon_type
DISArtillery			DISArtillery_type
DISMove			DISMove_type
DISStatus			DISStatus_type
DISRequest			DISRequest_type
DISNBC			DISNBC_type
DISDISMIJI			DISMIJI_type
DISPIREP			DISPIREP_type
DISDNAV			DISDNAV_type
DISFreeText			DISFreeText_type

5.1.15.2 DMC_PDU define statements

Table 143 DIS_types define statements

Name	Value	Description
KEYPAD_4	4	
KEYPAD_8	8	
KEYPAD_16	16	
KEYPAD_32	32	
ID_NUM_LENG	16	
RET_ID_LENG	32	
WHERE_LENG	8	
BITS_IN_BYTE	8	
SIGNAL_DATA_COMMON_BYTES	212	
DIS_HEADER_SIZE	4	
DMC_PDU	0xDC	
DMCDIS_PDU	ox1A	
DMC_OTHER_PDU	0	
DMC_ACKPPDU	1	
DMC_SPOT_PDU	2	
DMC_BAD_PDU	3	
DMC_RECON_PDU	4	
DMC_ARTY_PDU	5	
DMC_MOVE_PDU	6	
DMC_STATUS_PDU	7	
DMC_REQUEST_PDU	8	

DMC_NBC_PDU	9	
DMC_MIJI_PDU	10	
DMC_PIREP_PDU	11	
DMC_DNAV_PDU	12	
DMC_FREE_TEXT_PDU	13	
DMC_ATHS_PDU	14	
NUMBER_DMC_PDUS	15	
DEMC_HDR	8	
DMC_HDR_SPARE	4	
DMC_COMMON	232	
DSIDMC_HEADER_COMMON	240	
CEOI_LENG	8	
MAX_CEOI_RCVR	8	
ANNOTATION	64	
SIM_UNUSED_ARRAY_SIZE	13	
UNIT_LENG	16	
DMC_RECON_GND_ROUTE_PDU	0	
DMC_RECON_AIR_ROUTE_PDU	1	
DMC_RECON_BRIDGE_PDU	2	
DMC_RECON_LZ_PZ_PDU	3	
DMC_RECON_BP_OP_PDU	4	
DMC_RECON_CROSSING_PDU	5	
NUM_DMC_RECON_PDUS	6	
DMC_ARTY_REPEAT_PDU	0	
DMC_ARTY_CANCEL_PDU	1	
DMC_ARTY_CHECK_PDU	2	
DMC_ARTY_CNO_PDU	3	
DMC_ARTY_SHIFT_PDU	4	
DMC_ARTY_NW_MSN_PDU	5	
DMC_ARTY_MTO_PDU	6	
DMC_ARTY_SHOT_PDU	7	
DMC_ARTY_SPLASH_PDU	8	
DMC_ARTY_EOM_PDU	9	
DMC_ARTY_NEGLECT_PDU	0xA	
NUM_DMC_ARTY_PDU	0xB	
SHIFT_MSG	16	
PT_NUM	16	
FAILED_EQUIP	3	
DMC_NBC_1_PDU	0	
DMC_NBC_2_PDU	1	
DMC_NBC_3_PDU	2	
DMC_NBC_4_PDU	3	
DMC_NBC_5_PDU	4	
NUM_DMC_NBC_PDUS	5	
CLOUD_DESCR	16	
PGM_FREQ	4	
FREE_TEXT_LENG	256	

5.1.16 DMC_types.h

This header file contains definitions of various basic data elements, including how these elements are represented as communicated bits. The elements compose the fields which compose each PDU.

5.1.16.1.1 DMC_types structure DTG_type

Table 144 DMC_types structure DTG_type

Item Name	Type	bytes	Description
Hundredth	Char array	2	based on TWO_DIGITS
Second	Char array	2	based on TWO_DIGITS
Minute	Char array	2	based on TWO_DIGITS
Hour	Char array	2	based on TWO_DIGITS
Day	Char array	2	based on TWO_DIGITS
Month	Char array	2	based on TWO_DIGITS
Year	Char array	4	based on FOUR_DIGITS

5.1.16.1.2 DMC_types structure EncodingScheme_type

Table 145 DMC_types structure EncodingScheme_type

Item Name	Type	bytes	Description
Class	Integer	2	2 bit field
Scheme			14 bit field

5.1.16.1.3 DMC_types structure UTM_type

Table 146 DMC_types structure UTM_type

Item Name	Type	bytes	Description
Sector	Char array	2	based on TWO_DIGITS
Easting	Char array	7	based on UTM_MAP_COORD
Northing	Char array	7	based on UTM_MAP_COORD

5.1.16.1.4 DMC_types structure Speed_type

Table 147 DMC_types structure Speed_type

Item Name	Type	bytes	Description
Mph	Integer	2	1 bit field
Value			15 bit field

5.1.16.2 DMC_types define statements

Table 148 DMC_types define statements

Name	Value	Description
TWO_DIGITS	2	
FOUR_DIGITS	4	
ENCODESCHEME_VOICE	0	
ENCODESCHEME_RAWDATA	1	
ENCODESCHEME_APPSPEC	2	
ENCODESCHEME_POINTER	3	
UTM_MAP_COORD	7	
SPEED_IN_KPH	0	
SPEED_IN_MPH	1	
DMC_RADIOID	1	
DMC_PROTOCOLID	0xDC	
TERMINAL_UNKNOWN	0	
TERMINAL_PILOT_STN	1	
TERMINAL_CPG_STN	2	
TERMINAL_TOC_DMC	3	
TERMINAL_FSE	4	
PERSON_UNKNOWN	0	
PERSON_PILOT	1	
PERSON_CPG	2	
PERSON_G2	3	
PERSON_G3	4	
PERSON_S1	5	
PERSON_S2	6	
PERSON_FO	7	
PERSON_MO	8	
PERSON_ANY	9	
PRIORITY_ROUTINE	0	
PRIORTY_URGENT	1	
ACTIVITY_NONE	0	
ACTIVITY_STATIONARY	1	
ACTIVITY_MOVING	2	
ACTIVITY_DUG_IN	3	

ACTIVITY_RETREATING	4	
ACTIVITY_ADVANCING	5	
ACTIVITY_ATTACKING	6	
ACTIVITY_DAMAGED	7	
ACTIVITY_KILLED	8	
ACTIVITY_MOBILE_KILL	9	
TARGET_UNKNOWN	0	
TARGET_ADA	1	
TARGET_SAM	2	
TARGET_TANK	3	
TARGET_WHLD	4	
TARGET_TRKD	5	
TARGET_ACFT	6	
TARGET_TRPS	7	
TARGET_UNCL	8	
DIRECTION_N	0	
DIRECTION_NE	1	
DIRECTION_E	2	
DIRECTION_SE	3	
DIRECTION_S	4	
DIRECTION_SW	5	
DIRECTION_W	6	
DIRECTION_NW	7	
OBS_INT_OTHER	0	
OBS_INT_CONT_MSN	1	
OBS_INT_ENGA	2	
OBS_INT_HLD	3	
OBS_INT_RTS_IHLD	4	
OBS_INT_OBSV_TGT	5	
CLASSFORM_NULL	0	
CLASSFORM_W	1	
CLASSFORM_TYPE	2	
CLASSFORM_MLC	3	
CLASSFORM_OHC	4	
CLASSFORM_OB	5	
CLASSFORM_BLOCK	6	
OBSTACLES_NONE	0	
OBSTACLES_KEYPAD	1	
OBSTACLES_ADA	2	
OBSTACLES_TOWR_ANT	3	
OBSTACLES_WIRES	4	
OBSTACLES_TERR	5	
OBSTACLES_TREES	6	
OBSTACLES_BLDG	7	
OBSTACLES_NDST_CAMP	8	
OBSTACLES_OTHER	9	
BRDGTYP_UNKNOWN	0	
BRDGTYP_TRUSS	1	
BRDGTYP_GIRDER	2	

BRDGTYP_BEAM	3	
BRDGTYP_SLAB	4	
BRDGTYP_CLOS	5	
BRDGTYP_ARCHOP	6	
BRDGTYP_SUSPEN	7	
BRDGTYP_FLOATING	8	
BRDGTYP_SWING	9	
BRDGDMG_NONE	0	
BRDGDMG_LIGHT	1	
BRDGDMG_MODERATE	2	
BRDGDMG_SEVERE	3	
BRDGDMG_DESTROYED	4	
BRDGCONMAT_UNKNOWN	0	
BRDGCONMAT_KEYPAD	1	
BRDGCONMAT_METAL	2	
BRDGCONMAT_STONE	3	
BRDGCONMAT_CONCRETE	4	
ACT_LIKELY_NONE	0	
ACT_LIKELY_EXPECTED	1	
ACT_LIKELY_POSSIBLE	2	
ACT_LIKELY_UNLIKELY	3	
MSNSTAT_NULL	0	
MSNSTAT_REQUESTED	1	
MSNSTAT_READY	2	
MSNSTAT_SHOT	3	
MSNSTAT_SPLASH	4	
MSNTYPE_NULL	0	
MSNTYPE_FFE	1	
MSNTYPE_ADJ_FIRE	2	
MSNTYPE_IMMED_SUPPR	3	
MSNTYPE_SUPPR	4	
SHELL_NULL	0	
SHELL_HE	1	
SHELL_ICM	2	
SHELL_CPRHD	3	
SHELL_SMOKE	4	
SHELL_WP	5	
SHELL_CORD_ILLUM	6	
CNTL_NULL	0	
CNTL_AMC	1	
CNTL_WR	2	
CNTL_TOT	3	
FUZE_NULL	0	
FUZE_QUICK	1	
FUZE_TIME_DELAY	2	
FUZE_CONCR_PRCNG	3	
TRAJ_LOW	0	
TRAJ_HIGH	1	
TRAJ_OTHER	2	

DSPN_NULL	0	
DSPN_NOT_GIVEN	1	
DSPN_BURNING	2	
DSPN_CNO	3	
CSLT_NULL	0	
CSLT_NOT_GIVEN	1	
CSLT_GIVEN	2	
TASK_NULL	0	
TASK_TO	1	
TASK_HOLD	2	
TASK_CONT_MSN	3	
TASK_RENEZ_AT	4	
TASK_ENGA_TGT	5	
TASK_MOVING_TO	6	
TASK_HLDG_AT	7	
TASK_ARRVG_AT	8	
TASK_PASSG_THRU	9	
TASK_DEPRTG_FROM	10	
MVWHO_NULL	0	
MVWHO_MYPOS	1	
MVWHO_B2	2	
MVWHO_B3	3	
MVWHO_OP1	4	
MVWHO_OP2	5	
MVWHEN_NULL	0	
MVWHEN_IMMED	1	
MVWHEN_WHNRDY	2	
MVWHEN_AMC	3	
MVWHEN_DTG	4	
BADEQPT_NULL	0	
BADEQPT_ENG1	1	
BADEQPT_ENG2	2	
BADEQPT_GUNS	3	
BADEQPT_RADIO	4	
BADEQPT_DSGNTR	5	
BADEQPT_RKTPOD	6	
BADEQPT_MSLSTN	7	
BADEQPT_RADAR	8	
BADEQPT_LASER	9	
REPTTYP_NONE	0	
REPTTYP_STATUS	1	
REPTTYP_SPOIT	2	
REPTTYP_RECON	3	
REPTTYP_MOVEMENT	4	
RECTYP_NONE	0	
RECTYP_GND_RTE	1	
RECTYP_AIR_RTE	2	
RECTYP_LZ_PZ	3	
RECTYP_BP_OP	4	

RECTYP_CROSSING	5	
BURST_NONE	0	
BURST_UNKNOWN	1	
BURST_SURFACE	2	
BURST_AIR	3	
BURST_GND	4	
DLVRY_UNKNOWN	0	
DLVRY_ARTY	1	
DLVRY_MORTAR	2	
DLVRY_ROCKET	3	
DLVRY_MISSLE	4	
DLVRY_BOMB	5	
CLDHT_UNUSED	0	
CLDHT_DEGREES	1	
CLDHT_HT_IN_FT	2	
CLDHT_HT_IN_M	3	
NBCDESC_UNUSUAL	0	
NBCDESC_INITIAL	1	
NBCDESC_INCREASING	2	
NBCDESC_DECREASING	3	
NBCDESC_PEAD	4	
NBCDESC_SPECIAL	5	
NBCDESC_SERIES	6	
NBCDESC_VERIFICATION	7	
NBCDESC_SUMMARY	8	
MIJIDESC_UNKNOWN	0	
MIJIDESC_INTERMIT	1	
MIJIDESC_CONT	2	
MIJIDESC_IMITATIVE	3	
MIJIDESC_WARBLING	4	
MIJIDESC_MUSIC	5	
MIJIDESC_KEYPAD	6	
MIJITYPE_UNKNOWN	0	
MIJITYPE_MEACONING	1	
MIJITYPE_INTRUSION	2	
MIJITYPE_JAMMING	3	
MIJITYPE_INTERFERE	4	
VISI_UNLIMITED	0	
VISI_HALF_MILE	1	
VISI_ONE_MILE	2	
VISI_TWO_MILE	3	
VISI_THREE_MILE	4	
VISI_FOUR_MILE	5	
VISI_FIVE_MILE	6	
VISI_RESTR_NONE	0	
VISI_RESTR_HAZE	1	
VISI_RESTR_RAIN	2	
VISI_RESTR_SNOW	3	
VISI_RESTR_FOG	4	

VISI_RESTR_SMOKE	5	
VISI_RESTR_DRIZZLE	6	
VISI_RESTR_DUST	7	
VISI_RESTR_BLIZZARD	8	
CLDCVR_NONE	0	
CLDCVR_OVERCAST	1	
CLDCVR_BROKEN	2	
CLDCVR_SCATTERED	3	
TRBINT_NONE	0	
TRBINT_LIGHT	1	
TRBINT_MODERATE	2	
TRBINT_SEVERE	3	
TRBINT_EXTREME	4	
TRBFREQ_NONE	0	
TRBFREQ_OCCASION	1	
TRBFREQ_INTERMIT	2	
TRBFREQ_CONT	3	
ICEINT_NONE	0	
ICEINT_TRACE	1	
ICEINT_LIGHT	2	
ICEINT_MODERATE	3	
ICEINT_SEVERE	4	
ICETYP_NULL	0	
ICETYP_RIME	1	
ICETYP_CLEAR	2	
ACRFT_TYP_UNKNOWN	0	
ACRFT_TYP_RAH66	1	
ACRFT_TYP_AH64	2	
ACRFT_TYP_OH58	3	
ACRFT_TYP_AH1	4	
ACRFT_TYP_UH1	5	
ACRFT_TYP_CH47	6	
ACRFT_TYP_OV1	7	
ACRFT_TYP_OH6	8	
ACRFT_TYP_CH54	9	
ACRFT_TYP_UH60	10	
ACRFT_TYP_RPV	11	
ACRFT_STAT_UNKNOWN	0	
ACRFT_STAT_RECOVER	1	
ACRFT_STAT_DAMAGED	2	
ACRFT_STAT_DESTROYED	3	
PILOT_STAT_UNKNOWN	0	
PILOT_STAT_GOOD	1	
PILOT_STAT_WIA	2	
PILOT_STAT_KIA	3	
PILOT_STAT_POW	4	

5.1.17 dms.h

This header file contains parameters for the Digital Message Server routines.

5.1.17.1.1 dms structure msgdests_t

Table 149 dms structure msgdests_t

Item Name	Type	bytes	Description
name	Char array	8	based on MAXDESTSIZE

5.1.17.1.2 dms structure dms_shmem_t

Table 150 dms structure dms_shmem_t

Item Name	Type	bytes	Description
nbr_readers	Integer	4	
dests	Array of Struct	64	msgdest_t
exercise_id	Char	1	
pdu_length	Integer	4	
pdu	Char array	1500	based on MAXPDUSIZE

5.1.17.1.3 dms structure dms_outgoingmsg_t

Table 151 dms structure dms_outgoingmsg_t

Item Name	Type	bytes	Description
msg_type	Integer	4	
dm_size	Integer	4	
dm	Char array	1500	based on MAXPDUSIZE

5.1.17.1.4 dms structure log

Table 152 dms structure log

Item Name	Type	bytes	Description
pid	Integer	4	
exercise_id	Char	1	
dests	Array of Struct	64	msgdests_t

5.1.17.1.5 dms structure dms_msg_t

Table 153 dms structure dms_msg_t

Item Name	Type	bytes	Description
msg_type	Integer	4	
msg	Union	69	either shmem_addr or log Struct - 5.1.17.1.4

5.1.17.2 dms define statements

Table 154 dms define statements

Name	Value	Description
MAXMSGDESTS	8	# of names in PDU req
MAXDESTSIZE	8	size of name/group
MAX CLIENTS	8	
MAXPDU	32	
MAXPDUSIZE	1500	
MSG_LOGIN	1	
MSG_LOGOUT	2	
MSG_DMAVAIL	3	
MSG_DONE	4	
MSG_NEWDESTS	5	
MSG_FREE	6	
MSG_CLEANUP	7	
DMC_MAXMSGTYPE	10	
DMS_ENET_RECEIVE	1	
DMS_ENET_TRANSMIT	2	
FAN_OUT_Q_KEY		((key_t) 54321
OUTGOING_Q_KEY		((key_t) 54322
INCOMING_SHMEM_KEY		((key_t) 54323
DMS_PERMS	0666	

5.1.18 fcn_proto.h

In C, function prototyping (the defining the input arguments and resultant output from the function) is required. The compiler will check the prototyping of a function against the actual arguments used within the source code of the calling function and flag as a compiler error any incongruities. This header file prototypes all of the general use functions used in the DMC.

5.1.19 global_netif.h

This header file defines the globally used parameters for use by the ethernet interface functions, identifies and dimensions external data entities and prototypes the DIS and SIM network interface functions. Table 5.1.19.1 structures is not applicable.

5.1.19.2 global_netif define statements

Table 155 global_netif define statements

Name	Value	Description
DSAP	0xaa	
SSAP	0xaa	
CNTL	0x03	
SIM_SOCKET_NAME1	soc_1	
SIM_PROTOCOL_ID	0x08000800	
SIM_PROTOCOL_ID_B1	0x08	
SIM_PROTOCOL_ID_B2	0x00	
SIM_PROTOCOL_ID_B3	0x08	
SIM_ETHER_TYPE_B1	0x52	
SIM_ETHER_TYPE_B2	0x08	
SIM_BROADCAST_TYPE	0xFF	
DIS_BROADCAST_ADDR	0x89F9200	
DIS_PROTOCOL_ID	0x00000800	
DIS_PROTOCOL_ID_B1	0x00	
DIS_PROTOCOL_ID_B2	0x00	
DIS_PROTOCOL_ID_B3	0x00	
DIS_ETHER_TYPE_B1	0x52	
DIS_ETHER_TYPE_B2	0x09	
HEADER	0	
DATA	1	
MAXIOV	2	
TEST_PDU_FN_SIZE	40	
SITE_SIMULATOR_BYTES	4	
ETHER_TYPE_POS	6	
PROTOCOL_ID_POS	3	
READONLY	0	
SYS_ETHER_ADDR	"ETHER"	
SYS_IP_ADDR	"INET"	
SYS_IP_BROADCAST_ADDR	"BROADCAST"	

5.1.20 global_vars.h

This header file defines the globally used parameters for use by the DMC functions, identifies and dimensions external data entities and prototypes the PDU decode functions. Table 5.1.20.1 structures is not applicable.

5.1.20.2 global_vars define statements

Table 156 global_vars define statements

Name	Value	Description
TRUE	1	
FALSE	0	
UNUSED	0	
SIGREC	SIGUSR1	
SIGXMT	SIGUSR2	
USEC_PER_MS	1000	
MSEC_PER_SEC	1000	
MSEC_PER_HOUR	3600000	
USEC_PER_SEC	1000000.0	USEC_PER_MS*1000
SEC_PER_HOUR	3600	
USEC_PER_HOUR	3600000000	USEC_PER_SEC*SEC_PER_HOUR
DIS_TIME_UNITS	(double)0x80000000	= 0.5965
BAMS_PER_DEGREE	(float)0x80000000/ (float)180	

5.1.21 netif.h

This header file defines the globally used parameters for use by the network interface functions in sizing arrays and pointing to particular bytes within a PDU.

5.1.21.1.1 netif structure Int_Array

Table 157 dms structure Int_Array

Item Name	Type	bytes	Description
ethernet_address	Integer array	24	based on ENETADDRSZ

5.1.21.2 netif define statements

Table 158 netif define statements

Name	Value	Description
MAX_IFNAME_SIZE	10	
ETHER_HDR_SIZE	14	
DISADDRSZ	4	
ENETADDRSZ	6	
MAX_LINE_WIDTH	132	
ETHERTYPE_SIM	0x5208	
ETHERTYPE_DIS	0x5209	
PROMISCUOUS_MODE	0x01	

5.2 DMCC Include Files

The following table maps the DMCC CSCI include filesfor interprocess communication, the PDU Builder, and the digital message server, and CSC's and CSU's that use them:

Table 159 DMCC CSC/CSU Data File Map

CSC	CSU's	INCLUDE FILENAME	COMMENTS
dms	All	dms.h	Contains parameters and data structures for inter process communication
build pdu	All	dms.h	Contains parameters and data structures for inter process communication
ipc	All	dms.h	Contains parameters and data structures for inter process communication

6. CSCI data files.

6.1. The DMCC CSCI uses one external data file which contains an integer. This integer expresses the difference in minutes between Local Time and Zulu Time (also known as Greenwich Mean Time, or GMT).

Name:
"/tmp/dmcc_timezone_offset"

Format:
Contains a single integer value

Location:
Maintained in temporary files directory directory: /tmp.

Description:
The dmcc_timezone_offset file is a block oriented file containing a single integer. The contents of the file are ascii readable. The value represents the number of minutes that local time is ahead of Zulu time (that is, the number of minutes to be subtracted from local time to obtain Zulu). If the value of this integer is positive, then local time is ahead of Zulu time; if the value is negative, then local time is "behind" Zulu time.

GMT = ZULU.

7. Requirements traceability.

Requriements traceablility are covered under Section 4, Preliminary Design.

8. Notes.

GLOSSARY and ACRONYMS

DMCC:	Digital Message Communications Console
DMC:	Digital Message Communications
DMCS:	Digital Message Communications System
CEOI:	Communications Electronic Operational Identifier (Call Sign)
Sys Main:	System Main
SMD:	Situation Management Display
TMI:	Tactile Message Indicator
CIK:	Cockpit Interactive Keyboard
MOVCMD:	Movement Command
IPC:	Unix System V Interprocess Communications
GUI:	Graphical User Interface
MTO:	Message To Observer
CNO:	Can Not Observe
ARTY:	Artillery
BDA:	Battle Damage Assessment
PIREP:	Pilot Report
ENET:	Ethernet
SIMNET:	Simulation Network
DIS:	Distributed Interactive Simulation
PDU:	Protocol Data Unit
APDU:	Association Layer Protocol Data Unit
X	X-Windows Windowing System
OSF	Open Software Foundation
BX	Builder's Xcessory
CSC:	Computer Software Component
CSU:	Computer Software Unit
DMS:	Digital Message Server
NBC:	Nuclear, Biological, & Chemical Weapons
DTG:	Date Time Group
UTM:	Universal Transverse Mercator
CSCI:	Computer Software Configuration Item

10. Appendix A - DMCC Software Structure Charts.

The structure charts on this and the following pages depict collectively the overall structure of the DMCC CSCI.

10.1 DMCC CSCI

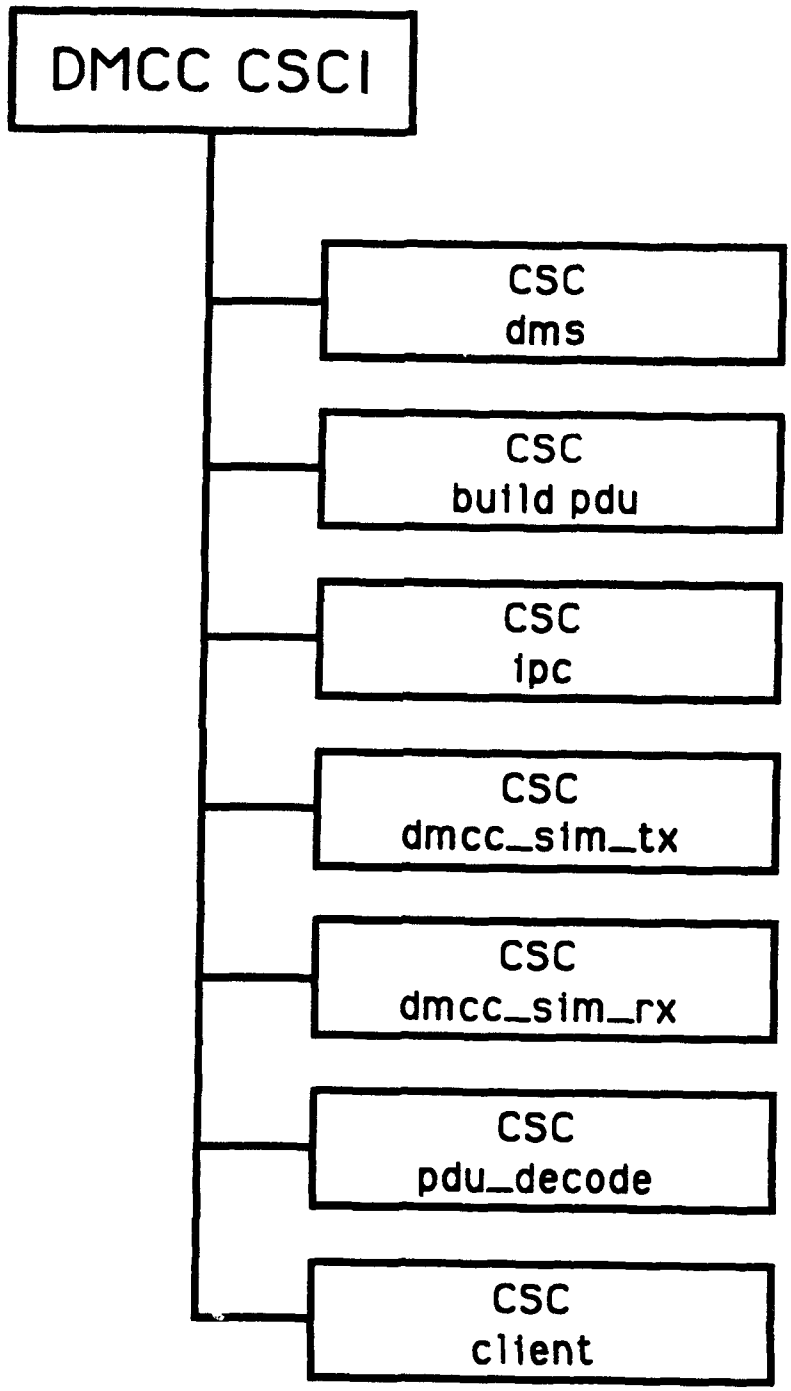


Figure 13 Top Level DMCC CSCI Structure

10.2 CSC dms

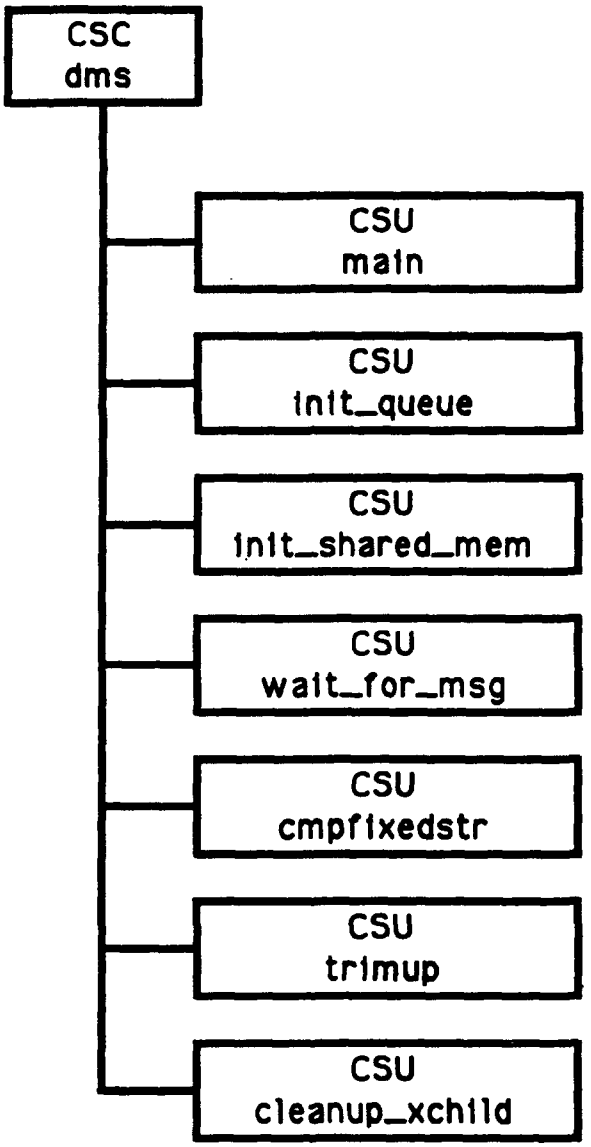


Figure14 CSC dms Structure

10.3 CSC build_pdu

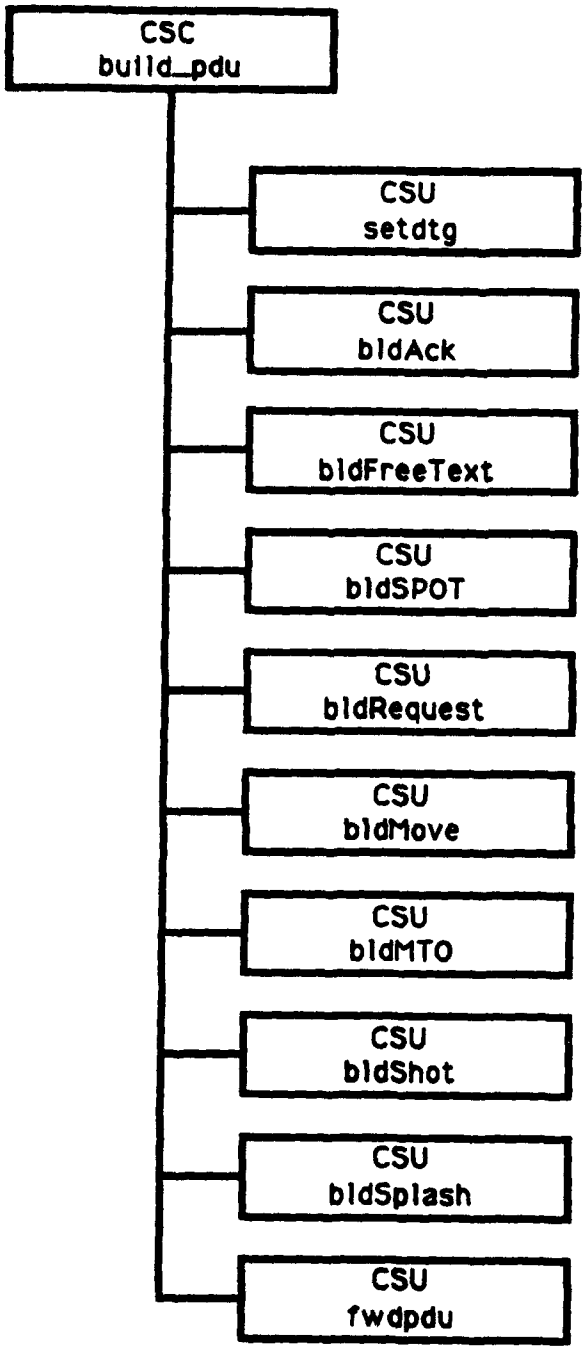


Figure 15 CSC build_pdu Structure

10.4 CSC ipc

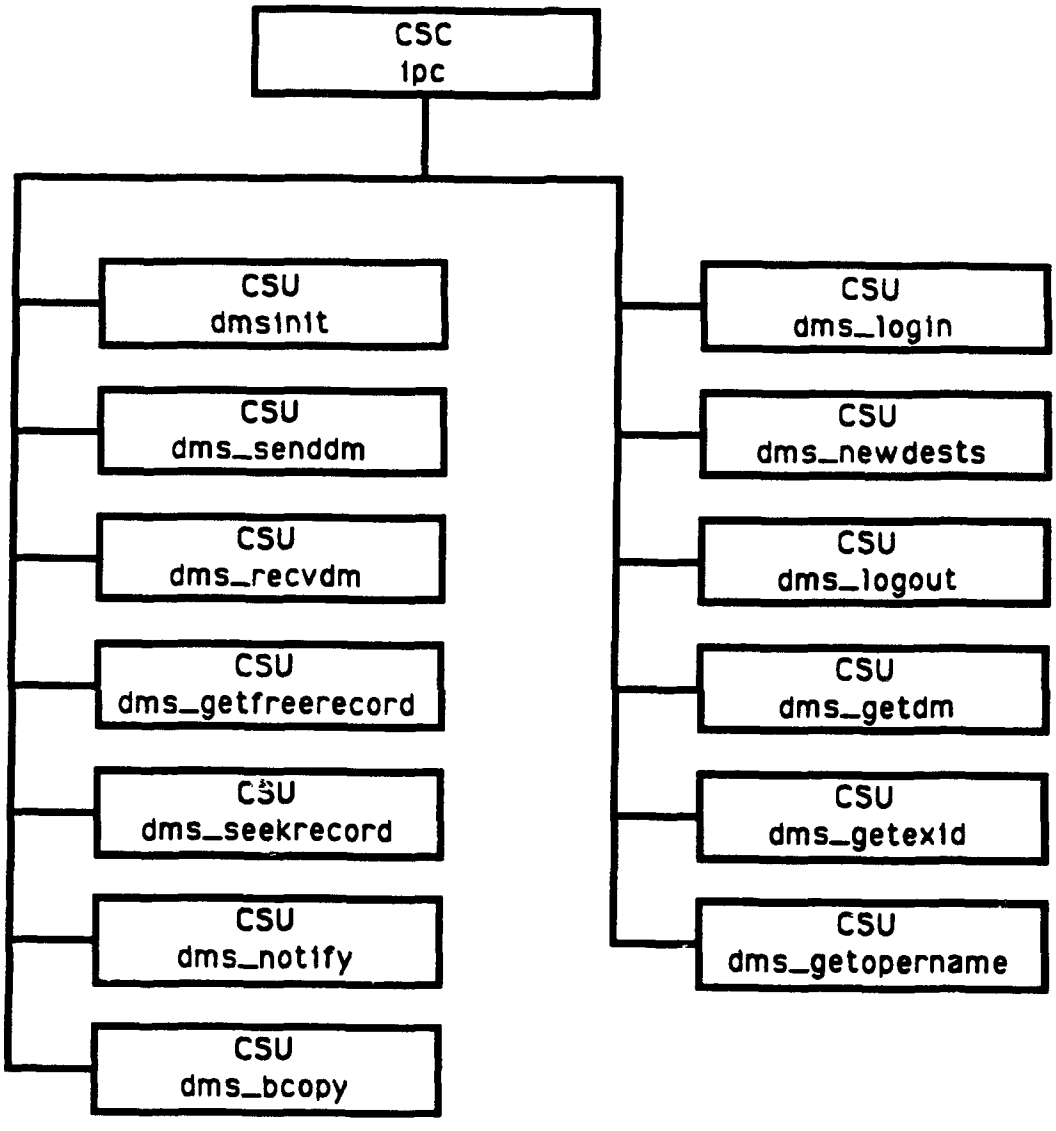


Figure 16 CSC ipc Structure

10.5 CSC dmcc_sim_tx

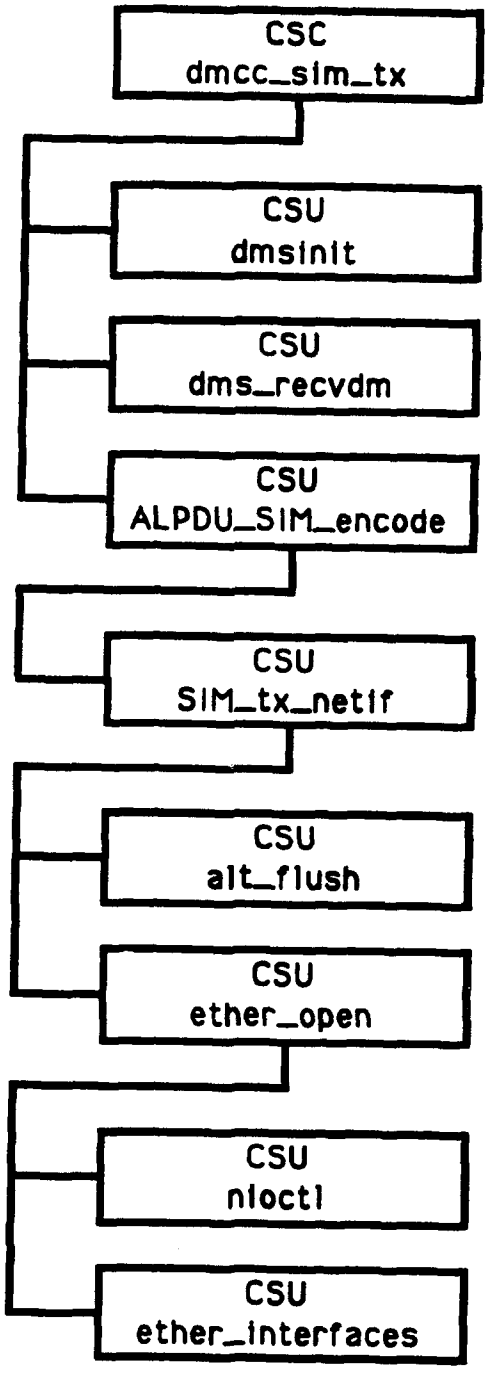


Figure 17 Sublevel CSC dmcc_sim_tx Structure

10.6 Sublevel CSC dmcc_sim_rx

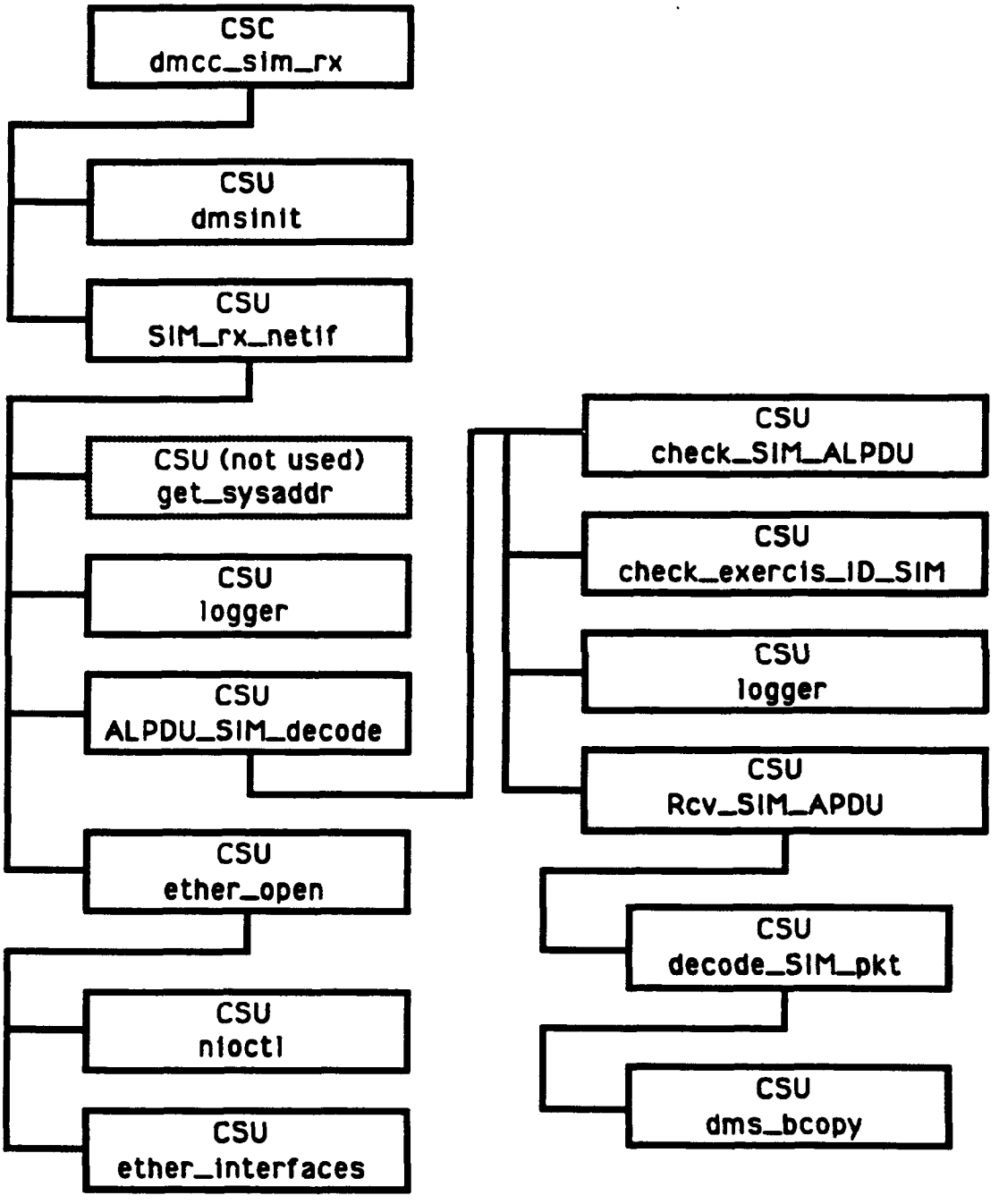


Figure 18 Sublevel CSC dmcc_sim_rx Structure

10.7 CSC pdu_decode

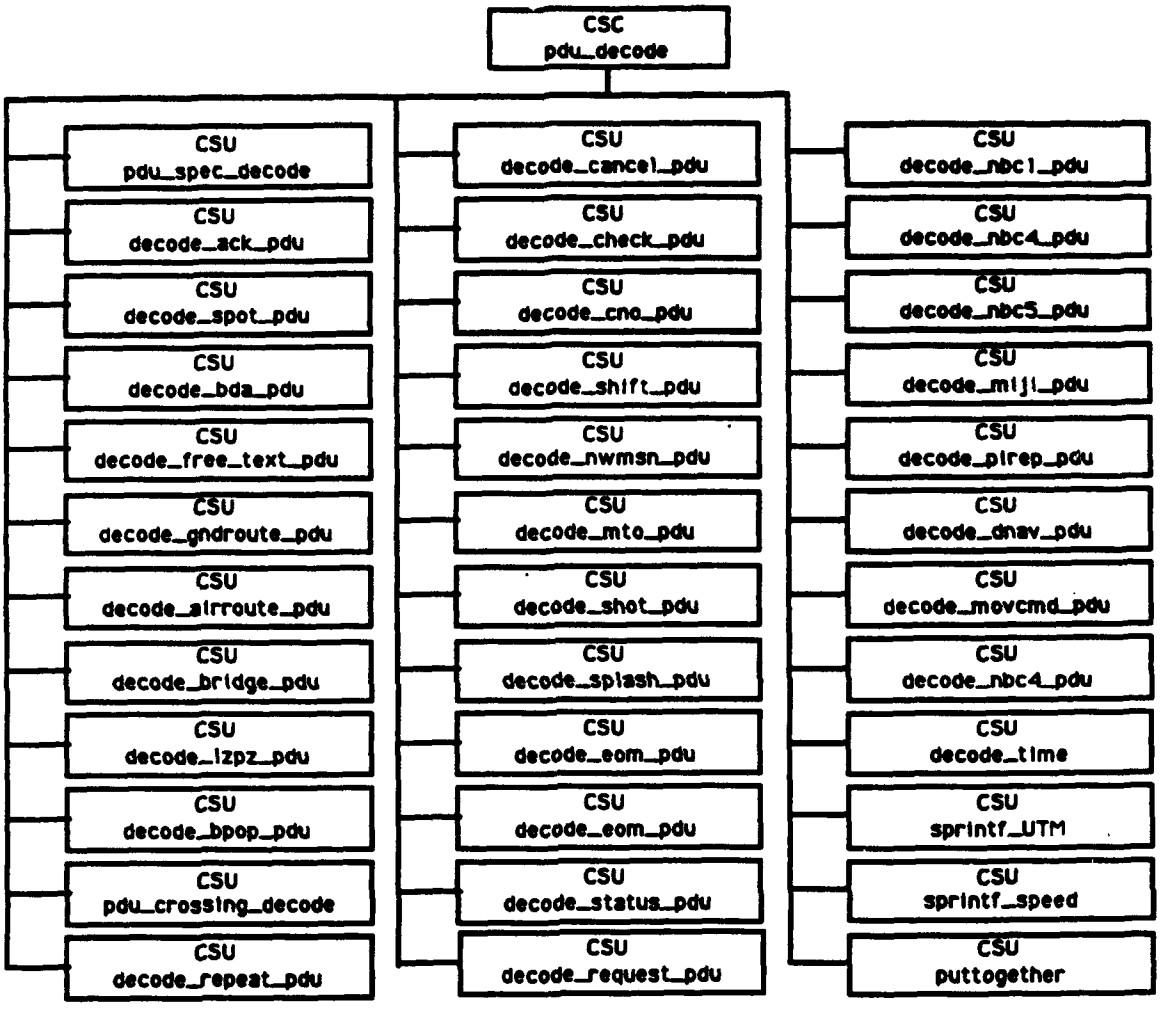


Figure 19 CSC pdu_decode Structure

10.8 CSC client

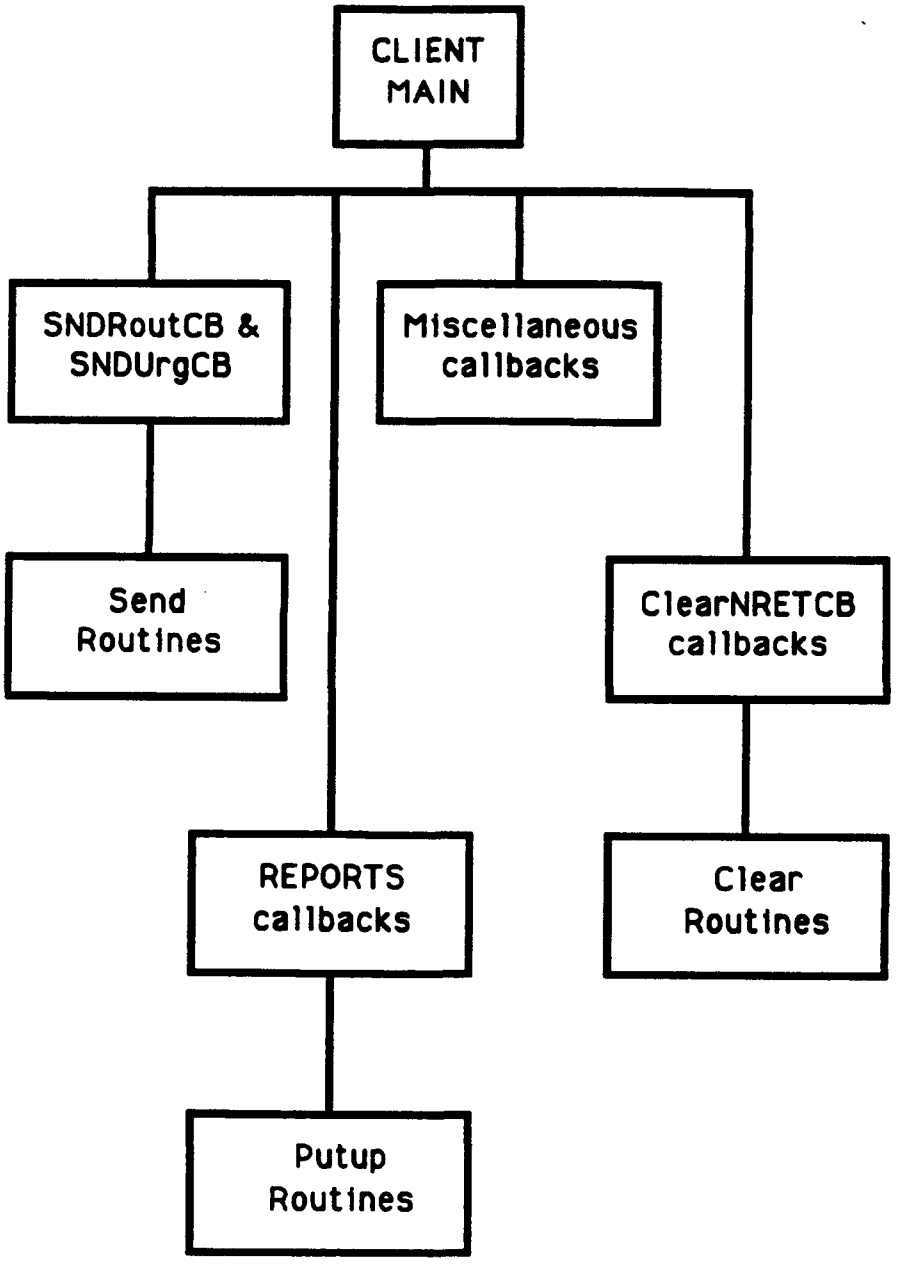


Figure 20 CSC client Structure

20. Appendix B - DMCC "C" Source Data Structures.

This is the set of "C" language header files (*.h) which describe the various datastructures and other data files . These files are located in the ADST Configuration Management Directory, under the following pathnames

a3/adst-cm/dmcc/include

a3/adst-cm/dmcc/gui/include

30. Appendix C - DMCC Client State Transition Diagram.

The diagram on the following pages shows the various states the CLIENT process can occupy, and the various actions which cause the process to change states. This diagram is an Excel 3.0 spreadsheet found in the MCC Software Development File under pathname IRM/ADST MacIIci #2/MCC SDF/Deliverable Documents/DMCC State Transition Diagram.

The X-Client Process State Event Matrix is shown on the following pages. This diagram shows actions and transitions from various window states bound to event flags representing operator intervention, such as clicking on a button region of a window. Each cell shows what actions are taken as a result of an event followed by a semicolon (;) character, which is followed by an indication of the next window to transition to (make active), if any. The top row is a list of the window events and the left column is a list of window states.

Table 160 Client State Transition Table

	Logon to Network Selected	Stand Alone Selected	Sys Main Selected	Address List Selected
Logon Window Active	Logon name and exercise ID to Network; go to SYS MAIN	No Logon; Go to SYS MAIN	n/a	n/a
Sys Main Window Active	n/a	n/a	n/a	Go To Address List Window
Address List Edit Window Active	n/a	n/a	Record Address List; Go to Sys Main	n/a
Location List Edit Window Active	n/a	n/a	Record Location List; Go to Sys Main	n/a
CEOI / Group List Edit Window Active	n/a	n/a	Record CEOI/ Group List; Go to Sys Main	n/a
Sys Main Msg Window Active	n/a	n/a	Go to Sys Main	n/a
Sys Main Msg Read Window Active	n/a	n/a	Go to Sys Main	n/a
Sys Main Rprt Window Active	n/a	n/a	Go to Sys Main	n/a

Table 160 Client State Transition Table, cont.

	CEOI/ Group List Selected	Location List Selected	MSGS Selected	RPRT Selected
Logon Window Active	n/a	n/a	n/a	n/a
Sys Main Window Active	Go to CEOI/ Group List Window	Go To Location List Window	Go to Sys Main Msg Window	Go To RPRT Window
Address List Edit Window Active	n/a	n/a	n/a	n/a
Location List Edit Window Active	n/a	n/a	n/a	n/a
CEOI / Group List Edit Window Active	n/a	n/a	n/a	n/a
Sys Main Msg Window Active	n/a	n/a	n/a	n/a
Sys Main Msg Read Window Active	n/a	n/a	n/a	n/a
Sys Main Rprt Window Active	n/a	n/a	n/a	n/a

Table 160 Client State Transition Table, cont.

	Up Selected	Down Selected	Add Selected	Delete Selected
Logon Window Active	n/a	n/a	n/a	n/a
Sys Main Window Active	n/a	n/a	n/a	n/a
Address List Edit Window Active	Round-Robin address highlight	Round-Robin address highlight	Add Address to Address List	Delete selected Address from Address List
Location List Edit Window Active	Round- Robin Location highlight	Rouund-Robin Location highlight	Add Location to Location List	Delete selected Location from Location List
CEOI / Group List Edit Window Active	Round-Robin Group highlight	Round Robin Group highlight	Add Group Name to Group List	Delete Group Name from Group List
Sys Main Msg Window Active	n/a	n/a	n/a	Delete Selected Message
Sys Main Msg Read Window Active	n/a	n/a	n/a	Delete Shown Message
Sys Main Rprt Window Active	n/a	n/a	n/a	n/a

Table 160 Client State Transition Table, cont.

	READ Selected	Logout Selected	Prev Selected	Next Selected
Logon Window Active	n/a	n/a	n/a	n/a
Sys Main Window Active	n/a	Logout; shut down client process	n/a	n/a
Address List Edit Window Active	n/a	n/a	n/a	n/a
Location List Edit Window Active	n/a	n/a	n/a	n/a
CEOI / Group List Edit Window Active	n/a	n/a	n/a	n/a
Sys Main Msg Window Active	Retrieve message; go to Sys Main Msg Read	n/a	Round-Robin message highlight	Round-Robin message highlight
Sys Main Msg Read Window Active	Go to Sys Main Msg Window	n/a	n/a	n/a
Sys Main Rprt Window Active	n/a	n/a	n/a	n/a

Table 160 Client State Transition Table, cont.

	Re-Use Selected	Reply Selected	Save & Exit Selected	Save & Return Selected
Logon Window Active	n/a	n/a	n/a	n/a
Sys Main Window Active	n/a	n/a	n/a	n/a
Address List Edit Window Active	n/a	n/a	n/a	n/a
Location List Edit Window Active	n/a	n/a	n/a	n/a
CEOI / Group List Edit Window Active	n/a	n/a	n/a	n/a
Sys Main Msg Window Active	Mark currently selected message for re-use; go to Sys Main Rprts	Mark currently selected message for reply; go to Sys Main Rprts	n/a	n/a
Sys Main Msg Read Window Active	Mark displayed message for re-use; go to Sys Main Rprts	Mark currently selected message for reply; go to Sys Main Rprts	n/a	n/a
Sys Main Rprt Window Active	n/a	n/a	Do not delete entered partial message; Go to SYS MAIN	n/a

Table 160 Client State Transition Table, cont.

	Clear& Return Selected	SPOT Selected	MTO Selected	SHOT Selected
Logon Window Active	n/a	n/a	n/a	n/a
Sys Main Window Active	n/a	n/a	n/a	n/a
Address List Edit Window Active	n/a	n/a	n/a	n/a
Location List Edit Window Active	n/a	n/a	n/a	n/a
CEOI / Group List Edit Window Active	n/a	n/a	n/a	n/a
Sys Main Msg Window Active	n/a	n/a	n/a	n/a
Sys Main Msg Read Window Active	n/a	n/a	n/a	n/a
Sys Main Rprt Window Active	n/a	Go to Spot Window	Go to MTO Window	Go to SHOT window

Table 160 Client State Transition Table, cont.

	SPLASH Selected	FREE TEXT Selected	REQT Selected	MOVCMD Selected
Logon Window Active	n/a	n/a	n/a	n/a
Sys Main Window Active	n/a	n/a	n/a	n/a
Address List Edit Window Active	n/a	n/a	n/a	n/a
Location List Edit Window Active	n/a	n/a	n/a	n/a
CEOI / Group List Edit Window Active	n/a	n/a	n/a	n/a
Sys Main Msg Window Active	n/a	n/a	n/a	n/a
Sys Main Msg Read Window Active	n/a	n/a	n/a	n/a
Sys Main Rprt Window Active	Go to SPLASH window	Go to FREE TEXT window	Go to REQUEST window	Go to MOVCMD window

Table 160 Client State Transition Table, cont.

	SND ROUT Selected	SND URG Selected	ADRS Selected	ADRS MORE Selected
Logon Window Active	n/a	n/a	n/a	n/a
Sys Main Window Active	n/a	n/a	n/a	n/a
Address List Edit Window Active	n/a	n/a	n/a	n/a
Location List Edit Window Active	n/a	n/a	n/a	n/a
CEOi / Group List Edit Window Active	n/a	n/a	n/a	n/a
Sys Main Msg Window Active	n/a	n/a	n/a	n/a
Sys Main Msg Read Window Active	n/a	n/a	n/a	n/a
Sys Main Rprt Window Active	n/a	n/a	n/a	n/a

Table 160 Client State Transition Table, cont.

	ENEMY TYPE Selected	ENEMY ACTIVITY Selected	ENEMY ACTIVITY MORE Selected	DIREC Selected
Logon Window Active	n/a	n/a	n/a	n/a
Sys Main Window Active	n/a	n/a	n/a	n/a
Address List Edit Window Active	n/a	n/a	n/a	n/a
Location List Edit Window Active	n/a	n/a	n/a	n/a
CEOI / [illegible] List Window Active	n/a	n/a	n/a	n/a
Sys Main Msg Window Active	n/a	n/a	n/a	n/a
Sys Main Msg Read Window Active	n/a	n/a	n/a	n/a
Sys Main Rprt Window Active	n/a	n/a	n/a	n/a

Table 160 Client State Transition Table, cont.

	OBS INT selected	SPEED Entered	NUMBER Selected	UNIT Selected
Logon Window Active	n/a	n/a	n/a	n/a
Sys Main Window Active	n/a	n/a	n/a	n/a
Address List Edit Window Active	n/a	n/a	n/a	n/a
Location List Edit Window Active	n/a	n/a	n/a	n/a
CEOI / Group List Edit Window Active	n/a	n/a	n/a	n/a
Sys Main Msg Window Active	n/a	n/a	n/a	n/a
Sys Main Msg Read Window Active	n/a	n/a	n/a	n/a
Sys Main Rprt Window Active	n/a	n/a	n/a	n/a

Table 160 Client State Transition Table, cont.

	REQ ADJ Selected	EAT Selected	EOM Selected	TASK Selected
Logon Window Active	n/a	n/a	n/a	n/a
Sys Main Window Active	n/a	n/a	n/a	n/a
Address List Edit Window Active	n/a	n/a	n/a	n/a
Location List Edit Window Active	n/a	n/a	n/a	n/a
CEOI / Group List Edit Window Active	n/a	n/a	n/a	n/a
Sys Main Msg Window Active	n/a	n/a	n/a	n/a
Sys Main Msg Read Window Active	n/a	n/a	n/a	n/a
Sys Main Rprt Window Active	n/a	n/a	n/a	n/a

Table 160 Client State Transition Table, cont.

	TASK MORE Selected	WHEN Selected	LCTN Selected	LCTN MORE Selected
Logon Window Active	n/a	n/a	n/a	n/a
Sys Main Window Active	n/a	n/a	n/a	n/a
Address List Edit Window Active	n/a	n/a	n/a	n/a
Location List Edit Window Active	n/a	n/a	n/a	n/a
CEOI / Group List Edit Window Active	n/a	n/a	n/a	n/a
Sys Main Msg Window Active	n/a	n/a	n/a	n/a
Sys Main Msg Read Window Active	n/a	n/a	n/a	n/a
Sys Main Rprt Window Active	n/a	n/a	n/a	n/a

Table 160 Client State Transition Table, cont.

	UTM Entered	TIME Entered	FREE TEXT ANNOTATION Entered
Logon Window Active	n/a		n/a
Sys Main Window Active	n/a		n/a
Address List Edit Window Active	n/a		n/a
Location List Edit Window Active	n/a		n/a
CEOI / Group List Edit Window Active	n/a		n/a
Sys Main Msg Window Active	n/a	n/a	n/a
Sys Main Msg Read Window Active	n/a	n/a	n/a
Sys Main Rprt Window Active	n/a	n/a	n/a

Table 160 Client State Transition Table, cont.

	Logon to Network Selected	Stand Alone Selected	Sys Main Selected	Address List Selected
MTO Window Active	n/a	n/a	Go to Sys Main	n/a
SHOT Window Active	n/a	n/a	Go to Sys Main	n/a
SPLASH Window Active	n/a	n/a	Go to Sys Main	n/a
MOVCMD Window Active	n/a	n/a	Go to Sys Main	n/a
REQUEST Window Active	n/a	n/a	Go to Sys Main	n/a
SPOT Window Active	n/a	n/a	Go to Sys Main	n/a
FREE TEXT Window Active	n/a	n/a	Go to Sys Main	n/a

Table 160 Client State Transition Table, cont.

	CEOI/ Group List Selected	Location List Selected	MSGS Selected	RPRT Selected
MTO Window Active	n/a	n/a	n/a	n/a
SHOT Window Active	n/a	n/a	n/a	n/a
SPLASH Window Active	n/a	n/a	n/a	n/a
MOVCMD Window Active	n/a	n/a	n/a	n/a
REQUEST Window Active	n/a	n/a	n/a	n/a
SPOT Window Active	n/a	n/a	n/a	n/a
FREE TEXT Window Active	n/a	n/a	n/a	n/a

Table 160 Client State Transition Table, cont.

	Up Selected	Down Selected	Add Selected	Delete Selected
MTO Window Active	n/a	n/a	n/a	n/a
SHOT Window Active	n/a	n/a	n/a	n/a
SPLASH Window Active	n/a	n/a	n/a	n/a
MOVCMD Window Active	n/a	n/a	n/a	n/a
REQUEST Window Active	n/a	n/a	n/a	n/a
SPOT Window Active	n/a	n/a	n/a	n/a
FREE TEXT Window Active	n/a	n/a	n/a	n/a

Table 160 Client State Transition Table, cont.

	READ Selected	Logout Selected	Prev Selected	Next Selected
MTO Window Active	n/a	n/a	n/a	n/a
SHOT Window Active	n/a	n/a	n/a	n/a
SPLASH Window Active	n/a	n/a	n/a	n/a
MOVCMD Window Active	n/a	n/a	n/a	n/a
REQUEST Window Active	n/a	n/a	n/a	n/a
SPOT Window Active	n/a	n/a	n/a	n/a
FREE TEXT Window Active	n/a	n/a	n/a	n/a

Table 160 Client State Transition Table, cont.

	Re-Use Selected	Reply Selected	Save & Exit Selected	Save & Return Selected
MTO Window Active	n/a	n/a	n/a	Do not delete entered partial MTO message, Go to SYS MAIN RPRTS
SHOT Window Active	n/a	n/a	n/a	Do not delete entered partial SHOT message; Go to SYS MAIN RPRTS
SPLASH Window Active	n/a	n/a	n/a	Do not delete entered partial SPLASH message; Go to SYS MAIN RPRTS
MOVCMD Window Active	n/a	n/a	n/a	Do not delete entered partial MOVCMD message; Go to SYS MAIN RPRTS
REQUEST Window Active	n/a	n/a	n/a	Do not delete entered partial REQUEST message; go to SYS MAIN RPRTS
SPOT Window Active	n/a	n/a	n/a	Do not delete entered partial SPOT message; Go to SYS MAIN RPRTS
FREE TEXT Window Active	n/a	n/a	n/a	Do not delete entered partial FREE TEXT message; Go to SYS MAIN RPRTS

Table 160 Client State Transition Table, cont.

	Clear& Return Selected	SPOT Selected	MTO Selected	SHOT Selected
MTO Window Active	Delete entered partial message; Go to SYS MAIN RPRTS	n/a	n/a	n/a
SHOT Window Active	Delete entered partial SHOT message; Go to SYS MAIN RPRTS	n/a	n/a	n/a
SPLASH Window Active	Delete entered partial SPLASH message; Go to SYS MAIN RPRTS	n/a	n/a	n/a
MOVCMD Window Active	Delete entered partial MOVCMD message; Go to SYS MAIN RPRTS	n/a	n/a	n/a
REQUEST Window Active	Delete entered partial REQUEST message; go to SYS MAIN RPRTS	n/a	n/a	n/a
SPOT Window Active	Delete entered partial SPOT message; go to SYS MAIN RPRTS	n/a	n/a	n/a
FREE TEXT Window Active	Delete entered partial FREE TEXT message; Go to SYS MAIN RPRTS	n/a	n/a	n/a

Table 160 Client State Transition Table, cont.

	SPLASH Selected	FREE TEXT Selected	REQT Selected	MOVCMD Selected
MTO Window Active	n/a	n/a	n/a	n/a
SHOT Window Active	n/a	n/a	n/a	n/a
SPLASH Window Active	n/a	n/a	n/a	n/a
MOVCMD Window Active	n/a	n/a	n/a	n/a
REQUEST Window Active	n/a	n/a	n/a	n/a
SPOT Window Active	n/a	n/a	n/a	n/a
FREE TEXT Window Active	n/a	n/a	n/a	n/a

Table 160 Client State Transition Table, cont.

	SND ROUT Selected	SND URG Selected	ADRS Selected	ADRS MORE Selected
MTO Window Active	Send MTO Report ROUTINE; Go to SYS MAIN RPRTS	Send MTO Report URGENT; Go to SYS MAIN RPRTS	Round-Robin the ADRS highlight	Toggle alternate address selections
SHOT Window Active	Send SHOT Report ROUTINE; Go to SYS MAIN RPRTS	Send SHOT Report URGENT; Go to SYS MAIN RPRTS	Round-Robin the ADRS highlight	Toggle alternate address selections
SPLASH Window Active	Send SPLASH Report ROUTINE; Go to SYS MAIN RPRTS	Send SPLASH Report URGENT; Go to SYS MAIN RPRTS	Round-Robin the ADRS highlight	Toggle alternate address selections
MOVCMD Window Active	Send MOVCMD Report ROUTINE; Go to SYS MAIN RPRTS	Send MOVCMD Report URGENT; Go to SYS MAIN RPRTS	Round-Robin the ADRS highlight	Toggle alternate address selections
REQUEST Window Active	Send REQUEST Report ROUTINE; Go to SYS MAIN RPRTS	Send REQUEST Report URGENT; Go to SYS MAIN RPRTS	Round-Robin the ADRS highlight	Toggle alternate address selections
SPOT Window Active	Send SPOT Report ROUTINE; Go to SYS MAIN RPRTS	Send SPOT Report URGENT; Go to SYS MAIN RPRTS	Round-Robin the ADRS highlight	Toggle alternate address selections
FREE TEXT Window Active	Send FREE TEXT REPORT ROUTINE; Go to SYS MAIN RPRTS	Send FREE TEXT REPORT URGENT; Go to SYS MAIN RPRTS	Round-Robin the ADRS highlight	Toggle alternate address selections

Table 160 Client State Transition Table, cont.

	ENEMY TYPE Selected	ENEMY ACTIVITY Selected	ENEMY ACTIVITY MORE Selected	DIREC Selected
MTO Window Active	n/a	n/a	n/a	n/a
SHOT Window Active	n/a	n/a	n/a	n/a
SPLASH Window Active	n/a	n/a	n/a	n/a
MOVCMD Window Active	n/a	n/a	n/a	n/a
REQUEST Window Active	n/a	n/a	n/a	n/a
SPOT Window Active	Round-Robin the ENEMY TYPE highlight	Round-Robin ENEMY ACTIVITY highlight	Toggle ENEMY ACTIVITY selections	Round-Robin the DIRECTION highlight
FREE TEXT Window Active	n/a	n/a	n/a	n/a

Table 160 Client State Transition Table, cont.

	OBS INT selected	SPEED Entered	NUMBER Selected	UNIT Selected
MTO Window Active	n/a	n/a	n/a	n/a
SHOT Window Active	n/a	n/a	n/a	n/a
SPLASH Window Active	n/a	n/a	n/a	n/a
MOVCMD Window Active	n/a	n/a	n/a	n/a
REQUEST Window Active	n/a	n/a	n/a	n/a
SPOT Window Active	Round-Robin OBSERVER INTENT highlight	Capture SPEED Text	Capture NUMBER Text	Capture UNIT Text
FREE TEXT Window Active	n/a	n/a	n/a	n/a

Table 160 Client State Transition Table, cont.

	REQ ADJ Selected	EAT Selected	EOM Selected	TASK Selected
MTO Window Active	Set MTO message type to REQ ADJ	Set MTO message type to Enter As Target	Set MTO message type to End of Mission	n/a
SHOT Window Active	n/a	n/a	n/a	n/a
SPLASH Window Active	n/a	n/a	n/a	n/a
MOVCMD Window Active	n/a	n/a	n/a	Round-Robin TASK highlight
REQUEST Window Active	n/a	n/a	n/a	n/a
SPOT Window Active	n/a	n/a	n/a	n/a
FREE TEXT Window Active	n/a	n/a	n/a	n/a

Table 160 Client State Transition Table, cont.

	TASK MORE Selected	WHEN Selected	LCTN Selected	LCTN MORE Selected
MTO Window Active	n/a	n/a	n/a	n/a
SHOT Window Active	n/a	n/a	n/a	n/a
SPLASH Window Active	n/a	n/a	n/a	n/a
MOVCMD Window Active	Toggle TASK selections	Round-Robin TASK selections	Round-Robin LCTN highlight	Toggle LCTN selections
REQUEST Window Active	n/a	n/a	n/a	n/a
SPOT Window Active	n/a	n/a	n/a	n/a
FREE TEXT Window Active	n/a	n/a	n/a	n/a

Table 160 Client State Transition Table, cont.

	UTM Entered	TIME Entered	FREE TEXT ANNOTATION Entered
MTO Window Active	n/a	n/a	Capture FREE TEXT Annotation
SHOT Window Active	n/a	n/a	Capture FREE TEXT Annotation
SPLASH Window Active	n/a	n/a	Capture FREE TEXT Annotation
MOVCMD Window Active	Capture UTMs	Capture TIME; change to ZULU if entered in LOCAL	Capture FREE TEXT Annotation
REQUEST Window Active	n/a	n/a	Capture FREE TEXT Annotation
SPOT Window Active	n/a	n/a	Capture FREE TEXT Annotation
FREE TEXT Window Active	n/a	n/a	Capture FREE TEXT Annotation

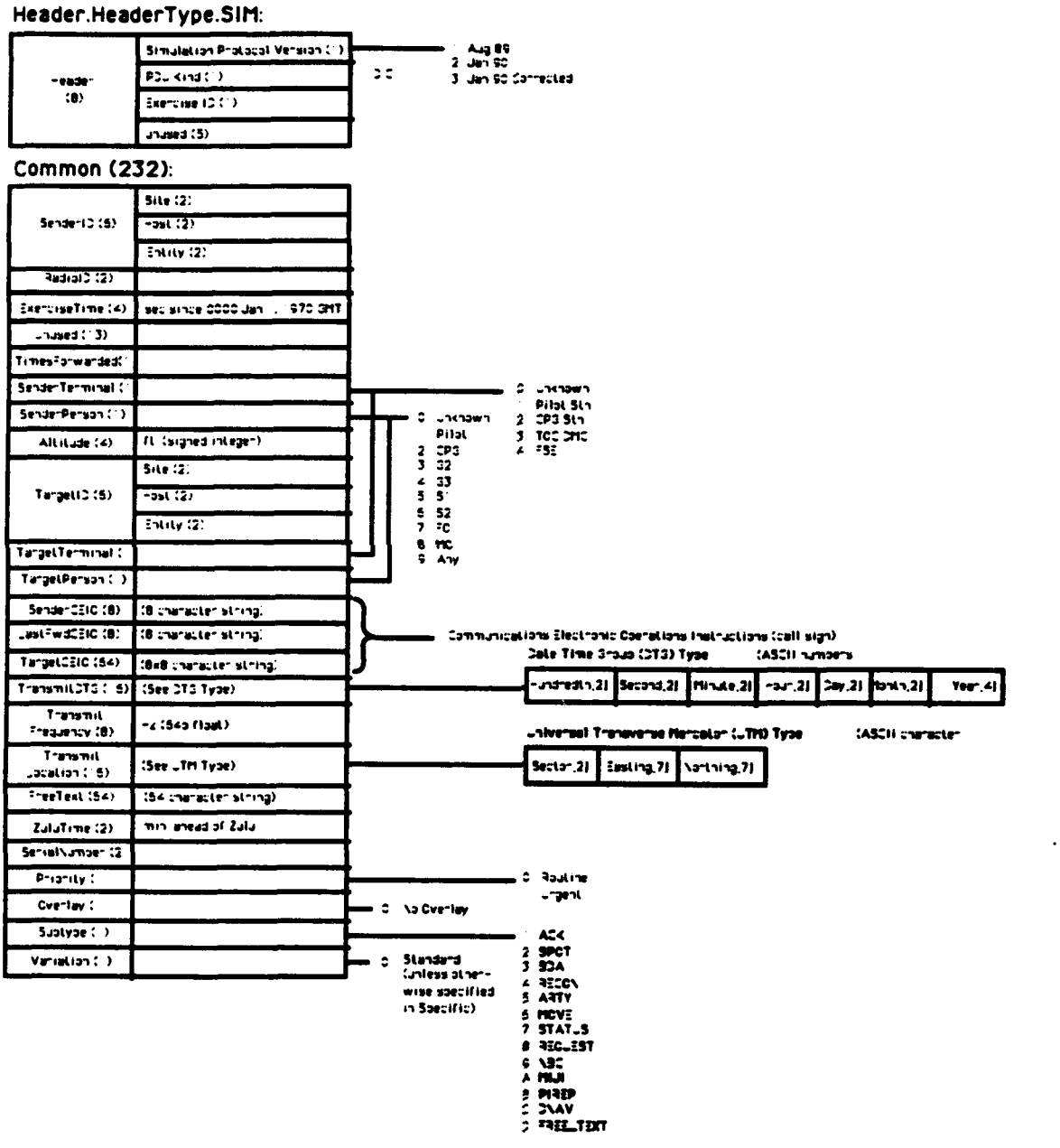
40. Appendix D - DMCC Protocol Data Units (PDUs).

The diagrams on the following pages graphically detail the message protocol data units used by the Digital Message Communications System. The DMCC software is capable of building, transmitting, receiving, decoding and displaying the SPOT, MTO, SHOT, SPLASH, REQUEST, MOVCMD, and FREE TEXT message PDUs. In addition, it is capable of receiving, decoding and displaying the BDA, RECON, ARTILLERY REPEAT, ARTILLERY CANCEL, ARTILLERY CNO ARTILLERY SHIFT, ARTILLERY NEW MISSION, ARTILLERY END MISSION, STATUS, NBC, PIREP, DNAV and MIJI message PDUs.

40.1 Simnet Header and common block

D C DMC PDU: Header & Common Block for Simnet

Rev. Date: 1/6/93



SIZE OF HEADER & COMMON BLOCK = 240

Figure 21 Simnet PDU Header & Common

40.2 DIS Header and Common Block

1A DMC (SIGNAL) PDU: DIS

Rev. Date: 1/6/93

Header_And_Common (240):

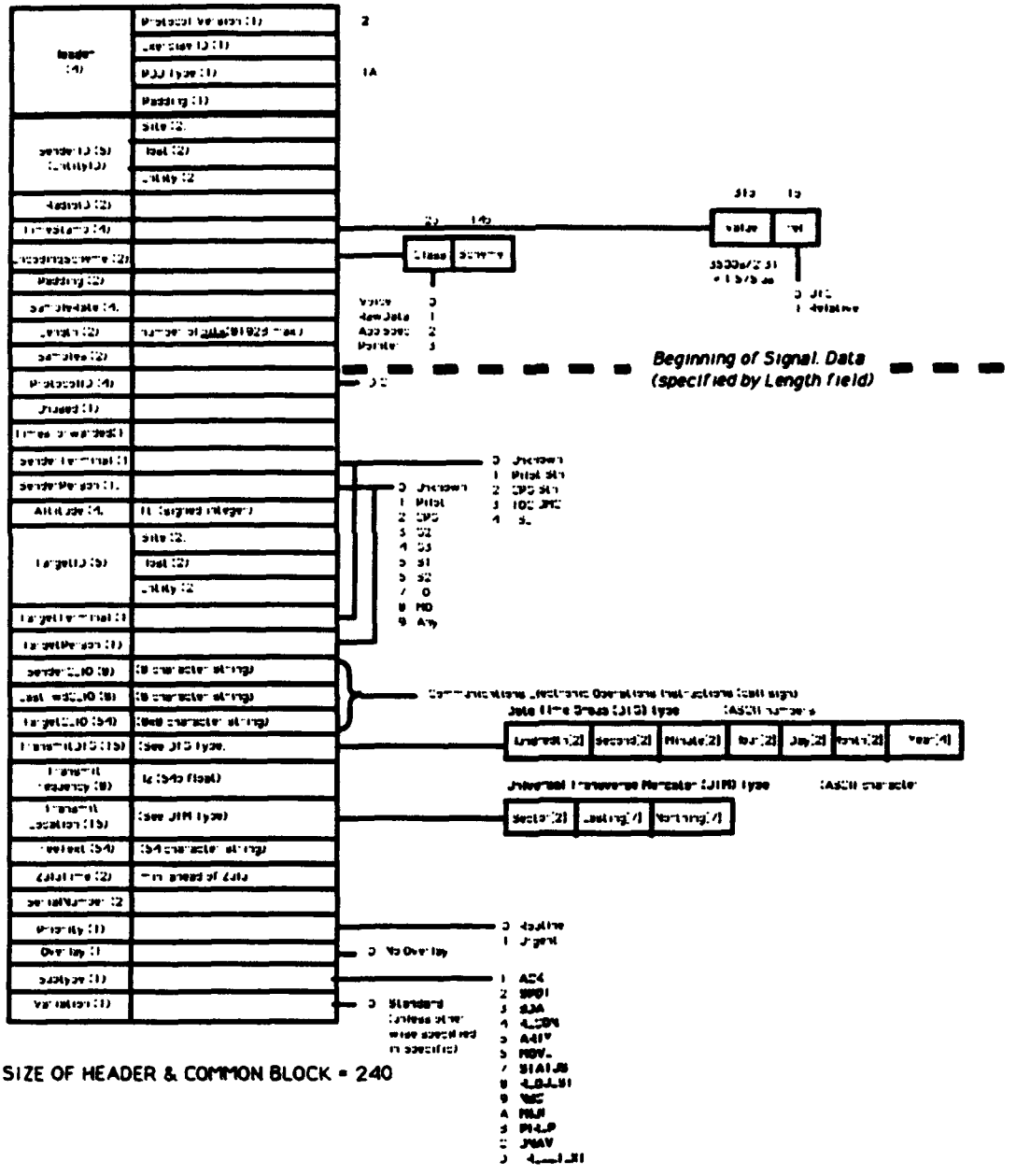


Figure 22 Simnet PDU Header & Common

40.3 BDA PDU Specific

Rev. Date:
12/16/92

03 Subtype = BDA
00 Variation = Standard

LEGEND
B = Bytes (default if not specified)
b = bits
all values specified in hex
fields are unsigned if not specified

Specific (40):

StrikeTime Start (16)	(See DTG Type)
StrikeTime End (16)	(See DTG Type)
TargetCategory (1)	
TargetType (1:	
Targets Destroyed (2)	
Percent Coverage (1:	
Unused (3)	

TargetType:
0 Not Entered
1 Unknown
2 ADA
3 SAM
4 Tank
5 Whld
6 Trkd
7 Acft
8 Trps
9 Uncl

Percent Coverage:
0 Not Enterec
1 0 %
2 25 %
3 50 %
4 75 %
5 100 %

TOTAL SIZE = 240 + 40 = 280 Bytes (includes Header & Common Block)

Figure 23 Battle Damage PDU Specific

40.4 MOVCMD PDU Specific

D C DMC PDU Specific:
06 Subtype = MOVE
00 Variation = Standard

Specific (48):

TargetID (16	(16 character strin
Task (1)	
Who (1)	
When (1)	
Unused (3)	
Zulu Time (2	Minutes ahead of Zul
Where (8)	(8 Character String
DTG (16)	(See DTG Type)

0 Not Entered
1 MOV TO
2 HOLD AT
3 CONT MSN
4 RENDZ AT
5 ENGAGE TGT AT
6 MVNG TO
7 HLDG AT
8 ARVING AT
9 PSNG THRU
A DEPRTNG FROM

0 Not Entered
1 ME
2 YOU
3 SEE BELOW

0 Not Entered
1 Immed
2 Whn Rdy
3 AMC
4 DTG

LEGEND
B = Bytes (default if not specified)
b = bits
all values specified in hex .
fields are unsigned if not specified

Rev. Date:
3/16/93

TOTAL SIZE = 240 + 48 = 288 Bytes (includes Header & Common Block)

Figure 24 MOVCMD PDU Specific

40.5 MIJI PDU Specific:

D C DMC PDU Specific:
0 A Subtype = MIJI
00 Variation = Standard

Specific (96):

Rev. Date:
12/15/92

Affected RadioFreq (8)	(64b float:
Programmed Freq (32)	(4x64b float)
KeypadEntry (16:	(16 character string)
JamStartDTG (16)	(See DTG Type)
JamEndDTG (16)	(See DTG Type)
Description (1:	
PercentLost (1)	
Type (1)	
Unused (5)	

LEGEND
B = Bytes (default if not specified)
b = bits
all values specified in hex
fields are unsigned if not specified

Description:
0 Not Entered
1 Unknown
2 Intermit
3 Cont
4 Imitative
5 Warbling
6 Music
7 Keypad

Type:
0 Not Enterec
1 Unknown
2 Meaconing
3 Intrusion
4 Jamming
5 Interfere

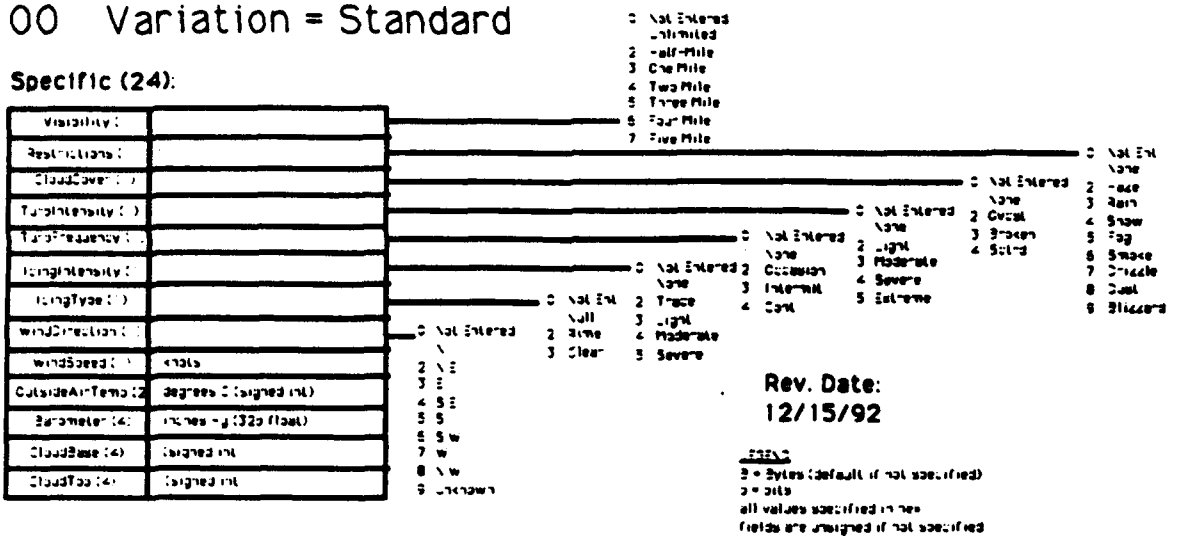
TOTAL SIZE = 240 + 96 = 336 Bytes (includes Header & Common Block)

Figure 25 MIJI PDU Specific

40.6 PIREP PDU Specific:

D C DMC PDU Specific:
0 B Subtype = PIREP
00 Variation = Standard

Specific (24):



TOTAL SIZE = 240 + 24 = 264 Bytes (includes Header & Common Block)

Figure 26 PIREP PDU Specific

40.7 RECON GND ROUTE PDU Specific:

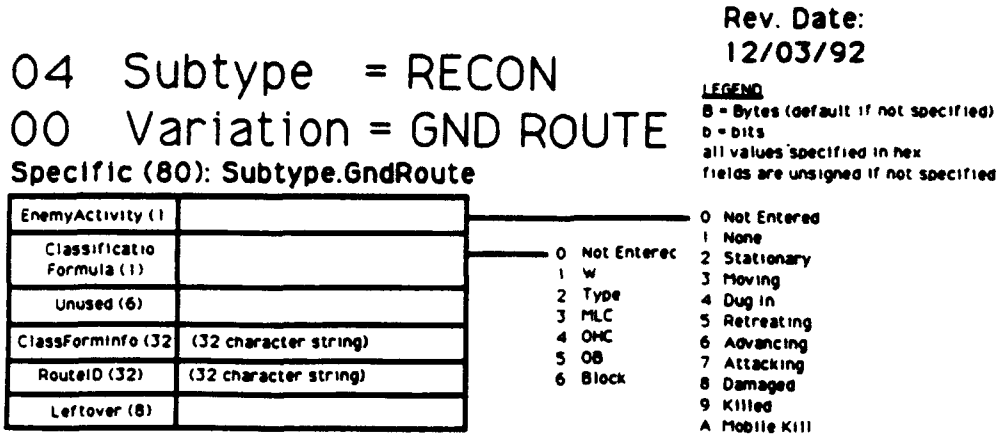


Figure 27 Gnd Route Recon PDU Specific

40.8 RECON AIR ROUTE Specific:

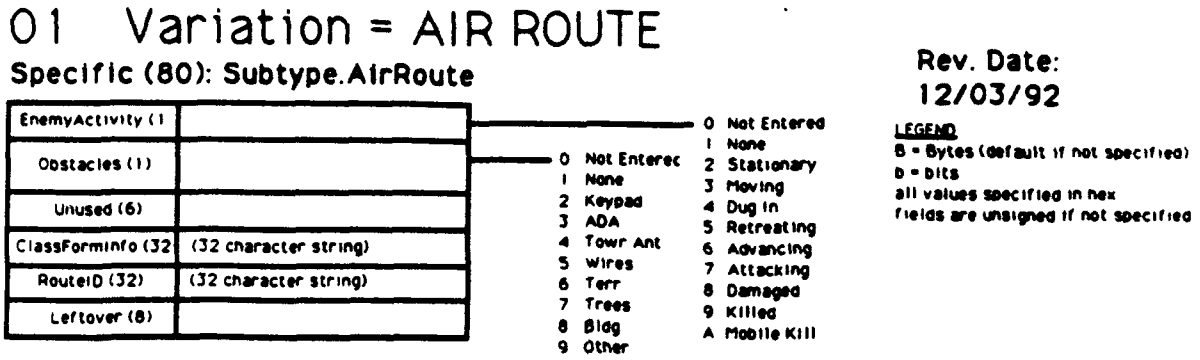


Figure 28 Air Route Recon PDU Specific

40.9 RECON Bridge Specific:

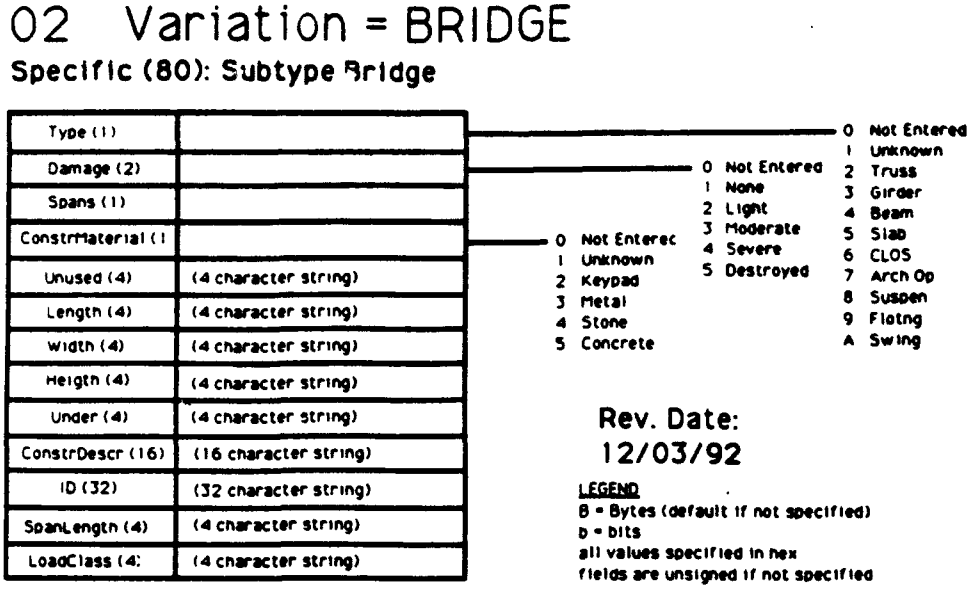


Figure 29 Bridge Recon PDU Specific

40.10 RECON LZ/PZ PDU Specific

D C DMC PDU Specific:
04 Subtype = RECON
03 Variation = LZ_PZ

Rev. Date:
11/23/92

LEGEND
B = Bytes (default if not specified)
b = bits
all values specified in hex
fields are unsigned if not specified

Specific (80): Subtype.LZ_PZ

ActivityLikely (1)	
Obstacles (1)	
Unused (6)	
Obstacles Descr (16)	(16 character string)
ID (16)	(16 character string)
LZ_Size (16)	(16 character string)
Axis (16)	(16 character string)
Leftover (8)	

0 Not Entered
1 None
2 Keypad
3 ADA
4 Towr Ant
5 Wires
6 Terr
7 Trees
8 Bldg
9 Other

0 Not Entered
1 None
2 Expected
3 Possible
4 Unlikely

Figure 30 LZ/PZ Recon PDU Specific

40.11 RECON BP-OP PDU Specific:

04 Variation = BP_OP

Specific (80): Subtype.BP_OP

EnemyActivity (1	
Obstacles (1)	
Unused (6)	
Obstacles Descr (16)	(16 character string)
ID (16)	(16 character string)
BP_Size (16)	(16 character string)
Axis (16)	(16 character string)
Leftover (8)	

0 Not Enterec
1 None
2 Keypad
3 ADA
4 Towr Ant
5 Wires
6 Terr
7 Trees
8 Bldg
9 Other

0 Not Entered
1 None
2 Stationary
3 Moving
4 Dug In
5 Retreating
6 Advancing
7 Attacking
8 Damaged
9 Killed
A Mobile Kill

Rev. Date:
11/23/92

LEGEND
B = Bytes (default if not specified)
b = bits
all values specified in hex
fields are unsigned if not specified

Figure 31 BP-OP Recon PDU Specific

40.12 RECON Crossing PDU Specific:

05 Variation = CROSSING

Specific (80): Subtype.Crossing

BankSlopeEntry (4)	(4 character string)
BankSlopeExit (4)	(4 character string)
CrossingLength (4)	(4 character string)
CrossingWidth (4)	(4 character string)
CrossingDepth (4)	(4 character string)
CurrentFlow (4)	(4 character string)
ID (8)	(8 character string)
Leftover (32)	

Rev. Date:
11/23/92

LEGEND
B = Bytes (default if not specified)
b = bits
all values specified in hex
fields are unsigned if not specified

Figure 32 Crossing Recon PDU Specific

40.13 ACK PDU Specific

D C DMC PDU Specific:
01 Subtype = ACK
00 Variation = Standard

Specific (48):

Original SenderID (6)	Site (2:
	Host (2)
	Entity (2:
OriginalSender Terminal (1)	
OriginalSende Person (1)	
AcknowledgeID (6	Site (2:
	Host (2)
	Entity (2:
Acknowledge Terminal (1)	
Acknowledge Person (1)	
Original SenderCEIO (8)	(8 character string)
Acknowledge CEIO (8)	(8 character strin
OriginalDTG(16:	(See DTG Type)

Rev. Date:
12/03/92

LEGEND
B = Bytes (default if not specified)
b = bits
all values specified in hex
fields are unsigned if not specified

TOTAL SIZE = 240 + 48 = 288 Bytes (includes Header & Common Block)

Figure 33 Acknowledge PDU Specific

40.14 SPOT PDU Specific

D C DMC PDU Specific:
02 Subtype = SPOT
00 Variation = Standard

Specific (88):

TimeSighted (16	(See DTG Type)
Observer Location (16)	(See UTM Type)
TargetQuantity(2)	
Observer Intentions (1)	
TargetType (1)	
TargetActivity (1	
TargetDirection (1	
TargetSpeed (2)	(See Speed Type)
Target Location (16)	(See UTM Type)
TargetUnit (16)	(16 character string)
Unused (16)	

Rev. Date:
12/03/92

0 Not Entered
1 Other
2 Cont Man
3 Engs
4 Hld
5 Rts Ihold
6 Obsv Tgt

0 Not Entered
1 Unknown
2 ADA
3 SAM
4 Tank
5 Whld
6 Trkd
7 Acft
8 Trps
9 Uncl

0 Not Entered
1 None
2 Stationary
3 Moving
4 Dug In
5 Retreating
6 Advancing
7 Attacking
8 Damaged
9 Killed
10 Mobile Kill

0 Not Enterec
1 N
2 N E
3 E
4 S E
5 S
6 S W
7 W
8 N W
9 Unknown

MPH | Value — Speed Type
1b 15b

LEGEND
B = Bytes (default if not specified)
b = bits
all values specified in hex
fields are unsigned if not specified

Figure 34 Acknowledge PDU Specific

40.15 NBC-1 PDU Specific

09 Subtype = NBC

00 Variation = NBC-1

Specific (64): Subtype.NBC_1

Description (1)	
BurstType (1)	
DeliveredBy (1)	
CloudHtUnits (1)	
CloudWidth (4)	in degrees (32b float)
CloudDescr (16)	(16 character string)
CloudHeight (2)	
FlashBangTime (2)	in seconds
Unused (4)	
StartDTG (16)	(See DTG Type)
EndDTG (16)	(See DTG Type)

CloudHtUnits (1): 0 Not Entered, 1 Degrees, 2 Ht in Ft, 3 Ht in M

DeliveredBy (1): 0 Not Entered, 1 Unknown, 2 Arty, 3 Mortar, 4 Rocket, 5 Missile, 6 Bomb

BurstType (1): 0 Not Entered, 1 None, 2 Unknown, 3 Surface, 4 Air, 5 Gnd

Description (1): 0 Not Entered, 1 Unusual, 2 Initial, 3 Increasing, 4 Decreasing, 5 Peak, 6 Special, 7 Series, 8 Verification, 9 Summary

Rev. Date: 11/30/92

LEGEND
B = Bytes (default if not specified)
b = bits
all values specified in hex
fields are unsigned if not specified

Figure 35 NBC-1 PDU Specific

40.16 NBC-4 PDU Specific

09 Subtype = NBC
03 Variation = NBC-4

Specific (64): Subtype.NBC_2

Description (1)	
BurstType (1)	
DeliveredBy (1)	
CloudHtUnits (1)	
CloudWidth (4)	in degrees (32b float)
CloudDescr (16)	(16 character string)
CloudHeight (2)	
DoseRate (2)	
Unused (2)	
Leftover (32)	

0 Not Entered
1 Degrees
2 Ht in Ft
3 Ht in M

0 Not Entered
1 Unknown
2 Arty
3 Mortar
4 Rocket
5 Missile
6 Bomb

0 Not Entered
1 None
2 Unknown
3 Surface
4 Air
5 Gnd

0 Not Entered
1 Unusual
2 Initial
3 Increasing
4 Decreasing
5 Peak
6 Special
7 Series
8 Verification
9 Summary

Rev. Date:
11/30/92

LEGEND
B = Bytes (default if not specified)
b = bits
all values specified in hex
fields are unsigned if not specified

Figure 36 NBC-4 PDU Specific

40.17 NBC-5 PDU Specific

09 Subtype = NBC

04 Variation = NBC-5

Specific (64): Subtype.NBC_Negative

Description (1)	
BurstType (1)	
DeliveredBy (1)	
CloudHtUnits (1)	
CloudWidth (4)	in degrees (32b float)
CloudDescr (16)	(16 character string)
CloudHeight (2)	
Leftover (38)	

CloudHtUnits (1):
0 Not Entered
1 Degrees
2 Ht in Ft
3 Ht in M

CloudDescr (16):
0 Not Entered
1 Unknown
2 Arty
3 Mortar
4 Rocket
5 Missile
6 Bomb

DeliveredBy (1):
0 Not Entered
1 None
2 Unknown
3 Surface
4 Air
5 Gnd

BurstType (1):
0 Not Entered
1 Unusual
2 Initial
3 Increasing
4 Decreasing
5 Peak
6 Special
7 Series
8 Verification
9 Summary

Rev. Date: 11/30/92

LEGEND
B = Bytes (default if not specified)
b = bits
all values specified in hex
fields are unsigned if not specified

Figure 37 NBC-5 PDU Specific

40.18 ARTILLERY REPEAT PDU Specific

D C DMC PDU Specific:
05 Subtype = ARTY
00 Variation = REPEAT
Specific (56): Subtype.Repeat

Rev. Date: 11/25/92

LEGEND
B = Bytes (default if not specified)
b = bits
all values specified in hex
fields are unsigned if not specified

Figure 38 Artillery Repeat PDU Specific

40.19 ARTILLERY CANCEL PDU Specific

D C DMC PDU Specific:
05 Subtype = ARTY
01 Variation = CANCEL
Specific (56): Subtype.Cancel

Rev. Date: 11/25/92

LEGEND
B = Bytes (default if not specified)
b = bits
all values specified in hex
fields are unsigned if not specifie

Figure 39 Artillery Cancel PDU Specific

40.20 ARTILLERY CHECK PDU Specific

D C DMC PDU Specific:
05 Subtype = ARTY
02 Variation = CHECK
Specific (56): Subtype.CheckFire

Rev. Date: 11/25/92

LEGEND
B = Bytes (default if not specified)
b = bits
all values specified in hex
fields are unsigned if not specifie

Figure 40 Artillery Check PDU Specific

40.21 ARTILLERY CNO PDU Specific

D C DMC PDU Specific:
05 Subtype = ARTY
03 Variation = CNO
Specific (56): Subtype.CNO

Rev. Date: 11/25/92

LEGEND
B = Bytes (default if not specified)
b = bits
all values specified in hex
fields are unsigned if not specifie

Figure 41 Artillery CNO PDU Specific

40.22 ARTILLERY SHIFT PDU Specific

D C DMC PDU Specific:
05 Subtype = ARTY

04 Variation = SHIFT
Specific (56): Subtype.Shift

Rev. Date:
11/25/92

LEGEND
B = Bytes (default if not specifiec
b = bits
all values specified in hex
fields are unsigned if not specifie

MisionID (16)	(16 character string)
TargetID (16)	(16 character string)
MissionStatus (1)	
FireforEffect (1)	
Unused (7)	
Message (16)	(16 character string)

MissionStatus (1):
0 Not Entered
1 Requested
2 Ready
3 Shot
4 Splash

FireforEffect (1):
0 FALSE (OFF)
1 TRUE (ON)

Figure 42 Artillery Shift PDU Specific

40.23 ARTILLERY SHOT PDU Specific

D C DMC PDU Specific:
05 Subtype = ARTY
07 Variation = SHOT
Specific (56): Subtype.Shot

Figure 43 Artillery Shot PDU Specific

40.24 ARTILLERY SPLASH PDU Specific

D C DMC PDU Specific:
05 Subtype = ARTY
08 Variation = SPLASH

Specific (56): Subtype.Splash

Rev. Date: 11/25/92

MisionID (16)	(16 character string)
TargetID (16)	(16 character string)
MissionStatus (1)	
RoundsFired (1)	
ImpactTime (1)	in seconds
Unused (5)	
Leftover (16)	

0 Not Entered
1 Requested
2 Ready
3 Shot
4 Splash

LEGEND
B = Bytes (default if not specified)
b = bits
all values specified in hex
fields are unsigned if not specified

Figure 44 Artillery Splash PDU Specific

40.25 ARTILLERY EOM PDU Specific

D C DMC PDU Specific:
05 Subtype = ARTY
09 Variation = EOM

Specific (56): Subtype.EndofMission

Rev. Date: 11/25/92

LEGEND
B = Bytes (default if not specifiec
b = bits
all values specified in hex
fields are unsigned if not specifie

MisionID (16)	(16 character string)
TargetID (16)	(16 character string)
MissionStatus (1)	
Disposition (1)	
Casualties (1)	
RecordTarget (1)	
CasualtyNumber (2)	
Unused (2)	
PointNumber (16)	(16 character string)

0 Not Entered
1 Requested
2 Ready
3 Shot
4 Splash

0 Not Entered
1 Burning
2 CND

0 Not Entered
1 Given

0 FALSE
1 TRUE

Figure 45 Artillery EOM PDU Specific

40.26 STATUS PDU Specific

D C DMC PDU Specific:
07 Subtype = STATUS
00 Variation = Standard

Specific (16):

Rev. Date:
11/23/92

Fuel (1)	in lbs
Elements (1)	
FailedEquip (3	(array of 3)
Hellfires (1	
Stingers (1)	
Rockets (1)	
Rounds (1)	
RequestType (1)	
Unused (5)	

FailedEquip: 0 Not Entered, 1 Engl, 2 Eng2, 3 Guns, 4 Radio, 5 Dgntr, 6 Rktpod, 7 Msl Stn, 8 Radar, 9 Laser

RequestType: 0 Standard, 1 Automatic

LEGEND
B = Bytes (default if not specified)
b = bits
all values specified in hex
fields are unsigned if not specified

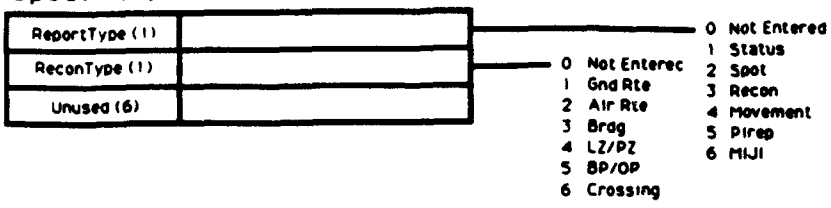
TOTAL SIZE = 240 + 16 = 256 Bytes (includes Header & Common Block)

Figure 46 Artillery Status PDU Specific

40.27 REQUEST PDU Specific

D C DMC PDU Specific:
08 Subtype = REQUEST
00 Variation = Standard

Specific (8):



TOTAL SIZE = 240 + 8 = 248 Bytes (includes Header & Common Block)

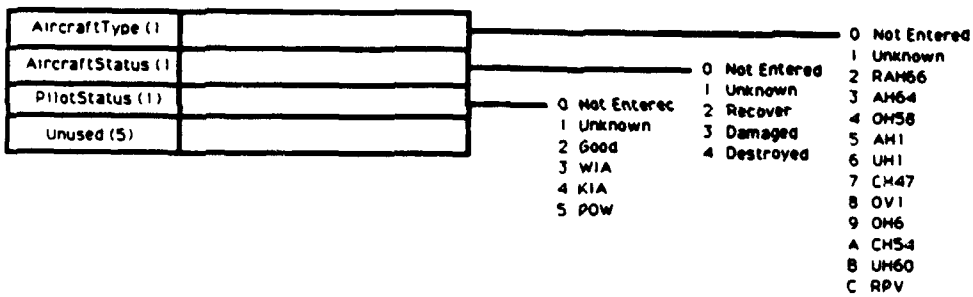
Figure 47 Artillery Request PDU Specific

40.28 DNAV PDU Specific

D C DMC PDU Specific: Rev. Date:
0 C Subtype = DNAV 11/23/92
00 Variation = Standard

LEGEND
B = Bytes (default if not specified)
b = bits
all values specified in hex
fields are unsigned if not specified

Specific (8):



TOTAL SIZE = 240 + 8 = 248 Bytes (includes Header & Common Block)

Figure 48 DNAV PDU Specific

40.29 FREE TEXT PDU Specific

D C DMC PDU Specific:
0 D Subtype = FREE TEXT
00 Variation = Standard

Rev. Date:
11/23/92

LEGEND
B = Bytes (default if not specified)
b = bits
all values specified in hex
fields are unsigned if not specified

Specific (256):

FreeText(256)	(256 character string)

TOTAL SIZE = 240 + 256 = 496 Bytes (includes Header & Common Block)

Figure 49 Free Text PDU Specific

50. Appendix E - DMCC Functional Development Tests.

The following pages detail the functional development tests which were used to verify proper operation of the DMCC software. These tests fall into several categories as noted here:

Developmental Thread Tests

1. Stand Alone IPC Thread Test
2. Dual Client Stand Alone Thread Test

Transmission and Reception Tests

3. Transmitted PDU Content Accuracy Test
4. PDU Reception Accuracy Text
5. DMCC X-terminal Configuration Test
6. Dual Host, Dual Client SIMNET Compliance Test

User Interface Tests:

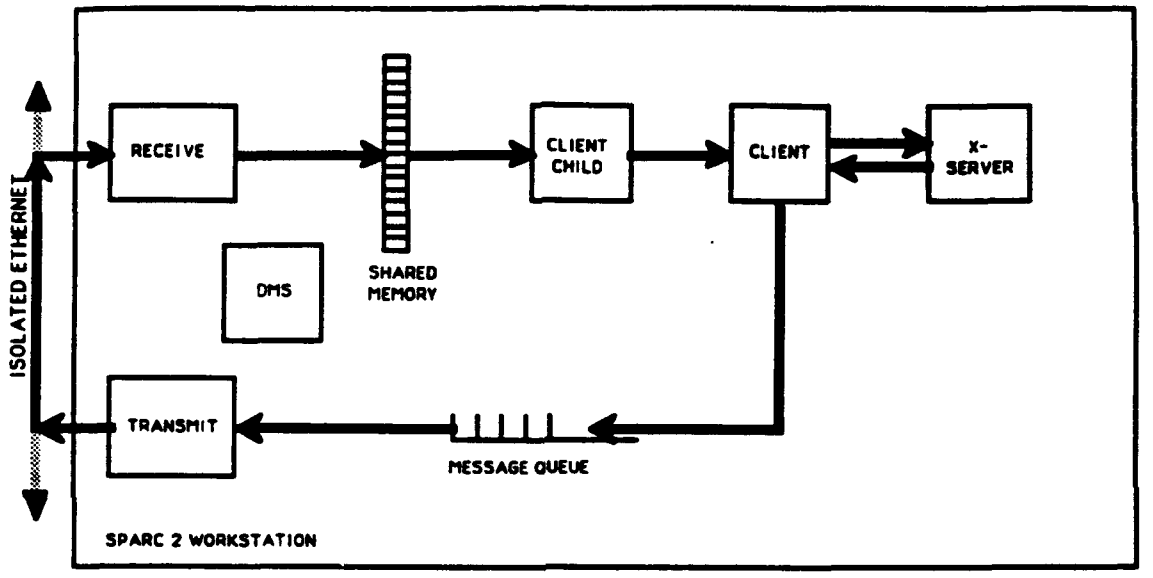
7. Window Content Verification Tests
8. Graphical User Interface Window Navigation Test
9. Re-Use and Reply; Acknowledge Tests
10. Miscellaneous Functional Tests

Developmental Thread Tests

50.1. Single Client Stand Alone IPC Thread Test

Purpose:

Verify operation of the Unix System V Interprocess Communication (IPC) elements in the DMCC software.



1. DMCC SINGLE CLIENT STAND ALONE DEVELOMPMENTAL THREAD TEST
(FREE TEXT MESSAGE;
HARD CODED DESTINATION;
NO CLIENT MESSAGE ARCHIVING OR RETRIEVAL)

Background:

In the Digital message Communications Console software, a message PDU is created in the client application and transmitted through the outgoing message queue to the concurrent ETHERNET TRANSMIT PROCESS. The TRANSMIT PROCESS then transmits the PDU on the Ethernet. The message is then received by the ETHERNET RECEIVE PROCESS, which asks the MESSAGE SERVER PROCESS where to put the message in shared memory. The RECEIVE process then informs the CLIENT CHILD PROCESS of the message's location through a message queue. The client child forked by the client then informs the client through an unnamed pipe that a new message has arrived for it. The client then retrieves the message from the shared memory and archives it for later retrieval and display.

Test Description:

This test will simulate transmission and reception of a Digital Message Protocol Data Unit by one DMCC CLIENT PROCESS. In this developmental test, no archiving of messages by the client is done, and the recipient address is hard-coded into the PDU, instead of being selectable by the user.

Acceptance Criteria:

A message must be successfully transmitted and received and the received Message must be identical to the transmitted one.

Test Procedure:

1. Start the DMCC developmental software using the START DMCC procedure.

2. Select the MSG1 window from the MAIN window.

3. Select the Free Text Message preparation window from the MSG1 window.

4. Enter a unique free text message (a string of 10 or 20 characters). Do not select an addressee for this message.

5. Transmit the message by clicking the mouse pointer on the SEND button.

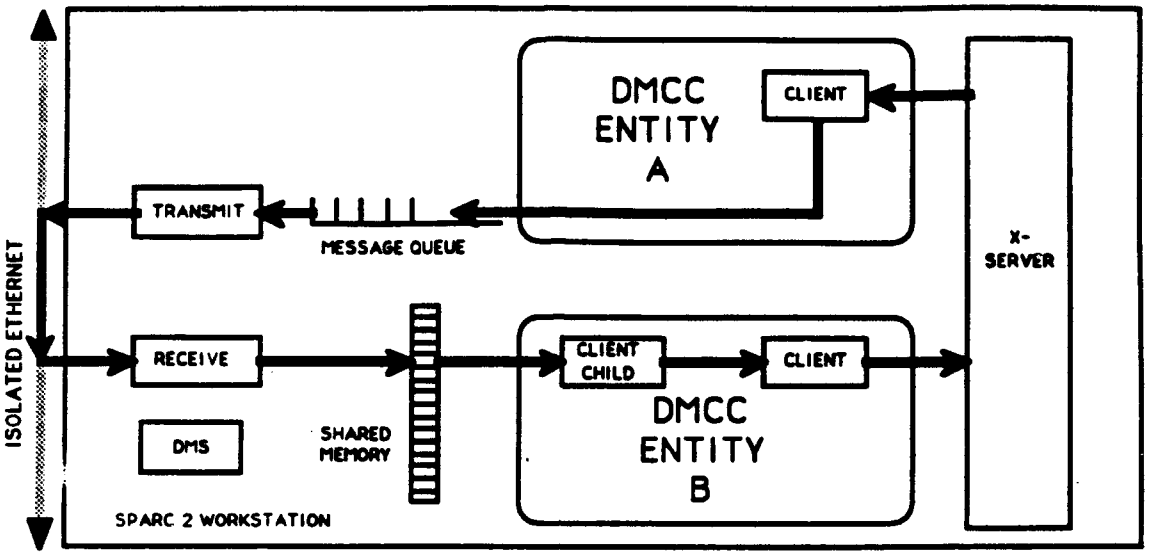
6. Observe that the message is received back from the Ethernet by the DMCC software.

7. Repeat steps 3 through 6, above, for several different length free text messages: 5 characters, 50 characters, 200 characters, and 255 characters. Allowable characters include _____.

50.2. Dual Client Stand Alone IPC Thread Test

Purpose:

Verify proper operation of the DMCC MESSAGE SERVER PROCESS.



2. DMCC DUAL CLIENT
STAND ALONE
DEVELOMPMENTAL THREAD TEST
(FREE TEXT MESSAGE;
OPERATOR SPECIFIED DESTINATION;
FULL CLIENT MESSAGE ARCHIVING & RETRIEVAL)

Background:

Each DMCC message contains at least one but no more than eight destinations or addresses. These destinations can take the form of the Call Sign (CEOI) of a destination entity, or a GROUP NAME for a group of DMCC entities to whom the digital message is collectively addressed. The CALL SIGN (CEOI) and GROUP NAMES are eight character alphanumeric identifiers.

However, in this implementation of the DMCC software, the graphical user interface does not allow for the entry of more than one simultaneous address for a particular message, be it a CALL SIGN (CEOI) or a GROUP NAME. The other seven destination fields in the Protocol Data Unit are left blank (filled with zeros).

Each DMCC Entity has a corresponding CALL SIGN (CEOI), and, optionally a list of up to seven GROUP NAMES to which it "claims" membership. More than one client may subscribe to messages in the same group. In this way, every client which subscribes to a group called "TEAM A" will receive messages transmitted to that group (in the same exercise).

During logon of a DMCC entity, the DMCC CLIENT PROCESS requests a logon from the message server. The CLIENT specifies to the message server what its CEOI is and what groups to which it belongs, as well as an EXERCISE ID. The CEOI and the GROUP NAMES are eight character alphanumeric strings. The message server process keeps a separate list of the CEOI, the GROUP NAMES and the Exercise ID for each client

In other words, the message server filters the messages according to destination and exercise.

Test Description:

This test will evaluate the performance of the message server. This process forms the bridge between the ETHERNET RECEIVE PROCESS and the clients. More than one client application may be running in the same workstation hardware at the same time, each corresponding to exactly one Digital Message Communications Entity. The message server receives a message from the ETHERNET RECEIVE process and sends it to the active client or clients to which it is addressed, and which are operating in the same exercise as the sending DMC entity.

Acceptance Criteria:

In this test, three DMCC entities (clients) are activated, and tests of sending messages between them are done.

The first test will check to see that a message properly addressed to the target Client and with the proper Exercise ID is correctly handled by the message server and received by the addressee client.

The third test will check to see that a message improperly addressed to the target Client but with the correct Exercise ID is correctly handled by the message server and not passed through shared memory to the client.

The second test will check to see that a message properly addressed to the target Client but with the wrong Exercise ID is correctly handled by the message server and not passed through shared memory to the client.

Test Procedure:

1. Start the DMCC software using the START DMCC procedure. Log a client on to the DMS using the logon name "**BINKY**" and the Exercise ID of 1.

2. Start another client using the START CLIENT procedure. Log this client into the Digital Message Server using the logon name "**AKBAR**" and an exercise ID of 1.

3. Start a third client using the START CLIENT procedure. Log this client into the DMS using the logon name "**BONGO**" and the exercise ID of 2.

4. Using the **BINKY** client, transfer to the ADDRESS LIST EDIT window and add the names "**BONGO**", "**AKBAR**" and "**JEFF**" to the list.

5. Using the **AKBAR** client, transfer to the ADDRESS LIST EDIT window and add the names "**BINKY**", "**BONGO**", AND "**JEFF**"

ADDRESS LISTS FOR THE TWO CLIENTS IN THIS TEST

BINKY CLIENT	AKBAR CLIENT	BONGO CLIENT
BONGO	BINKY	
AKBAR	BONGO	
JEFF	JEFF	

6. Using the **BINKY** client, transfer to the SPOT report window and send a spot report to the **AKBAR** client. Using the **AKBAR** client, transfer to the MSG screen and view the received messages. The spot report from the Binky client should be listed in the mailbox. Select this message and click READ to read it. **You should see the spot report you sent from the BINKY client.**

7. Using the **BINKY** client, transfer to the Reports window send a Spot report to the address **JEFF**.

8. Using the AKBAR client, transfer to the MSG screen and check to see that the second message from the BINKY client was NOT received.

9. Start another client and log on as **JEFF**, with Exercise ID 2. Go to the address list edit window and enter the address **BONGO**.

10. Using the **JEFF** client in Exercise 2, send a spot report to **BONGO**.

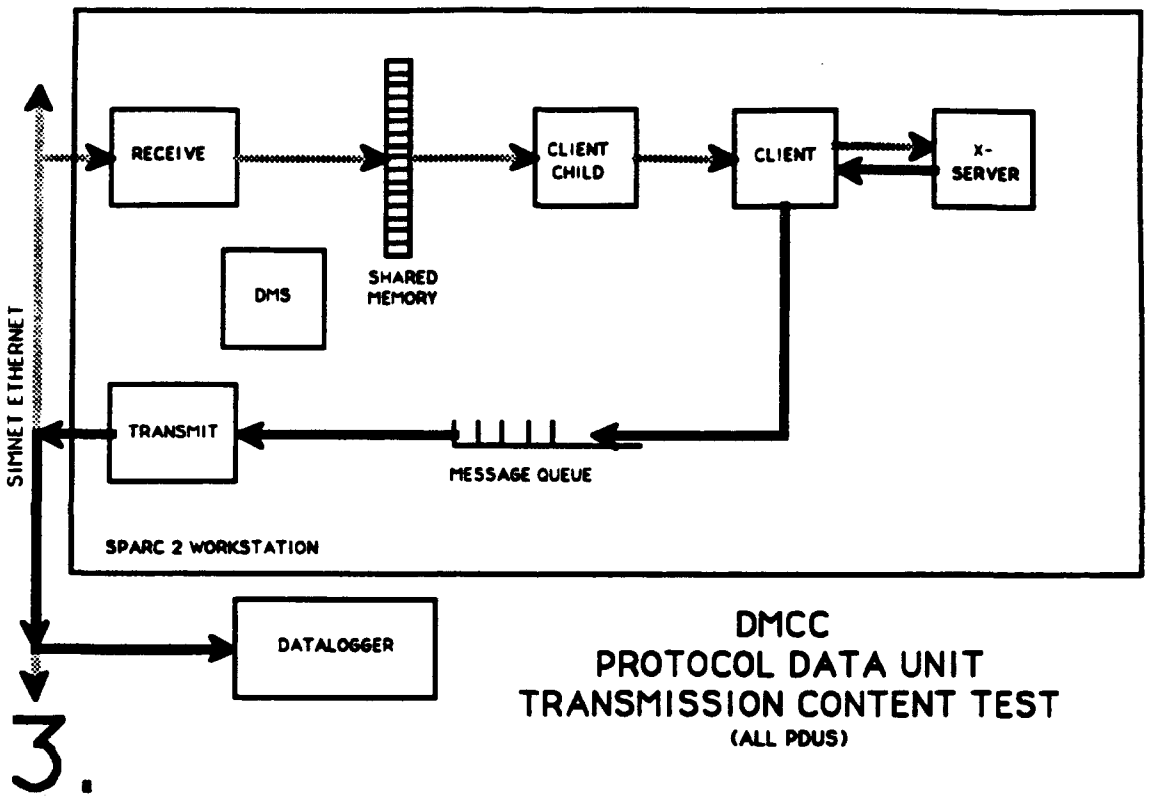
11. Transfer to the BONGO client and check to see that the message from JEFF was NOT received (since JEFF is in Exercise 2, and BONGO is in Exercise 1.)

DMCC Transmission and Reception Tests

50.3. Transmitted PDU Content Accuracy Test.

Purpose:

Verify correct transmission of message PDUs.



Background:

In this implementation, the Digital Message Communications Console is capable of sending seven different kinds of messages.

Test Description:

In this test, each of the different kinds of messages will be transmitted by the DMCC onto the network and logged by the Datalogger for manual inspection.

Acceptance Criteria:

Each Message PDU will be examined for content correctness.

Test Procedure:

1. Start the DMCC software using the DMCC start procedure.

2. Start a Client, and log on to the Digital Message Server as REAGAN, with Exercise ID 1.

3. Start the Datalogger and set it to capture Digital Message PDUs only.

4. Using the REAGAN client, send a spot report to the datalogger.

5. Send a MTO report to the datalogger.

6. Send a SHOT report to the datalogger.

7. Send a SPLASH report to the datalogger.

8. Send a REQUEST report to the datalogger.

9. Send a MOVCMD report to the datalogger.

10. Send a FREE TEXT report to the datalogger.

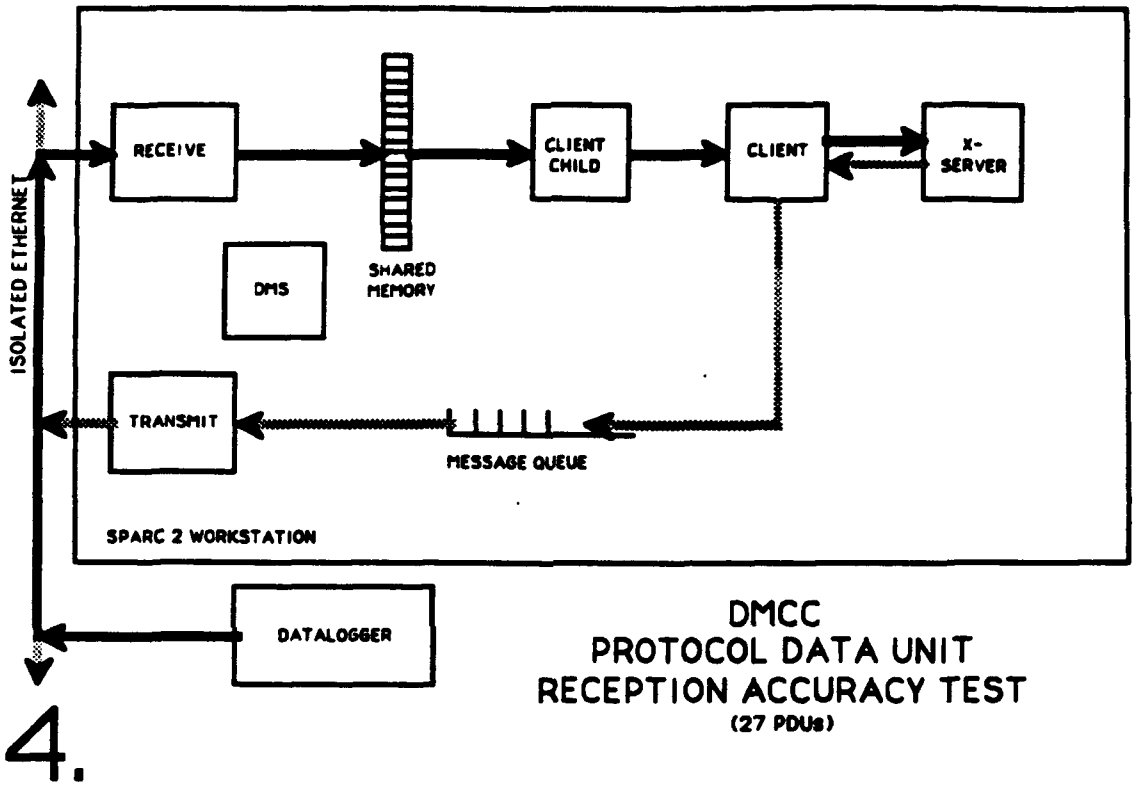
11. Stop the datalogger.

12. Print out the PDUs which were received by the datalogger from the REAGAN client.

13. Inspect the PDUs received from the REAGAN DMCC client for content correctness. **You should see the exact fields which you sent from the DMCC for each message type.**

50.4. DMCC PDU Reception Accuracy Test.

Purpose: To verify that the DMCC software properly decodes PDUs it receives from the network.



Background:

In this implementation, the DMCC software can receive, record, retrieve and display 28 different kinds of messages, most of which are actually incapable of being transmitted by the DMCC itself. These are messages which will (maybe) be transmitted by the simulated Comanche helicopter mission equipment package software, and represent digital communications between the helicopter aircrew and ground force elements and command centers.

Test Description:

For this test, the PDU Builder will be used to manufacture digital message PDUs which will be stored in a file. These will then be transmitted onto the network by the Anna Tool for reception, storage, and retrieval by a Digital Message Communications Console. The Message Read function of the DMCC will be

used to examine the contents of the received message, and these contents will be checked against the PDU.

Acceptance Criteria:

Each phony PDU will be constructed ahead of time and the contents noted. These PDUs will be contained in datalogger PDU files which can be read by the Datalogger and transmitted on the net. As each PDU is sent by the Datalogger and received by the DMCC, the PDU contents are checked against the displayed information. The test will be considered to be passed when each type of message is correctly received and displayed by the DMCC software.

Test Procedure:

1. Using the PDU Builder, prepare a set of PDU files for each PDU type in the Digital Message Protocol. Vary the fields of data between each of the PDUs of a particular type. Prepare enough PDUs to fully test each enumerated type in each kind of PDU. For example, in the **MOVCMD** PDU, the **TASK** enumerated type field contains 11 selections: Not Entered, To, Hold Cont Ms, Rendez At, Enga Tgt, Moving to, Hldg At, Arrivg At, Passg Thru, and Departg From. The **WHO** enumerated type field contains six selections, and the **WHEN** field contains five. For this example, 11 PDUs would need to be prepared, since 11 is the number of selections in the task with the most selections. Each PDU would have one of the 11 selections for the **TASK** field. The selections for the WHO and WHEN fields are to be varied as well, but some of the selections will have to be used more than once since there are only 5 selections and 11 different PDUs.

Build these PDUs with the DESTINATION field set to **MADONNA**, and the exercise ID set to 1.

Build some of the PDUs of each type with priority set to URGENT and some set to ROUTINE.

Build these PDUs with the date time group set to a constant date: 0705 March 3, 1993.

2. Build a set of 12 SPLASH PDUs to test the Date Time Group feature, by varying the date field to include each month. Vary the time portion of the field to allow tests of 0000 hours, 0600, 1159 and 2359 hours. Vary the seconds field. Obviously, an exhaustive test of all combinations of time is not practical; the attempt here is to just verify that the date and time are going thru for a representative selection of the possible cases.

3. Using the PDU Viewer, print the contents of each PDU out on paper.

4. Connect the DMCC to SIMNET.

5. Start the DMCC software using the DMCC Start Procedure.

6. Log on a CLIENT to the digital message server under the name **MADONNA**, using Exercise ID 1.

7. Set the Datalogger to log DMC PDUs.

8. Using the Anna Tool, transmit 15 PDUs into the DMCC.

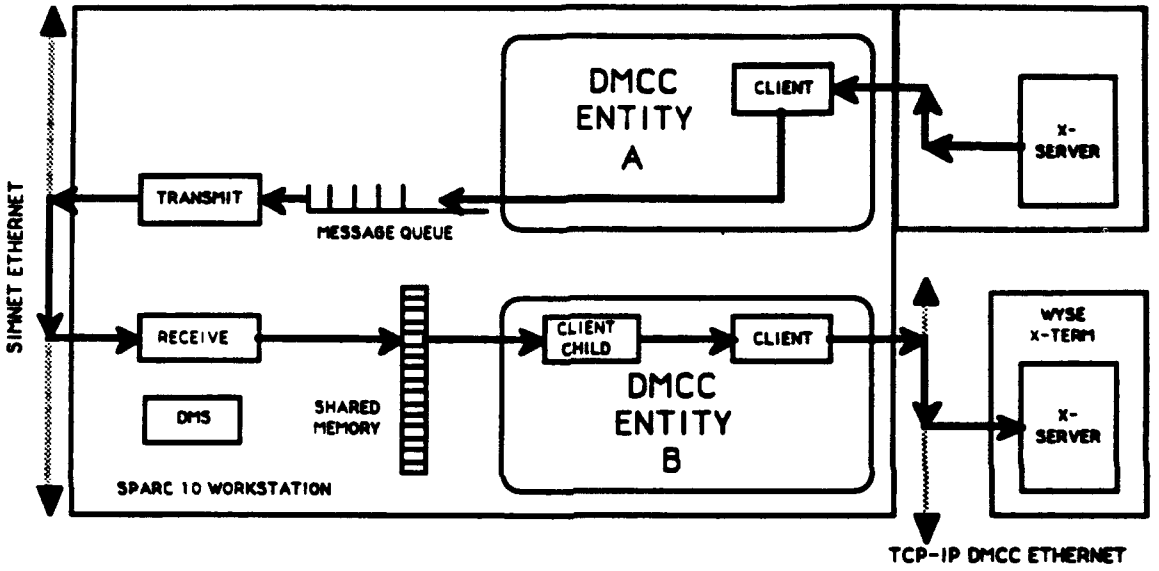
9. Using the DMCC, transfer to the MSG1 screen, select a PDU from the Message List and click READ.

10. Check the contents of the message as shown on the DMCC MSG READ window against the known contents of the PDU from the PDU Viewer. **For all PDUs, you should notice no discrepancies between the message contents and the PDU printout.**

50.5. DMCC X-terminal Configuration Test

Purpose:

This test confirms the DMCC operates properly in a dual client, single platform environment. It involves two client processes, one whose x-server resides in the workstation, and one whose x-server resides in a X-terminal.



5. DMCC
X-TERMINAL CONFIGURATION TEST
(7 MESSAGE TYPES + ACKNOWLEDGE;
OPERATOR SPECIFIED DESTINATION;
FULL CLIENT MESSAGE ARCHIVING & RETRIEVAL)

Background:

The Digital Message Communications Console software operates in a client/server X environment. Individual client applications interface with the network through the Digital Message Server and the network interface software. More than one DMCC client application may be simultaneously running in one workstation. Individual DMCC application windows may be displayed on the console display of the workstation or on X-terminals connected through a normal TCP-IP Ethernet to the workstation.

Test Description:

This test verifies operation of two client applications in the workstation at the same time, one whose server is resident in the workstation and one whose server is resident in a Wyse X-terminal.

Acceptance Criteria:

This test is considered to be passed if proper operation of the send and receive functions is observed for each of the two DMCC client entities.

Test Procedure:

1. Start the DMCC software in the workstation using the DMCC start up procedure.

2. At the console of the Workstation, log a client on to the Digital Message Server using the DMCC call sign SEAHAWK, and Exercise ID 1.

3. At the X-terminal, log a DMCC client onto the Digital Message Server using the call sign NATE and Exercise ID 1.

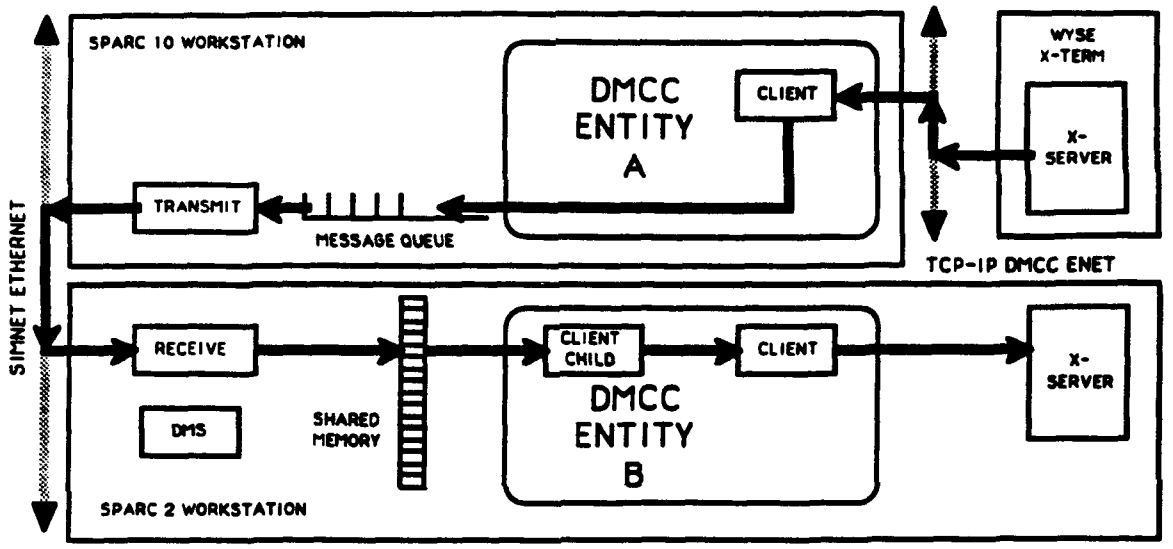
4. Using the X-term client, NATE, send some digital messages to the client SEAHAWK. Verify that they are received properly by SEAHAWK.

5. Using the Workstation client, SEAHAWK, send some various digital messages to NATE. Verify that they are received properly by NATE.

50.6. Dual Host Dual Client SIMNET Compliance Test

Purpose:

The purpose of this test is to verify operation of the DMCC software in the context of separate host platforms.



6. DMCC DUAL HOST DUAL CLIENT SIMNET COMPLIANCE TEST
(SELECTED MESSAGE TYPES; OPERATOR SPECIFIED DESTINATION; FULL CLIENT MESSAGE ARCHIVING & RETRIEVAL)

Background:

The DMCC is intended to be able to provide simulated digital communications between message entities (client applications) hosted on one machine or on several machines connected to the SIMNET ETHERNET or to the Distributed Interactive Simulation network via the protocol gateway.

Test Description:

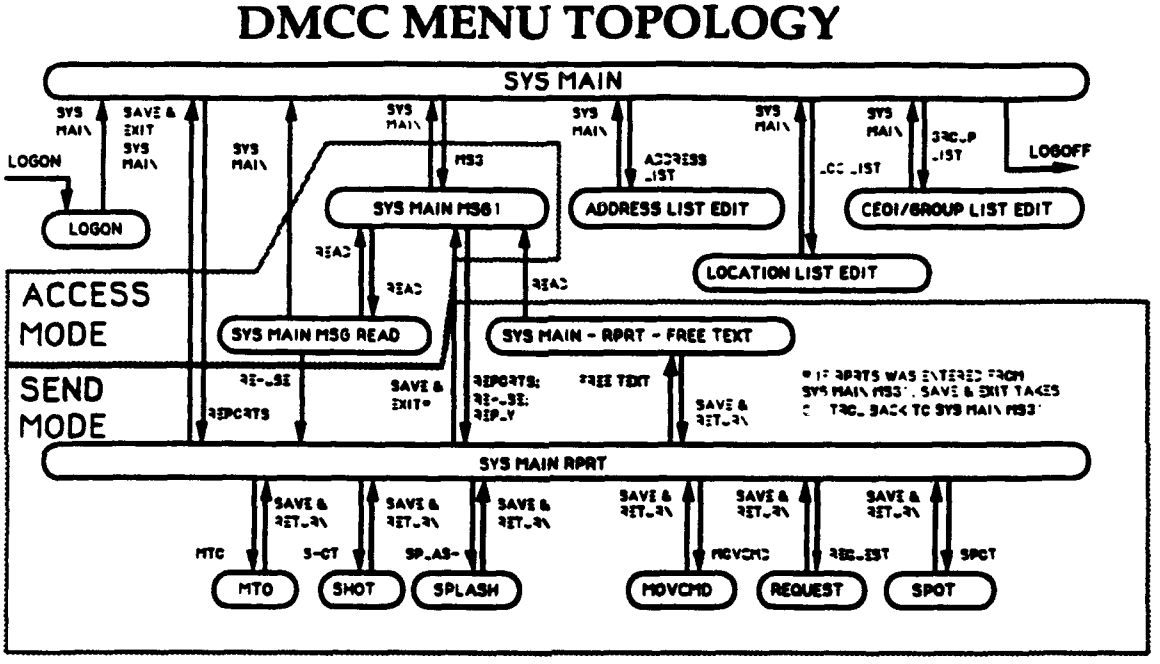
In this test, two host platforms are used to test message interoperability on the actual SIMNET network. The Datalogger is used to capture the session for playback.

User Interface Tests

50.7. Window Content Verification Test

Purpose:

The purpose of this test is to verify that each window in the DMCC client application contains the widgets it needs to do its job.



Background:

The Digital Message Communications Console software uses a OSF/Motif Graphical User Interface running in a X-Windows windowing environment under SunOS, Sun Microsystems UNIX operating system. The Graphical User Interface consists of a number of Windows which contain buttons for clicking with the mouse, areas for entry of text, lists of names, addresses and locations, and graphical objects designed to simulate to some extent the appearance of the Communications Subsystem of the RAH-66 Comanche Helicopter Mission Equipment Package. These objects are called WIDGETS in the vernacular of the OSF/Motif Graphical User Interface.

Acceptance Criteria:

This test shall be deemed to have been passed if all required Widgets are present in all of the DMCC GUI windows.

Test Description:

This test simply checks each of the 13 screens in the DMCC GUI for the correct widget content.

Note:

It should be noted that there do exist graphical icons and features in some windows which do not have functionality in the context of the Digital Message Communications Console software. These are provided to increase the level of fidelity of the simulation of the Comanche Mission Equipment Package. Those objects are not included in the Window Widget List in appendix WW.

Test Procedure:

1. Start the DMCC software using the DMCC start procedure.

2. Examine the logon screen for the proper widget content. The widget content of the logon screen and the other screens in the DMCC is found in the DMCC Graphical User Interface Widget Content List, which is an appendix to this document. **Check that each widget identified in the Widget List is present in this window.**

3. Log a client on to the Message Server under the client name HOMER and the exercise ID 1.

4. Examine the Sys Main screen for widget content.

5. Transfer to the CEOI/Group List window and examine it for widget content.

6. Transfer back to the Sys Main window then to the Location List Window. Examine the Location List Window for widget content.

7. Transfer back to the Sys Main Window and then transfer to the Address List Window. Examine the Address list window for widget content. Enter HOMER and click the ADD button.

8. Transfer back to the Sys Main Window and then transfer to the MSGS window. Examine the MSGS window for widget content.

9. Transfer to the Reports window and exam it for widget content.

10. Transfer to the MTO window and examine it for widget content.

11. Transfer back to the REPORTS window and then to the SHOT window. Examine the SHOT window for widget content.

12. Transfer back to the REPORTS window and then to the SPLASH window. Examine the SPLASH window for widget content.

13. Transfer back to the REPORTS window and then to the MOVCMD window. Examine the MOVCMD window for widget content.

14. Transfer back to the REPORTS window and then to the REQUEST window. Examine the REQUEST window for widget content.

15. Transfer back to the REPROTS window and then to the SPOT window. Examine the SPOT window for widget content.

16. Transfer back to the REPORTS window and then to the FREE TEXT window. Examine the FREE TEXT window for widget content.

17. Enter the a free text message, select HOMER as the addressee, and click SEND. This will cause the client to send the free text message to itself, so you can select it for viewing.

18. Wait for 5 seconds.

19. Click the READ button to get back to the MSG window. Select the message which was just received, and press click READ again.

20. Examine the READ window for widget content.

21. Click SYS MAIN to get back to the SYS MAIN window.

22. Click LOGOFF to Log off.

50.8. Graphical User Interface Window Navigation Test

Purpose

The purpose of this test is to ensure that all window navigation pathways exist and are operational.

Background

The DMCC uses a X-Windowing system, with various X-Windows representing various "screens" or "menus" in the Comanche Mission Equipment Package. Control is passed from one window to another upon certain actions by the user.

Acceptance Criteria

Acceptance criteria for this test are that all window navigation pathways exist and are operational.

Test Description

This test involves starting a DMCC client process, logging it on to the Server, and moving about the window structure to verify each pathway.

Test Procedure

Start the DMCC using the DMCC Start Procedure. Logon a client as **MARGE**. The following paragraphs describe pathways between windows. Check that each pathway exists and is operational.

Click ADDRESS LIST and check that control passes to the ADDRESS LIST EDIT MENU.

Click SYS MAIN and verify that control passes to the SYS MAIN MENU window.

Click LOC LIST and verify that control passes to the LOCATION LIST EDIT window.

Click SYS MAIN and verify that control passes to the SYS MAIN window.

Click CEOI/GROUP LIST and verify that control passes to the CEOI/GROUP LIST EDIT window.

Click SYS MAIN and verify that control passes to the SYS MAIN window.

Click MSGS and verify that control passes to the MSGS window.

Click SYS MAIN and verify that control passes to the SYS MAIN window.

Click MSGS and verify that control passes to the MSGS window.

Click RPRTS and verify that control passes to the RPRTS window.

Click MTO and verify that control passes to the MTO window.

Click SAVE & RETURN and verify that control passes to the RPRTS window.

Click MTO and verify that control passes to the MTO window.

Click CLEAR & RETURN and verify that control passes to the RPRTS window.

Click SHOT and verify that control passes to the SHOT window.

Click SAVE & RETURN and verify that control passes to the RPRTS window.

Click SHOT and verify that control passes to the SHOT window.

Click CLEAR & RETURN and verify that control passes to the RPRTS window.

Click SPLASH and verify that control passes to the SPLASH window.

Click SAVE & RETURN and verify that control passes to the RPRTS window.

Click SPLASH and verify that control passes to the SPLASH window.

Click CLEAR & RETURN and verify that control passes to the RPRTS window.

Click MOVCMD and verify that control passes tot the MOVCMD window.

Click SAVE & RETURN and verify that control passes to the RPRTS window.

Click MOVCMD and verify that control passes to the MOVCMD window.

Click CLEAR & RETURN and verify that control passes to the RPRTS window.

Click REQUEST and verify that control passes to the REQUEST window.

Click SAVE & RETURN and verify that control passes to the RPRTS window.

Click REQUEST and verify that control passes to the REQUEST window.

Click CLEAR & RETURN and verify that control passes to the RPRTS window.

Click SPOT and verify that control passes to the SPOT window.

Click SAVE & RETURN and verify that control passes to the RPRTS window.

Click SPOT and verify that control passes to the SPOT window.

Click CLEAR & RETURN and verify that control passes to the RPRTS window.

Click FREE TEXT and verify that control passes to the FREE TEXT window.

Click SAVE & RETURN and verify that control passes to the RPRTS window.

Click FREE TEXT and verify that control passes to the FREE TEXT window.

Click CLEAR & RETURN and verify that control passes to the RPRTS window.

Click SYS MAIN and verify that control passes to the SYS MAIN window.

Click MSGS and verify that control passes to the MSGS window.

Click REPORTS and verify that control passes to the RPRTS window.

Click MTO and verify that control passes to the MTO window.

Click SYS MAIN and verify that control passes to the SYS MAIN window.

Click MSGS and verify that control passes to the MSGS window.

Click RPRTS and verify that control passes to the RPRTS window.

Click SHOT and verify that control passes to me SHOT window.

Click SYS MAIN and verify that control passes to the SYS MAIN window.

Click MSGS and verify that control passes to the MSGS window.

Click RPRTS and verify that control passes to the RPRTS window.

Click SPLASH and verify that control passes to the SPLASH window.

Click SYS MAIN and verify that control passes to the SYS MAIN window.

Click MSGS and verify that control passes to the MSGS window.

Click RPRTS and verify that control passes to the RPRTS window.

Click MOVCMD and verify that control passes to the MOVCMD window.

Click SYS MAIN and verify that control passes to the SYS MAIN window.

Click MSGS and verify that control passes to the MSGS window.

Click RPRTS and verify that control passes to the RPRTS window.

Click REQUEST and verify that control passes to the REQUEST window.

Click SYS MAIN and verify that control passes to the SYS MAIN window.

Click MSGS and verify that control passes to the MSGS window.

Click RPRTS and verify that control passes to the RPRTS window.

Click SPOT and verify that control passes to the SPOT window.

Click SYS MAIN and verify that control passes to the SYS MAIN window.

Click MSGS and verify that control passes to the MSGS window.

Click RPRTS and verity that control passes to the RPRTS window.

Click FREE TEXT and verify that control passes to the FREE TEXT window.

Click SYS MAIN and verify that control passes to the SYS MAIN window.

50.9. Re-use and Reply; Acknowledge Tests

Purpose:

Verify proper operation of the Reuse (forwarding) and Reply functions in the DMCC Client.

Background:

In the Digital Message Communications System, a mechanism is provided to allow messages which are received to be forwarded to other DMCC entities. This is done by allowing the operator to exit the MESSAGES screen directly to the REPORTS menu, where he can prepare another message to accompany the forwarded message as it is sent on.

The DMCC also allows the user to REPLY to a message directly without having to select the addressee.

The DMCC software is sends an ACKNOWLEDGE PDU to the sender of messages which are read by the user. This acknowledge PDU causes an ACKNOWLEDGE pop-up window to be displayed on the sender's terminal.

Test Description:

This three client test will demonstrate proper operation of the Re-use (Forward) function by sending a message from one client to a second client, and then having the second client forward that message, along with another message, to a third client.

Several REPLY messages will be sent as well.

Checks for proper transmission and reception of Acknowledge PDUs will be undertaken as well.

Acceptance Criteria:

This test will be deemed to have been passed if messages are properly forwarded and replied to, and if ACKNOWLEDGE messages are sent and received correctly.

Test Procedure:

1. Start the DMCC software using the START DMCC procedure.

2. Log on to the Server as TOC.

3. In the TOC client, enter the ADDRESS LIST EDIT window and add the CEOI "LEADER" to the list. Return to the SYS MAIN window.

4. Start a second client and log it on to the server as SANTA. Add the names TOC and WINGMAN to LEADER's address list.

5. Start a third client named WINGMAN.

6. Go back to the TOC client and click MSGS to enter the Messages window.

7. Click RPRTS to enter the REPORTS window.

8. Click FREE TEXT to enter the FREE TEXT window.

9: Enter the following free text message:

PROCEED WAYPT 4 AWAIT BEGINNING OF HF JAMMING.

10. Send the message to LEADER.

11. Now go to the LEADER client and access the message sent by the TOC client.

12. Dismiss the message window. Make sure the message is still highlighted in the mailbox.

13. Click REPLY.

14. In the REPORTS screen, select FREE TEXT .

15. Go back to the TOC client and see if an acknowledge PDU was received from LEADER.

16. Go to the LEADER client and enter the following FREE TEXT message:

WILL COMPLY. WILL AWAIT FRIENDLY MIJI ACTIVITY

16. Click Send Routine.

17. Go to the TOC client and verify that the reply message has arrived.

18. Go to the mailbox and read the reply message.

19. Dismiss the reply message.

20. Click REPLY, and go to the FREE TEXT REPORT screen.

21. Enter the following free text message:

IGNORE MEACONING ON 1022 MHZ

22. Click SEND URGENT.

23. Go to the LEADER client and access the message from TOC.

24. Go to TOC and see if an ACK was received from LEADER.

25. Go to LEADER. Read the message, then dismiss it.

25. Click REPLY

26. Go to free text and enter the following message:

WILL AWAIT JAMMING; IGNORE MEACONING

27. Click Send Routine.

28. The message from TOC should still be highlighted in the mailbox.

29. Click RE-USE.

30. Go to Free Text.

31. Enter the following Free Text Message:

WINGMAN, PLEASE SEE ENCLOSED AND COMMENT ASAP. STATUS OF RADIOS. JAMMING TO BEGIN 1125.

32. Click SEND URGENT.

33. Go to the WINGMAN client and see if the messages arrived. There should be the original message from the TOC client, plus the new one from LEADER.

34. Access and read the messages.

35. Go back to LEADER and see if LEADER got ACKS back for EACH of the two messages.

50.10. Miscellaneous Functional Tests

Purpose:

The purpose of this test is to verify miscellaneous operational features of the Digital Message Communications Console software. These include:

Message Deletion
Client Death
Message Prioritization
Operator errors entering time
Operator errors entering UTMs
Escape key handling
Control key handling
Reception of messages during report preparation
Reception of messages while an alert box is active
Case insensitivity of target CEOI/Group Names
Default Addressing for Reply
Empty Message Send Handling

50.11. Functional Stress Tests

Message Server Message Queue Overload
Shared Memory Overload
Client Message Queue Overload
High Fan Out Messages
Free Text String Overload
Free Text Annotation Overload
Eight Client Test
Alternate Group Name Message Routing

50.12 Functional Test... Window Widgets

DMCC Window Widget Content List

All DMCC windows contain the Free Text field, simulate Cockpit Interactive Keyboard buttons (left arrow, right arrow, DEL and CLEAR ALL. The following paragraphs detail the window widgets in addition to these, for each window type.

LOGON WINDOW:

- LOGON TO NETWORK button
- STAND ALONE button
- LOGON NAME Text Field
- EXERCISE ID Text Field

SYS MAIN WINDOW

- ADDRESS LIST button
- LOCATION LIST button
- CEOI/GROUP LIST BUTTON
- LOGOUT BUTTON
- MSGS BUTTON
- RPRT BUTTON
- SYS MAIN BUTTON

ADDRESS LIST EDIT WINDOW

- Add button
- Delete button
- Save & Exit button
- Sys Main button
- UP button
- DOWN button
- New Address Text Field
- Address List

LOCATION LIST EDIT WINDOW

- ADD button
- DELETE button

SAVE & EXIT button
SYS MAIN button
UP button
DOWN button
New Location Text Field
Location List

CEOI/GROUP LIST EDIT WINDOW

ADD button
DELETE button
SAVE & EXIT button
SYS MAIN button
UP button
DOWN button
NEW CALL SIGN or GROUP NAME text field
Group List

SYS MAIN MSG1 WINDOW

Message Queue List
PREV button
NEXT button
RPRT button
READ button
DELETE button
RE USE button
REUSE & INCLUDE button
REPLY button
Envelope Icon
Envelope Icon Number
SAVE & EXIT button

SYS MAIN MSG READ WINDOW

READ button
DELETE button
SYS MAIN button
NEW MESSAGE display

SYS MAIN REPORTS WINDOW

SPOT button
MTO button
SHOT button
SPLASH button

FREE TXT button
REQT button
MOVCMD button
SYS MAIN button

MTO WINDOW

ADRS selection ribbon & button
REQ ADJ button
EAT button
EOM button
SAVE & RETURN button
CLEAR & RETURN button
SND * ROUT button
SND * URG button

* Note: for the MTO window, the SND ROUT and SND URG buttons will contain the text of what kind of MTO message has been selected for transmission. For example, if the REQ ADJ button has been selected, the buttons will say SND ROUT REQ ADJ and SND URG REQ ADJ.

SHOT WINDOW

ADRS selection ribbon & button
FREE TEXT annotation field
SAVE & RETURN button
CLEAR & RETURN button
SND ROUT button
SND URG button
SYS MAIN button

SPLASH WINDOW

ADRS selection ribbon & button
FREE TEXT annotation field
SAVE & RETURN button
CLEAR & RETURN button
SND ROUT button
SND URG button
SYS MAIN button

MOVCMD WINDOW

TASK selection ribbon & button
WHEN selection ribbon & button
LCTN selection ribbon & button
ADRS selection ribbon & button
FREE TEXT annotation field
SAVE & RETURN button
CLEAR & RETURN button
SND ROUT button
SND URG button
SYS MAIN button

REQUEST WINDOW

TYPE selection ribbon & button
RECON TYPE selection ribbon & button
ADRS selection ribbon & button
FREE TEXT annotation field
SAVE & RETURN button
CLEAR & RETURN button
SND ROUT button
SND URG button
SYS MAIN button

SPOT WINDOW

ENEMY TYPE selection ribbon
ENEMY TYPE button
ENEMY ACTIVITY selection ribbon
DIREC seletion ribbon
DIRC button
OBS INT selection field
OBS INT button
SPEED text entry field
NUM text entry field
UNIT text entry field
FREE TEXT ANNOTATION field
ADRS selection ribbon & button
SAVE & RETURN button
CLEAR & RETURN button
SND ROUT button
SND URG button
SYS MAIN button

FREE TEXT WINDOW.

FULL FREE TEXT field
FREE TEXT ANNOTATION field
ADRS selection ribbon & button
SAVE & RETURN button
CLEAR & RETURN button
SND ROUT button
SND URG button
SYS MAIN button

REUSE WINDOW

ADRS selection ribbon & button
FREE TEXT annotation field
SAVE & RETURN button
CLEAR & RETURN button
SND ROUT button
SND URG button
SYS MAIN button

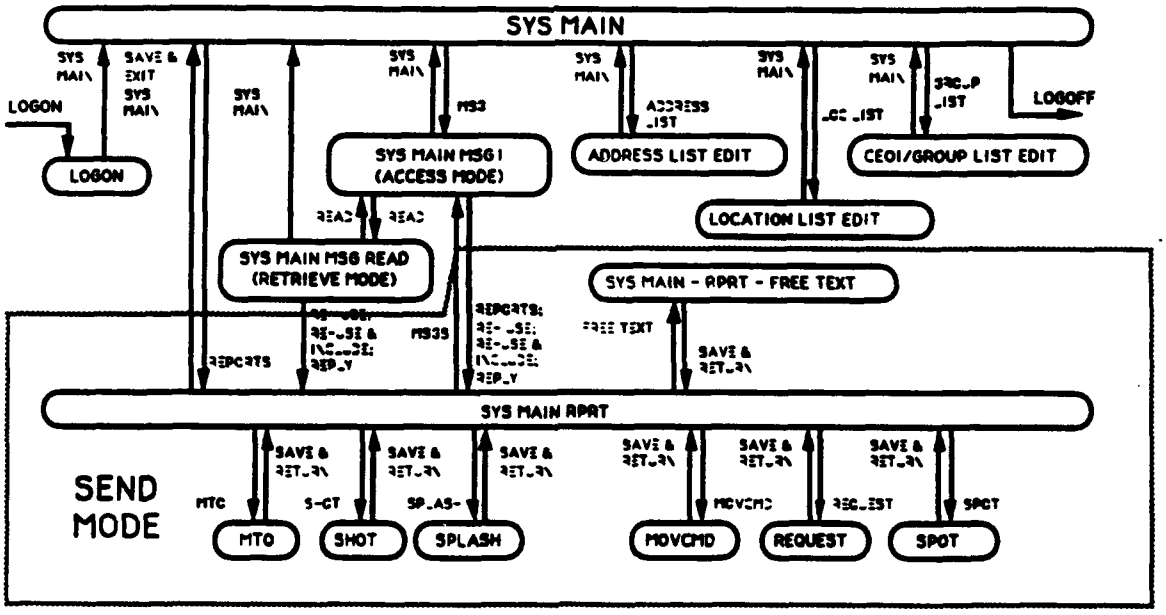
60. Appendix F - DMCC Graphical User Interface.

OPERATIONAL SPECIFICATION
FOR THE
AIRNET DMCC GRAPHICAL USER INTERFACE (GUI)

60.1 Purpose and Scope:

The AIRNET Digital Message Communications Console uses a OSF/Motif® Graphical User Interface running in an X-Windows/UNIX environment on a Sun Microsystems dedicated workstation or client-server X-terminal platform to closely mimic the operation of the messaging subsystem of the Mission Equipment Package on the RAH-66 Comanche reconnaissance/attack helicopter. This interface uses Motif Widgets and icons to effect the simulation of displays and buttons on the System Management Display (SMD), Touchscreen Menu Items (TMI), and Cockpit Interactive Keyboard (CIK) of the Mission Equipment Package. A mouse input device is used to manipulate a mouse pointer on the DMCC workstation or X-terminal display, and to simulate depressing cockpit switches through "clicking" on the screen icons which represent those switches. This document specifies the operation of this Graphical User Interface in the context of the overall DMCC software system specification.

DMCC WINDOW TOPOLOGY



In this document, **Displays, Switches and Icons** are identified in **THIS COURIER FONT.**

60.2 General Description of Operation:

60.2.1 Interface Elements

The accessing of incoming messages and the control of outgoing messages on the AIRNET Digital Message Communications Console (DMCC) is accomplished by depressing simulated cockpit **Touchscreen Menu Items (TMI)**, activation of **Soft Bezel Switches (SBSs)** on the simulated **System Management Display (SMD)**, and by entering information in the simulated **Cockpit Interactive Keyboard (CIK)**. The limited CIK will be simulated by using the workstation keyboard with a small window on the DMCC Console Display simulating the CIK display. The X-terminal QWERTY keyboard will be used for entry of textual information.

60.2.2 Menu Windows

Fourteen menus will be used by the DMCC GUI:

LOGON
SYS MAIN
ADDRESS LIST EDIT
CEOI/GROUP LIST EDIT
MSG1
MSG READ
RPRT
FREE TEXT
SPOT
REQUEST
MOVCMD
SPLASH
SHOT
MTO

60.3 DMCC Incoming Message Control

60.3.1 Overview

The DMCC provides a EMAIL-like interface for the listing, access, forwarding, and acknowledgment of incoming Digital Message Communications. Control of the incoming messages is accomplished "in" a number of screen menus which are described in the sections covering the **SYS MAIN** and **SYS MAIN MSG** menu pages.

60.3.2 Call Sign

Each Digital Message/Communications Entity is given a call sign by its operator.

60.3.3 Groups

Each entity may optionally specify that it belongs to up to 7 entity groups.

60.3.4 Addresses

Each Message includes a list of DMC entities (up to 8 entities, identified by their call signs, and/or groups, identified by their group names) which are the addressees or target recipient entities or groups.

60.3.5 Reception

Each DMC entity can receive only those messages transmitted to it; i.e., messages which are not addressed to its call sign or to a group of which it is a member are ignored by the entity. No alert or acknowledgment of such messages occurs.

60.3.6 TOC - to - TOC Communications

Thus, for TOC to TOC communications, the TOC which wants to transmit a message to the other TOC must include that TOC's call sign (or the name of a group of which the target TOC is a member) in the list of target recipients.

60.3.7 Message Reception

When a digital message arrives at a DMC entity which is addressed to that entity or is addressed to a group of which that entity claims membership, then that message is received and stored and an alert box is displayed telling the origin of the message and its type (see Sys Main, below).

60.3.8 Sender ID & Time Stamp

Each transmitted message is marked with the CEOI (Computer Electronic Operations Instructions) or **CALL SIGN** of the transmitting entity and the Date Time Group which tells the time of message transmission.

60.3.9 Acknowledge

Upon SELECTION of a message, the DMCC will transmit an Acknowledge message to the sender, which includes the DTG of the message. Acknowledge messages themselves are not acknowledged.

60.3.10 The Alert Box

An **ALERT BOX** will appear on the transmitter's screen which says "**MESSAGE ACKNOWLEDGED BY <CEOI>**". Note that the acknowledger ID is not the **GROUP** to which the message was sent, but the individual **CALL SIGN** of the acknowledger. This alert box can then be dismissed by an **OK** button.

60.3.11 Message Forwarding (RE_USE):

When a message is selected from the **MAILBOX**, a **RE-USE TMI** will become active. This switch will allow the operator to tranfer to the **REPORTS** menu, where he will be able to generate a new message of preformatted or free-text type, to be sent to the forward recipient (the person the message is being forwarded to) of the original message, along with the original message.

60.3.12 Reply

When a message is selected from the **MAILBOX**, a **REPLY TMI** will become active. This switch will allow the operator to tranfer to the **REPORTS** menu, where he will be able to generate a new message of preformatted or free-text type, to be sent to the sender of the original message. The **ADRS** bezel will already have selected the sender of the first message as the addressee of the new message to be sent back.

60.4 DMCC Outgoing Message Control (REPORTS)

60.4.1 Available Reports

Available reports shall be presented and accessed via the soft bezel switches. For all types of reports transmitted by the DMCC, the DMCC shall append information to the report selected by the operator. These items include: the sender's identity, the Date Time Group (DTG) telling the date and time the report was sent, and where the sending DMCC was when the report was sent (TOC, FSE, or BATTLEMASTER). Although these pieces of information shall not be seen by the operator of the DMCC, they shall appear on the report received by the addressee individual or group.

60.4.2 Reports Function

The REPORTS function is the means by which the DMCC operator constructs and sends messages out to other DMC entities. The MESSAGES function handles incoming digital message traffic and was discussed above.

60.4.3 Accessing the Reports Window

The standard way of entering the SYS MAIN RPRT menu is by using the REPORTS soft bezel. SYS MAIN RPRT may be exited by pressing the SAVE & EXIT TMI. There are pre-formatted reports, plus a free text report and a "REQUEST REPORT".

60.4.4 Menu Wundows

The following pages describe in detail the operation of each of the MENUS which can appear in the shell window widget resembling the RAH-66 SMD, TMI and CIK.

60.4.5 Individual Reports Windows

In this implementation, seven report message types will be sent from the DMCC to the helicopter simulation entities, including:

SPOT
MTO
SHOT
SPLASH
REQUEST
MOVCMD
FREE TEXT

TYPES OF REPORTS TRANSMITTED BY THE DMCC

60.4.6 CIK MENU

The CIK allows the operator to enter alphanumeric data into the system. on board the actual Comanche aircraft. The functionality of the CIK in the Digital Message/Communications Console will be limited to simulation of the CIK's display on the screen of the X-terminal, but the entry of actual alphanumeric information into the DMCC will take place though the QWERTY keyboard of the X-term. Four other keys on the CIK are implemented, the left and right arrow, the Enter key, and CLEAR ALL.

60.4.7 LOGON

The LOGON menu will be the inital menu seen by the user upon starting the DMCC software. It will prompt the user for the CEOI (an eight character alphanumeric call sign).

60.4.7.1 LOGON TO NETWORK

The LOGON TO NETWORK SMD bezel on this screen will allow the user to logon to the Digital Message Communications System. The program will wait for a period of time

60.4.7.2 STAND ALONE OPERATION

This bezel will allow the user to turn on the DMCC client except that it will not transmit or receive messages.

60.4.7.3 SYS MAIN Window

After the user has selected one of the modes of operation, control will pass back to the SYS MAIN screen.

60.4.8 ADDRESS LIST EDIT

The ADDRESS LIST is the list of CALL SIGNS and GROUPS to which the operator can *send* messages from the various reports screens.

60.4.8.1 Call Signs, Group Names

It contains 8 possible CALL SIGNS (8 character alphanumeric strings) and GROUP NAMES (also 8 character alphanumeric strings).

60.4.8.2 Initial Members

Initially, the list will have no members. Up to 8 members can be added to the list.

60.4.8.3 ADDRESS LIST

The LIST is a round-robin tabulated menu of the current GROUPS and CALL SIGNS to which the console can send messages.

60.4.8.4 Adding and Deleting Addresses

This menu allows the operator to ADD and DELETE various CALL SIGNS and GROUP NAMES from this list, using the TEXT WIDGET, the ADD key, and the DELETE key.

60.4.8.5 ADDRESS TEXT WIDGET

The text widget is used for entering addresses for addition to the list.

60.4.8.6 SIZING

An 8 character text widget allows the entry of new CALL SIGNS or GROUP NAMES . Its contents are cleared when the ADD function is activated.

60.4.8.7 UP BUTTON

The UP button moves the highlight up the LIST, round-robining to the bottom when the top is reached. If there is 0 or 1 CALL SIGN or GROUP NAME in the list, it does nothing.

60.4.8.8 DOWN BUTTON

The DOWN button moves the highlight down the LIST, round-robining to the top when the bottom is reached. If there is 0 or 1 CALL SIGN or GROUP NAME in the list, it does nothing.

60.4.8.9 ADD BUTTON

The ADD button adds the new CALL SIGN or GROUP NAME from the TEXT WIDGET to the LIST which is displayed in the round-robin selectable list. The ADD button will only add a text string to the list if the string contains a character other than the space character or the null character.

60.4.8.10 DEL BUTTON

The DEL key deletes the selected CALL SIGN or GROUP NAME from the ADDRESS LIST.

60.4.9 CEOI/GROUP LIST EDIT

The **GROUP LIST** is the list containing the individual **CEOI** or **CALL SIGN** of the operator, plus up to 7 **GROUP NAMES** for groups of which the operator is a member (both are 8 character alphanumeric strings).

60.4.9.1 MESSAGE RECEPTION

The station will *receive* messages addressed to the **CALL SIGN** of the station and messages sent to groups to which the operator claims membership.

60.4.9.2 EDITING THE LIST

This screen allows the operator to edit the list of **CALL SIGNS** and **GROUP NAMES** for messages he or she wishes to *receive*.

60.4.9.3 FUNCTIONAL DESCRIPTION

This screen is functionally identical to the **ADDRESS LIST EDIT** menu, except that it manages a separate list of names.

60.4.10 LOCATION LIST EDIT

The **LOCATION LIST** is the list containing alphanumeric codes of up to 8 characters for up to 8 battlefield locations for use in the **MOVCMD** report.

60.4.10.1 FUNCTIONAL DESCRIPTION

This screen is functionally identical to the **ADDRESS LIST EDIT** menu, except that it manages a separate list of names.

60.4.11 SYS MAIN

60.4.11.1 NEW MESSAGE Indicator

When the DMCC receives a new message, an alert box will pop up on the screen and will announce the type, sender and priority of the incoming message. The warnings will be different depending on the priority of the message. The alert box will be dismissed by the operator by clicking a box in the alert labelled DISMISS. The MESGS indicator will show the number of messages (that have not been read) in the IN-BOX , the type of message just received (according the

table immediately below this paragraph) and its priority (U for Urgent, R for Routine). The MESGS indicator is the number which appears in the envelope.

Receive Message Types	Message Identifier
SPOT	SPT
STATUS	STA
MOVEMENT	MVT
BDA	BDA
LZ/PZ RECON	LZR
AIR ROUTE RECON	ARR
GROUND ROUTE RECON	GRR
BP/OP RECON	BPR
CROSSING RECON	CRR
BRIDGE RECON	BRR
FREE TEXT	FTX
ARTY REPEAT	ARP
ARTY CANCEL	ACL
ARTY CHECK	ACH
ARTY CNO	ACN
ARTY SHIFT	ASH
ARTY NEW MSN	ANM
ARTY END MSN	AEM
REQUEST REPORT	RQR
NBC	NBC
SPLASH	SPS
MIJI	MIJ
NBC	NBC
PIREP	PRP
DNAV	DNV

60.4.11.2 SYS MAIN TMI BUTTONS

The SYS MAIN menu will have TMI buttons to transfer control to the ADDRESS LIST screen and the CEOI/GROUP LIST screen. It will have a soft bezel to transfer control to the SYS MAIN MSG screen.

It will also contain a button to LOGOFF.

60.5 SYS MAIN MSG1 Window

When the DMCC operator wants to read a message he or she has just received, or dispose of a message he or she has just read, or do something with a message he or she received some time ago and had kept, the operator will start with the **MESSAGES MAIN MENU.**

60.5.1 THE IN-BOX

A numbered list of all the messages in the **IN-BOX** shall be displayed, with the message type, the sender, the time the message was sent, and the priority given for each message. The list shall be arranged first by priority and second by time of message receipt, with the most urgent messages appearing in a group at the top of the list. Urgent messages shall be marked with a **U**, while routine priority messages shall be marked with an **R**. Message priority shall be determined by the sender of the message. Upon entering the **MESSAGES** main menu, the top message in the list shall be highlighted.

60.5.2 ENVELOPE ICON

There shall be an **ENVELOPE ICON** at the bottom of the simulated SMD window that contains the **MSGS** indicator which indicates the number of new messages **that have not been read.** This icon shall appear when a new message is received by the DMCC. When a message has been read but has not been disposed of, it shall still exist as an active message in the **IN-BOX** but will not be reflected in the number on the envelope.

60.5.3 MSG READ

Pressing **READ** shall change the window from the list of available messages to the display of the contents of the highlighted message in the Sys Main Mesg Read screen. When **READ** is ON, a second press of the **READ** button will dismiss the contents and return to the list of messages available. This option shall only be available if messages are available to be acted upon. The second line of the **CIK** shall be blank until **READ** is on at which time the second line shall read "**ON**".

60.5.4 Message Display Formats

Display formats for the information contained in the various reports are textual information, displayed in the **SYS MAIN MSG READ** submenu. They are found at the end of this document.

60.5.5 PRE MSG and NEXT MSG Bezels

If the operator decides to read a message in the **IN BOX**, he or she will use the **PRE MSG** and the **NEXT MSG** bezels to move the highlight onto the desired message, then will press **READ MSG**. Upon entering the **READ MSG** function the contents of the message highlighted on the **MESSAGES** main menu are displayed on the screen.

60.5.6 DELETE MSG TMI

Pressing the **DELETE MSG TMI** shall delete a message from the **IN-BOX** . This option shall only be available if messages are available to be acted upon.

60.5.7 SAVE & EXIT

The **SAVE & EXIT** switch shall save any changes made while in the **MSG** menu and shall return control to the **SYS MAIN** menu.

60.5.8 RE-USE

Pressing the **RE-USE TMI** shall allow the operator to forward the same message to someone else. Control will transfer to the **REPORTS** menu, from where a new message may be appended to the original message.

60.5.9 REPLY

When a message is selected from the **MAILBOX**, a **REPLY TMI** will become active. This switch will allow the operator to tranfer to the **REPORTS** menu, where he will be able to generate a new message of preformatted or free-text type, to be sent to the sender of the original message. The **ADRS** bezel will already have selected the sender of the first message as the addressee of the new message to be sent back.

60.5.10 PREV, NEXT

Pressing the **PREV** bezel and the **NEXT** bezel shall move the highlight up and down the list of messages in the **IN-BOX** to select a message on which to perform some action. The highlight shall jump to the top message when the **NEXT** bezel is pressed and the highlight is on the bottom most message. Likewise, when the topmost message is highlighted and the **PREV** bezel is pressed, the highlight shall move to the bottom-most item on the list. This option shall only be available if messages are available to be acted upon.

60.6 RPRT Windows

The **RPRT MENU** is the master menu for sending messages (called REPORTS) from the DMCC to the other DMC-capable simulation entities in the AIRNET simulation facility or elsewhere in the network. The following paragraphs describe the various widgets in this menu and their uses and effects.

60.6.1 SAVE & EXIT

This switch shall save any changes made while in the RPRT MENU and shall return to the SYS MAIN MENU or whatever function was being used when RPRT was invoked.

60.6.2 SHOT

Upon selection of this switch, the system shall display the **SHOT TMI's** and **SMD** page. (see below).

60.6.3 SPLASH

Upon selection of this switch, the system shall display the **SPLASH TMIs** and **SMD** page (see below).

60.6.4 MTO

Upon selection of this switch, the system shall display the **MTO TMIs and SMD** page (see below).

60.6.5 SPOT

The SPOT report is implemented in the DMCC even though no ground person is anticipated to want to send a **SPOT** report. This is to allow the assessment of the DMCC archetecture for in-cab simulated communications, and to provide a simple means of integration testing. Upon selection of this switch, the system shall display the **SPOT TMIs and SMD** page (see below).

60.6.6 MOVCMD

Upon selection of this switch the DMCC shall display the **MOVCMD TMIs** and the **MOVCMD SMD** page.

60.6.7 REQT

Upon selection of this **REQUEST** switch the DMCC shall display the **RPRT REQT TMIs** and the **REQT SMD** page.

60.6.8 FREE TXT

Upon selection of this switch the DMCC shall display the **RPRT FREE TXT TMIs** and the **FREE TXT SMD** page.

60.7 RPRT Window Functionality

The following sections detail the screens which are used to send messages from the DMCC. Each screen shares the following features:

60.7.1 COMMON RPRTS WINDOW FUNCTIONs

60.7.1.1 ADRS

Once the screen function is entered, the DMCC operator shall select an addressee using the **ADRS** soft bezel and the **MORE** bezel. This will take the form of a tape type list of 6 possible addressees. The addressees will be listed in boxes on this tape, and the operator will use the **ADRS** key to cycle a highlight along this tape., wiht the highlight round robining to the beginning when the key is pressed while the highlight is at the end. The **MORE** key will be used to display the remaining addressees if there are more than 6, and will also be used to go back to the first set of selections when the remaining ones are being displayed.

60.7.1.2 TAPE WIDGET

This round-robin tape widget will be used to perform screen specific choices also, as outlined below.

60.7.1.3 TEXT MSG

Pressing the **TEXT MSG TMI** shall allow the DMCC operator to add any comments to the report, by entering it on the **KEYBOARD** and pressing the **KEYBOARD RETURN** key or clicking on the **ENTER** key in the **CIK**. If no text message is entered, the text message display area of the screen shall remain blank.

60.7.1.4 SEND ROUTIN

The **SEND ROUTIN TMI** shall operate as described in the **SPOT REPORT** section.

60.7.1.5 SEND URGENT

The **SEND URGENT TMI** shall operate as described in the **SPOT REPORT** section.

60.7.1.6 CLR & RETURN

The **CLR & RETURN TMI** shall terminate the report function without saving any changes made and the display shall return to the **SYS MAIN RPRT** menu.

60.7.1.7 SAVE & RETURN

Pressing the **SAVE & RETURN** shall save any changes made while in the current function and the display shall return to the **RPRT** menu.

60.7.1.8 ENUMERATED TYPES DEFAULT VALUE

If a selection is not made for any of the selection fields in any of the reports, then no highlight will appear on the SMD. (In this case, enumerated type 0 for that type will be transmitted, meaning NOT ENTERED. In the case where a message is received containing NOT ENTERED for a field, the message read display for that field shall be blank.

60.7.2 SPOT REPORT

The SPOT menu will have round-robin highlight menu choices for the following:

ENEMY TYPE (ADA, SAM, TANK, WHLD, TRKD, ACFT, TRPS, UNCL)

ENEMY ACTIVITY (STATIONARY, MOVING, DUG-IN, RETREATING, ATTACKING DMGD, KILLED, MOBIL_KILL);

DIRECTION (N, NE, E, SE, S, SW, W, NW);

OBS INT (CONT_MSN, ENGA, RTS IHLD, OBSV TGT).

60.7.2.1 Enemy Speed

Text fields for entry of enemy **SPEED** (3 char) and **UNIT** (8 char) will be provided.

60.7.2.2 Text Msg TMI

The **TEXT MSG TMI** will allow text to be entered in the 23 character CIK test widget for editing free text to be included within the message.

60.7.2.3 MPH/KPH Switch

A **TMI** toggle switch shall select between **MPH** and **KPH**.

60.7.2.4 Enemy Type

Each time the **ENEMY TYPE** button is pressed, the GUI software will make a call to the Date Time Group routine and record the **DTG**. This **DTG** represents the **EFFECTIVE TIME** or **TIME SIGHTED**, and will be passed as an argument to the to the **SPOT PDU BUILDER** when **SEND** occurs.

60.7.2.5 Enemy Number

A 3 character decimal integer field shall be provided for entering the **NUMBER** of the enemy.

60.7.2.6 Enemy Location

The lower CIK text widget shall be labeled LOCN and will be used for entry of a UTM coordinate of up to 16 alphanumeric characters.

60.7.3 MTO

The MTO menu has three TMIs to select pre-formatted messages for transmission.

60.7.3.1 REQ ADJ

The REQ ADJ TMI will select a REQUEST ADJUST message for transmission.

60.7.3.2 EAT

The **EAT? TMI** will select an **ENTER AS TARGET?** message for transmission.

60.7.3.3 EOM

The **EOM? TMI** will select an **END OF MISSION?** message for transmission.

60.7.3.4 TEXT MSG

The **TEXT MSG TMI shall operate as described in the SPOT REPORT section.**

60.7.4 SHOT

The **SHOT** report menu involves no buttons, other than the standard ones which all message screens use as described above. It sends the **SHOT** message and then displays the words **SHOT REPORT SENT** for 2 seconds in a small pop up window, and returns to the calling menu.

60.7.5 SPLASH

The **SPLASH** report menu sends the **SPLASH** message and then displays the words **SPLASH REPORT SENT** for 2 seconds in a pop up window, and returns to the calling menu.

60.7.6 MOVCMD

The Comanche Movement report shall be used to direct units to a particular location to perform a particular activity, or to inform others of one's intent to move somewhere. It is intended to replace the traditional verbal or written FRAGO (Fragmentary Order) which has been used to convey timely changes to missions which were previously outlined in OPORDs (Operational Orders). The **MOVCMD** report shall also be used to hand over targets to a team member for servicing when speedy message composition is not a factor.

60.7.6.1 TASK

The operator DMCC shall select in a tape widget the desired movement task using the **TASK** soft bezel and the **MORE** bezel exactly as prescribed for selecting an addressee. The options under **TASK** shall include "**MOV TO**", "**HOLD AT**", "**CONT MSN**", "**RENDZ AT**", "**ENGAGE TGT AT**", "**MVNG TO**", "**HLDNG AT**", "**ARVING AT**", "**PSNG THRU**", and "**DPRTING FROM**".

60.7.6.2 When

Pressing the **WHEN** bezel shall round-robin a highlight among the options given for when the movement will take place. Options shall include "**IMMED**", "**WHEN RDY**", "**AMC**", and "**TIME**". When the highlight is in moved to the last option, "**TIME**", the alphanumeric **KEYBOARD** shall be enabled for text input of a time. The time is entered in the form

hh:mm L

or

hh:mm Z.

L stands for Local Time and Z stands for Zulu, or Greenwich Mean Time, the time at the Prime Meridian in Greenwich, England.

The simulated **CIK** shall display the prompt "**TIME?**" on the first line, with the operator entering the time on the second line.

If the operator entered the time in Local Time, the DMCC must change the time to ZULU time prior to transmission. This is done by accessing a file containing

the ZULU OFFSET, which is the number of minutes Local Time is AHEAD of ZULU time. This file is changed whenever the timezone of the DMCC changes.

60.7.6.3 Location

Pressing the LOCN bezel shall round-robin a highlight among the options given for where the movement will take place. Options shall include "MY POSN", and those positions entered in the location list edit screen.

60.7.7 REQT REPORT

The REQUEST REPORT function shall be used when the DMCC operator wants to ask another DMC entity to send him a specific kind of report. Figure 6.6.5.22 outlines the REQT REPT.

60.7.7.1 Report Type Selection

A round robin horizontal tape selection of the type of report to be requested shall be presented. The options shall include

NONE
STATUS
SPOT
RECON
MOVEMENT
PIREP
MIJI

60.7.7.2 Recon Type Selection

When the Recon report type is selected, another horzontal tape selection for the type of recon report desired. The options shall include:

GND RTE
AIR RTE
BRDG
LZ/PZ
BP/OP
CRSG

60.7.8 FREE TEXT

The FREE TEXT message window will basically be a 256 character editable text widget.

60.8 Message Display Formats

Note: These drawings provide row and column numbers which are not displayed on the actual message read window. Dots are also not displayed; they are provided here for clarity of character position.

SPOT REPORT

```
   0          1          2          3
   01234567890123456789012345678901234

 0 .SPOT MESSAGE......*URGENT*........
 1 ....................................
 2 .SENDER ...........XXXXXXXX........
 3 .SENT TO...........XXXXXXXX........
 4 .FWD BY............XXXXXXXX........
 5 .MSG SENT..........26 1745 JUN 95..
 6 .XMIT LCN.. .......NB 0624 0332....
 7 .XMIT ALT..... ....12000 FEET......
 8   ..... .. ..................
 9   .GT QUAN..........nnnn............
10   .GT TYPE..........TRKD............
11 .TGT ACTIV.........MVNG............
12 .TGT SPEED.........20 MPH..........
13 .TGT DIRN..........NW..............
14 .TGT UNIT..........aaaaaaaaaaaaaaaa
15 .TGT LCN...........NB 2322 2323....
16 .TIME TGT SGHTD....26 1744 JUN 95..
17 .OBS LCN AT CNTCT..NB 1233 4333..
18 .OBS INTENT........ENGA............
19 ....................................
20 .FREE TEXT ANNOTATION•••...........
```

FREE TEXT REPORT

```
    0          1          2          3
    01234567890123456789012345678901234

 0  .FREE TEXT MESSAGE.ROUTINE.........
 1  ....................................
 2  .SENDER ...........XXXXXXXX........
 3  .SENT TO...........XXXXXXXX........
 4  .FWD BY............XXXXXXXX........
 5  .MSG SENT..........26 1745 JUN 95..
 6  .XMIT LCN..........NB 0624 0332....
 7  .XMIT ALT..........1200 FEET.......
 8  ....................................
 9  ...aaaaaaaaaaaaaaaaaaaaaaaa......
10  ...aaaaaaaaaaaaaaaaaaaaaaaa......
11  ...aaaaaaaaaaaaaaaaaaaaaaaa......
12  ...aaaaaaaaaaaaaaaaaaaaaaaa......
13  ...aaaaaaaaaaaaaaaaaaaaaaaa......
14  ...aaaaaaaaaaaaaaaaaaaaaaaa......
15  ...aaaaaaaaaaaaaaaaaaaaaaaa......
16  ...aaaaaaaaaaaaaaaaaaaaaaaa......
17  ...aaaaaaaaaaaaaaaaaaaaaaaa......
18  ...aaaaaaaaaaaaaaaaaaaaaaaa......
19  ....................................
20  .FREE TEXT ANNOTATION•••...........
```

ARTILLERY CANCEL REPORT

```
   0          1          2          3
   01234567890123456789012345678901234

 0 .ARTY CANCEL.RPRT..ROUTINE.........
 1 ...................................
 2 .SENDER ...........XXXXXXXX........
 3 .SENT TO...........XXXXXXXX........
 4 .FWD BY............XXXXXXXX........
 5 .MSG SENT..........26 1745 JUN 95..
 6 .XMIT LCN..........NB 0624 0332....
 7 .XMIT ALT..........1200 FEET.......
 8 ...................................
 9 ...................................
10 ...................................
11 ...................................
12 ...................................
13 ............CANCEL.................
14 ...................................
15 ...................................
16 ...................................
17 ...................................
18 ...................................
19 ...................................
20 .FREE TEXT ANNOTATION•••...........
```

ARTILLERY REPEAT REPORT

```
   0         1         2         3
   01234567890123456789012345678901234

 0 .ARTY REPEAT REPT..ROUTINE.........
 1 .....................................
 2 .SENDER ...........XXXXXXXX........
 3 .SENT TO...........XXXXXXXX........
 4 .FWD BY............XXXXXXXX........
 5 .MSG SENT..........26 1745 JUN 95..
 6 .XMIT LCN..........NB 0624 0332....
 7 .XMIT ALT..........1200 FEET.......
 8 .....................................
 9 .....................................
10 .....................................
11 .....................................
12 .....................................
13 ............REPEAT.................
14 .....................................
15 .....................................
16 .....................................
17 .....................................
18 .....................................
19 .....................................
20 .FREE TEXT ANNOTATION•••...........
```

ARTILLERY CHECK REPORT

```
   0         1         2         3
   01234567890123456789012345678901234

 0 .ARTY CHECK REPORT.ROUTINE.........
 1 .....................................
 2 .SENDER ...........XXXXXXXX........
 3 .SENT TO...........XXXXXXXX........
 4 .FWD BY............XXXXXXXX........
 5 .MSG SENT..........26 1745 JUN 95..
 6 .XMIT LCN..........NB 0624 0332....
 7 .XMIT ALT..........1200 FEET.......
 8 .....................................
 9 .....................................
10 .....................................
11 .....................................
12 .....................................
13 ............CHECK ..................
14 .....................................
15 .....................................
16 .....................................
17 .....................................
18 .....................................
19 .....................................
20 .FREE TEXT ANNOTATION•••...........
```

ARTILLERY CNO REPORT

```
   0          1          2          3
   01234567890123456789012345678901234

 0 .ARTY CNO REPORT...ROUTINE.........
 1 ....................................
 2 .SENDER ...........XXXXXXXX........
 3 .SENT TO...........XXXXXXXX........
 4 .FWD BY............XXXXXXXX........
 5 .MSG SENT..........26 1745 JUN 95..
 6 .XMIT LCN..........NB 0624 0332....
 7 .XMIT ALT..........1200 FEET.......
 8 ....................................
 9 .MISSION ID........AAAAAAAAAAAAAAAA
10 .TARGET ID........AAAAAAAAAAAAAAAA
11 .MISSION STATUS....AAAAAAAAAAAAA...
12 .FIRE FOR EFFECT...NO..............
13 ....................................
14 ....................................
15 ........CAN NOT OBSERVE............
16 ....................................
17 ....................................
18 ....................................
19 ....................................
20 .FREE TEXT ANNOTATION•••...........
```

ARTILLERY SHIFT REPORT

```
   0          1          2          3
   01234567890123456789012345678901234

 0 .ARTY SHIFT REPORT.ROUTINE.........
 1 .....................................
 2 .SENDER ...........XXXXXXX.........
 3 .SENT TO...........XXXXXXX.........
 4 .FWD BY............XXXXXXX.........
 5 .MSG SENT..........26 1745 JUN 95..
 6 .XMIT LCN..........NB 0624 0332....
 7 .XMIT ALT..........1200 FEET.......
 8 .....................................
 9 .MISSION ID........AAAAAAAAAAAAAAAA
10 .TARGET ID.........AAAAAAAAAAAAAAAA
11 .MISSION STATUS....AAAAAAAAAAAAA...
12 .FIRE FOR EFFECT...NO..............
13 .....................................
14 .....................................
15 ..............SHIFT.................
16 .....................................
17 .........AAAAAAAAAAAAAAA...........
18 .....................................
19 .....................................
20 .FREE TEXT ANNOTATION•••...........
```

ARTILLERY NEW MISSION REPORT

```
   0         1         2         3
   01234567890123456789012345678901234

 0 .ARTY NEW MSN RPRT.ROUTINE.........
 1 .....................................
 2 .SENDER ...........XXXXXXXX........
 3 .SENT TO...........XXXXXXXX........
 4 .FWD BY............XXXXXXXX........
 5 .MSG SENT..........26 1745 JUN 95..
 6 .XMIT LCN..........NB 0624 0332....
 7 .XMIT ALT..........1200 FEET.......
 8 .....................................
 9 ...* * NEW ARTILLERY MISSION * *...
10 .....................................
11 .MISSION ID........AAAAAAAAAAAAAAAA
 2 .TARGET ID.........AAAAAAAAAAAAAAAA
13 .MISSION STATUS....AAAAAAAAAAAAA...
14 .MISSION TYPE......IMMED SUPPR.....
15 .SHELL.............WP..............
16 .CONTROL...........WR..............
17 .FUZE..............TIME DELAY......
18 .TRAJ..............LOW.............
19 .FIRE FOR EFFECT...YES.............
20 .FREE TEXT ANNOTATION•••...........
```

ARTILLERY MESSAGE TO OBSERVER REPORT

```
   0          1          2          3
   01234567890123456789012345678901234

 0 .ARTY MTO REPORT...ROUTINE.........
 1 .....................................
 2 .SENDER ...........XXXXXXXX........
 3 .SENT TO...........XXXXXXXX........
 4 .FWD BY............XXXXXXXX........
 5 .MSG SENT..........26 1745 JUN 95..
 6 .XMIT LCN..........NB 0624 0332....
 7 .XMIT ALT..........1200 FEET.......
 8 .....................................
 9 .MISSION ID........AAAAAAAAAAAAAAAA
10 .TARGET ID.........AAAAAAAAAAAAAAAA
11 .MISSION STATUS....AAAAAAAAAAAAA...
12 .....................................
13 ........MESSAGE TO OBSERVER........
14 .....................................
15 .REQUEST ADJUST....YES.............
16 .ENTER AS TARGET...NO..............
17 .END MISSION.......NO..............
18 .....................................
19 .....................................
20 .FREE TEXT ANNOTATION•••...........
```

ARTILLERY SHOT REPORT

```
   0         1         2         3
   01234567890123456789012345678901234

 0 .ARTY SHOT REPORT..ROUTINE.........
 1 .....................................
 2 .SENDER ...........XXXXXXXX........
 3 .SENT TO...........XXXXXXXX........
 4 .FWD BY............XXXXXXXX........
 5 .MSG SENT..........26 1745 JUN 95..
 6 .XMIT LCN..........NB 0624 0332....
 7 .XMIT ALT..........1200 FEET.......
 8 .....................................
 9 .MISSION ID........AAAAAAAAAAAAAAAA
10 .TARGET ID.........AAAAAAAAAAAAAAAA
11 .MISSION STATUS....AAAAAAAAAAAAA...
12 .....................................
13 .....................................
14 .....................................
15 .......* * SHOT FIRED * *..........
16 .....................................
17 .....................................
18 .....................................
19 .....................................
20 .FREE TEXT ANNOTATION•••...........
```

ARTILLERY SPLASH REPORT

```
   0         1         2         3
   01234567890123456789012345678901234

 0 .ARTY SPLASH RPRT..ROUTINE.........
 1 ....................................
 2 .SENDER ...........XXXXXXX........
 3 .SENT TO...........XXXXXXX........
 4 .FWD BY............XXXXXXX........
 5 .MSG SENT..........26 1745 JUN 95..
 6 .XMIT LCN..........NB 0624 0332....
 7 .XMIT ALT..........1200 FEET.......
 8 ....................................
 9 .MISSION ID........AAAAAAAAAAAAAAAA
10 .TARGET ID.........AAAAAAAAAAAAAAAA
11 .MISSION STATUS....AAAAAAAAAAAA...
12 ....................................
13 ....................................
14 ..........* * SPLASH * *...........
15 ....................................
16 ....................................
17 ....................................
18 ....................................
19 ....................................
20 .FREE TEXT ANNOTATION•••...........
```

ARTILLERY END OF MISSION REPORT

```
   0          1          2          3
   01234567890123456789012345678901234

 0 .ARTY EOM REPORT...ROUTINE.........
 1 ....................................
 2 .SENDER ...........XXXXXXXX........
 3 .SENT TO...........XXXXXXXX........
 4 .FWD BY............XXXXXXXX........
 5 .MSG SENT..........26 1745 JUN 95..
 6 .XMIT LCN..........NB 0624 0332....
 7 .XMIT ALT..........1200 FEET.......
 8 ....................................
 9 .MISSION ID........AAAAAAAAAAAAAAAA
10 .TARGET ID.........AAAAAAAAAAAAAAAA
11 .MISSION STATUS....AAAAAAAAAAAAA...
12 ....................................
13 ......* * END OF MISSION * *.......
14 ....................................
15 .DISPOSITION.......BURNING..........
16 .RECORD AS TARGET..NO...............
17 .CASUALTIES........3000.............
18 .POINT NUMBER.....AAAAAAAAAAAAAAAA.
19 ....................................
20 .FREE TEXT ANNOTATION•••............
```

NUCLEAR, BIOLOGICAL, CHEMICAL TYPE 1 REPORT

```
   0         1         2         3
   01234567890123456789012345678901234

 0 .NBC-1 REPORT......URGENT..........
 1 ....................................
 2 .SENDER ...........XXXXXXX........
 3 .SENT TO...........XXXXXXX........
 4 .FWD BY............XXXXXXX........
 5 .MSG SENT..........26 1745 JUN 95..
 6 .XMIT LCN..........NB 0624 0332....
 7 .XMIT ALT..........1200 FEET.......
 8 ....................................
 9 .....>>>> NUCLEAR BLAST <<<<.......
10 ....................................
11 .DESCRIPTION.......INCREASING......
12 .BURST TYPE........SURFACE.........
13 .DELIVERED BY......ROCKET..........
14 .CLOUD HGHT UNITS..DEGREES.........
15 .CLOUD HEIGHT......35..............
16 .CLOUD DESCRIPTION.AAAAAAAAAAAAAAA
17 .FLASH TO BANG.....55.SECONDS......
18 .START TIME........26 1743 JUN 95..
19 .STOP TIME.........26 1744 JUN 95..
20 .FREE TEXT ANNOTATION•••...........
```

NUCLEAR, BIOLOGICAL, CHEMICAL TYPE 4 REPORT

```
   0          1          2          3
   01234567890123456789012345678901234

 0 .NBC-4 REPORT......URGENT..........
 1 ....................................
 2 .SENDER ...........XXXXXXXX........
 3 .SENT TO...........XXXXXXXX........
 4 .FWD BY............XXXXXXXX........
 5 .MSG SENT..........26 1745 JUN 95..
 6 .XMIT LCN..........NB 0624 0332....
 7 .XMIT ALT..........1200 FEET.......
 8 ....................................
 9 .*.CHEMICAL OR BIOLOGICAL ATTACK.*.
10 ....................................
11 .DESCRIPTION.......INCREASING......
12 .BURST TYPE........SURFACE..........
13 .DELIVERED BY......ROCKET...........
14 .CLOUD HGHT UNITS..DEGREES..........
15 .CLOUD HEIGHT......35...............
16 .CLOUD DESCRIPTION.AAAAAAAAAAAAAAA
17 .DOSE RATE.........300..............
18 ....................................
19 ....................................
20 .FREE TEXT ANNOTATION••.............
```

NUCLEAR, BIOLOGICAL, CHEMICAL TYPE 5 REPORT

```
   0          1          2          3
   01234567890123456789012345678901234

 0 .NBC-5 REPORT......ROUTINE.........
 1 .....................................
 2 .SENDER ...........XXXXXXXX........
 3 .SENT TO...........XXXXXXXX........
 4 .FWD BY............XXXXXXXX........
 5 .MSG SENT..........26 1745 JUN 95..
 6 .XMIT LCN..........NB 0624 0332....
 7 .XMIT ALT..........1200 FEET.......
 8 .....................................
 9 .....................................
10 .....................................
11 .....................................
12 .....................................
13 ..........NBC NEGATIVE.............
14 .....................................
15 .....................................
16 .....................................
17 .....................................
18 .....................................
19 .....................................
20 .FREE TEXT ANNOTATION••••..........
```

```
STATUS REPORT

   0         1         2         3
   01234567890123456789012345678901234

 0 .STATUS REPORT.....ROUTINE.........
 1 .....................................
 2 .SENDER ...........XXXXXXXX........
 3 .SENT TO...........XXXXXXXX........
 4 .FWD BY............XXXXXXXX........
 5 .MSG SENT..........26 1745 JUN 95..
 6 .XMIT LCN..........NB 0624 0332....
 7 .XMIT ALT..........1200 FEET.......
 8 .....................................
 9 .FUEL..............XXXX LBS.......
10 .FAILED EQUPT......aaaaaaaaaaaaaaaa
11 .FAILED EQUPT......aaaaaaaaaaaaaaaa
12 .FAILED EQUPT......aaaaaaaaaaaaaaaa
13 .HELLFIRES.........XX..............
14 .STINGERS..........XX..............
15 .ROCKETS...........XX..............
16 .ROUNDS............XXXX............
17 .REQUEST TYPE......AUTOMATIC.......
18 .....................................
19 .....................................
20 .FREE TEXT ANNOTATION•••...........
```

```
REQUEST REPORT: STATUS, SPOT, MOVEMENT

    0         1         2         3
    01234567890123456789012345678901234

 0  .REQUEST REPORT....ROUTINE.........
 1  ....................................
 2  .SENDER ............XXXXXXXX........
 3  .SENT TO............XXXXXXXX........
 4  .FWD BY.............XXXXXXXX........
 5  .MSG SENT...........26 1745 JUN 95..
 6  .XMIT LCN...........NB 0624 0332....
 7  .XMIT ALT...........1200 FEET.......
 8  ....................................
 9  .REPORT DESIRED....STATUS...........
10  ....................................
11  ....................................
12  ....................................
13  ....................................
14  ....................................
15  ....................................
16  ....................................
17  ....................................
18  ....................................
19  ....................................
20  .FREE TEXT ANNOTATION•••............
```

REQUEST REPORT: RECON

```
   0         1         2         3
   01234567890123456789012345678901234

 0 .REQUEST REPORT....ROUTINE.........
 1 ....................................
 2 .SENDER ...........XXXXXXX........
 3 .SENT TO...........XXXXXXX........
 4 .FWD BY............XXXXXXX........
 5 .MSG SENT..........26 1745 JUN 95..
 6 .XMIT LCN..........NB 0624 0332....
 7 .XMIT ALT..........1200 FEET.......
 8 ....................................
 9 .REPORT DESIRED....RECON...........
10 .TYPE DESIRED......AIR ROUTE.......
11 ....................................
12 ....................................
13 ....................................
14 ....................................
15 ....................................
16 ....................................
17 ....................................
18 ....................................
19 ....................................
20 .FREE TEXT ANNOTATION•••...........
```

BATTLE DAMAGE ASSESSMENT REPORT

```
   0          1          2          3
   01234567890123456789012345678901234

 0 .BDA REPORT.........ROUTINE.........
 1 .....................................
 2 .SENDER ............XXXXXXXX........
 3 .SENT TO............XXXXXXXX........
 4 .FWD BY.............XXXXXXXX........
 5 .MSG SENT...........26 1745 JUN 95..
 6 .XMIT LCN...........NB 0624 0332....
 7 .XMIT ALT...........1200 FEET.......
 8 .....................................
 9 .STRIKE START......26 1740 JUN 95..
10 .STRIKE END........26 1742 JUN 95..
11 .TARGET CATEGORY....................
12 .TARGET TYPE.......TANK.............
13 .TARGETS DESTRYD...4................
14 .PERCENT COVERAGE..100..............
15 .....................................
16 .....................................
17 .....................................
18 .....................................
19 .....................................
20 .FREE TEXT ANNOTATION•••............
```

GROUND ROUTE RECON REPORT

```
   0          1          2          3
   01234567890123456789012345678901234

 0 .GND RTE RECON RPT.ROUTINE.........
 1 .....................................
 2 .SENDER ...........XXXXXXXX........
 3 .SENT TO...........XXXXXXXX........
 4 .FWD BY............XXXXXXXX........
 5 .MSG SENT..........26 1745 JUN 95..
 6 .XMIT LCN..........NB 0624 0332....
 7 .XMIT ALT..........1200 FEET.......
 8 .....................................
 9 .ENEMY ACTIVITY....ATTACKING.......
10 .CLSFCTN FORMULA...MLC.............
11 .CLSFCTN INFO......AAAAAAAAAAAAAAAA
12 .ROUTE ID..........AAAAAAAAAAAAAAAA
13 .....................................
14 .....................................
15 .....................................
16 .....................................
17 .....................................
18 .....................................
19 .....................................
20 .FREE TEXT ANNOTATION••••..........
```

AIR ROUTE RECON REPORT

```
   0          1          2          3
   01234567890123456789012345678901234

 0 .AIR RTE RECON RPT.ROUTINE.........
 1 .....................................
 2 .SENDER ............XXXXXXXX........
 3 .SENT TO............XXXXXXXX........
 4 .FWD BY.............XXXXXXXX........
 5 .MSG SENT...........26 1745 JUN 95..
 6 .XMIT LCN...........NB 0624 0332....
 7 .XMIT ALT...........1200 FEET.......
 8 .....................................
 9 .ENEMY ACTIVITY....ATTACKING........
10 .....................................
11 .OBSTACLES.........TREES............
12 .....................................
13 .ROUTE ID..........AAAAAAAAAAAAAAA
14 .....................................
15 .....................................
16 .....................................
17 .....................................
18 .....................................
19 .....................................
20 .FREE TEXT ANNOTATION•••............
```

BRIDGE RECON REPORT

```
   0         1         2         3
   01234567890123456789012345678901234

 0 .BRIDGE RECON RPT..ROUTINE.........
 1 ...................................
 2 .SENDER ...........XXXXXXXX........
 3 .SENT TO...........XXXXXXXX........
 4 .FWD BY............XXXXXXXX........
 5 .MSG SENT..........26 1745 JUN 95..
 6 .XMIT LCN..........NB 0624 0332....
 7 .XMIT ALT..........1200 FEET.......
 8 .BRIDGE TYPE.......GIRDER..........
 9 .DAMAGE............SEVERE..........
10 .SPANS.............6...............
11 .CONST MATL........METAL...........
12 .LENGTH............AAAA............
13 .WIDTH.............AAAA............
14 .HEIGHT............AAAA............
15 .UNDER.............AAAA............
16 .CONSTR DESCR......AAAAAAAAAAAAAAAA
17 .ID................AAAAAAAAAAAAAAAA
18 .SPAN LENGTH.......AAAA............
19 .LOAD CLASS........AAAA............
20 .FREE TEXT ANNOTATION•••...........
```

Note: Only first 16 chars in 32 char PDU field used for bridge ID

LANDING ZONE/PICKUP ZONE RECON REPORT

```
   0          1          2          3
   01234567890123456789012345678901234

 0 .LZ/PZ RECON RPT. .ROUTINE.........
 1 ...................................
 2 .SENDER ...........XXXXXXXX........
 3 .SENT TO...........XXXXXXXX........
 4 .FWD BY............XXXXXXXX........
 5 .MSG SENT..........26 1745 JUN 95..
 6 .XMIT LCN..........NB 0624 0332....
 7 .XMIT ALT..........1200 FEET.......
 8 ...................................
 9 .ACTIVITY LIKELY...EXPECTED........
10 .OBSTACLES.........TWR/ANT.........
11 .OBST. DESCR.......AAAAAAAAAAAAAAAA
12 .LZ/PZ ID..........AAAAAAAAAAAAAAAA
13 .LZ/PZ SIZE........AAAAAAAAAAAAAAAA
14 ...................................
15 ...................................
16 ...................................
17 ...................................
18 ...................................
19 ...................................
20 .FREE TEXT ANNOTATION•••...........
```

BP/OP RECON REPORT

```
   0          1          2          3
   01234567890123456789012345678901234

 0 .BP/OP RECON RPT...ROUTINE.........
 1 ...................................
 2 .SENDER ...........XXXXXXXX........
 3 .SENT TO...........XXXXXXXX........
 4 .FWD BY............XXXXXXXX........
 5 .MSG SENT..........26 1745 JUN 95..
 6 .XMIT LCN..........NB 0624 0332....
 7 .XMIT ALT..........1200 FEET.......
 8 ...................................
 9 .ENEMY ACTIVITY....EXPECTED........
10 .OBSTACLES.........WIRES...........
11 .OBST. DESCR.......AAAAAAAAAAAAAAAA
12 .BP/OP ID..........AAAAAAAAAAAAAAAA
13 .BP/OP SIZE........AAAAAAAAAAAAAAAA
14 .AXIS..............AAAAAAAAAAAAAAAA
15 ...................................
16 ...................................
17 ...................................
18 ...................................
19 ...................................
20 .FREE TEXT ANNOTATION•••...........
```

CROSSING REPORT

```
   0         1         2         3
   01234567890123456789012345678901234

 0 .CROSSNG RECON RPT.ROUTINE.........
 1 .....................................
 2 .SENDER ...........XXXXXXX.........
 3 .SENT TO...........XXXXXXX.........
 4 .FWD BY............XXXXXXX.........
 5 .MSG SENT..........26 1745 JUN 95..
 6 .XMIT LCN..........NB 0624 0332....
 7 .XMIT ALT..........1200 FEET.......
 8 .....................................
 9 .BANK SLOPE ENTRY..AAAA............
10 .BANK SLOPE EXIT...AAAA............
11 .CROSSING LENGTH...AAAA............
12 .CROSSING WIDTH....AAAA............
13 .CROSSING DEPTH....AAAA............
14 .CURRENT FLOW......AAAA............
15 .CROSSING ID.......AAAAAAAA........
16 .....................................
17 .....................................
18 .....................................
19 .....................................
20 .FREE TEXT ANNOTATION•••...........
```

DOWNED AIR VEHICLE REPORT

```
   0          1          2          3
   01234567890123456789012345678901234

 0 .DNAV REPORT.......URGENT..........
 1 ....................................
 2 .SENDER ...........XXXXXXX........
 3 .SENT TO...........XXXXXXX........
 4 .FWD BY............XXXXXXX........
 5 .MSG SENT..........26 1745 JUN 95..
 6 .XMIT LCN..........NB 0624 0332....
 7 .XMIT ALT..........1200 FEET.......
 8 ....................................
 9 .AIRCRAFT TYPE.....OH-58...........
10 .AIRCRAFT STATUS...RECOVER.........
11 .PILOT STATUS......GOOD............
12 ....................................
13 ....................................
14 ....................................
15 ....................................
16 ....................................
17 ....................................
18 ....................................
19 ....................................
20 .FREE TEXT ANNOTATION•••...........
```

Movement Command (MOVCMD) REPORT

```
    0         1         2         3
    01234567890123456789012345678901234

 0  .MOVCMD REPORT,....URGENT.........
 1  .....................................
 2  .SENDER ...........XXXXXXXX........
 3  .SENT TO...........XXXXXXXX........
 4  .FWD BY............XXXXXXXX........
 5  .MSG SENT..........26 1745 JUN 95..
 6  .XMIT LCN..........NB 0624 0332....
 7  .XMIT ALT..........1200 FEET.......
 8  .....................................
 9  .TARGET ID.........AAAAAAAAAAAAAAA
10  .TASK..............CONT MSN........
11  .WHO...............B3...............
12  .WHEN..............1245 Z; 0445 L..
13  .WHERE.............AAAAAAAA........
14  .....................................
15  .....................................
16  .....................................
17  .....................................
18  .....................................
19  .....................................
20  .FREE TEXT ANNOTATION•••...........
```

PIREP REPORT

```
   0          1          2          3
   01234567890123456789012345678901234

 0 .PILOT REPORT......ROUTINE.........
 1 .....................................
 2 .SENDER ...........XXXXXXX.........
 3 .SENT TO...........XXXXXXX.........
 4 .FWD BY............XXXXXXX.........
 5 .MSG SENT..........26 1745 JUN 95..
 6 .XMIT LCN..........NB 0624 0332....
 7 .XMIT ALT..........1200 FEET.......
 8 .....................................
 9 .VISIB..HALF MILE..BLIZZARD........
10 .CLDS SCTRD, BASE 12000, TOP.30000.
11 .....................................
12 .TEMP.. -5.C.......BAR 29.56"......
13 .....................................
14 .WIND 80 KNOTS, NORTH EAST.........
15 .....................................
16 .CONT, EXTREME TURBULENCE..........
17 .....................................
18 .SEVERE CLEAR ICING................
19 .....................................
20 .FREE TEXT ANNOTATION•••...........
```

MIJI REPORT

```
    0          1          2          3
    01234567890123456789012345678901234

 0 .MIJI REPORT.....ROUTINE.........
 1 ..................................
 2 .SENDER ..........XXXXXXXX........
 3 .SENT TO..........XXXXXXXX........
 4 .FWD BY...........XXXXXXXX........
 5 .MSG SENT.........26 1745 JUN 95..
 6 .XMIT LCN.........NB 0624 0332....
 7 .XMIT ALT.........1200 FEET.......
 8 ..................................
 9 .AFFECTED FREQ.....xxx.xxxxxx kHz..
10 .PROGRAMD FREQ.1...xxx.xxxxxx kHz..
11 .PROGRAMD FREQ.2...xxx.xxxxxx kHz..
12 .PROGRAMD FREQ.3...xxx.xxxxxx kHz..
13 .PROGRAMD FREQ.4...xxx.xxxxxx kHz..
14 .START TIME.........26 1711 JUN 95..
15 .STOP TIME.........  .26 1711 JUN 95..
16 .DESCRIPTION.......WARBLING.........
17 .PERCENT LOST......XXX PERCENT.....
18 .TYPE..............JAMMING.........
19 ..................................
20 .FREE TEXT ANNOTATION•••...........
```

NOTE: Frequencies come in as 64 bit floating point numbers which express cycles per second (Hertz). This is to be changed to kiloHertz, (kHz) if the number is greater than or equal to 1000 and less than a million; it is to be changed to Megahertz (MHz) if the value is greater than or equal to a million and less than a billion; and it is to be changed to GigaHertz (GHz) if it is over a billion. The notations Hz, kHz, MHz, and GHz (case exactly as shown here) are to be used. One to three decimal places are to be to the left of the decimal point; exactly 6 (with right filled zeroes) are to be to the right.

ACKNOWLEDGEMENT

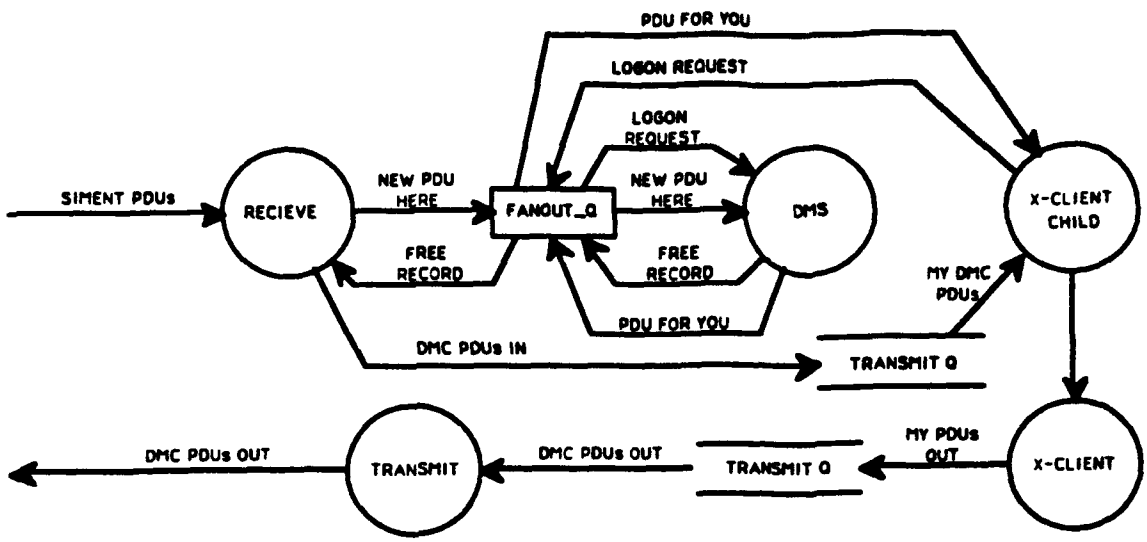
```
   0          1          2          3
   01234567890123456789012345678901234

 0 .ACKNOWLEDGEMENT...ROUTINE.........
 1 ...................................
 2 .SENDER ...........XXXXXXXX........
 3 .SENT TO...........XXXXXXXX........
 4 .FWD BY............XXXXXXXX........
 5 .MSG SENT..........26 1745 JUN 95..
 6 .XMIT LCN..........NB 0624 0332....
 7 .XMIT ALT..........1200 FEET.......
 8 ...................................
 9 .YOUR MESSAGE TO...AAAAAAAA........
10 .WAS RECEIVED AT...26 1743 JUN 95..
11 ...................................
12 ...................................
13 ...................................
14 ...................................
15 ...................................
16 ...................................
17 ...................................
18 ...................................
19 ...................................
20 .FREE TEXT ANNOTATION•••...........
```

70. Appendix G - Data Flow Diagrams.

The following diagrams contain data flow diagrams for the DMCC System and the DMCC X-Client Process.

70.1 Top Level DMCC Inter-Process Data Flow Diagram



70.2 DMCC X-Client Process Data Flow Diagram