

To the Graduate Council:

I am submitting herewith a thesis written by James Gordon Barrett III entitled "Image Transfer Over a Manufacturing Protocol Network." I have examined the final copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Electrical Engineering.

Asa O. Bishop, Jr.

Asa O. Bishop, Jr., Major Professor

We have read this thesis
and recommend its acceptance:

Marshall D. Pace

William J. McClure

Accepted for the Council:

C. W. Mink

Vice Provost
and Dean of The Graduate School

**IMAGE TRANSFER OVER A
MANUFACTURING AUTOMATION
PROTOCOL NETWORK**

A Thesis

Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

**James Gordon Barrett III
December 1990**

DEDICATION

This thesis is dedicated to my parents and my wife. My parents' example of perseverance and determination continues to have a strong influence on my life. My wife, in addition to contributing to our monetary support, has been an endless source of encouragement in my studies.

ACKNOWLEDGEMENTS

I would like to thank the Martin Marietta Corporation for supporting me on a research assistantship. I would also like to thank the Electrical and Computer Engineering Department of the University of Tennessee, Knoxville for a teaching assistantship. My thanks is extended to Systems Integration Specialists Company, Inc. for the opportunity to take their MMS-EASE programming class.

I would especially like to thank Dr. Asa O. Bishop, Professor of Electrical and Computer Engineering and the major advisor of my graduate committee. His guidance and advice in my research have been greatly appreciated. Special thanks also goes to Dr. William McClain and Dr. Marshall Pace, Professors of Electrical and Computer Engineering, for their part in my degree and for serving on my graduate committee.

For support above and beyond the call of duty, my gratitude goes to Mr. Robert K. Robinson, an engineer from Systems Integration Specialists Company, Inc. His timely help and advice on obscure aspects of MMS-EASE saved me months of wasted effort. I am indebted to Mr. Jason Hartman, the manager of the Manufacturing Protocol Network used in the research, for his suggestions and help in learning the network. My thanks to Mr. Rick Igou, an electrical engineer from the Martin Marietta Corporation, who provided equipment, reviews, and advice on this research. Finally, I would like to thank my wife, Jessica, who has patiently supported and encouraged me in whatever I do.

ABSTRACT

In just a few short years, image capture and image processing in general have progressed from somewhat impractical, expensive, and specialized applications into the mainstream of office and production computing. Scanners, image capture boards, facsimile machines, and other image related digitizers have been used for a variety of purposes in many office and laboratory environments. The increasing use of these devices has reduced the problem of data acquisition, however, the problem of data isolation is intensified. This thesis addresses the distribution of captured or live video images over the Manufacturing Automation Protocol (MAP) network.

In a manufacturing context, layouts and specifications are usually generated in a design lab, but used on the plant floor. Multiple hardcopies of a particular drawing may cause confusion and mistakes by using an old revision. A solution to this problem is an image server on a network for remote access to electronic documentation. With advances in the capabilities of image analysis, a remote vision system for quality control or monitoring purposes is now practical.

This thesis examines the ramifications of image transfer and manipulation using the Manufacturing Message Specification (MMS) over a well defined MAP network at the Electrical and Computer Engineering Department of the University of Tennessee. Two distinct methods were used; file transfer and domain downloading. In addition to static images, real time image capture was explored in anticipation of the expanding applications of machine vision in manufacturing.

TABLE OF CONTENTS

CHAPTER	PAGE
1. INTRODUCTION	1
2. EXPERIMENTAL CONFIGURATION	3
2.1 Overview	3
2.2 MAP 2.X Broadband Configuration	3
2.3 MAP 3.0 Carrierband Configuration	6
2.4 The Image Server	6
2.5 The Image Client	9
3. METHODOLOGY	11
3.1 Overview	11
3.2 SISCO's MMS-EASE	12
3.3 The Targa-M8 Video Buffer	15
3.4 The Everex Viewpoint VGA Video Buffer	16
3.5 The File Transfer Method	17
3.6 The Domain Upload Method	28
4. RESULTS	38
4.1 Overview	38
4.2 MAP 2.X Broadband Results	39
4.3 MAP 3.0 Carrierband Results	40
5. CONCLUSION	41
5.1 Overview	41
5.2 Future Work	43
LIST OF REFERENCES	45
APPENDIXES	47
APPENDIX A. INI HEAD END CONFIGURATION	48
APPENDIX B. INI NMC CONFIGURATION	49
APPENDIX C. INI MP-400 CARD CONFIGURATION	56
APPENDIX D. CONCORD 1200 CARD CONFIGURATION	58

CHAPTER	PAGE
APPENDIX E. TRUEVISION TARGA-M8 CONFIGURATION	62
APPENDIX F. COMPILER CONFIGURATION	66
APPENDIX G. STANDARD CODE USED IN ALL PROGRAMS ...	67
APPENDIX H. FILE TRANSFER SOURCE CODE	134
H.1 MAP 2.X Server Code	134
H.2 MAP 2.X Client Code	149
H.3 MAP 3.0 Server Code	165
H.4 MAP 3.0 Client Code	166
APPENDIX I. DOMAIN UPLOAD SOURCE CODE	167
I.1 MAP 2.X Server Code	167
I.2 MAP 2.X Client Code	182
I.3 MAP 3.0 Server Code	198
I.4 MAP 3.0 Client Code	199
VITA	200

LIST OF FIGURES

FIGURE	Page
2.1. MAP 2.X Network Layout	4
2.2. MAP 3.0 Carrierband Network Layout	7
3.1. Memory Windowing of a Video Buffer	15
3.2. Flow Diagram of Connection	19
3.3. Flow Diagram of File Open	21
3.4. Flow Diagram of File Read	23
3.5. Flow Diagram of File Close and Disconnect	26
3.6. Bit Packing Data Compression Scheme	28
3.7. Flow Diagram of Domain Creation and Connection	30
3.8. Flow Diagram of Domain Initiate Upload	32
3.9. Flow Diagram of Domain Upload	34
3.10. Flow Diagram of Terminate Upload and Disconnect	37
C.1. INI MP-400 TBC Jumper Settings	57
D.1. Concord 1200 Bus Controller Jumper Settings	59
E.1. Targa-M8 Switch Settings	63

LIST OF TABLES

TABLE	PAGE
3.1. Source Code Changes From MAP 2.X to MAP 3.0	13
3.2. Targa Tools Function Calls	16
3.3. Everex VGA Function Calls	17
3.4. MMS-EASE Connection Oriented Functions	18
3.5. MMS-EASE File Open Functions	20
3.6. MMS-EASE File Read Functions	22
3.7. MMS-EASE File Close Functions	25
3.8. MMS-EASE Disconnect Functions	25
3.9. MMS-EASE Initiate Upload Domain Functions	31
3.10. MMS-EASE Upload Domain Functions	33
3.11. MMS-EASE Terminate Upload Domain Functions	36
4.1. MAP 2.X Broadband Results Without Compression	39
4.2. MAP 2.X Broadband Results With Compression	39
4.3. MAP 3.0 Carrierband Results Without Compression	40
4.4. MAP 3.0 Carrierband Results With Compression	40
5.1. Percentage Data Measurements	43
B.1. Configuration of Block Sizes	50
B.2. XLI Configuration	51
B.3. DLI Configuration	52

TABLE	PAGE
B.4. DL Configuration	53
B.5. XPORT Configuration	54
C.1. INI TBC Jumpers	56
D.1. Concord 1200 Bus Controller Jumpers	58
D.2. Contents of MAP3CNFG.DAT File	60
E.1. Targa-M8 Jumpers	62
E.2. Targa-M8 Software Configuration Option Values	64
E.3. Camera-to-Targa Cable Pinout	65
E.4. Targa-to-Monitor Cable Pinout	65
F.1. Compiler Command Line Options	66
F.2. Linker Command Line Options	66

CHAPTER 1

INTRODUCTION

With the introduction of the MAP network came the first real possibilities of a completely integrated and nonproprietary environment for automated manufacturing. Specifically designed to eliminate the "Islands of Automation" problem, the MAP network can allow more than just machine-to-machine communication [1]. By using some of the lower level MMS services, machine-to-human communication is also possible.

The primary purpose of a network is the orderly distribution of data to authorized users in an efficient and transparent manner [2]. In manufacturing, a large part of that data may be meaningless unless it is in a graphical form (circuit diagrams, part drawings, machine configurations, etc.). Although a drawing file may contain information about lines and vectors in the drawing, most people cannot visualize the image from a list of numbers. Simple access to the image itself, rather than the vector file, is highly desirable.

Working within the MMS protocol, an image transfer mechanism was implemented using commercially available equipment for office personal computers with a video graphics array (VGA) adapter and network card. One use for this capability could be the transfer of up-to-date plans, photographs, or specifications from the engineering lab to a fabrication station for machine

configuration by an operator. Another use could be the monitoring and automatic adjustment of numerically-controlled machine tools using image processing edge-finding algorithms to compensate for working-surface wear. Other uses include automated process monitoring using a variety of existing and emerging machine vision techniques.

The primary problem of graphical electronic documentation is the large amount of data required to define the image. Depending on the resolution and the number of colors, images may be as large as 10 megabytes (MB). Transferring these images across the MAP network in a restricted time frame can be a challenge. The images used in the comparison of transmission schemes were 200 kilobytes (KB) in size.

In Chapter 2 the hardware and software configurations used in the experiments is discussed. Chapter 3 presents the methodology used in the experiments and the relationship between the client and server entities. Chapter 4 gives the results of the tests, and Chapter 5 draws some conclusions from the results. Several appendices are also included for specific hardware configurations, cable diagrams, and source code.

CHAPTER 2

EXPERIMENTAL CONFIGURATION

2.1 Overview

For comparison purposes, the software was developed to run over both MAP 2.X and MAP 3.0 network configurations. For the 2.X network, Industrial Networking Incorporated (INI) MP-400 network cards were used on a broadband network using an INI head end. The 3.0 network used Concord Communications 1200 controller cards with external carrierband modems. The following sections describe in detail the configurations.

2.2 MAP 2.X Broadband Configuration

The Overall Network Layout

The configuration for the research under MAP 2.X was organized as shown in Figure 2.1.

The INI Head End Remodulator

The head end used was an INI MHR-40. The frequency pair used was 6'-FM1' (83.75MHz - 89.75MHz transmit and 276MHz - 282MHz receive) as the

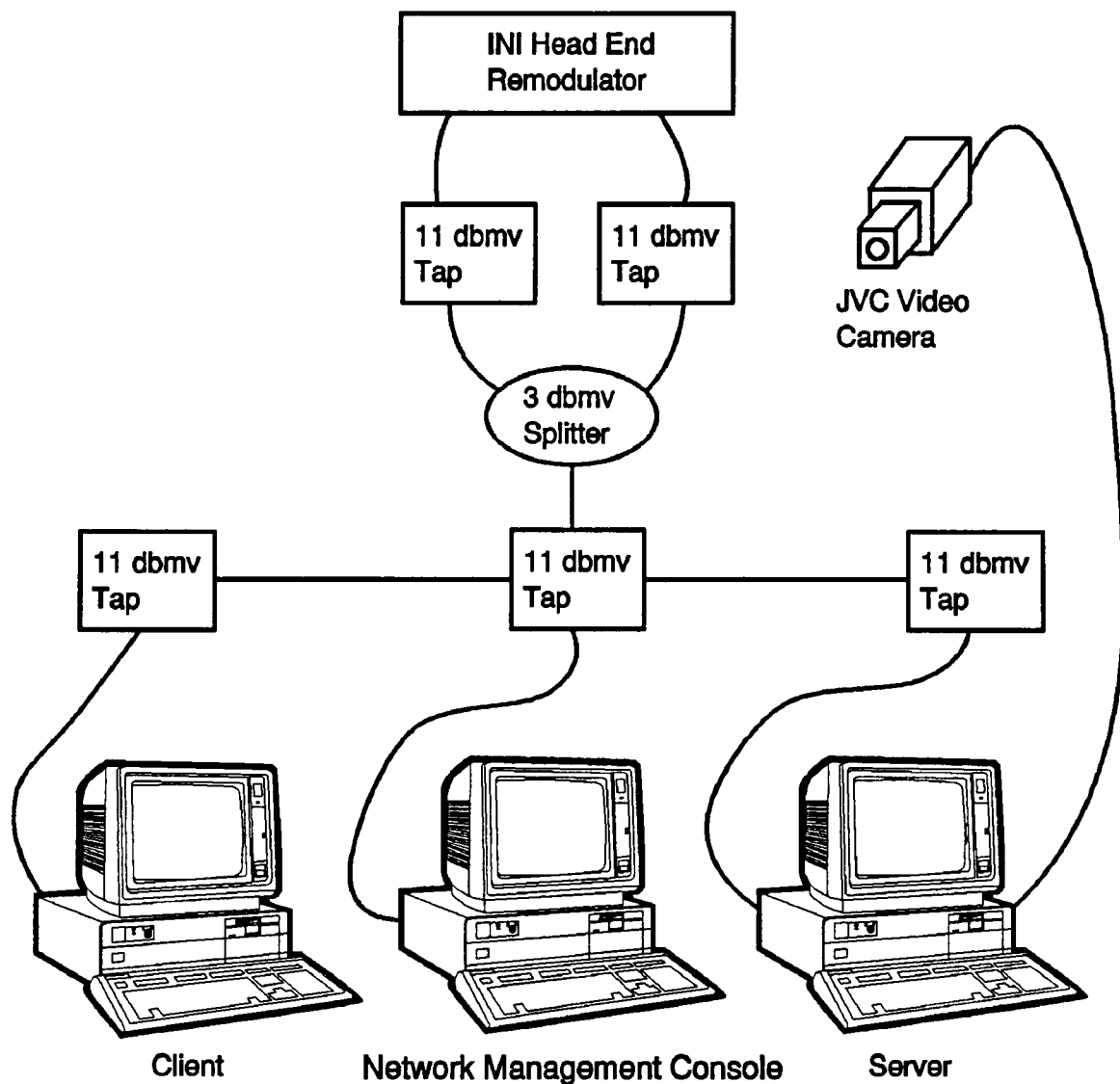


Figure 2.1. MAP 2.X Network Layout

INI MP-400 card set uses fixed frequency modems with that setting. The frequency pair used is one of three set aside for MAP by the Federal Communications Commission (FCC) [3]. See Appendix A for specific configuration information.

The Network Management Console

The INI MP-400 network cards are configured and downloaded from a central file server called the network management console. Overall network settings, individual network card configurations, and the CASE level network address mechanism are handled at this console. See Appendix B for specific configuration information.

The INI MP-400 Network Card

The network card-sets on both ends of the connection were INI MP-400's. A set consists of a Token Bus Controller (TBC) and a TBM-12 internal 10 Mbps broadband modem. The TBC card is driven by an Intel 80186 running at 10MHz and has 64 KB of buffer memory available as user controlled transfer space [4]. The program PROM revision on both TBC cards was level 4.0.4. See Appendix C for jumper configuration.

2.3 MAP 3.0 Carrierband Configuration

The Overall Network Layout

The layout of the network for the research under MAP 3.0 using carrier band was organized as shown in Figure 2.2.

The Concord 1200 Network Card

The network card sets for the MAP 3.0 network were Concord Communications series 1200's. These nodes consisted of a series 1200 bus controller for the PC and a model L2-MOC1-2000 Concord external 5 Mbps carrierband modem. The revision of the MAP software stack and the ROMs on the bus controllers was level 1.1c. The bus controller is powered by an Intel 80186 running at 10 MHz and has 128 KB of multi-port memory for buffer management [5]. See Appendix D for configuration information.

2.4 The Image Server

The host machine for the image server was an Everex IBM AT compatible personal computer running at 16MHz with 1MB of memory (640K is required for operation). This machine contained a networking card, the Targa-M8 image capture board, a monochrome video card, and a 40 MB hard disk.

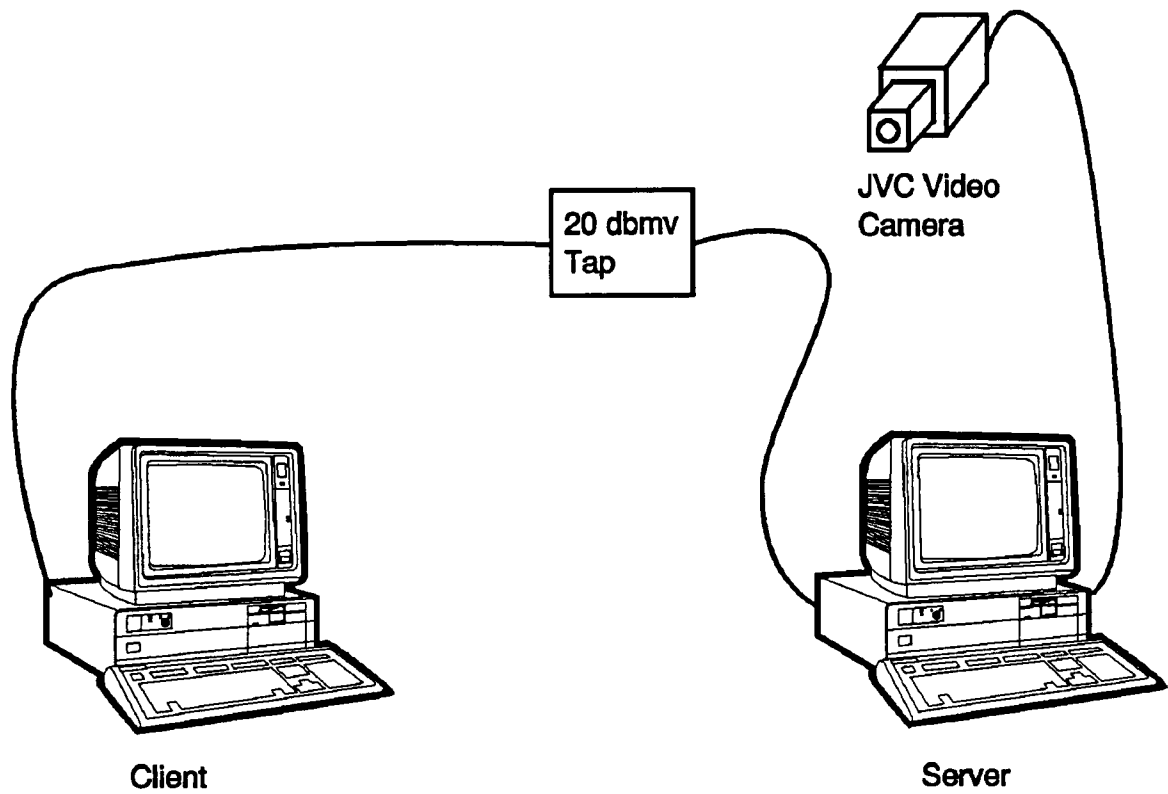


Figure 2.2. MAP 3.0 Carrierband Network Setup

The Monochrome Video Card

The Targa-M8 does not directly support normal MS-DOS video output. The monochrome card was the boot video display and was also used for prompts and status messages.

The Truevision Targa-M8 Image Capture Board

The image capture board used was the Targa-M8 from Truevision. This board is capable of capturing images at 256 levels of grey and at a resolution of 512 horizontal lines by 482 vertical lines. The inputs to the board consist of four RS-170 black and white composite video channels or one analog RGB channel for separate capture of the red, green, and blue components of an image. The output of the board can be RS-170 or analog RGB [6].

This board was chosen because of its intrinsic similarity to the 256 color mode of the IBM VGA video standard (and many Super VGA boards). Like the VGA, each pixel is represented by an 8 bit value that is an index into a color palette table. The palette table is made up of 256 24-bit values; 8 bits each for red, green, and blue. The main difference between the two video boards is that the VGA uses only the lower 6 bits of each color in the palette.

Though the Targa-M8 has color capabilities, they can only be realized by capturing the red, green, and blue portions of an image separately. Since the primary goal of this project was to explore the maximum frame rate of the

system, a single composite input was used. See Appendix E for configuration information and cable pin-outs.

The JVC TK-870U RGB Video Camera

The video source for the Targa-M8 was a JVC TK-870U RGB camera. This camera has both composite color and RGB color outputs. The lens used on the camera was a JVC HZ-C611AFU with auto-focus, auto-iris, and remote zoom.

2.5 The Image Client

The host machine for the image client was an Everex IBM PC/AT compatible personal computer running at 16MHz with 1MB of memory (640K is required for operation). This machine contained the networking card and an Everex Viewpoint VGA card.

The Everex Viewpoint VGA Card

The Everex VGA card is a sixteen-bit, three-quarter length card that provides full VGA compatibility as defined by IBM as well as several extended modes not defined in the VGA standard [7]. The main reason for choosing the Everex card is that one of the extended modes, 512 by 480 by 256 colors, is an excellent match for the mode used to capture images on the Targa board (512

by 400 by 256 grey levels).

The match in horizontal resolution and number of bits-per-pixel means that practically no mapping is necessary when transmitting data from the Targa video buffer to the Everex video buffer even if the number of vertical lines is different. In fact, to take a page (64KB) of memory from the Targa and place it in a page of the Everex card requires only that the data be reverse-ordered (the last pixel becomes the first, and the first pixel becomes the last).

By loading the palette registers of the VGA with a set of linear grey scales corresponding to the palette of the Targa board, a reverse data dump from the Targa video memory to the network and on to VGA memory is possible. Since nothing is swapped to disk, the fastest transfer times, given the equipment used, may be obtained.

CHAPTER 3

METHODOLOGY

3.1 Overview

The implementation of the image transfer mechanism was developed using the Systems Integration Specialists Company, Inc. (SISCO) MMS Embedded Application Service Element (MMS-EASE) interface library. The MMS-EASE library provides MMS services for programming in the 'C' language. The problem of image transfer was solved with two logically different methods under MMS. The first method was file transfer, and the second was a domain upload.

Both of these methods are block oriented. In a transfer under either method, the client prepares the server for a transfer by sending a preamble request with information on what data it wants to receive, and then requests blocks of data until the server indicates an end of transfer. The primary difference in the methods is the logical organization of the data at the server, and the format in which the client requests the data.

The size of the blocks is determined as part of the negotiation that takes place when the client first connects to the server and is generally the largest common size that both nodes support. For this research, the block size was controlled by the programmer.

3.2 SISCO's MMS-EASE

MMS-EASE implements a subset of the International Standards Organization (ISO) Draft International Standard (DIS) 9506 protocols of MMS under MAP. The product consists of a library of 'C' function calls and data structures that allow portable MMS code to be developed. In addition to primitive level interfaces to the variable access, context management, domain, file, and semaphore services of MMS, variable access and file functions can be supported using a higher level "virtual machine" interface [8].

"Primitive" in the MMS-EASE sense does not connote obsolete or unusable functions, but that the service provided is very basic and has no inherent capability. In the case of file transfer, primitive means that the functions provide the ability to pass filenames and blocks of data but do not interact with the host file system. The programmer is responsible for opening, reading, writing, closing, or otherwise manipulating the host machine file system. This provides the programmer with a great deal of flexibility.

The "virtual machine", by contrast, implements a level of logical functionality. Again using the file functions as an example, an entire file download or upload may be accomplished by calling a single function and waiting until it is completed. While in some cases the virtual machine is simpler to use, like most high level functions a sacrifice is made in performance and flexibility.

An advantage to using an interface library like MMS-EASE is that it provides device independence from various network boards, host machines, and even MAP network versions. The image transfer software was first developed under MAP 2.X and then ported to MAP 3.0. Because the function names did not change in MMS-EASE, only a dozen lines of code specific to the network version and network card had to be changed. Table 3.1 shows the specific changes that were necessary.

Table 3.1. Source Code Changes From MAP 2.X to MAP 3.0

MAP 2.X	MAP 3.0
<code>#include "suicase.h"</code>	<code>#include "suicacse.h"</code>
<code>case_size = B_SIZE+OVRHEAD</code>	<code>_msgsize = B_SIZE+OVRHEAD</code>
<code>SUIC_int_no = 2</code>	<code>s_int_no = 3</code>
	<code>s_int_en = TRUE</code>
	<code>s_dma_chan = 3</code>
	<code>s_dma_en = TRUE</code>
<code>boot_talk_en = FALSE</code>	<code>s_boot_talk_en = TRUE</code>
<code>printen = 1</code>	

The two versions used for this research were MMS-EASE-400-001 level 4.05 for the INI series of PC based cards under MAP 2.X and MMS-EASE-120-001 level 4.05 for the Concord 1200 series of PC based cards under MAP 3.0. Hosts supported include the IBM PC family, VAX, micro VAX, and Motorola Delta series. Network cards supported include the Concord 1200 and 1400

series, Interlan NP-642, DEC/MAP, DECnet, and Computrol LP-25 and LQ-24. The INI MP-400 cards do not support MAP 3.0 and SISCO no longer supports them.

Unlike the usual method of top-down coding, the MMS-EASE system must be written in an event-driven manner [8]. This means that program flow is driven by events from the network that may occur at any time, and in some cases, in any order. Events that are important to the program are filtered out for special attention. To do this in a non-multitasking operating system such as MS-DOS, it is necessary to set up the program in an overall polling loop. The loop first checks for user input, and if none is detected calls the network service function. In MMS-EASE, the network service function is "comm_serve".

The functions that start with "u_" in the programs have names that are defined by MMS-EASE, but the contents of the functions are written by the applications programmer. These functions are called when the appropriate message is received from the network. For instance, when a file open indication is received at a server, comm_serve calls the "u_fopen_ind" function, which may then perform whatever actions the programmer finds necessary to open the file and prepare for a file read or write.

3.3 The Targa-M8 Video Buffer

The Targa-M8 video buffer is 256 KB in size. When in the 512 by 400 mode (as in these experiments), only 200 KB are actually used (there is another mode that allows four 256 by 256 images to be captured which uses all 256 KB). Since there is only 384 KB of address space available for memory mapped devices on IBM compatible PC's, the 256 KB of memory is mapped into a 64 KB "window" at address A0000 hex [9]. The window can be moved in the Targa address space so that all the buffer memory is available 64 KB at a time. Figure 3.1 shows this process in respect to the IBM PC architecture.

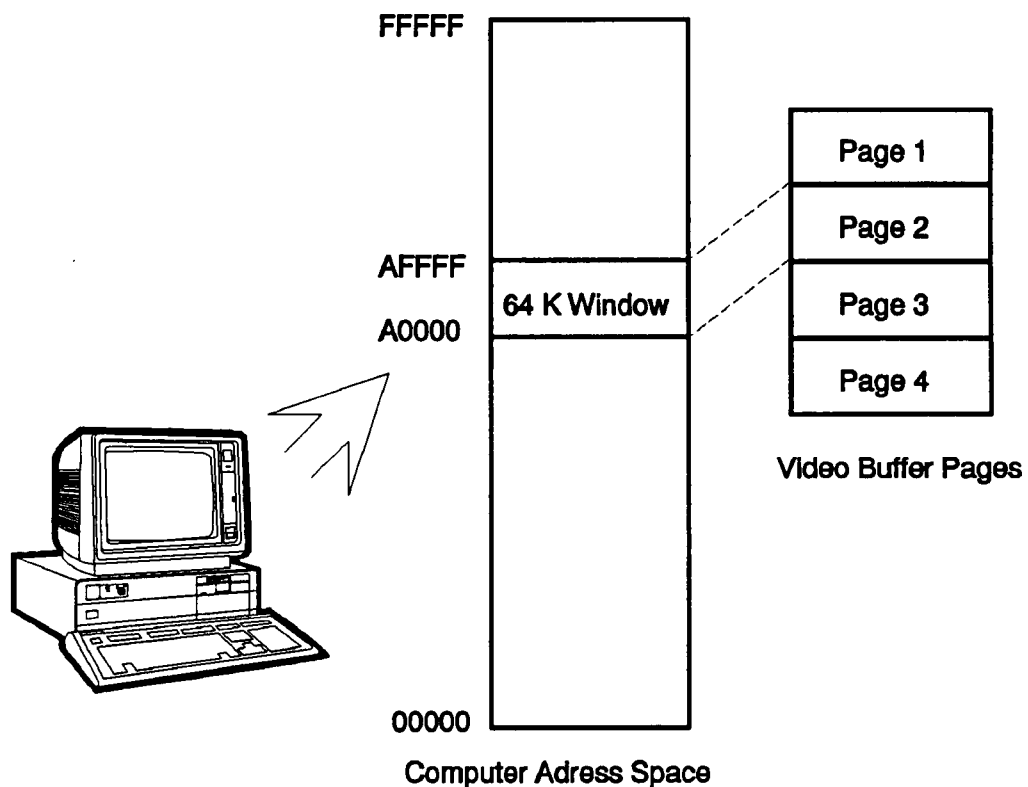


Figure 3.1. Memory Windowing of Video Buffer

The manipulation of the Targa-M8 board was accomplished by using the Targa Tools product. This is a set of 'C' library routines for simple control of the image capture board. Display modes, image capture triggering, data transfer, page selection, and other operations can be done by issuing function calls from the library [10]. The functions used and their purposes are given in Table 3.2.

Table 3.2. Targa Tools Function Calls

Function	Purpose of Function
GetRow	Fill a buffer with a row of pixels
GetScrSize	Return actual screen size
GrabFrame	Capture an Image
GraphInit	Initialize Targa board
SetBLiveMode	Display what the camera is viewing
SetDispMode	Display the captured image
SetPageMode	Set desired display mode

3.4 The Everex Viewpoint VGA Video Buffer

The Everex VGA card is similar to the Targa in that the video buffer uses an almost identical windowing scheme. The Everex VGA can have up to 512 KB of video memory, although only 256 KB were necessary for the resolution used in the experiments. The address of the 64 KB window is again A0000 hex as shown in Figure 3.1. The implementation of the paging

mechanism by Everex made necessary the coding of the routines for paging and extended mode setup in assembly [7]. The assembly functions were written to be called as external functions to 'C'. The routines for setting up the palette registers for the grey scale were coded directly in the 'C' language. The functions used and their purposes are shown in Table 3.3.

Table 3.3. Everex VGA Function Calls

Function	Purpose of Function
SetExtendedMode	Set graphics mode (resolution etc.)
SetExtendedPage	Set video page (64K window location)
LoadDacs256	Set palette for 256 levels of grey
LoadDacs64	Set palette for 64 levels of grey

3.5 The File Transfer Method

The file transfer method of image transfer was implemented by using the primitive file functions of MMS-EASE to treat a video buffer as a "memory file". By mapping the video memory of the Targa-M8 to a logical MMS concept, the video buffer can be downloaded as a file by any program that incorporates a normal MMS file transfer capability. Instead of retrieving data from a disk drive, it is instead taken from video memory.

Connecting to the Server

Before any other network functions can take place a connection must be established between the client and the server. Table 3.4 lists the functions necessary to accomplish a connection, and Figure 3.2 shows the flow of execution.

Both client and server must activate and register their respective "network names" on program startup. Network names are a form of shorthand for network addresses (see Appendix B for more information on setting up these names). A registered name is "visible" to other stations on the network. In addition, the server must "post a listen" on a channel in order to indicate that it is willing to receive a connection request.

Table 3.4. MMS-EASE Connection Oriented Functions

Function Name	Purpose of Function
mlp_act_ar_name	Activate logical name
mlp_reg_ar_name	Register logical name
mlp_ass_listen	Post listen on a channel
mv_init	Request connection
u_init_ind	Indication of connection
mv_init_resp	Response to connection
u_mv_init_conf	Confirmation of connection

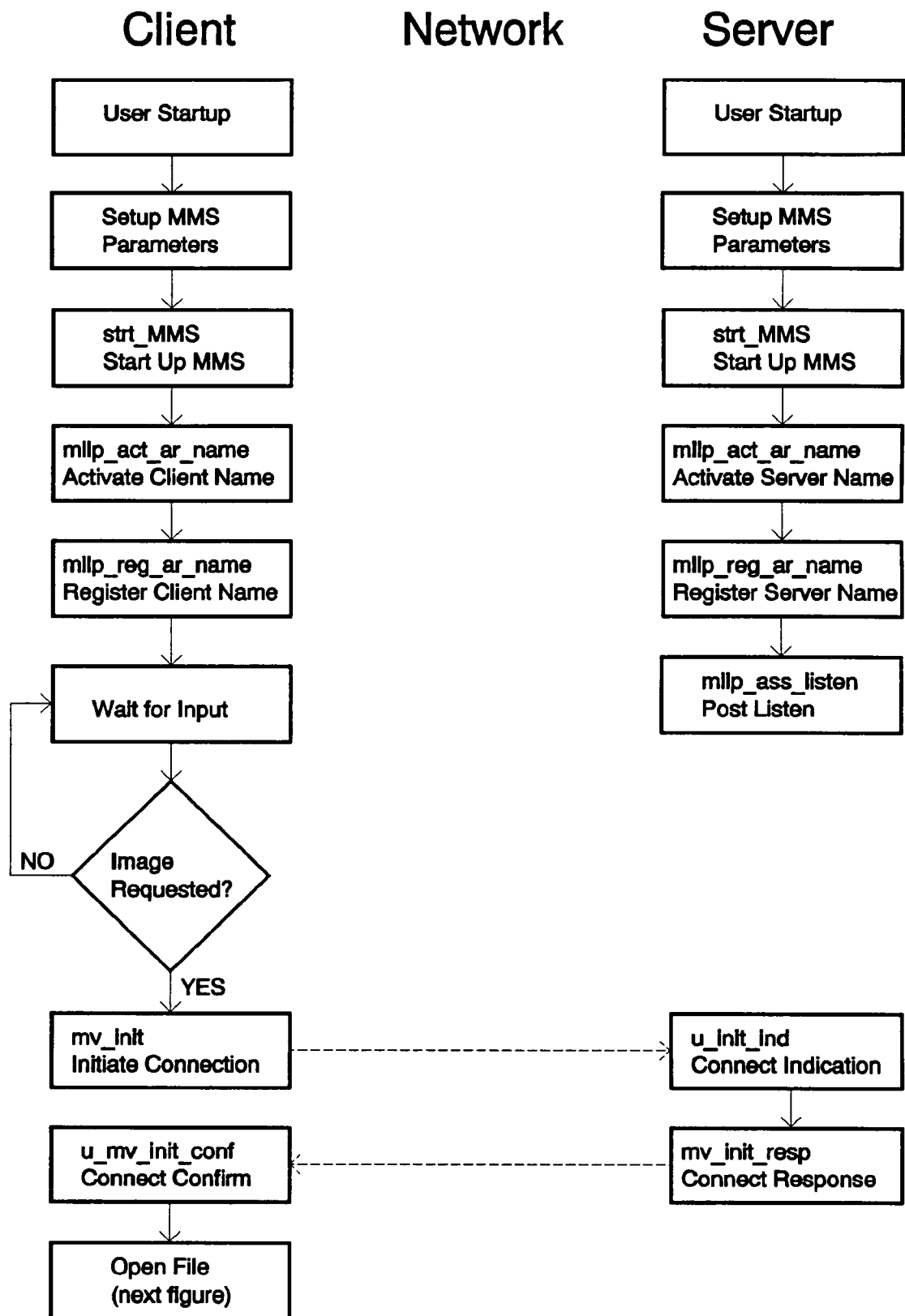


Figure 3.2. Flow Diagram of Connection

The client program, after being directed by the user to receive an image, issues an `mv_init` call which appears at the server as a `u_init_ind` function. The server responds with a `mv_init_resp` which generates a `u_mv_init_conf` at the client. After processing the confirm, the client and the server are connected.

Opening the File

The first logical step in reading or writing file data in almost any operating system is to open the file. The file open is used primarily to let the server know which file is to be operated on. The functions necessary for opening a file under MMS-EASE are listed in Table 3.5, and Figure 3.3 shows the flow of execution.

Table 3.5. MMS-EASE File Open Functions

Function Name	Purpose of Function
<code>mp_fopen</code>	Request file open
<code>u_fopen_ind</code>	Indication of file open
<code>mp_fopen_resp</code>	Response to file open
<code>u_mp_fopen_conf</code>	Confirmation of file open

The client first supplies a file name to `mp_fopen` which results in a `u_fopen_ind` at the server. The server then initializes the Targa board and

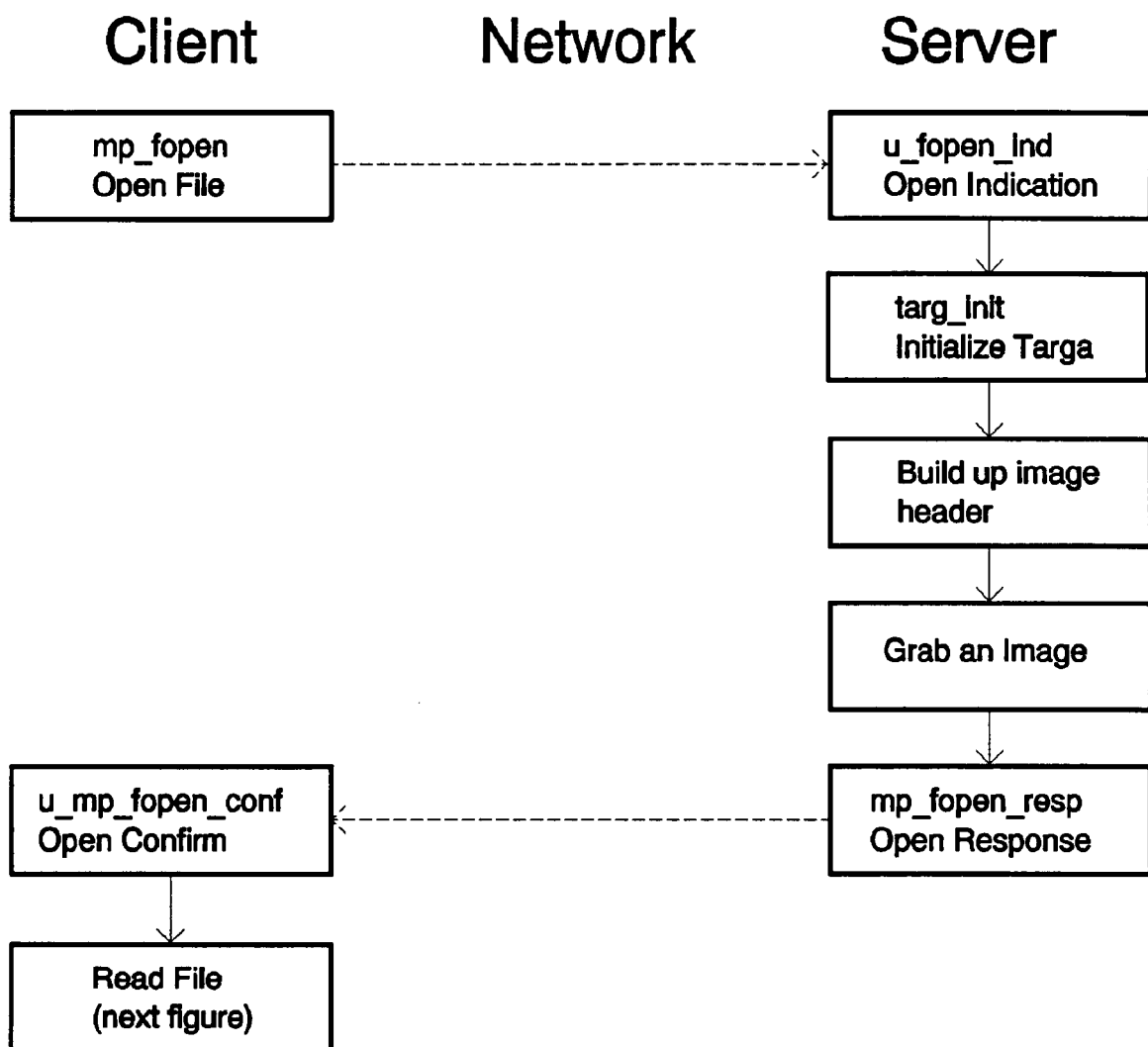


Figure 3.3. Flow Diagram of File Open

builds up a file header. The file header contains information about the image such as horizontal and vertical resolution (as queried from the Targa board), compression methods, and the version of the software that produced it. Some of this information (compression method and resolution) is necessary to interactively display the image during the transfer and for other programs to correctly display the image from a file.

The buffer for an anticipated read is setup so that when it is received the header may be immediately sent. Finally, an image is captured from the Targa board and the server calls `mp_fopen_resp`. The client receives a `u_mp_fopen_conf` and can proceed to read the file.

Reading the File

Once the file is opened, the client can request file data. The functions necessary to read from the file are listed in Table 3.6, and Figure 3.4 shows the flow of execution.

Table 3.6. MMS-EASE File Read Functions

Function Name	Purpose of Function
<code>mp_fread</code>	Request file read (one block)
<code>u_fread_ind</code>	Indication of file read
<code>mp_fread_resp</code>	Response to file read
<code>u_mp_fread_conf</code>	Confirmation of file read

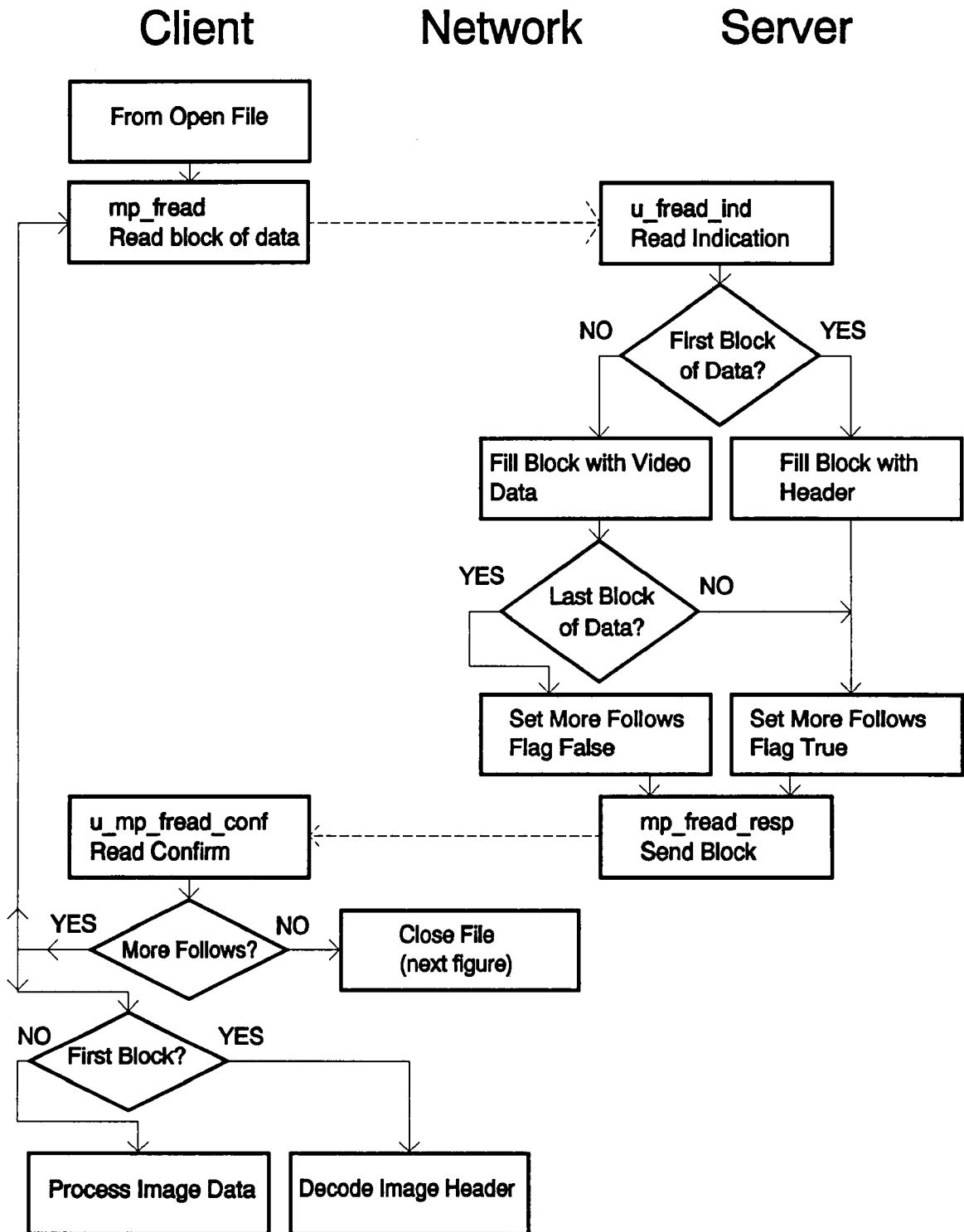


Figure 3.4. Flow Diagram of File Read

The client begins by calling `mp_fread`. The server is notified by a `u_fread_ind` and checks to see if this is the first block requested and therefore the file header. If so, the server copies the data in the header to a holding buffer and sets the "more follows" flag to `TRUE`. The server calls `mp_fread_resp` to send the holding buffer (and the "more follows" flag) to the client. The client, upon receiving a `u_mp_fread_conf`, checks the condition of "more follows". If there is more data, the client immediately sends for the next block. If the block sent was the header, the client copies the information to a header structure so that the information is available for use. The client will continue to request blocks in this manner until the server sets the "more follows" flag to `FALSE`. All blocks past the first are treated as image data. When the last block is finally sent, the client will proceed to the file close section.

Closing the File

After the file has been read, it should be closed so that the server may release file handles, buffers, and other resources. The functions necessary for a file close are listed in Table 3.7 and Figure 3.5 shows the flow of execution.

To close the file the client calls `mp_fclose`. The server receives a `u_fclose_ind` and immediately calls `mp_fclose_resp`. After the client receives the resulting `u_mp_fclose_conf`, it is ready to break the connection.

Table 3.7. MMS-EASE File Close Functions

Function Name	Purpose of Function
mp_fclose	Request file close
u_fclose_ind	Indication of file close
mp_fclose_resp	Response to file close
u_mp_fclose_conf	Confirmation of file close

Breaking the Connection

The final stage in the image transfer is to break the connection between the client and the server. The functions required for disconnection are listed in Table 3.8 and Figure 3.5 shows the flow of execution.

Table 3.8. MMS-EASE Disconnect Functions

Function Name	Purpose of Function
mp_conclude	Request disconnect
u_conclude_ind	Indication of disconnect
mp_conclude_resp	Response to disconnect
u_mp_conclude_conf	Confirmation of disconnect
u_mp_conclude_done	Resources for connection freed

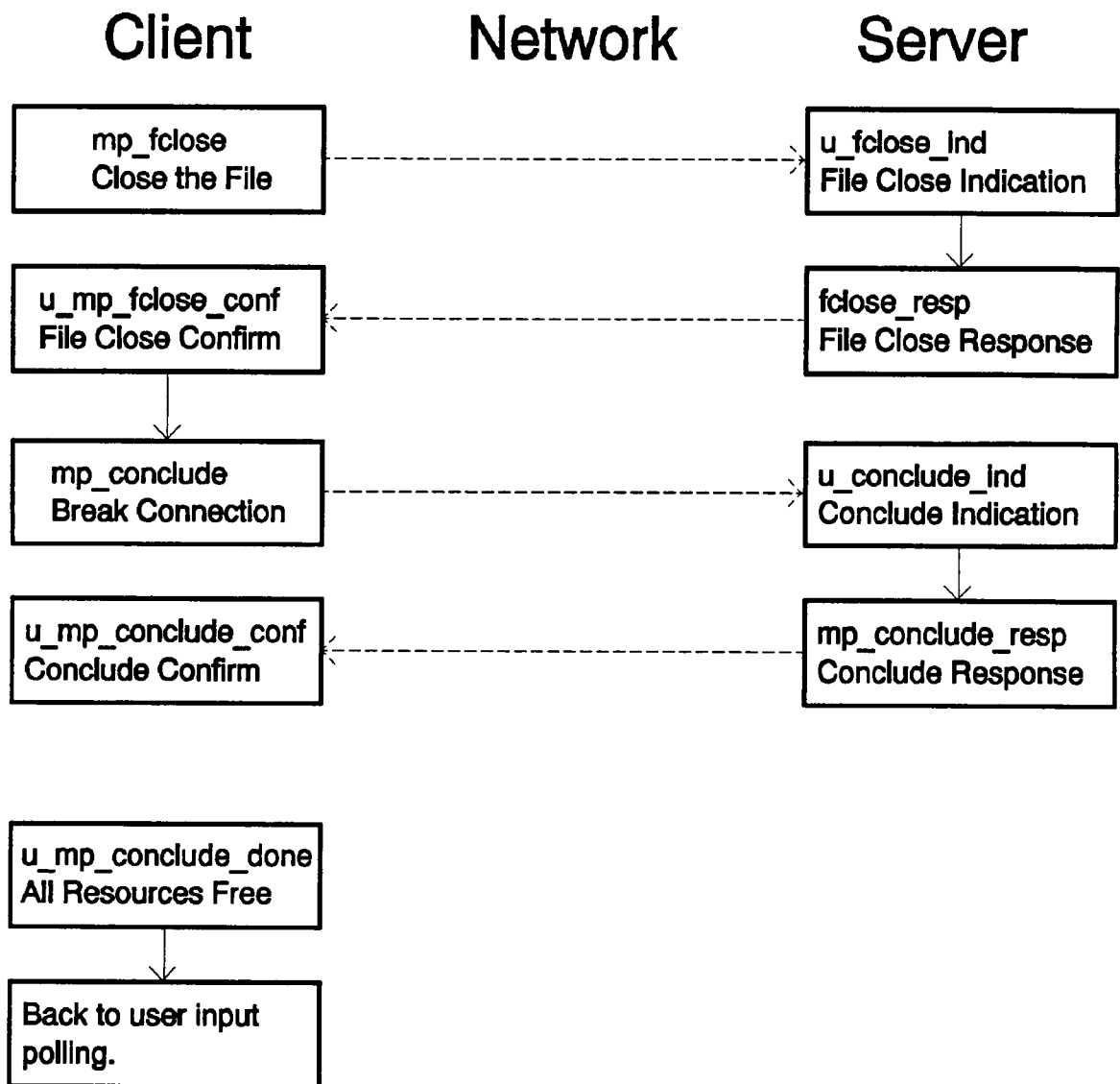


Figure 3.5. Flow Diagram of File Close and Disconnect

The client requests a disconnect by calling `mp_conclude` which is received at the server as a `u_conclude_ind`. The server responds with a `mp_conclude_resp` which results in a `u_mp_conclude_resp` at the client. The channel that connected the client to the server is not free, however, until the client receives a `u_mp_conclude_done` from the MMS-EASE system.

Data Compression

It was observed that between a request for the next block of data and the actual arrival of the data, there was a time lag. In order to increase the throughput of the image transfer, an elementary data compression scheme was devised to put that time to productive use.

Since the VGA standard can only display 64 true shades of grey (6 bits each for red, green, and blue : $2^6 = 64$), only the most significant 6 bits of each 8-bit pixel were needed [11]. By using a bit shifting algorithm as shown in Figure 3.6, every 4 pixels could be transferred in 3 bytes, a space savings of one byte in four, or 25%.

Since the system is event driven, the flow of control in the file-read section of code shown in Figure 3.4 does not change. Whenever the server switches video pages during the normal course of the transfer, it compresses the entire page. Thus, when data requests are received from the client, the data to send it is already compressed. Because the client decompresses each block

as it is received, the client and server must use block sizes that are evenly divisible by three (four pixels equal three bytes).

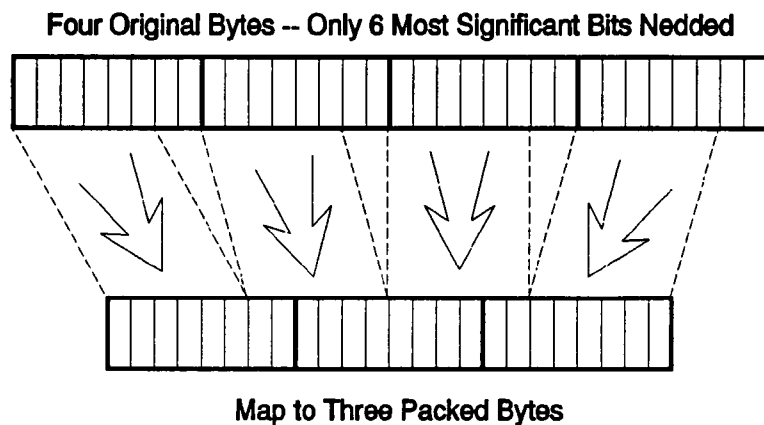


Figure 3.6. Bit Packing Data Compression Scheme

3.6 The Domain Upload Method

Domains under MMS can represent logical devices or areas of control on a given machine. The purpose of domains is similar to the concept of a directory on most computer file systems. A domain is a way to organize logical functions [12]. The domain functions, like the file functions used previously, were implemented with primitive MMS-EASE functions.

The basic functions allow for creating, transferring, and otherwise manipulating the logical domain structure, but they do not provide the logic and flow necessary to do practical work with the domain. Once again, the

opportunity exists to map the video memory to a logical MMS concept, such as an "image domain".

Creating the Domain and Connecting to the Server

The domain under which the Targa-M8 was organized had to be created and initialized on the server before a connection was established. The function necessary to create the domain is "ms_add_named_dom". As with the file transfer method, in the domain upload method a connection must be established between the client and the server. Figure 3.7 shows the flow of domain creation and connection. The connection-oriented functions are the same as those in Table 3.4, page 18.

The "ms_add_named_dom" function returns a pointer to a structure which contains control information for the domain. Of primary importance in the structure are two variables - "state" and "loadable". In order for the client to upload the domain, the "state" variable must be set to DOM_READY, and the "loadable" variable must be set to TRUE. After the domain is created, the rest of the startup routines are the same as in file transfer.

Initiating the Domain Upload

In much the same way a file must be opened before it is read, an initiating procedure is necessary before uploading a domain. The name of the domain to be manipulated is provided to the server from this mechanism.

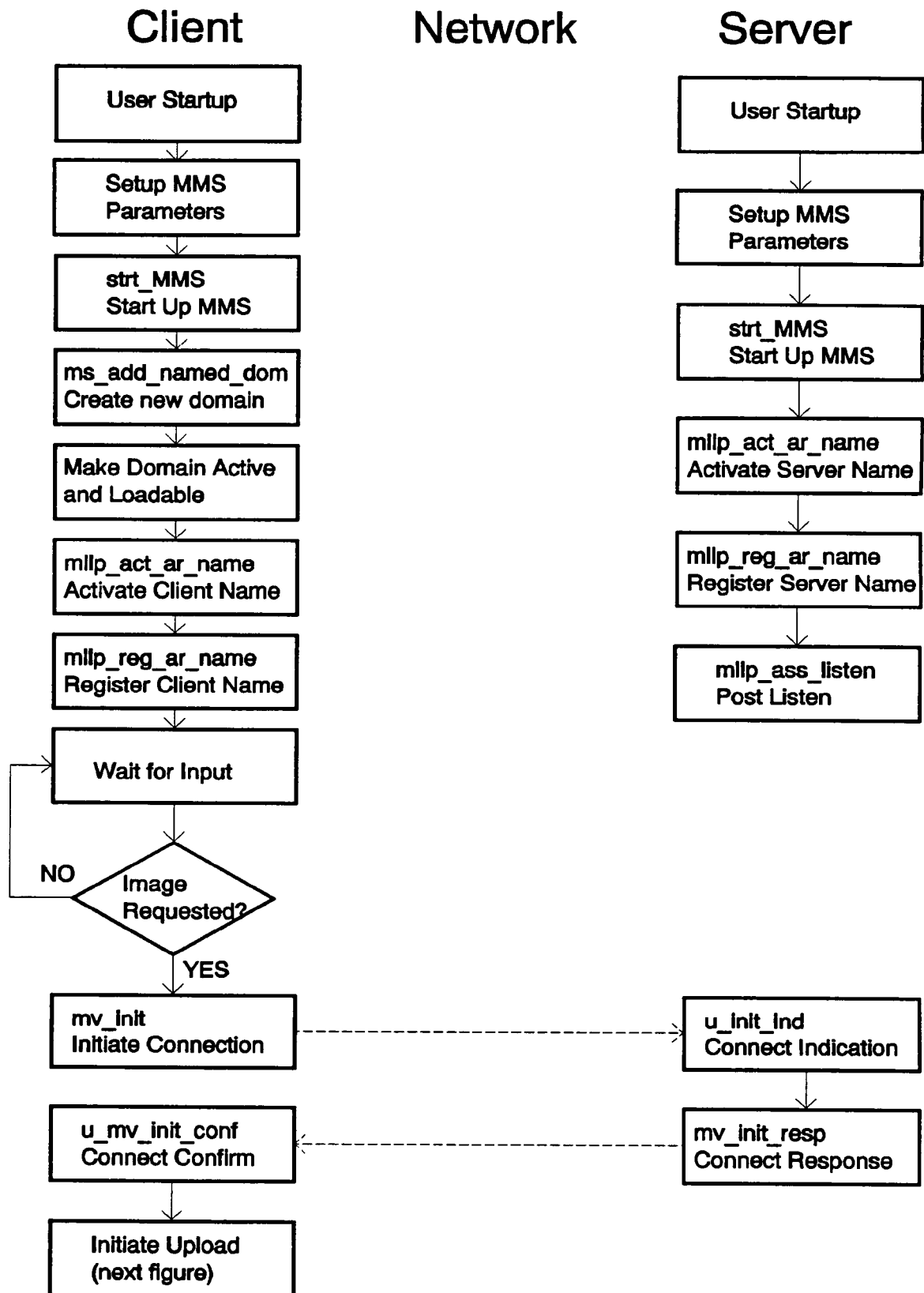


Figure 3.7. Flow Diagram of Domain Creation and Connection

The functions required to initiate a domain upload are listed in Table 3.9, and Figure 3.8 shows the flow of execution.

Table 3.9. MMS-EASE Initiate Upload Domain Functions

Function Name	Purpose of Function
mp_initupl	Requests a domain upload
u_initupl_ind	Indication of upload request
mp_initupl_resp	Response to upload request
u_mp_initupl_conf	Confirmation of upload request

The client first supplies the domain name to mp_initupl which results in a u_initupl_ind at the server. The server then initializes the Targa board and builds up a header. This header is exactly the same as that used in the file transfer method.

The buffer for an anticipated upload block is set up so that when it is received the header may be sent immediately. As before, an image is captured from the Targa board. The server then calls mp_initupl_resp, which the client receives as a u_mp_initupl_conf.

Uploading the Domain

When the domain has been initialized for upload, the client can request blocks of data.

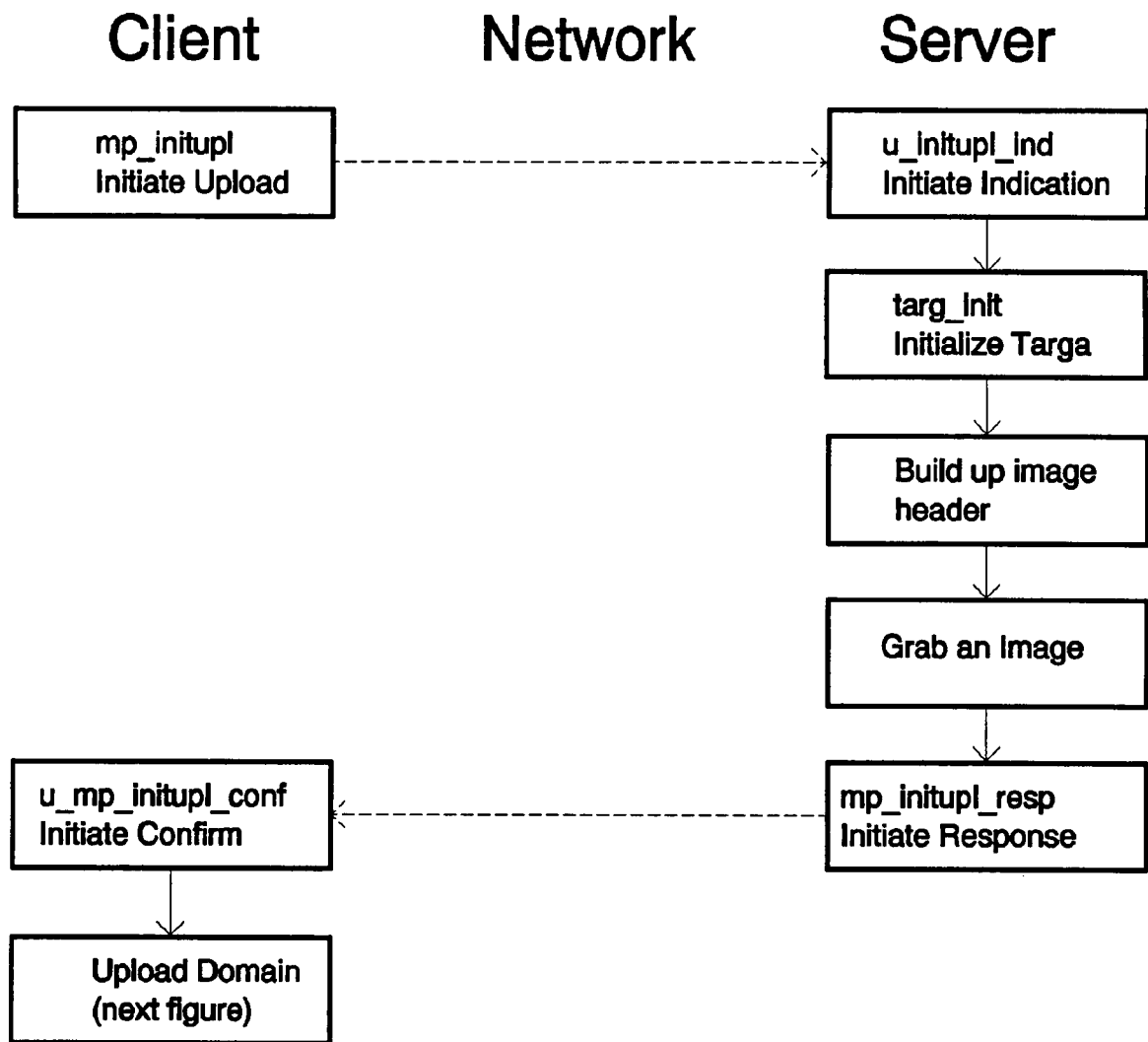


Figure 3.8. Flow Diagram of Domain Initiate Upload

The functions required for uploading the domain are listed in Table 3.10, and Figure 3.9 shows the flow of execution.

Table 3.10. MMS-EASE Upload Domain Functions

Function Name	Purpose of Function
mp_upload	Requests a block of domain data
u_upload_ind	Indication of block request
mp_upload_resp	Response to block request
u_mp_upload_conf	Confirmation of block request

The client begins by calling mp_upload. The server is notified by a u_upload_ind; it then checks to see if this is the first block requested and, therefore, the file header. If so, the server copies the data in the header to a holding buffer and sets the "more follows" flag to TRUE. The server calls mp_upload_resp to send the holding buffer (and the "more follows" flag) to the client. The client, upon receiving a u_mp_upload_conf, checks the condition of "more follows". If there is more data, the client immediately sends for the next block. If the block sent was the header, the client copies the information to a header structure so that the information is available for use. The client will continue to request blocks until the server sets the "more follows" flag to FALSE. All blocks past the first are treated as image data. When the last block is finally sent, the client proceeds to the terminate upload section.

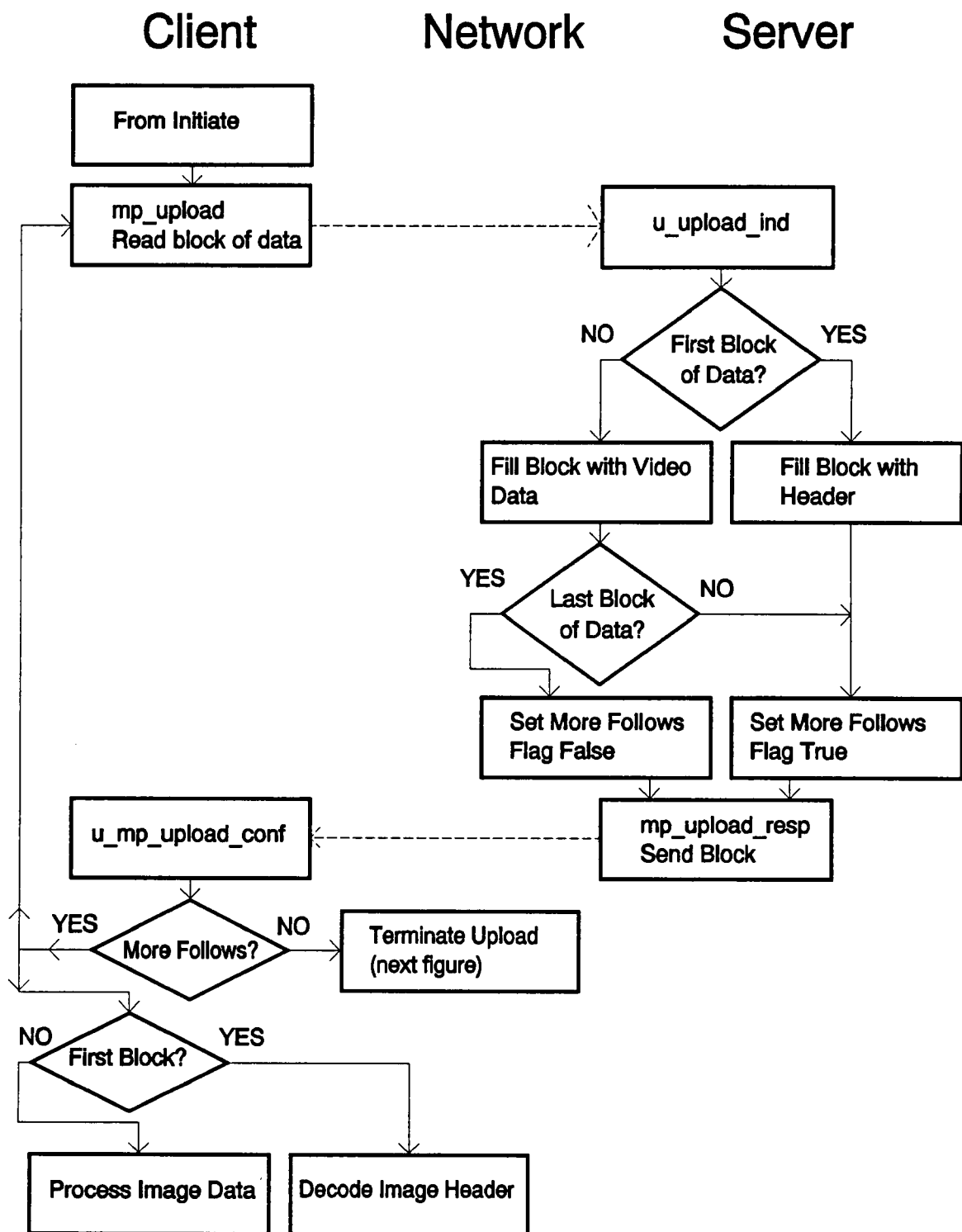


Figure 3.9. Flow Diagram for Domain Upload

The client begins by calling `mp_upload`. The server is notified by a `u_upload_ind`; it then checks to see if this is the first block requested and, therefore, the file header. If so, the server copies the data in the header to a holding buffer and sets the "more follows" flag to `TRUE`. The server calls `mp_upload_resp` to send the holding buffer (and the "more follows" flag) to the client. The client, upon receiving a `u_mp_upload_conf`, checks the condition of "more follows". If there is more data, the client immediately sends for the next block. If the block sent was the header, the client copies the information to a header structure so that the information is available for use. The client will continue to request blocks until the server sets the "more follows" flag to `FALSE`. All blocks past the first are treated as image data. When the last block is finally sent, the client proceeds to the terminate upload section.

Terminating the Upload

Similar to the closing of a file, the upload must be terminated after all data has been acquired. The functions necessary to terminate an upload are listed in Table 3.11, and Figure 3.10 shows the flow of execution.

The client first calls `mp_termupl`, which is received on the server as a `u_termupl_ind`. The server responds with a `mp_termupl_resp`, which generates a `u_mp_termupl_conf` at the client.

Table 3.11. MMS-EASE Terminate Upload Domain Functions

Function Name	Purpose of Function
mp_termupl	Requests termination of upload
u_termupl_ind	Indication of termination
mp_termupl_resp	Response to termination
u_mp_termupl_conf	Confirmation of termination

Breaking the Connection

The disconnect proceeds as before in the file transfer section. Figure 3.10 shows the flow of the process.

Data Compression

The data compression scheme was also used in the domain upload. The process was the same as in the file transfer method.

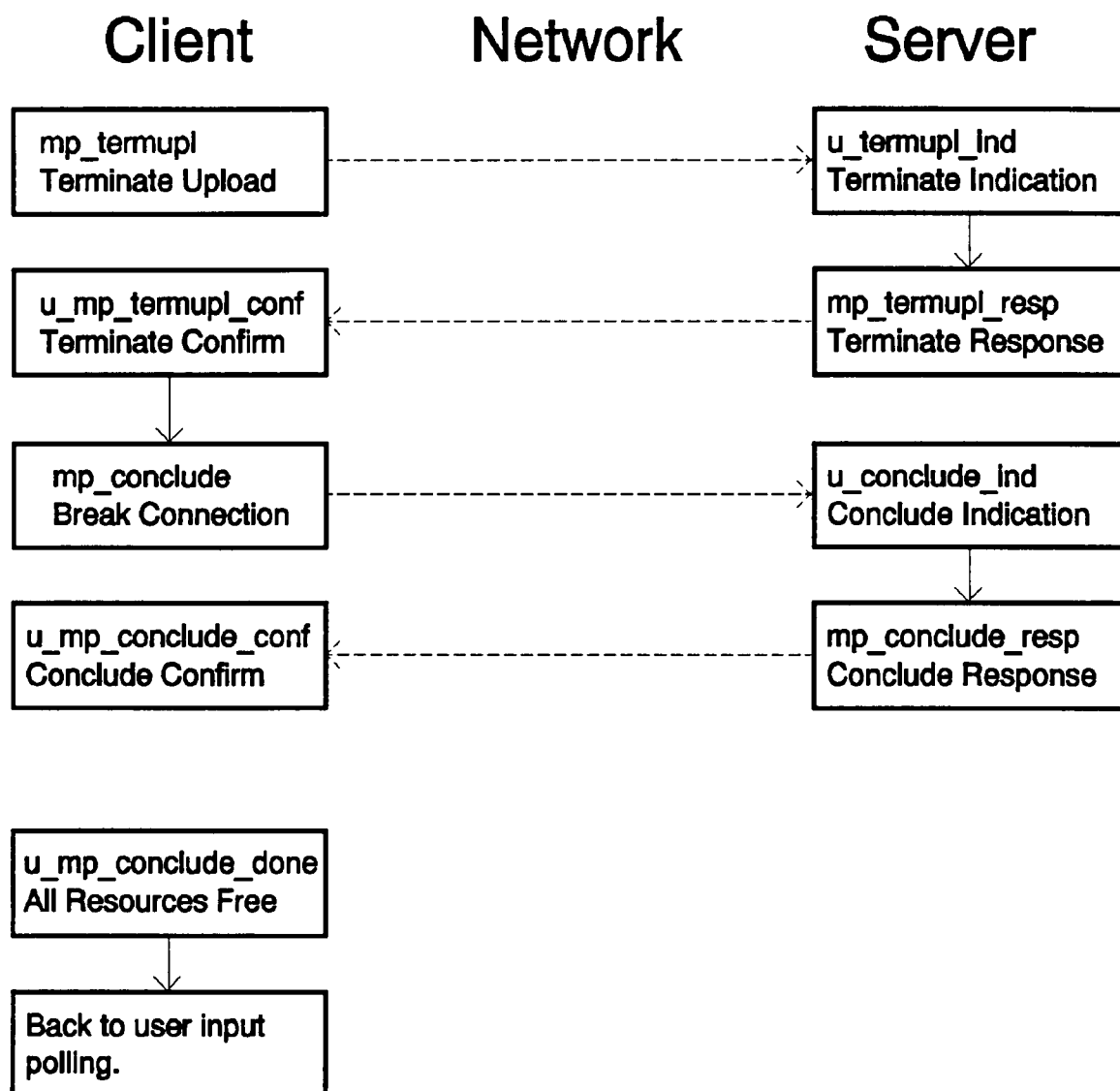


Figure 3.10. Flow Diagram of Terminate Upload and Disconnect

CHAPTER 4

RESULTS

4.1 Overview

The results for both the MAP 2.X and the MAP 3.0 networks were measured in terms of the number of seconds required to transfer an image and the throughput in kilobytes per second. Both real and effective values are given for throughput since compression of the data results in a higher perceived speed.

Measurements were made using three different block sizes (255 bytes, 513 bytes, and 1002 bytes) to determine the effect of block size on the overall transmission time of a 200 KB image. Each block size was used both with and without data compression for a total of six variations under each network version. Specific measurements for each transfer method were made, and as expected, were identical between file transfer and domain upload. The reason for this phenomenon is that the actual functionality of moving the data is the same in both methods -- only the organization of the data at the server is different. The tables in the following sections, therefore, apply to both file transfer and domain upload methods. Tables 4.1 and 4.2 show the results of the experiments under MAP 2.X using 10 Mbps broadband technology. Tables 4.3 and 4.4 show the results under MAP 3.0 using 5 Mbps carrierband technology.

4.2 MAP 2.X Broadband Results

Table 4.1. MAP 2.X Broadband Results Without Compression

Measurement	Block Size		
	255 Bytes	513 Bytes	1002 Bytes
Transfer Time	65 sec	40 sec	25 sec
Real Throughput	3.07 KBps	5.00 KBps	8.00 KBps
Effective Throughput	3.07 KBps	5.00 KBps	8.00 KBps

Table 4.2. MAP 2.X Broadband Results With Compression

Measurement	Block Size		
	255 Bytes	513 Bytes	1002 Bytes
Transfer Time	51 sec	31 sec	20 sec
Real Throughput	2.94 KBps	4.83 KBps	7.50 KBps
Effective Throughput	3.92 KBps	6.45 KBps	10.00 KBps

4.3 MAP 3.0 Carrierband Results

Table 4.3. MAP 3.0 Carrierband Results Without Compression

Measurement	Block Size		
	255 Bytes	513 Bytes	1002 Bytes
Transfer Time	108 sec	60 sec	39 sec
Real Throughput	1.85 KBps	3.33 KBps	5.12 KBps
Effective Throughput	1.85 KBps	3.33 KBps	5.12 KBps

Table 4.4. MAP 3.0 Carrierband Results With Compression

Measurement	Block Size		
	255 Bytes	513 Bytes	1002 Bytes
Transfer Time	80 sec	45 sec	29 sec
Real Throughput	1.87 KBps	3.33 KBps	5.17 KBps
Effective Throughput	2.50 KBps	4.44 KBps	6.89 KBps

CHAPTER 5

CONCLUSION

5.1 Overview

Two methods, file transfer and domain upload, were explored in this thesis to assess the advantages of each particular method of organizing image data. Of the two methods, file transfer offers the more flexible solution. In order to upload a domain correctly, the client must know exactly how the data is represented on the server. In a file transfer, however, any MMS program with file transfer capabilities can obtain the data. To a file transfer program, the video buffer data appears to be a file. The primary benefit gained from that property is that a non-graphics capable computer can still acquire the image and perform analysis on it as the situation demands.

Since the difference in these methods is organizational and not operational, transfer times under each technique were identical. It is surmised that transfer time could be significantly reduced by further increasing the block size used. Unfortunately, for the releases of MMS-EASE used, blocks of this size had not previously been encountered, and a limitation surfaced that would not allow more than 1002 bytes of data to be sent at one time. This limit is not an inherent property of MMS or MMS-EASE but is the consequence of the

insufficient size of a static ASN.1 encoding buffer. A future release of MMS-EASE can be expected to address this problem.

If the trend observed when going from 255-byte blocks to 513-byte blocks to 1002-byte blocks holds for even larger block sizes, transfer times below 10 seconds would seem reasonable since MAP allows a block size of over 7KB. As seen in the results section, when the block size is increased from 255 to 513 bytes, throughput increases between 1.6 and 1.8 times. When block size is increased from 513 to 1002 bytes, throughput is increased between 1.5 and 1.6 times. An investigation of this phenomenon with a frame analyzer reveals the reason for the significant difference in throughput. The length of time required to complete a "send data - acknowledge data - request more data - acknowledge request" sequence and the number of bytes required to send the sequence were observed. It was found that a significant amount of overhead was induced by the various layers of the network stack, and the actual percentage of data sent as a function of the total number of bytes varied. The larger the block size, the less percentage overhead. Another source of overhead is caused by ASN.1 encoding. Each data block sent is encoded as "user specific data" by ASN.1. This tag induces approximately 30 bytes of overhead for each data block sent. When the number of blocks is increased, both ASN.1 overhead and encoding time are increased. Table 5.1 shows the percentages observed for the three block sizes.

Table 5.1. Percentage Data Measurements

Data Size	Bytes Required to Send	Percentage Data
255	566	45.1%
513	979	52.4%
1002	1549	64.7%

Until a companion standard for image capture is written, mimicking a file with the video buffer of an image capture card provides the maximum in ease-of-use and compatibility with existing programs.

5.2 Future Work

Future work on this topic could encompass several areas. First, the addition of color support should be considered. Many documents use multiple colors for emphasis or clarity, and this quality is lost when captured or displayed as a monochromatic image. Preferably, a standard format for image transfer would be developed that allows variable resolution, color palette support, and image compression.

A second area of work could be that of throughput improvement. As already mentioned, increasing the block size used in transmission should significantly help matters in this area. Another way to improve transfer times would be to use better compression algorithms. For the best results (e.g. speed)

in a continuous image capture application, the compression algorithm should be implemented in hardware (and accessed through the presentation layer for compatibility).

The logical next step for this kind of research would be directed towards full motion video. A delayed-action, full-motion video configuration with limited duration can already be set up under MAP. By using large amounts of expanded memory on the server, images could be captured and dumped to memory in quick succession. The set of images could then be transferred to the client and played back at full speed.

LIST OF REFERENCES

LIST OF REFERENCES

- [1] Kaminski, Michael A. and Praveen, Dwivedi, "General Motors Manufacturing Automation Protocol, An Overview", General Motors Technical Center, Warren, MI., 1987.
- [2] Tanenbaum, Andrew S., Computer Networks, Prentice-Hall, Englewood Cliffs, N.J., 1988.
- [3] IEEE Standards for Local Area Networks -- Token-Passing Bus Access Method and Physical Layer Specifications, The IEEE, Inc., 1984.
- [4] Industrial Networking Incorporated, MP-400 Installation Guide, 1987.
- [5] Concord Communications, Inc., Series 1200 for PC Bus Systems Installation Instructions and Product Description, 1989.
- [6] Truevision Inc., TARGA-M8 User's Guide, 1987.
- [7] Everex Systems Inc., Viewpoint VGA Graphics Adapter Owner's Manual and Reference Guide, 1989.
- [8] Systems Integration Specialists Company Inc, MMS-EASE Reference Manual Revision 8, 1988.
- [9] Dettmann, Terry R., DOS Programmer's Reference, Que Corporation, Carmel, Indiana, 1988.
- [10] Truevision Inc, TARGA Software Tools Notebook, 1988
- [11] Hogan, Thom, The Programmer's PC Sourcebook, Microsoft Press, Redmond, 1988.
- [12] MMS Application Interface Specification -- MAP_Attachment 6 to MAP_Appendix 7, 1988.
- [13] Microsoft Corporation, Microsoft C 5.1 Optimizing Compiler User's Guide, 1987.

APPENDIXES

APPENDIX A

INI HEAD END CONFIGURATION

A Tektronix 2710 spectrum analyzer was used to set the transmit level of the head end so that the network cards' receive level was 3 dbmv. The measurement was obtained by putting one of the INI MP-400 cards into a "squawk mode" to get a continuous broadcast across the network. The "squawk mode" was activated by removing jumper W11 on the INI MP-400 TBC. The INI cards transmit at 47 dbmv.

APPENDIX B

INI NMC CONFIGURATION

To set up a software stack for a card, the user first must choose "Install a Station" from the top level menu of the NMC. Within this menu, the user should fill in the "Station #" and "Station Name". The station number is actually the stations' network address found on the back of each INI card. The station name is a convenient alias for the station address. There is also a "Net ID" and "Comments" section. The net id needs to be changed only if a non-standard network number is being used. The comment field can be anything the user wants, such as the building location of the station. After that information is entered, the user should press F2 and choose the card type from the list provided. The type selected was "MP_400" for the card type and "MP_TBC" for the load type.

After returning to the main menu, select "Configure a Station". After entering the station name (as given in "Install a Station" menu), press the carriage return key three times followed by F2. Within this menu are several choices. In this investigation, the MP-400 cards used in both the client and the server were configured identically. The sections below show the settings that were used.

"Buffers" Menu

The settings in this section were changed in order to observe the effect of block size on the speed of the image transfers. The data link buffers were not used by the software and were never changed. The configurations shown are for the different sizes of the transport buffers, which were the data carrying blocks. Block sizes of 255 bytes, 513 bytes, and 1002 bytes were used. The somewhat odd sizes were used in order to facilitate the simple compression algorithm incorporated into the code.

It should be noted that the transport receive buffers, which are the data carrying buffers, are configured 100 bytes larger than the amount of data to be transmitted. This is because MMS-EASE uses some of the block for its own overhead and for ASN.1 encoding information. Table B.1 shows the settings used for the INI MP-400 cards.

Table B.1: Configuration of Block Sizes

Buffer Type	Size	High Thresh	Low Thresh
Control Block	254	60	40
Data Link Receive	1120	5	2
Transport Receive	1120	5	2

"XLI" Menu

The software that was developed did not use any of the information in this section, and the configuration was as shown below in Table B.2.

Table B.2. XLI Configuration

ENTRY	SIZE	TASKID	SUBB- ADR	CLIENT TAG	USED
0	00 00	0000	0000	00000000	N
1	00 00	0000	0000	00000000	N
2	00 00	0000	0000	00000000	N
3	00 00	0000	0000	00000000	N
4	00 00	0000	0000	00000000	N
5	00 00	0000	0000	00000000	N
6	00 00	0000	0000	00000000	N
7	00 00	0000	0000	00000000	N
8	00 00	0000	0000	00000000	N
9	00 00	0000	0000	00000000	N
10	00 00	0000	0000	00000000	N
11	00 00	0000	0000	00000000	N
12	00 00	0000	0000	00000000	N
13	00 00	0000	0000	00000000	N
14	00 00	0000	0000	00000000	N
15	00 00	0000	0000	00000000	N

"DLI" Menu

The Data Link Interface menu is similar to the transport interface. Again, none of these parameters were used, and the section was set up as shown in Table B.3.

Table B.3. DLI Configuration

ENTRY	SAP	CONN	BUFFERS	TASKID	SUBADR	USED
0	00	FRAMES	RCV_POOL	0000	0000	N
1	00	FRAMES	RCV_POOL	0000	0000	N
2	00	FRAMES	RCV_POOL	0000	0000	N
3	00	FRAMES	RCV_POOL	0000	0000	N
4	00	FRAMES	RCV_POOL	0000	0000	N
5	00	FRAMES	RCV_POOL	0000	0000	N
6	00	FRAMES	RCV_POOL	0000	0000	N
7	00	FRAMES	RCV_POOL	0000	0000	N
8	00	FRAMES	RCV_POOL	0000	0000	N
9	00	FRAMES	RCV_POOL	0000	0000	N
10	00	FRAMES	RCV_POOL	0000	0000	N
11	00	FRAMES	RCV_POOL	0000	0000	N

"DL" Menu

Several operating parameters of the Data Link layer are configured in this section. The settings used are shown in Table B.4.

Table B.4. DL Configuration

Parameter	Value
MAX_RETRY_LIMIT	00003
MIN_RX_BUFLN	1100
TX_BUF_AREA	4000
HI_PRI_TOKEN_HOLD	31250
MAX_AC_4	300000
MAX_AC_2	125000
MAX_AC_0	62500
MAX_RING_MAINT	2000000
RING_MAINT_TIMER_INIT	2000000
MAX_INTER_SOLICIT	255
DATA_LINK_BUFFER_AREA	4C00

"XPORT" Menu - Several operating parameters of the transport layer are configured in this section. The settings used are shown in Table B.5

Table B.5. XPORT Configuration

Parameter	Value
MAX_PDUSZ_LOG2	9
REF_WAIT_TIMER	0
MIN_RCV_WIND	15
SENDER_SUFFIX	00 01
RECEIVER_SUFFIX	00 01
MAX_CREDITS	1
USE_CHECKSUM	N
USE_EXPEDITED	Y
SEQ_SPACE	31 SEQ-16 CRT
DTLT_RETRIES	5
DFLT_RETRIES	10
AK_DALLY_TIME	2
HEARTBEAT_TIME	15
GIVEUP_TIME	10
INACTIVITY_TIME	120
RETRANS_CC_TIME	10
RETRANS_CR_TIME	10
RETRANS_DR_TIME	10
RETRANS_DT_TIME	5
RETRANS_XP_TIME	5
MAX_CONNECTIONS	32

After all the parameters have been checked or changed, press F2 to update the changes.

Finally, select "Update Global Case DIB" from the top level menu. This

is where stations are given "logical names". A logical name is used as a shorthand not only for the station address, but for a particular process on that station. First, the user should press F4 to change to the data entry window. The logical name desired for a station, the station name it applies to, and the CSSAP number is then entered. The CSSAP number is used to distinguish processes with more than one logical name on the same station. After the information has been entered, the user can press F6 to store the logical name.

APPENDIX C

INI MP-400 CARD CONFIGURATION

Each card was set up for the host machines as shown in Figure C.1.

The purposes of the various jumpers are shown in Table C.1.

Table C.1. INI TBC Jumpers

Jumper	Purpose
W1	Enables parity checking
W2 and W3	Selects EPROM size
W4, W5, W6, and W7	Selects address of card memory page
W8, W9, and W10	Selects IO Address
W11, W12, W13, and W14	Selects download or "Squawk Mode"
W15, W16, and W17	Interrupt selection
W18	Clock selection
W19	Factory test jumper.

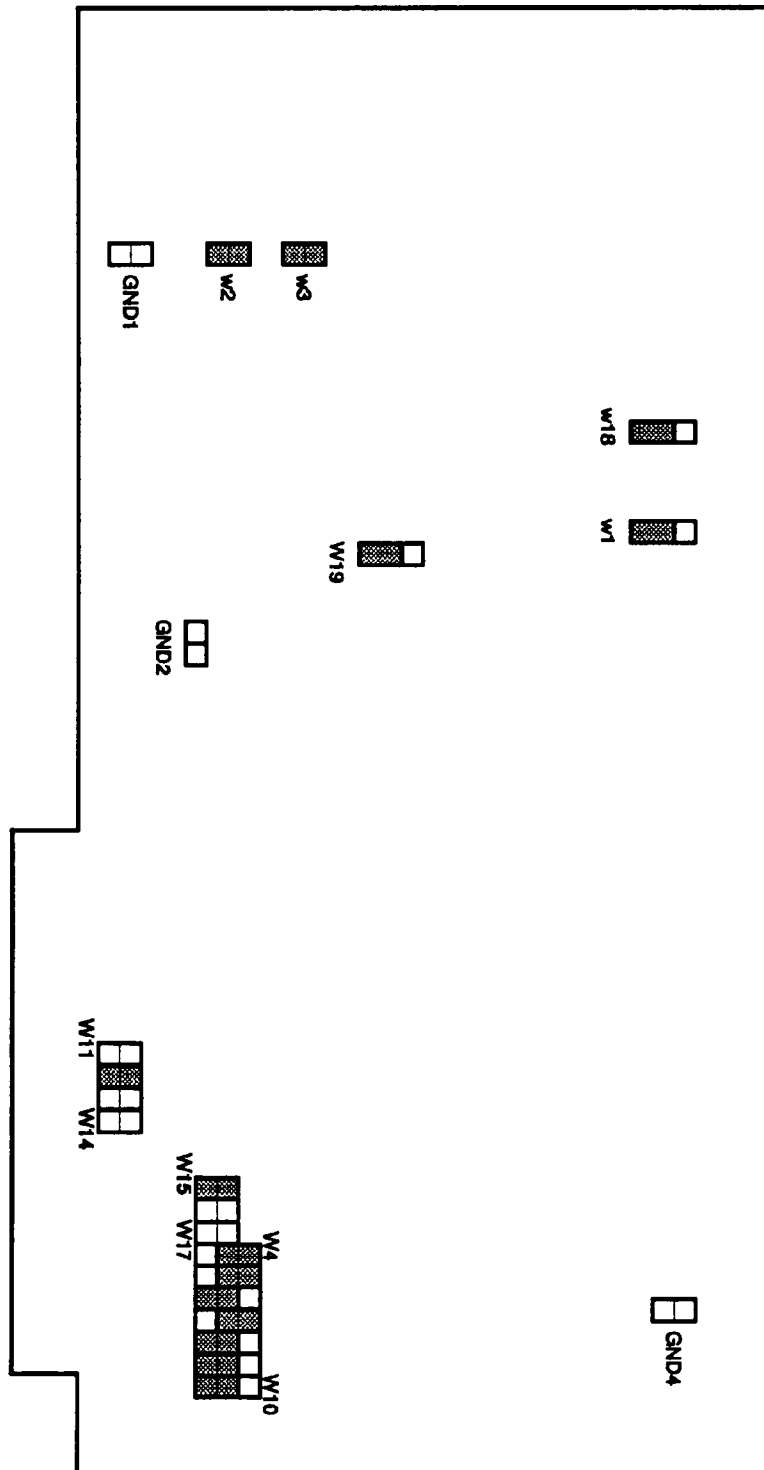


Figure C.1. INI MP-400 TBC Jumper Settings

APPENDIX D

CONCORD 1200 CARD CONFIGURATION

Unlike the INI MP-400 cards, the software stack configuration for the Concord 1200 cards is done at each host machine. For the hardware configuration, Table D.1 shows the jumpers on the Concord cards, and Figure D.1 shows the actual configuration used on both ends of the connection.

Table D.1. Concord 1200 Bus Controller Jumpers

Jumper	Purpose
S3	Selects modem type (internal or external)
S4	Selects 512K EPROM
S5	Selects internal clock
S6	Selects 512K EPROM
S7	Auto init enabled
S8	Not used
S9	Hardware revision register
S10 and S22	Selects DMA chan 2
S11	Selects IRQ 3
S12	Selects IRQ 4
S13	Selects IRQ 5
S14	Selects IRQ 6
S15	Selects IRQ 7
S16 and S19	Selects DMA chan 1
S17 and S18	Selects DMA chan 3
S20 and S21	Selects +- 12 pins
S23	Selects IRQ 2
Switch 1	Selects board base address

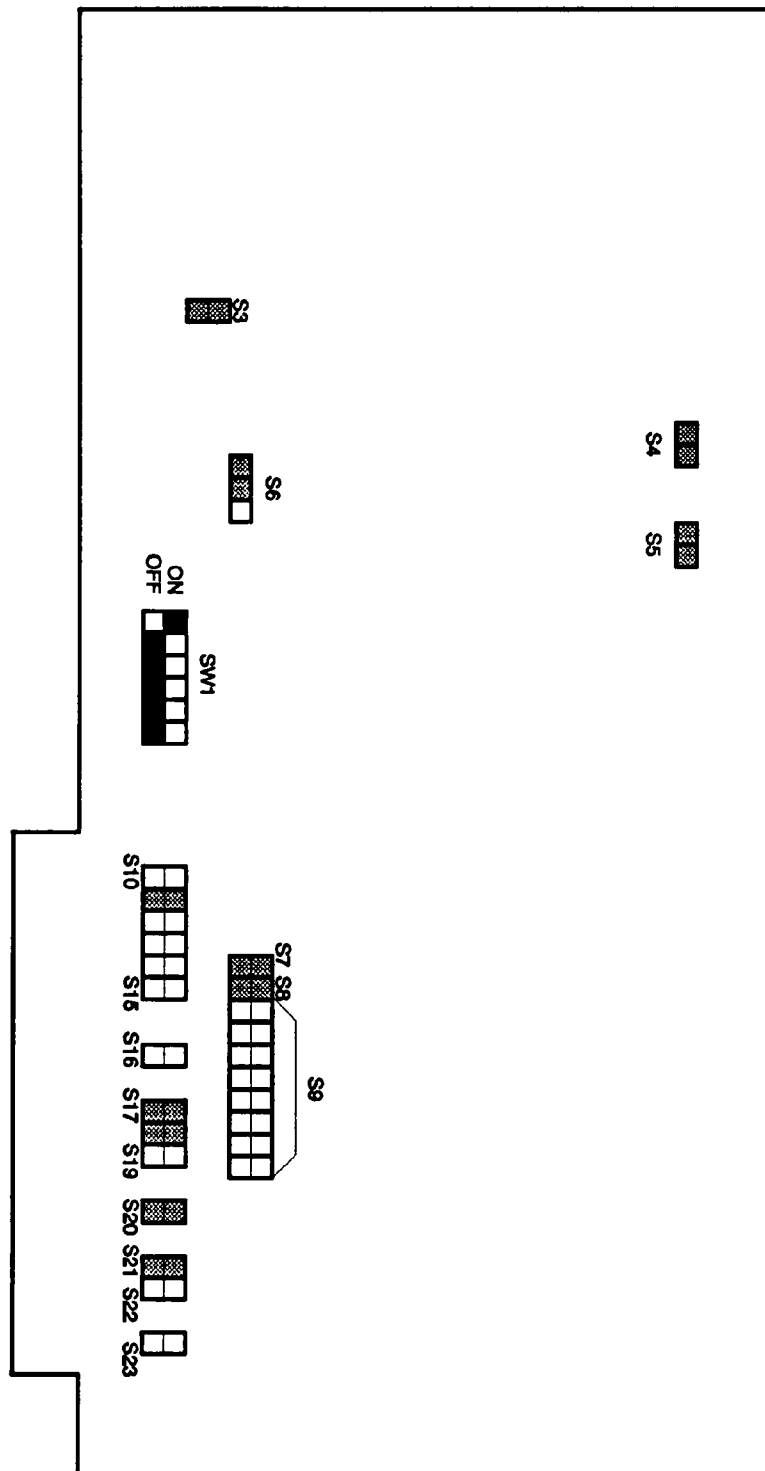


Figure D.1. Concord 1200 Bus Controller Jumper Settings

For the software stack configuration, a utility is supplied by Concord. Before the stack can be configured, however, a file named "MAP3CNFG.DAT" must be set up with the address of the card and the addresses of any other cards that the host can "talk" to. Table D.2 shows the contents of the file used for this thesis.

Table D.2. Contents of MAP3CNFG.DAT File

net1 =	39 84 0f 45 4e 45 00 00 69 01 01 f4 00 01 00
net2 =	39 84 0f 45 4e 45 00 00 69 01 01 f4 00 02 00
net3 =	39 84 0f 45 4e 45 00 00 69 01 01 f4 00 03 00
net4 =	39 84 0f 45 4e 45 00 00 69 01 01 f4 00 04 00

After the addresses have been set the following line must be executed at the DOS prompt:

MAPUR10 STACK=MAP30.DLD NAME=<ADD_NAME>

Where <ADD_NAME> should be replaced with the name of the address of the host card as given in Table D.2. For the network used in this thesis, the server was named "net1" and the client was named "net2". A configuration screen will come up. Before changing any settings, the user should type "vcon" to verify the controller and modem. The transmit level was set to 47, the channel to 6'-FM1', and the slot time to 128. After verifying these settings, the

user should type "sa" to save the configuration. The program should then be exited by typing "exit". Finally, since MMS-EASE looks for a stack named "DLDPROD.CNV" by default, MAP30.DLD should be copied to that name.

APPENDIX E

TRUEVISION TARGA-M8 CONFIGURATION

The Targa-M8 configuration consisted of two parts; hardware and software. The hardware configuration is used to set the video buffer address, I/O port, and output termination for the board. Table E.1 shows the uses for the switches in the Targa-M8, and Figure E.1 shows how the board was actually configured.

Table E.1. Targa-M8 Switch Settings

Switch Block	Switch	Purpose
1	1	Factory use only
1	2, 3, and 4	Select video buffer address
1	5, 6, 7, and 8	Select I/O port address
1	9 and 0	Factory use only
2	1	Video 0 termination
2	2	Video 1 termination
2	3	Video 2 termination
2	4	Video 3 termination
2	5	Synch on green termination

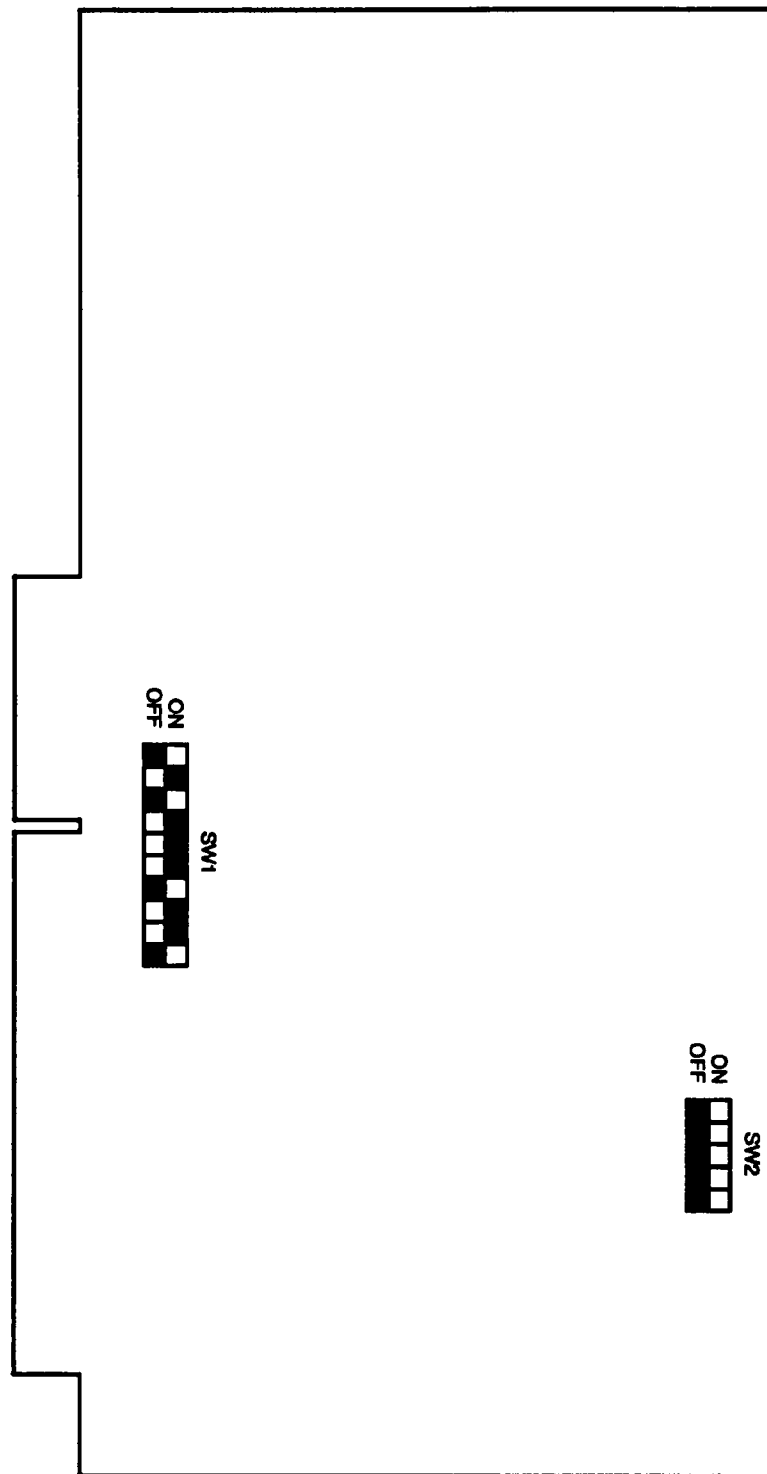


Figure E.1. Targa-M8 Switch Settings

For the software configuration, a utility program called "trueinit.exe" (version 3.0) must be run. This program consists of a series of screens with various parameters. Table E.2 shows the parameters used in this research.

Table E.2. Targa-M8 Software Configuration Option Values

Parameter	Value
Targa Type	Targa M8
Memory Address	0xA000
I/O Address	0x0220
Overscan	No
Comp Video Adapter	No
Display Lines	400
Interlace Mode	Interlaced
Genlock	Lock When Live
CV or RGB	CV
VCR or Camera	Broadcast/Camera
Gain	50
Offset	50

Custom cables were used both from the camera to the Targa-M8 and from the Targa-M8 to the display monitor. The Targa-to-monitor cable was made so that it could be used with standard multi-synch monitors typically used for VGA cards (such as the NEC Multi-Synch II). Table E.3 shows the pin connections of the camera-to-Targa cable, and Table E.4 shows the pin connections of the Targa-to-monitor cable. All pins not explicitly shown have no connection.

Table E.3. Camera-to-Targa Cable Pinout

Camera Male DB9	Targa-M8 Male Low-Density DB15	Signal
5	8	Blue
4	7	Green
3	6	Red
9	5	Ground
6	13	Composite Video

Table E.4. Targa-to-Monitor Cable Pinout

Targa-M8 Male DB9	Monitor Female High-Density DB15	Signal
1	5 and 10	Ground
2	6, 7, and 8	Ground
3	1	Red
4	2	Green
5	3	Blue
8	13	Horizontal Synch
9	14	Vertical Synch

APPENDIX F

COMPILER CONFIGURATION

Microsoft C version 5.1 was used to compile the programs. MMS-EASE required several overrides to the default parameters used by the compiler. The parameters that were used to compile the code are shown in Tables F.1 and F.2 [13].

Table F.1. Compiler Command Line Options

Switch	Purpose
/AL	Select large model code generation
/W2	Warning level 2
/c	Compile only, no link

Table F.2. Link command line options

Switch	Purpose
/LI	Include line numbers
/MAP:60000	List public symbols in map file
/ST:8196	Set stack to 8K
/SEGMENTS:200	Allow 200 segments
/NOD	No default libraries
/NOE	No extended dictionary
/NOI	Preserve case sensitivity

APPENDIX G

STANDARD CODE USED IN ALL PROGRAMS

Three source code modules must be used in every MMS-EASE program; FKEY.C, GETVAL5.C, and MMSOP_EN.C. Because unnecessary functions in these modules were deleted to conserve memory and the possibility that they may change in a future release of MMS-EASE, they are included here. Also, the files SENDCONN.C and RECVCONN.C contain the connection-oriented routines used in all server and client programs, respectively. PACKBITS.H and PACKBITS.C together make up the module that handles data compression and decompression. The file EV678.ASM contains the assembly language source code that controls the Everex VGA card, and it was used by both client programs. Finally, a file named "SUIC.DIB" must be present for MMS-EASE to use the Concord 1200 card sets. A different version of this file is used for the client and the server.

"FKEY.C"

```

/*****
/* SISCO SOFTWARE MODULE HEADER *****/
/*****
/* (c) Copyright Systems Integration Specialists Company, Inc.,
/* 1986, 1987, 1988 All Rights Reserved
/*
/* MODULE NAME : fkey.c
/* PRODUCT(S) : MMSEASE
/*
/* MODULE DESCRIPTION :
/* This module contains function key handling functions, and
/* other misc. key handling functions.
/*
/* GLOBAL FUNCTIONS DEFINED IN THIS MODULE :
/* static SHORT nlfn ()
/* VOID wait_msg (str)
/* VOID wait_key ()
/* VOID flush_keys ()
/* VOID list_bytes (ptr,len)
/*
/* MODIFICATION LOG :
/* Date Who Rev Comments
/* -----
/* 08/15/88 4.00 MMSEASE 4.0 release. See MODL40.DOC for
/* history.
*****/

#include "glbtypes.h"
#include "sysincs.h"
#include "conio.h"

#include "gvaldefs.h"
#include "scrndefs.h"
#include "ctype.h"

VOID flush_keys ( VOID );
static SHORT nlfn ();
SHORT (*servefun)( VOID ) = nlfn; /* pointer to function to be called */

/*****
/* nlfn
/* function to just return. Used when need to set service routine to
/* a dummy function
*****/

static SHORT nlfn ()
{
    return(0);
}

/*****
/* wait_msg
/* Function to display a message and wait for a key hit
*****/

VOID wait_msg (str)
BYTE *str;
{
    flush_keys ();
    printf ("\n %s",str);
    while (!kbhit())
        (*servefun) (); /* execute background task */
    flush_keys ();
}
```

```

/*****
/*                                wait_key                                */
*****/

VOID wait_key ()
{
    while (kbhit())
        (*servefun) (); /* execute background task */
    flush_keys ();
}

/*****
/*                                flush_keys                                */
/* Function to flush the keyboard character buffer                        */
*****/

VOID flush_keys ()
{
    SHORT i;

    while (kbhit())
        i = getch ();
}

/*****
/*                                list_bytes                                */
*****/

VOID list_bytes (ptr,len)
BYTE *ptr;
SHORT len;
{
    CLEAR_EOL;
    while (len)
    {
        printf (" %02X",*(ptr++) & 0xFF);
        len--;
    }
}

/*****
/*                                wait_debug_log                            */
*****/

VOID wait_debug_log (fptr,str)
FILE *fptr;
BYTE *str;
{
    flush_keys ();
    fprintf (fptr,"\n %s",str);
    fflush(fptr);
    if (fptr == stdout)
    {
        while (kbhit())
            (*servefun) (); /* execute background task */
    }
    flush_keys ();
}

```

```

/*****
/*          log_hex_bytes          */
/* This function provides a hex dump facility. */
*****/

#define cvt_nibble(a) ((a) > 9 ? (a) + 'A' - 10 : (a) + '0')
#define DEBUG_BUF_LEN 100

VOID log_hex_bytes (dest,ptr,len)
FILE *dest;
BYTE *ptr; /* pointer to hex data */
SHORT len; /* length of data */
{
    register SHORT i;
    register SHORT ascii_index;
    register SHORT hex_index;
    BYTE debug_buf[DEBUG_BUF_LEN+1]; /* output buffer */
    SHORT curlen;
    BYTE ch;
    UBYTE hex;
    SHORT addr = 0;

    /* format hex data */
    while (len)
    {
        curlen = min(len,16); /* # char's in current line */
        sprintf (debug_buf," %04X ",addr); /* data offset */
        hex_index = 7; /* where to put hex data */
        ascii_index = 58; /* where to put ascii conversion */
        debug_buf[ascii_index++] = '*';

        for (i = 0; i < DEBUG_BUF_LEN; ++i)
            debug_buf[i] = ' ';
        fprintf (dest,"%s","\n");

        for (i = 0; i < curlen; i++) /* for each byte in this line */
        {
            ch = *ptr++; /* get next character */

            hex = (ch >> 4) & 0x0f; /* write HEX chars for the byte */
            hex = cvt_nibble(hex);
            debug_buf[hex_index++] = hex;
            hex = ch & 0x0f;
            hex = cvt_nibble(hex);
            debug_buf[hex_index++] = hex;
            debug_buf[hex_index++] = ' ';

            if (isprint(ch)) /* print ASCII portion */
                debug_buf[ascii_index] = ch;
            else
                debug_buf[ascii_index] = '.'; /* just put a '.' there */
            ascii_index++;

            if (i == 7) /* half way through line */
                debug_buf[hex_index++] = ' '; /* put an extra space here */
        } /* end of line */

        debug_buf[ascii_index++] = '*';
        debug_buf[ascii_index++] = 0;
        for (i = hex_index; i < 58; i++)
            debug_buf[i] = ' '; /* clear the buffer between hex and ascii data */
        /* write out the line */
        fprintf (dest,"%s",debug_buf);

        addr += 16; /* prepare for next line */
        len -= curlen;
    }
}

```

```

/*****
/*                               ask                               */
/* Ask a yes/no question and return the answer.                  */
*****/

SHORT ask (question, default_ans)

BYTE *question;
SHORT default_ans;
{
  BYTE ans [20];
  SHORT ret_ans;

  printf ("%s", question);
  flush_keys ();
  ans [0] = '?';
  strget (ans);
  if (ans [0] == 'n' || ans [0] == 'N')
    ret_ans = 0;
  else
  {
    if (ans [0] == 'y' || ans [0] == 'Y')
      ret_ans = 1;
    else
      ret_ans = default_ans;
  }

  return (ret_ans);
}

```

"GETVAL5.C"

```

/*****
/* SISCO SOFTWARE MODULE HEADER *****/
/*****
/* (c) Copyright Systems Integration Specialists Company, Inc.,
/* 1986, 1987, 1988 All Rights Reserved
/*
/*
/* MODULE NAME : getval5.c
/* PRODUCT(S) : MMSEASE
/*
/*
/* MODULE DESCRIPTION :
/* This module contains routines to get input from the user.
/* Each routine accepts a pointer to an appropriate variable type
/* and writes the variable if the user enters appropriate data at
/* the console. The routines return 1 if successful, else 0.
/*
/*
/* GLOBAL FUNCTIONS DEFINED IN THIS MODULE :
/* SHORT strget(BYTE[]);
/* VOID entry1(SHORT,BYTE[]);
/*
/*
/* MODIFICATION LOG :
/* Date Who Rev Comments
/* -----
/* 08/16/88 RKR 4.00 Added stdlib.h for VMS so atof would work
/* 08/15/88 4.00 MMSEASE 4.0 release. See MODL40.DOC for
/* history.
*****/

#include "glbtypes.h"

#include "sysincs.h"

#include <conio.h>
#include <math.h>
#include <ctype.h>

#include "gvaldefs.h"
#include "scrndefs.h"
#include "fkeydefs.h"

extern SHORT (*servefun()); /* pointer to function to be called */

#ifdef LINT_ARGS
SHORT strget(BYTE *);
VOID entry1(SHORT,BYTE[]);
#else
SHORT strget ();
VOID entry1 ();
#endif

```

```

/*****
/*                               strget                               */
/* Function to get a character string. WARNING: Input string must be */
/* 200 characters long!! (Otherwise, there is the possibility of the */
/* user writing over adjacent data space.)                          */
*****/

```

```

SHORT strget (str)
BYTE str[];
{
    SHORT row;
    SHORT col;
    BYTE line[200];

    SAVESCR;
    entry1(200,line);
    if(strlen(line))
    {
        strcpy(str,line);
        return(1);
    }
    else
        return(0);
}

```

```

/*****
/*                               entry1                               */
/* This function allows a character string or a fixed length to be */
/* read in. It checks for backspacing (erasing of previous data, an */
/* escape function (abort input), and scrolling of the inputted string. */
/* This function requires the following information be passed to it */
/* in the following format:                                          */
/*                                                                     */
/* entry1(row, column, maxwidth, *str_ptr)                          */
/*                                                                     */
/* where:                                                            */
/*   row & column - indicate the coordinates of the data           */
/*                  string to be read in                            */
/*   maxwidth      - indicates the maximum length of the            */
/*                  data string                                      */
/*   str_ptr       - is the name of the character array              */
/*                  the data string is to be placed in              */
*****/

```

```

VOID entry1(maxlen, str_ptr)
SHORT maxlen;
BYTE str_ptr[];
{
    register SHORT i,i1;
    SHORT exit;

```

```

maxlen -= 1; /* eliminates overwrite off buffers */

```

```

    i1 = 0;
    i = 0;
    exit = 0;
    str_ptr[0] = '\x0';
    while (exit == 0)
    {
        if (kbhit())
        {
            str_ptr[i] = (BYTE) getch();

```

```

        str_ptr[i+1] = '\x0';
        switch (str_ptr[i])
        {
        case '\x0' :           /* check for function key      */
            getch();           /* discard if found */
            break;

        case '\x08' :           /* check for a backspace */
            if (i > 0)
            {
                str_ptr[--i] = '\x20';
                str_ptr[i+1] = '\x0';
                str_ptr[i] = '\x0';
            }
            break;

        case CR :               /* check for a carriage return */
            exit = 1;
            str_ptr[i] = '\x0';
            ADD_LF;
            break;

        case '\x1b' :           /* check for an escape sequence */
            exit = 1;           /* discard information and exit */
            for (0; i < maxlen; i++)
                str_ptr[i] = '\x20';
            str_ptr[maxlen] = '\x0';
            str_ptr[0] = '\x0';
            break;

        default :               /* addition data */
            if (i >= maxlen) /* data greater than buffer */
            {
                /* scroll buffer */
                for (i1=0; i1 < maxlen; ++i1)
                    str_ptr[i1] = str_ptr[i1+1];
                str_ptr[maxlen] = '\x0';
            }
            else
                ++i;           /* data less than buffer length */
        }
        /* add in data to end of string */

    /* print data string on console */

        if (!exit)
        {
            RESTORESCR;
            PRINTSTR;
        }
        }
        (*servefun)();           /* perform function while waiting */
    }
}

```

```

/*****
/*                               intget                               */
/* Function to get an integer value from the user. Works perfectly well */
/* for inputting unsigned integers also, where the input data is simply */
/* cast as a SHORT in the call.                                         */
*****/

SHORT intget (in)
SHORT *in;
{
    SHORT x,i,j,len;
    BYTE tmp[50];

    SAVESCR;

    entry1(50,tmp);
    len = strlen(tmp);          /* check for numeric string */
    for (i = 0; i <= len && len && (isdigit(tmp[i]) || tmp[i] == '-'); ++i)
        ;
    if(i && (i == len))
    {
        x = (SHORT) atoi(tmp);
        *in = x;
        return(1);
    }
    else
        return(0);
}

```

"MMSOP_EN.C"

```

/*****
/* SISCO SOFTWARE MODULE HEADER *****/
/*****
/*
/* MODULE NAME : mmsop_en.c
/* PRODUCT(S) : MMSEASE
/*
/* MODULE DESCRIPTION :
/* This module is used to initialize the tables used in defining
/* the supported operation set, and MUST be re-compiled if the
/* supported command set is reduced (or expanded) via the subset
/* creation mechanism. This module also defines and initializes
/* the supported services parameters.
/*
/* The following tables are initialized:
/*
/* *operation enable table (mmsop_en [86]):
/* used to check if an MMS operation is to be supported
/* as requestor and/or responder, and if the data element
/* containing the opcode is of the proper form.
/*
/* *request decode function table (mms_req_decode_fun [86]):
/* used to define which function will be executed to
/* decode a request.
/*
/* *response decode function table (mms_rsp_decode_fun [86]):
/* used to define which function will be executed to
/* decode a response.
/*
/* **indication service function table (mms_ind_serve_fun [86]):
/* used to define which function will be executed to
/* service an indication.
/*
/* **confirmation service function table (mms_conf_serve_fun [106]):
/* used to define which function will be executed to
/* service a confirmation.
/*
/* * - these tables are not normally modified by the user.
/* ** - these tables may be modified by the user.
/*
/* GLOBAL FUNCTIONS DEFINED IN THIS MODULE :
/* NONE
/*
/* MODIFICATION LOG :
/* Date Who Rev Comments
/* -----
/* 09/06/88 RKR 4.01 Fixed the order of INITIATE, CONCLUDE, and
/* CANCEL in the mmsop_en array and
/* mms_op_string
/* 08/26/88 MDE 4.01 Corrected FDIR_SSI from 0x10 to 0x04
/* 08/15/88 4.00 MMSEASE 4.0 release. See MODL40.DOC for
/* history.
/*****/

#include "glbtypes.h"
#include "sysincs.h"
#include "mms_user.h"

#ifdef FILE_SEND
#include "fts_enab.h"
#endif

#ifdef FILE_RECV
#include "ftr_enab.h"

```

```

#endif
#ifdef DOM_SEND
#include "dts_enab.h"
#endif

#ifdef DOM_RECV
#include "dtr_enab.h"
#endif

extern VOID mms_status_req ();
extern VOID mms_status_rsp ();
extern VOID mms_ustatus_req ();
extern VOID mms_namelist_req ();
extern VOID mms_namelist_rsp ();
extern VOID mms_identify_req ();
extern VOID mms_identify_rsp ();
extern VOID mms_getcl_req ();
extern VOID mms_getcl_rsp ();

extern VOID mms_init_download_req ();
extern VOID mms_init_download_rsp ();
extern VOID mms_download_req ();
extern VOID mms_download_rsp ();
extern VOID mms_term_download_req ();
extern VOID mms_term_download_rsp ();
extern VOID mms_init_upload_req ();
extern VOID mms_init_upload_rsp ();
extern VOID mms_upload_req ();
extern VOID mms_upload_rsp ();
extern VOID mms_term_upload_req ();
extern VOID mms_term_upload_rsp ();
extern VOID mms_rddwn_req ();
extern VOID mms_rddwn_rsp ();
extern VOID mms_rdupl_req ();
extern VOID mms_rdupl_rsp ();
extern VOID mms_load_domain_req ();
extern VOID mms_load_domain_rsp ();
extern VOID mms_store_domain_req ();
extern VOID mms_store_domain_rsp ();
extern VOID mms_delete_domain_req ();
extern VOID mms_delete_domain_rsp ();
extern VOID mms_get_dom_attr_req ();
extern VOID mms_get_dom_attr_rsp ();

extern VOID mms_create_pi_req ();
extern VOID mms_create_pi_rsp ();
extern VOID mms_delete_pi_req ();
extern VOID mms_delete_pi_rsp ();
extern VOID mms_start_req ();
extern VOID mms_stop_req ();
extern VOID mms_stop_rsp ();
extern VOID mms_resume_req ();
extern VOID mms_resume_rsp ();
extern VOID mms_reset_req ();
extern VOID mms_reset_rsp ();
extern VOID mms_kill_req ();
extern VOID mms_kill_rsp ();
extern VOID mms_get_pi_req ();
extern VOID mms_get_pi_rsp ();

extern VOID mms_read_req ();
extern VOID mms_read_rsp ();
extern VOID mms_write_req ();
extern VOID mms_write_rsp ();
extern VOID mms_info_rpt_req ();
extern VOID mms_get_var_req ();
extern VOID mms_get_var_rsp ();
extern VOID mms_def_var_req ();

```

```

extern VOID mms_def_var_rsp ();
extern VOID mms_del_var_req ();
extern VOID mms_del_var_rsp ();
extern VOID mms_get_type_req ();
extern VOID mms_get_type_rsp ();
extern VOID mms_def_type_req ();
extern VOID mms_def_type_rsp ();
extern VOID mms_del_type_req ();
extern VOID mms_del_type_rsp ();
extern VOID mms_takectrl_req ();
extern VOID mms_takectrl_rsp ();
extern VOID mms_relctrl_req ();
extern VOID mms_relctrl_rsp ();
extern VOID mms_rsstat_req ();
extern VOID mms_rsstat_rsp ();
extern VOID mms_rspool_req ();
extern VOID mms_rspool_rsp ();
extern VOID mms_rsendry_req ();
extern VOID mms_rsendry_rsp ();
extern VOID mms_defsem_req ();
extern VOID mms_defsem_rsp ();
extern VOID mms_delsem_req ();
extern VOID mms_delsem_rsp ();

extern VOID mms_output_req ();
extern VOID mms_output_rsp ();
extern VOID mms_input_req ();
extern VOID mms_input_rsp ();

extern VOID mms_geteca_req ();
extern VOID mms_geteca_rsp ();
extern VOID mms_repecs_req ();
extern VOID mms_repecs_rsp ();
extern VOID mms_evnot_req ();
extern VOID mms_ackevnot_req ();
extern VOID mms_ackevnot_rsp ();
extern VOID mms_getas_req ();
extern VOID mms_getas_rsp ();

extern VOID mms_jwrite_req ();
extern VOID mms_jwrite_rsp ();
extern VOID mms_jread_req ();
extern VOID mms_jread_rsp ();
extern VOID mms_jinit_req ();
extern VOID mms_jinit_rsp ();
extern VOID mms_jstat_req ();
extern VOID mms_jstat_rsp ();

extern VOID mms_obtain_file_req ();
extern VOID mms_obtain_file_rsp ();
extern VOID mms_file_open_req ();
extern VOID mms_file_open_rsp ();
extern VOID mms_file_read_req ();
extern VOID mms_file_read_rsp ();
extern VOID mms_file_close_req ();
extern VOID mms_file_close_rsp ();
extern VOID mms_file_rename_req ();
extern VOID mms_file_rename_rsp ();
extern VOID mms_file_delete_req ();
extern VOID mms_file_delete_rsp ();
extern VOID mms_file_dir_req ();
extern VOID mms_file_dir_rsp ();

extern VOID mms_conclude_req ();
extern VOID mms_conclude_rsp ();
extern VOID mms_cancel_req ();
extern VOID mms_cancel_rsp ();
extern VOID mms_init_dec ();

```

```

extern VOID mms_rsp_not_supp ();
extern VOID mms_req_not_supp ();
extern VOID u_conf_not_supp ();
extern VOID u_ind_not_supp ();
extern VOID mms_null_pdu_dec ();

/*****
/*****
/* The following defines are used in initializing the mmsop_en array. */
/* Each entry of the array consists of 8 bits: 00ijklmn, where the */
/* bottom 6 bits mean the following: */
/* */
/* i - if 1, then response is NULL */
/* j - if 1, then request is NULL */
/* k - if 1, then response is constructor; otherwise, primitive. */
/* l - if 1, then request is constructor; otherwise, primitive. */
/* m - if 1, then response is supported; otherwise not. */
/* n - if 1, then request is supported; otherwise not. */
/* */
/* By "supported" is meant that the executable program contains code to */
/* handle that op code. */
/*****
/* */
/* resp req */
#define PRIM_PRIM 0x00 /* primitive resp and req */

#define PRIM_CSTR 0x04 /* prim resp, constructor req */
#define CSTR_PRIM 0x08 /* constructor resp, prim req */
#define CSTR_CSTR 0x0C /* cstr resp, cstr req */
#define NULL_NULL 0x30 /* primitive resp and req */
#define NULL_PRIM 0x20 /* prim resp, constructor req */
#define PRIM_NULL 0x10 /* constructor resp, prim req */
#define NULL_CSTR 0x24 /* prim resp, constructor req */
#define CSTR_NULL 0x18 /* constructor resp, prim req */

/*****
/* initialize the operation enable table */
/*****
BYTE mmsop_en [86] = /* opcode is index into table */
{
    MMS_STATUS_EN      | CSTR_PRIM, /* 00 STATUS */
    MMS_GETNAMES_EN     | CSTR_CSTR, /* 01 GET_NAMLIST */
    MMS_IDENT_EN        | CSTR_NULL, /* 02 IDENTIFY */
    0,                  /* 03 RENAME */
    MMS_READ_EN         | CSTR_CSTR, /* 04 READ */
    MMS_WRITE_EN        | CSTR_CSTR, /* 05 WRITE */
    MMS_GETVAR_EN       | CSTR_CSTR, /* 06 GET_VARDEF */
    MMS_DEFVAR_EN       | NULL_CSTR, /* 07 DEF_VARNAM */
    0,                  /* 08 DEF_SCATTERED */
    0,                  /* 09 GET_SCATTERED */
    MMS_DELVAR_EN       | CSTR_CSTR, /* 10 DEL_VARNAM */
    0,                  /* 11 DEF_VARLIST */
    0,                  /* 12 GET_VARLIST */
    0,                  /* 13 DEL_VARLIST */
    MMS_DEFTYPE_EN      | NULL_CSTR, /* 14 DEF_TYPENAM */
    MMS_GETTYPE_EN      | CSTR_CSTR, /* 15 GET_TPEDEF */
    MMS_DELTTYPE_EN     | CSTR_CSTR, /* 16 DEL_TYPENAM */
    MMS_INPUT_EN        | PRIM_CSTR, /* 17 INPUT */
    MMS_OUTPUT_EN       | NULL_CSTR, /* 18 OUTPUT */
    MMS_TAKECTRL_EN     | CSTR_CSTR, /* 19 TAKE_CONTROL */
    MMS_RELCTRL_EN      | NULL_CSTR, /* 20 REL_CONTROL */
    MMS_DEFINE_SEM_EN   | NULL_CSTR, /* 21 DEF_SEMAPHORE */
    MMS_DELETE_SEM_EN   | NULL_CSTR, /* 22 DEL_SEMAPHORE */
    MMS_REP_SEMSTAT_EN  | CSTR_CSTR, /* 23 REP_SEM_STATUS */
    MMS_REP_SEMPOOL_EN  | CSTR_CSTR, /* 24 REP_SEM_POOL_STATUS */
    MMS_REP_SEMENTRY_EN | CSTR_CSTR, /* 25 REP_SEM_ENTRY_STATUS */
    MMS_INIT_DWN_EN     | CSTR_CSTR, /* 26 INIT_DOWNLOAD */
    MMS_DWN_LOAD_EN     | CSTR_PRIM, /* 27 DOWN_LOAD */

```

```

MMS_TERM_DWN_EN      NULL_CSTR, /* 28 TERM_DOWNLOAD */
MMS_INIT_UPL_EN      CSTR_PRIM, /* 29 INIT_UPLOAD */
MMS_UP_LOAD_EN       CSTR_PRIM, /* 30 UP_LOAD */
MMS_TERM_UPL_EN      NULL_PRIM, /* 31 TERM_UPLOAD */
MMS_RDDWN_EN         NULL_CSTR, /* 32 REQ_DOWNLOAD */
MMS_RDUPL_EN         NULL_CSTR, /* 33 REQ_UPLOAD */
MMS_LOAD_DOM_EN      NULL_CSTR, /* 34 LOAD_DOMAIN */
MMS_STR_DOM_EN       NULL_CSTR, /* 35 STORE_DOMAIN */
MMS_DEL_DOM_EN       NULL_PRIM, /* 36 DELETE_DOMAIN */
MMS_GET_DOM_EN       CSTR_PRIM, /* 37 GET_DOMAIN */
MMS_CRE_PI_EN        NULL_CSTR, /* 38 CREATE_PROGRAM_INVOCATION */
MMS_DEL_PI_EN        NULL_PRIM, /* 39 DEL_PROGRAM_INVOCATION */
MMS_START_EN         NULL_CSTR, /* 40 START */
MMS_STOP_EN          NULL_CSTR, /* 41 STOP */
MMS_RESUME_EN        NULL_CSTR, /* 42 RESUME */
MMS_RESET_EN         NULL_CSTR, /* 43 RESET */
MMS_KILL_EN          NULL_CSTR, /* 44 KILL */
MMS_GET_PI_EN        CSTR_PRIM, /* 45 GET_PROGRAM_INVOCATION */
MMS_OBTAINFILE_EN    NULL_CSTR, /* 46 OBTAIN_FILE */
0, /* 47 MMS_EVMGMT_SERVICE */
0, /* 48 MMS_EVMGMT_SERVICE */
MMS_GETECA_EN        CSTR_CSTR, /* 49 GET_EVENT_CONDITION_ATTR */
MMS_REPCS_EN         CSTR_CSTR, /* 50 REPORT_EVENT_COND_STATUS */
0, /* 51 MMS_EVMGMT_SERVICE */
0, /* 52 MMS_EVMGMT_SERVICE */
0, /* 53 MMS_EVMGMT_SERVICE */
0, /* 54 MMS_EVMGMT_SERVICE */
0, /* 55 MMS_EVMGMT_SERVICE */
0, /* 56 MMS_EVMGMT_SERVICE */
0, /* 57 MMS_EVMGMT_SERVICE */
0, /* 58 MMS_EVMGMT_SERVICE */
0, /* 59 MMS_EVMGMT_SERVICE */
0, /* 60 MMS_EVMGMT_SERVICE */
0, /* 61 MMS_EVMGMT_SERVICE */
MMS_ACKEVNOT_EN      NULL_CSTR, /* 62 ACK_EVENT_NOTIFICATION */
MMS_GETAS_EN         CSTR_CSTR, /* 63 GET_ALARM_SUMMARY */
0, /* 64 MMS_EVMGMT_SERVICE */
MMS_JREAD_EN         CSTR_CSTR, /* 65 READ_JOURNAL */
MMS_JWRITE_EN        NULL_CSTR, /* 66 WRITE_JOURNAL */
MMS_JINIT_EN         PRIM_CSTR, /* 67 INITIALIZE_JOURNAL */
MMS_JSTAT_EN         CSTR_CSTR, /* 68 REPORT_JOURNAL_STATUS */
0, /* 69 CREATE_JOURNAL */
0, /* 70 DELETE_JOURNAL */
MMS_GETCL_EN         CSTR_CSTR, /* 71 GET_CAPABILITY_LIST */
MMS_FOPEN_EN         CSTR_CSTR, /* 72 FILE_OPEN */
MMS_FREAD_EN         CSTR_PRIM, /* 73 FILE_READ */
MMS_FCLOSE_EN        NULL_PRIM, /* 74 FILE_CLOSE */
MMS_FRENAME_EN       NULL_CSTR, /* 75 FILE_RENAME */
MMS_FDELETE_EN       NULL_CSTR, /* 76 FILE_DELETE */
MMS_FDIR_EN          CSTR_CSTR, /* 77 FILE_DIR */
0, /* 78 ADDITIONAL SERVICE */
MMS_INFO_EN          CSTR_CSTR, /* 79 INFO_RPT */
MMS_USTATUS_EN       CSTR_CSTR, /* 80 UNSOLICITED_STATUS */
MMS_EVNOT_EN         CSTR_CSTR, /* 81 EVENT_NOTIFICATION */
0, /* 82 NOT A MMS_OPCODE */
MMS_INIT_EN          CSTR_CSTR, /* 83 INITIATE */
MMS_CONCLUDE_EN      NULL_NULL, /* 84 CONCLUDE */
MMS_CANCEL_EN        PRIM_PRIM /* 85 CANCEL */
};

```

```

/*****/
/*****/
/* The following defines are used in initializing the request and */
/* response decode functions, the indication and confirmation service */
/* funtions, and the supported service indicators. */
/*****/

```

```

/*****
/* CONFIRMED SERVICES
*****/

#if (MMS_STATUS_EN & REQ_EN)
#define MMS_STATUS_RSP_DFUN mms_status_rsp
#define MMS_STATUS_CONF_SFUN u_mp_status_conf
#else
#define MMS_STATUS_RSP_DFUN mms_rsp_not_supp
#define MMS_STATUS_CONF_SFUN u_conf_not_supp
#endif

#if (MMS_STATUS_EN & RESP_EN)
#define MMS_STATUS_REQ_DFUN mms_status_req
#define MMS_STATUS_IND_SFUN u_status_ind
#define STATUS_SSI 0x80
#else
#define MMS_STATUS_REQ_DFUN mms_req_not_supp
#define MMS_STATUS_IND_SFUN u_ind_not_supp
#define STATUS_SSI 0x00
#endif

#if (MMS_GETNAMES_EN & REQ_EN)
#define MMS_GETNAMES_RSP_DFUN mms_namelist_rsp
#define MMS_GETNAMES_CONF_SFUN u_mp_namelist_conf
#else
#define MMS_GETNAMES_RSP_DFUN mms_rsp_not_supp
#define MMS_GETNAMES_CONF_SFUN u_conf_not_supp
#endif

#if (MMS_GETNAMES_EN & RESP_EN)
#define MMS_GETNAMES_REQ_DFUN mms_namelist_req
#define MMS_GETNAMES_IND_SFUN u_namelist_ind
#define GETNAMES_SSI 0x40
#else
#define MMS_GETNAMES_REQ_DFUN mms_req_not_supp
#define MMS_GETNAMES_IND_SFUN u_ind_not_supp
#define GETNAMES_SSI 0x00
#endif

#if (MMS_IDENT_EN & REQ_EN)
#define MMS_IDENT_RSP_DFUN mms_identify_rsp
#define MMS_IDENT_CONF_SFUN u_mp_ident_conf
#else
#define MMS_IDENT_RSP_DFUN mms_rsp_not_supp
#define MMS_IDENT_CONF_SFUN u_conf_not_supp
#endif

#if (MMS_IDENT_EN & RESP_EN)
#define MMS_IDENT_REQ_DFUN mms_null_pdu_dec
#define MMS_IDENT_IND_SFUN u_ident_ind
#define IDENT_SSI 0x20
#else
#define MMS_IDENT_REQ_DFUN mms_req_not_supp
#define MMS_IDENT_IND_SFUN u_ind_not_supp
#define IDENT_SSI 0x00
#endif

```

```

#if (MMS_RENAME_EN & REQ_EN)
#define MMS_RENAME_RSP_DFUN mms_rsp_not_supp
#define MMS_RENAME_CONF_SFUN u_conf_not_supp
#else
#define MMS_RENAME_RSP_DFUN mms_rsp_not_supp
#define MMS_RENAME_CONF_SFUN u_conf_not_supp
#endif
#if (MMS_RENAME_EN & RESP_EN)
#define MMS_RENAME_REQ_DFUN mms_req_not_supp
#define MMS_RENAME_IND_SFUN u_ind_not_supp
#define RENAME_SSI 0x10
#else
#define MMS_RENAME_REQ_DFUN mms_req_not_supp
#define MMS_RENAME_IND_SFUN u_ind_not_supp
#define RENAME_SSI 0x00
#endif

#if (MMS_READ_EN & REQ_EN)
#define MMS_READ_RSP_DFUN mms_read_rsp
#define MMS_READ_CONF_SFUN u_mp_read_conf
#define MMS_MV_READ_CONF_SFUN u_mv_read_conf
#else
#define MMS_READ_RSP_DFUN mms_rsp_not_supp
#define MMS_READ_CONF_SFUN u_conf_not_supp
#define MMS_MV_READ_CONF_SFUN u_conf_not_supp
#endif
#if (MMS_READ_EN & RESP_EN)
#define MMS_READ_REQ_DFUN mms_read_req
#define MMS_READ_IND_SFUN u_read_ind
#define READ_SSI 0x08
#else
#define MMS_READ_REQ_DFUN mms_req_not_supp
#define MMS_READ_IND_SFUN u_ind_not_supp
#define READ_SSI 0x00
#endif

#if (MMS_WRITE_EN & REQ_EN)
#define MMS_WRITE_RSP_DFUN mms_write_rsp
#define MMS_WRITE_CONF_SFUN u_mp_write_conf
#define MMS_MV_WRITE_CONF_SFUN u_mv_write_conf
#else
#define MMS_WRITE_RSP_DFUN mms_rsp_not_supp
#define MMS_WRITE_CONF_SFUN u_conf_not_supp
#define MMS_MV_WRITE_CONF_SFUN u_conf_not_supp
#endif
#if (MMS_WRITE_EN & RESP_EN)
#define MMS_WRITE_REQ_DFUN mms_write_req
#define MMS_WRITE_IND_SFUN u_write_ind
#define WRITE_SSI 0x04
#else
#define MMS_WRITE_REQ_DFUN mms_req_not_supp
#define MMS_WRITE_IND_SFUN u_ind_not_supp
#define WRITE_SSI 0x00
#endif

```

```

#if (MMS_GETVAR_EN & REQ_EN)
#define MMS_GETVAR_RSP_DFUN mms_get_var_rsp
#define MMS_GETVAR_CONF_SFUN u_mp_getvar_conf
#else
#define MMS_GETVAR_RSP_DFUN mms_rsp_not_supp
#define MMS_GETVAR_CONF_SFUN u_conf_not_supp
#endif
#if (MMS_GETVAR_EN & RESP_EN)
#define MMS_GETVAR_REQ_DFUN mms_get_var_req
#define MMS_GETVAR_IND_SFUN u_getvar_ind
#define GETVAR_SSI 0x02
#else
#define MMS_GETVAR_REQ_DFUN mms_req_not_supp
#define MMS_GETVAR_IND_SFUN u_ind_not_supp
#define GETVAR_SSI 0x00
#endif

#if (MMS_DEFVAR_EN & REQ_EN)
#define MMS_DEFVAR_RSP_DFUN mms_null_pdu_dec
#define MMS_DEFVAR_CONF_SFUN u_mp_defvar_conf
#else
#define MMS_DEFVAR_RSP_DFUN mms_rsp_not_supp
#define MMS_DEFVAR_CONF_SFUN u_conf_not_supp
#endif
#if (MMS_DEFVAR_EN & RESP_EN)
#define MMS_DEFVAR_REQ_DFUN mms_def_var_req
#define MMS_DEFVAR_IND_SFUN u_defvar_ind
#define DEFVAR_SSI 0x01
#else
#define MMS_DEFVAR_REQ_DFUN mms_req_not_supp
#define MMS_DEFVAR_IND_SFUN u_ind_not_supp
#define DEFVAR_SSI 0x00
#endif

#if (MMS_DEFSCAT_EN & REQ_EN)
#define MMS_DEFSCAT_RSP_DFUN mms_rsp_not_supp
#define MMS_DEFSCAT_CONF_SFUN u_conf_not_supp
#else
#define MMS_DEFSCAT_RSP_DFUN mms_rsp_not_supp
#define MMS_DEFSCAT_CONF_SFUN u_conf_not_supp
#endif
#if (MMS_DEFSCAT_EN & RESP_EN)
#define MMS_DEFSCAT_REQ_DFUN mms_req_not_supp
#define MMS_DEFSCAT_IND_SFUN u_ind_not_supp
#define DEFSCAT_SSI 0x80
#else
#define MMS_DEFSCAT_REQ_DFUN mms_req_not_supp
#define MMS_DEFSCAT_IND_SFUN u_ind_not_supp
#define DEFSCAT_SSI 0x00
#endif

#if (MMS_GETSCAT_EN & REQ_EN)
#define MMS_GETSCAT_RSP_DFUN mms_rsp_not_supp
#define MMS_GETSCAT_CONF_SFUN u_conf_not_supp
#else
#define MMS_GETSCAT_RSP_DFUN mms_rsp_not_supp
#define MMS_GETSCAT_CONF_SFUN u_conf_not_supp
#endif
#if (MMS_GETSCAT_EN & RESP_EN)
#define MMS_GETSCAT_REQ_DFUN mms_req_not_supp
#define MMS_GETSCAT_IND_SFUN u_ind_not_supp
#define GETSCAT_SSI 0x40
#else
#define MMS_GETSCAT_REQ_DFUN mms_req_not_supp
#define MMS_GETSCAT_IND_SFUN u_ind_not_supp
#define GETSCAT_SSI 0x00
#endif

```

```

#if (MMS_DELVAR_EN & REQ_EN)
#define MMS_DELVAR_RSP_DFUN mms_del_var_rsp
#define MMS_DELVAR_CONF_SFUN u_mp_delvar_conf
#else
#define MMS_DELVAR_RSP_DFUN mms_rsp_not_supp
#define MMS_DELVAR_CONF_SFUN u_conf_not_supp
#endif
#if (MMS_DELVAR_EN & RESP_EN)
#define MMS_DELVAR_REQ_DFUN mms_del_var_req
#define MMS_DELVAR_IND_SFUN u_delvar_ind
#define DELVAR_SSI 0x20
#else
#define MMS_DELVAR_REQ_DFUN mms_req_not_supp
#define MMS_DELVAR_IND_SFUN u_ind_not_supp
#define DELVAR_SSI 0x00
#endif

#if (MMS_DEFVLIST_EN & REQ_EN)
#define MMS_DEFVLIST_RSP_DFUN mms_rsp_not_supp
#define MMS_DEFVLIST_CONF_SFUN u_conf_not_supp
#else
#define MMS_DEFVLIST_RSP_DFUN mms_rsp_not_supp
#define MMS_DEFVLIST_CONF_SFUN u_conf_not_supp
#endif
#if (MMS_DEFVLIST_EN & RESP_EN)
#define MMS_DEFVLIST_REQ_DFUN mms_req_not_supp
#define MMS_DEFVLIST_IND_SFUN u_ind_not_supp
#define DEFVLIST_SSI 0x10
#else
#define MMS_DEFVLIST_REQ_DFUN mms_req_not_supp
#define MMS_DEFVLIST_IND_SFUN u_ind_not_supp
#define DEFVLIST_SSI 0x00
#endif

#if (MMS_GETVLIST_EN & REQ_EN)
#define MMS_GETVLIST_RSP_DFUN mms_rsp_not_supp
#define MMS_GETVLIST_CONF_SFUN u_conf_not_supp
#else
#define MMS_GETVLIST_RSP_DFUN mms_rsp_not_supp
#define MMS_GETVLIST_CONF_SFUN u_conf_not_supp
#endif
#if (MMS_GETVLIST_EN & RESP_EN)
#define MMS_GETVLIST_REQ_DFUN mms_req_not_supp
#define MMS_GETVLIST_IND_SFUN u_ind_not_supp
#define GETVLIST_SSI 0x08
#else
#define MMS_GETVLIST_REQ_DFUN mms_req_not_supp
#define MMS_GETVLIST_IND_SFUN u_ind_not_supp
#define GETVLIST_SSI 0x00
#endif

#if (MMS_DELVLIST_EN & REQ_EN)
#define MMS_DELVLIST_RSP_DFUN mms_rsp_not_supp
#define MMS_DELVLIST_CONF_SFUN u_conf_not_supp
#else
#define MMS_DELVLIST_RSP_DFUN mms_rsp_not_supp
#define MMS_DELVLIST_CONF_SFUN u_conf_not_supp
#endif
#if (MMS_DELVLIST_EN & RESP_EN)
#define MMS_DELVLIST_REQ_DFUN mms_req_not_supp
#define MMS_DELVLIST_IND_SFUN u_ind_not_supp
#define DELVLIST_SSI 0x04
#else
#define MMS_DELVLIST_REQ_DFUN mms_req_not_supp
#define MMS_DELVLIST_IND_SFUN u_ind_not_supp
#define DELVLIST_SSI 0x00
#endif

```

```

#if (MMS_DEFTYPE_EN & REQ_EN)
#define MMS_DEFTYPE_RSP_DFUN mms_null_pdu_dec
#define MMS_DEFTYPE_CONF_SFUN u_mp_deftype_conf
#define MMS_MV_DEFTYPE_CONF_SFUN u_mv_deftype_conf
#else
#define MMS_DEFTYPE_RSP_DFUN mms_rsp_not_supp
#define MMS_DEFTYPE_CONF_SFUN u_conf_not_supp
#define MMS_MV_DEFTYPE_CONF_SFUN u_conf_not_supp
#endif
#if (MMS_DEFTYPE_EN & RESP_EN)
#define MMS_DEFTYPE_REQ_DFUN mms_def_type_req
#define MMS_DEFTYPE_IND_SFUN u_deftype_ind
#define DEFTYPE_SSI 0x02
#else
#define MMS_DEFTYPE_REQ_DFUN mms_req_not_supp
#define MMS_DEFTYPE_IND_SFUN u_ind_not_supp
#define DEFTYPE_SSI 0x00
#endif

#if (MMS_GETTYPE_EN & REQ_EN)
#define MMS_GETTYPE_RSP_DFUN mms_get_type_rsp
#define MMS_GETTYPE_CONF_SFUN u_mp_gettype_conf
#else
#define MMS_GETTYPE_RSP_DFUN mms_rsp_not_supp
#define MMS_GETTYPE_CONF_SFUN u_conf_not_supp
#endif
#if (MMS_GETTYPE_EN & RESP_EN)
#define MMS_GETTYPE_REQ_DFUN mms_get_type_req
#define MMS_GETTYPE_IND_SFUN u_gettype_ind
#define GETTYPE_SSI 0x01
#else
#define MMS_GETTYPE_REQ_DFUN mms_req_not_supp
#define MMS_GETTYPE_IND_SFUN u_ind_not_supp
#define GETTYPE_SSI 0x00
#endif

#if (MMS_DELTYPE_EN & REQ_EN)
#define MMS_DELTYPE_RSP_DFUN mms_del_type_rsp
#define MMS_DELTYPE_CONF_SFUN u_mp_delttype_conf
#else
#define MMS_DELTYPE_RSP_DFUN mms_rsp_not_supp
#define MMS_DELTYPE_CONF_SFUN u_conf_not_supp
#endif
#if (MMS_DELTYPE_EN & RESP_EN)
#define MMS_DELTYPE_REQ_DFUN mms_del_type_req
#define MMS_DELTYPE_IND_SFUN u_delttype_ind
#define DELTYPE_SSI 0x80
#else
#define MMS_DELTYPE_REQ_DFUN mms_req_not_supp
#define MMS_DELTYPE_IND_SFUN u_ind_not_supp
#define DELTYPE_SSI 0x00
#endif

```

```

#if (MMS_INPUT_EN & REQ_EN)
#define MMS_INPUT_RSP_DFUN mms_input_rsp
#define MMS_INPUT_CONF_SFUN u_mp_input_conf
#else
#define MMS_INPUT_RSP_DFUN mms_rsp_not_supp
#define MMS_INPUT_CONF_SFUN u_conf_not_supp
#endif
#if (MMS_INPUT_EN & RESP_EN)
#define MMS_INPUT_REQ_DFUN mms_input_req
#define MMS_INPUT_IND_SFUN u_input_ind
#define INPUT_SSI 0x40
#else
#define MMS_INPUT_REQ_DFUN mms_req_not_supp
#define MMS_INPUT_IND_SFUN u_ind_not_supp
#define INPUT_SSI 0x00
#endif

#if (MMS_OUTPUT_EN & REQ_EN)
#define MMS_OUTPUT_RSP_DFUN mms_null_pdu_dec
#define MMS_OUTPUT_CONF_SFUN u_mp_output_conf
#else
#define MMS_OUTPUT_RSP_DFUN mms_rsp_not_supp
#define MMS_OUTPUT_CONF_SFUN u_conf_not_supp
#endif
#if (MMS_OUTPUT_EN & RESP_EN)
#define MMS_OUTPUT_REQ_DFUN mms_output_req
#define MMS_OUTPUT_IND_SFUN u_output_ind
#define OUTPUT_SSI 0x20
#else
#define MMS_OUTPUT_REQ_DFUN mms_req_not_supp
#define MMS_OUTPUT_IND_SFUN u_ind_not_supp
#define OUTPUT_SSI 0x00
#endif

#if (MMS_TAKECTRL_EN & REQ_EN)
#define MMS_TAKECTRL_RSP_DFUN mms_takectrl_rsp
#define MMS_TAKECTRL_CONF_SFUN u_mp_takectrl_conf
#else
#define MMS_TAKECTRL_RSP_DFUN mms_rsp_not_supp
#define MMS_TAKECTRL_CONF_SFUN u_conf_not_supp
#endif
#if (MMS_TAKECTRL_EN & RESP_EN)
#define MMS_TAKECTRL_REQ_DFUN mms_takectrl_req
#define MMS_TAKECTRL_IND_SFUN u_takectrl_ind
#define TAKECTRL_SSI 0x10
#else
#define MMS_TAKECTRL_REQ_DFUN mms_req_not_supp
#define MMS_TAKECTRL_IND_SFUN u_ind_not_supp
#define TAKECTRL_SSI 0x00
#endif

#if (MMS_RELCTRL_EN & REQ_EN)
#define MMS_RELCTRL_RSP_DFUN mms_null_pdu_dec
#define MMS_RELCTRL_CONF_SFUN u_mp_relctrl_conf
#else
#define MMS_RELCTRL_RSP_DFUN mms_rsp_not_supp
#define MMS_RELCTRL_CONF_SFUN u_conf_not_supp
#endif

#if (MMS_RELCTRL_EN & RESP_EN)
#define MMS_RELCTRL_REQ_DFUN mms_relctrl_req
#define MMS_RELCTRL_IND_SFUN u_relctrl_ind
#define RELCTRL_SSI 0x08
#else
#define MMS_RELCTRL_REQ_DFUN mms_req_not_supp
#define MMS_RELCTRL_IND_SFUN u_ind_not_supp
#define RELCTRL_SSI 0x00
#endif

```

```

#if (MMS_DEFINE_SEM_EN & REQ_EN)
#define MMS_DEFINE_SEM_RSP_DFUN mms_null_pdu_dec
#define MMS_DEFINE_SEM_CONF_SFUN u_mp_defsem_conf
#else
#define MMS_DEFINE_SEM_RSP_DFUN mms_rsp_not_supp
#define MMS_DEFINE_SEM_CONF_SFUN u_conf_not_supp
#endif

#if (MMS_DEFINE_SEM_EN & RESP_EN)
#define MMS_DEFINE_SEM_REQ_DFUN mms_defsem_req
#define MMS_DEFINE_SEM_IND_SFUN u_defsem_ind
#define DEFSEM_SSI 0x04
#else
#define MMS_DEFINE_SEM_REQ_DFUN mms_req_not_supp
#define MMS_DEFINE_SEM_IND_SFUN u_ind_not_supp
#define DEFSEM_SSI 0x00
#endif

#if (MMS_DELETE_SEM_EN & REQ_EN)
#define MMS_DELETE_SEM_RSP_DFUN mms_null_pdu_dec
#define MMS_DELETE_SEM_CONF_SFUN u_mp_delsem_conf
#else
#define MMS_DELETE_SEM_RSP_DFUN mms_rsp_not_supp
#define MMS_DELETE_SEM_CONF_SFUN u_conf_not_supp
#endif

#if (MMS_DELETE_SEM_EN & RESP_EN)
#define MMS_DELETE_SEM_REQ_DFUN mms_delsem_req
#define MMS_DELETE_SEM_IND_SFUN u_delsem_ind
#define DELSEM_SSI 0x02
#else
#define MMS_DELETE_SEM_REQ_DFUN mms_req_not_supp
#define MMS_DELETE_SEM_IND_SFUN u_ind_not_supp
#define DELSEM_SSI 0x00
#endif

#if (MMS_REP_SEMSTAT_EN & REQ_EN)
#define MMS_REP_SEMSTAT_RSP_DFUN mms_rsstat_rsp
#define MMS_REP_SEMSTAT_CONF_SFUN u_mp_rsstat_conf
#else
#define MMS_REP_SEMSTAT_RSP_DFUN mms_rsp_not_supp
#define MMS_REP_SEMSTAT_CONF_SFUN u_conf_not_supp
#endif

#if (MMS_REP_SEMSTAT_EN & RESP_EN)
#define MMS_REP_SEMSTAT_REQ_DFUN mms_rsstat_req
#define MMS_REP_SEMSTAT_IND_SFUN u_rsstat_ind
#define RSSTAT_SSI 0x01
#else
#define MMS_REP_SEMSTAT_REQ_DFUN mms_req_not_supp
#define MMS_REP_SEMSTAT_IND_SFUN u_ind_not_supp
#define RSSTAT_SSI 0x00
#endif

#if (MMS_REP_SEMPOOL_EN & REQ_EN)
#define MMS_REP_SEMPOOL_RSP_DFUN mms_rspool_rsp
#define MMS_REP_SEMPOOL_CONF_SFUN u_mp_rspool_conf
#else
#define MMS_REP_SEMPOOL_RSP_DFUN mms_rsp_not_supp
#define MMS_REP_SEMPOOL_CONF_SFUN u_conf_not_supp
#endif

```

```

#if (MMS_REP_SEMPOOL_EN & RESP_EN)
#define MMS_REP_SEMPOOL_REQ_DFUN mms_rspool_req
#define MMS_REP_SEMPOOL_IND_SFUN u_rspool_ind
#define RSPPOOL_SSI 0x80
#else
#define MMS_REP_SEMPOOL_REQ_DFUN mms_req_not_supp
#define MMS_REP_SEMPOOL_IND_SFUN u_ind_not_supp
#define RSPPOOL_SSI 0x00
#endif

#if (MMS_REP_SEMENTRY_EN & REQ_EN)
#define MMS_REP_SEMENTRY_RSP_DFUN mms_rsentry_rsp
#define MMS_REP_SEMENTRY_CONF_SFUN u_mp_rsentry_conf
#else
#define MMS_REP_SEMENTRY_RSP_DFUN mms_rsp_not_supp
#define MMS_REP_SEMENTRY_CONF_SFUN u_conf_not_supp
#endif

#if (MMS_REP_SEMENTRY_EN & RESP_EN)
#define MMS_REP_SEMENTRY_REQ_DFUN mms_rsentry_req
#define MMS_REP_SEMENTRY_IND_SFUN u_rsentry_ind
#define RSEMENTRY_SSI 0x40
#else
#define MMS_REP_SEMENTRY_REQ_DFUN mms_req_not_supp
#define MMS_REP_SEMENTRY_IND_SFUN u_ind_not_supp
#define RSEMENTRY_SSI 0x00
#endif

#if (MMS_INIT_DWN_EN & REQ_EN)
#define MMS_INIT_DWN_RSP_DFUN mms_init_download_rsp
#define MMS_INIT_DWN_CONF_SFUN u_mp_initdown_conf
#else
#define MMS_INIT_DWN_RSP_DFUN mms_rsp_not_supp
#define MMS_INIT_DWN_CONF_SFUN u_conf_not_supp
#endif

#if (MMS_INIT_DWN_EN & RESP_EN)
#define MMS_INIT_DWN_REQ_DFUN mms_init_download_req
#define MMS_INIT_DWN_IND_SFUN u_initdown_ind
#define INITDWN_SSI 0x20
#else
#define MMS_INIT_DWN_REQ_DFUN mms_req_not_supp
#define MMS_INIT_DWN_IND_SFUN u_ind_not_supp
#define INITDWN_SSI 0x00
#endif

#if (MMS_DWN_LOAD_EN & REQ_EN)
#define MMS_DWN_LOAD_RSP_DFUN mms_download_rsp
#define MMS_DWN_LOAD_CONF_SFUN u_mp_download_conf
#else
#define MMS_DWN_LOAD_RSP_DFUN mms_rsp_not_supp
#define MMS_DWN_LOAD_CONF_SFUN u_conf_not_supp
#endif

#if (MMS_DWN_LOAD_EN & RESP_EN)
#define MMS_DWN_LOAD_REQ_DFUN mms_download_req
#define MMS_DWN_LOAD_IND_SFUN u_download_ind
#define DOWNLOAD_SSI 0x10
#else
#define MMS_DWN_LOAD_REQ_DFUN mms_req_not_supp
#define MMS_DWN_LOAD_IND_SFUN u_ind_not_supp
#define DOWNLOAD_SSI 0x00
#endif

```

```

#if (MMS_TERM_DWN_EN & REQ_EN)
#define MMS_TERM_DWN_RSP_DFUN mms_null_pdu_dec
#define MMS_TERM_DWN_CONF_SFUN u_mp_termdown_conf
#else
#define MMS_TERM_DWN_RSP_DFUN mms_rsp_not_supp
#define MMS_TERM_DWN_CONF_SFUN u_conf_not_supp
#endif

#if (MMS_TERM_DWN_EN & RESP_EN)
#define MMS_TERM_DWN_REQ_DFUN mms_term_download_req
#define MMS_TERM_DWN_IND_SFUN u_termdown_ind
#define TERMDOWN_SSI 0x08
#else
#define MMS_TERM_DWN_REQ_DFUN mms_req_not_supp
#define MMS_TERM_DWN_IND_SFUN u_ind_not_supp
#define TERMDOWN_SSI 0x00
#endif

#if (MMS_INIT_UPL_EN & REQ_EN)
#define MMS_INIT_UPL_RSP_DFUN mms_init_upload_rsp
#define MMS_INIT_UPL_CONF_SFUN u_mp_initupl_conf
#else
#define MMS_INIT_UPL_RSP_DFUN mms_rsp_not_supp
#define MMS_INIT_UPL_CONF_SFUN u_conf_not_supp
#endif

#if (MMS_INIT_UPL_EN & RESP_EN)
#define MMS_INIT_UPL_REQ_DFUN mms_init_upload_req
#define MMS_INIT_UPL_IND_SFUN u_initupl_ind
#define INITUPL_SSI 0x04
#else
#define MMS_INIT_UPL_REQ_DFUN mms_req_not_supp
#define MMS_INIT_UPL_IND_SFUN u_ind_not_supp
#define INITUPL_SSI 0x00
#endif

#if (MMS_UP_LOAD_EN & REQ_EN)
#define MMS_UP_LOAD_RSP_DFUN mms_upload_rsp
#define MMS_UP_LOAD_CONF_SFUN u_mp_upload_conf
#else
#define MMS_UP_LOAD_RSP_DFUN mms_rsp_not_supp
#define MMS_UP_LOAD_CONF_SFUN u_conf_not_supp
#endif

#if (MMS_UP_LOAD_EN & RESP_EN)
#define MMS_UP_LOAD_REQ_DFUN mms_upload_req
#define MMS_UP_LOAD_IND_SFUN u_upload_ind
#define UPLOAD_SSI 0x02
#else
#define MMS_UP_LOAD_REQ_DFUN mms_req_not_supp
#define MMS_UP_LOAD_IND_SFUN u_ind_not_supp
#define UPLOAD_SSI 0x00
#endif

#if (MMS_TERM_UPL_EN & REQ_EN)
#define MMS_TERM_UPL_RSP_DFUN mms_null_pdu_dec
#define MMS_TERM_UPL_CONF_SFUN u_mp_termupl_conf
#else
#define MMS_TERM_UPL_RSP_DFUN mms_rsp_not_supp
#define MMS_TERM_UPL_CONF_SFUN u_conf_not_supp
#endif

```

```

#if (MMS_TERM_UPL_EN & RESP_EN)
#define MMS_TERM_UPL_REQ_DFUN mms_term_upload_req
#define MMS_TERM_UPL_IND_SFUN u_termupl_ind
#define TERMUPL_SSI 0x01
#else
#define MMS_TERM_UPL_REQ_DFUN mms_req_not_supp
#define MMS_TERM_UPL_IND_SFUN u_ind_not_supp
#define TERMUPL_SSI 0x00
#endif

#if (MMS_RDDWN_EN & REQ_EN)
#define MMS_RDDWN_RSP_DFUN mms_null_pdu_dec
#define MMS_RDDWN_CONF_SFUN u_mp_rddwn_conf
#else
#define MMS_RDDWN_RSP_DFUN mms_rsp_not_supp
#define MMS_RDDWN_CONF_SFUN u_conf_not_supp
#endif

#if (MMS_RDDWN_EN & RESP_EN)
#define MMS_RDDWN_REQ_DFUN mms_rddwn_req
#define MMS_RDDWN_IND_SFUN u_rddwn_ind
#define RDDWN_SSI 0x80
#else
#define MMS_RDDWN_REQ_DFUN mms_req_not_supp
#define MMS_RDDWN_IND_SFUN u_ind_not_supp
#define RDDWN_SSI 0x00
#endif

#if (MMS_RDUPL_EN & REQ_EN)
#define MMS_RDUPL_RSP_DFUN mms_null_pdu_dec
#define MMS_RDUPL_CONF_SFUN u_mp_rdupl_conf
#else
#define MMS_RDUPL_RSP_DFUN mms_rsp_not_supp
#define MMS_RDUPL_CONF_SFUN u_conf_not_supp
#endif

#if (MMS_RDUPL_EN & RESP_EN)
#define MMS_RDUPL_REQ_DFUN mms_rdupl_req
#define MMS_RDUPL_IND_SFUN u_rdupl_ind
#define RDUPL_SSI 0x40
#else
#define MMS_RDUPL_REQ_DFUN mms_req_not_supp
#define MMS_RDUPL_IND_SFUN u_ind_not_supp
#define RDUPL_SSI 0x00
#endif

#if (MMS_LOAD_DOM_EN & REQ_EN)
#define MMS_LOAD_DOM_RSP_DFUN mms_null_pdu_dec
#define MMS_LOAD_DOM_CONF_SFUN u_mp_loaddom_conf
#else
#define MMS_LOAD_DOM_RSP_DFUN mms_rsp_not_supp
#define MMS_LOAD_DOM_CONF_SFUN u_conf_not_supp
#endif

#if (MMS_LOAD_DOM_EN & RESP_EN)
#define MMS_LOAD_DOM_REQ_DFUN mms_load_domain_req
#define MMS_LOAD_DOM_IND_SFUN u_loaddom_ind
#define LOADDOM_SSI 0x20
#else
#define MMS_LOAD_DOM_REQ_DFUN mms_req_not_supp
#define MMS_LOAD_DOM_IND_SFUN u_ind_not_supp
#define LOADDOM_SSI 0x00
#endif

```

```

#if (MMS_STR_DOM_EN & REQ_EN)
#define MMS_STR_DOM_RSP_DFUN mms_null_pdu_dec
#define MMS_STR_DOM_CONF_SFUN u_mp_storedom_conf
#else
#define MMS_STR_DOM_RSP_DFUN mms_rsp_not_supp
#define MMS_STR_DOM_CONF_SFUN u_conf_not_supp
#endif

#if (MMS_STR_DOM_EN & RESP_EN)
#define MMS_STR_DOM_REQ_DFUN mms_store_domain_req
#define MMS_STR_DOM_IND_SFUN u_storedom_ind
#define STRDOM_SSI 0x10
#else
#define MMS_STR_DOM_REQ_DFUN mms_req_not_supp
#define MMS_STR_DOM_IND_SFUN u_ind_not_supp
#define STRDOM_SSI 0x00
#endif

#if (MMS_DEL_DOM_EN & REQ_EN)
#define MMS_DEL_DOM_RSP_DFUN mms_null_pdu_dec
#define MMS_DEL_DOM_CONF_SFUN u_mp_deldom_conf
#else
#define MMS_DEL_DOM_RSP_DFUN mms_rsp_not_supp
#define MMS_DEL_DOM_CONF_SFUN u_conf_not_supp
#endif
#if (MMS_DEL_DOM_EN & RESP_EN)
#define MMS_DEL_DOM_REQ_DFUN mms_delete_domain_req
#define MMS_DEL_DOM_IND_SFUN u_deldom_ind
#define DELDOM_SSI 0x08
#else
#define MMS_DEL_DOM_REQ_DFUN mms_req_not_supp
#define MMS_DEL_DOM_IND_SFUN u_ind_not_supp
#define DELDOM_SSI 0x00
#endif

#if (MMS_GET_DOM_EN & REQ_EN)
#define MMS_GET_DOM_RSP_DFUN mms_get_dom_attr_rsp
#define MMS_GET_DOM_CONF_SFUN u_mp_getdom_conf
#else
#define MMS_GET_DOM_RSP_DFUN mms_rsp_not_supp
#define MMS_GET_DOM_CONF_SFUN u_conf_not_supp
#endif
#if (MMS_GET_DOM_EN & RESP_EN)
#define MMS_GET_DOM_REQ_DFUN mms_get_dom_attr_req
#define MMS_GET_DOM_IND_SFUN u_getdom_ind
#define GETDOM_SSI 0x04
#else
#define MMS_GET_DOM_REQ_DFUN mms_req_not_supp
#define MMS_GET_DOM_IND_SFUN u_ind_not_supp
#define GETDOM_SSI 0x00
#endif

#if (MMS_CRE_PI_EN & REQ_EN)
#define MMS_CRE_PI_RSP_DFUN mms_null_pdu_dec
#define MMS_CRE_PI_CONF_SFUN u_mp_crepi_conf
#else
#define MMS_CRE_PI_RSP_DFUN mms_rsp_not_supp
#define MMS_CRE_PI_CONF_SFUN u_conf_not_supp
#endif
#if (MMS_CRE_PI_EN & RESP_EN)
#define MMS_CRE_PI_REQ_DFUN mms_create_pi_req
#define MMS_CRE_PI_IND_SFUN u_crepi_ind
#define CREPI_SSI 0x02
#else
#define MMS_CRE_PI_REQ_DFUN mms_req_not_supp
#define MMS_CRE_PI_IND_SFUN u_ind_not_supp
#define CREPI_SSI 0x00
#endif

```

```

#if (MMS_DEL_PI_EN & REQ_EN)
#define MMS_DEL_PI_RSP_DFUN mms_null_pdu_dec
#define MMS_DEL_PI_CONF_SFUN u_mp_delpi_conf
#else
#define MMS_DEL_PI_RSP_DFUN mms_rsp_not_supp
#define MMS_DEL_PI_CONF_SFUN u_conf_not_supp
#endif
#if (MMS_DEL_PI_EN & RESP_EN)
#define MMS_DEL_PI_REQ_DFUN mms_delete_pi_req
#define MMS_DEL_PI_IND_SFUN u_delpi_ind
#define DELPI_SSI 0x01
#else
#define MMS_DEL_PI_REQ_DFUN mms_req_not_supp
#define MMS_DEL_PI_IND_SFUN u_ind_not_supp
#define DELPI_SSI 0x00
#endif

#if (MMS_START_EN & REQ_EN)
#define MMS_START_RSP_DFUN mms_null_pdu_dec
#define MMS_START_CONF_SFUN u_mp_start_conf
#else
#define MMS_START_RSP_DFUN mms_rsp_not_supp
#define MMS_START_CONF_SFUN u_conf_not_supp
#endif
#if (MMS_START_EN & RESP_EN)
#define MMS_START_REQ_DFUN mms_start_req
#define MMS_START_IND_SFUN u_start_ind
#define START_SSI 0x80
#else
#define MMS_START_REQ_DFUN mms_req_not_supp
#define MMS_START_IND_SFUN u_ind_not_supp
#define START_SSI 0x00
#endif

#if (MMS_STOP_EN & REQ_EN)
#define MMS_STOP_RSP_DFUN mms_null_pdu_dec
#define MMS_STOP_CONF_SFUN u_mp_stop_conf
#else
#define MMS_STOP_RSP_DFUN mms_rsp_not_supp
#define MMS_STOP_CONF_SFUN u_conf_not_supp
#endif
#if (MMS_STOP_EN & RESP_EN)
#define MMS_STOP_REQ_DFUN mms_stop_req
#define MMS_STOP_IND_SFUN u_stop_ind
#define STOP_SSI 0x40
#else
#define MMS_STOP_REQ_DFUN mms_req_not_supp
#define MMS_STOP_IND_SFUN u_ind_not_supp
#define STOP_SSI 0x00
#endif

#if (MMS_RESUME_EN & REQ_EN)
#define MMS_RESUME_RSP_DFUN mms_null_pdu_dec
#define MMS_RESUME_CONF_SFUN u_mp_resume_conf
#else
#define MMS_RESUME_RSP_DFUN mms_rsp_not_supp
#define MMS_RESUME_CONF_SFUN u_conf_not_supp
#endif
#if (MMS_RESUME_EN & RESP_EN)
#define MMS_RESUME_REQ_DFUN mms_resume_req
#define MMS_RESUME_IND_SFUN u_resume_ind
#define RESUME_SSI 0x20
#else
#define MMS_RESUME_REQ_DFUN mms_req_not_supp
#define MMS_RESUME_IND_SFUN u_ind_not_supp
#define RESUME_SSI 0x00
#endif

```

```

#if (MMS_RESET_EN & REQ_EN)
#define MMS_RESET_RSP_DFUN mms_null_pdu_dec
#define MMS_RESET_CONF_SFUN u_mp_reset_conf
#else

#define MMS_RESET_RSP_DFUN mms_rsp_not_supp
#define MMS_RESET_CONF_SFUN u_conf_not_supp
#endif
#if (MMS_RESET_EN & RESP_EN)
#define MMS_RESET_REQ_DFUN mms_reset_req
#define MMS_RESET_IND_SFUN u_reset_ind
#define RESET_SSI 0x10
#else
#define MMS_RESET_REQ_DFUN mms_req_not_supp
#define MMS_RESET_IND_SFUN u_ind_not_supp
#define RESET_SSI 0x00
#endif

#if (MMS_KILL_EN & REQ_EN)
#define MMS_KILL_RSP_DFUN mms_null_pdu_dec
#define MMS_KILL_CONF_SFUN u_mp_kill_conf
#else
#define MMS_KILL_RSP_DFUN mms_rsp_not_supp
#define MMS_KILL_CONF_SFUN u_conf_not_supp
#endif
#if (MMS_KILL_EN & RESP_EN)
#define MMS_KILL_REQ_DFUN mms_kill_req
#define MMS_KILL_IND_SFUN u_kill_ind
#define KILL_SSI 0x08
#else
#define MMS_KILL_REQ_DFUN mms_req_not_supp
#define MMS_KILL_IND_SFUN u_ind_not_supp
#define KILL_SSI 0x00
#endif

#if (MMS_GET_PI_EN & REQ_EN)
#define MMS_GET_PI_RSP_DFUN mms_get_pi_rsp
#define MMS_GET_PI_CONF_SFUN u_mp_getpi_conf
#else
#define MMS_GET_PI_RSP_DFUN mms_rsp_not_supp
#define MMS_GET_PI_CONF_SFUN u_conf_not_supp
#endif
#if (MMS_GET_PI_EN & RESP_EN)
#define MMS_GET_PI_REQ_DFUN mms_get_pi_req
#define MMS_GET_PI_IND_SFUN u_getpi_ind
#define GETPI_SSI 0x04
#else
#define MMS_GET_PI_REQ_DFUN mms_req_not_supp
#define MMS_GET_PI_IND_SFUN u_ind_not_supp
#define GETPI_SSI 0x00
#endif

#if (MMS_OBTAINFILE_EN & REQ_EN)
#define MMS_OBTAINFILE_RSP_DFUN mms_null_pdu_dec
#define MMS_OBTAINFILE_CONF_SFUN u_mp_obtfile_conf
#else
#define MMS_OBTAINFILE_RSP_DFUN mms_rsp_not_supp
#define MMS_OBTAINFILE_CONF_SFUN u_conf_not_supp
#endif
#if (MMS_OBTAINFILE_EN & RESP_EN)
#define MMS_OBTAINFILE_REQ_DFUN mms_obtain_file_req
#define MMS_OBTAINFILE_IND_SFUN u_obtfile_ind
#define OBTFILE_SSI 0x02
#else
#define MMS_OBTAINFILE_REQ_DFUN mms_req_not_supp
#define MMS_OBTAINFILE_IND_SFUN u_ind_not_supp
#define OBTFILE_SSI 0x00
#endif

```

```

#if (MMS_DEFEC_EN & REQ_EN)
#define MMS_DEFEC_RSP_DFUN mms_rsp_not_supp
#define MMS_DEFEC_CONF_SFUN u_conf_not_supp
#else
#define MMS_DEFEC_RSP_DFUN mms_rsp_not_supp
#define MMS_DEFEC_CONF_SFUN u_conf_not_supp
#endif
#if (MMS_DEFEC_EN & RESP_EN)
#define MMS_DEFEC_REQ_DFUN mms_req_not_supp
#define MMS_DEFEC_IND_SFUN u_ind_not_supp
#define DEFEC_SSI 0x01
#else
#define MMS_DEFEC_REQ_DFUN mms_req_not_supp
#define MMS_DEFEC_IND_SFUN u_ind_not_supp
#define DEFEC_SSI 0x00
#endif

#if (MMS_DELEC_EN & REQ_EN)
#define MMS_DELEC_RSP_DFUN mms_rsp_not_supp
#define MMS_DELEC_CONF_SFUN u_conf_not_supp
#else
#define MMS_DELEC_RSP_DFUN mms_rsp_not_supp
#define MMS_DELEC_CONF_SFUN u_conf_not_supp
#endif
#if (MMS_DELEC_EN & RESP_EN)
#define MMS_DELEC_REQ_DFUN mms_req_not_supp
#define MMS_DELEC_IND_SFUN u_ind_not_supp
#define DELEC_SSI 0x80
#else
#define MMS_DELEC_REQ_DFUN mms_req_not_supp
#define MMS_DELEC_IND_SFUN u_ind_not_supp
#define DELEC_SSI 0x00
#endif

#if (MMS_GETECA_EN & REQ_EN)
#define MMS_GETECA_RSP_DFUN mms_geteca_rsp
#define MMS_GETECA_CONF_SFUN u_mp_geteca_conf
#else
#define MMS_GETECA_RSP_DFUN mms_rsp_not_supp
#define MMS_GETECA_CONF_SFUN u_conf_not_supp
#endif
#if (MMS_GETECA_EN & RESP_EN)
#define MMS_GETECA_REQ_DFUN mms_geteca_req
#define MMS_GETECA_IND_SFUN u_geteca_ind
#define GETECA_SSI 0x40
#else
#define MMS_GETECA_REQ_DFUN mms_req_not_supp
#define MMS_GETECA_IND_SFUN u_ind_not_supp
#define GETECA_SSI 0x00
#endif

#if (MMS_REPECS_EN & REQ_EN)
#define MMS_REPECS_RSP_DFUN mms_repecs_rsp
#define MMS_REPECS_CONF_SFUN u_mp_repecs_conf
#else
#define MMS_REPECS_RSP_DFUN mms_rsp_not_supp
#define MMS_REPECS_CONF_SFUN u_conf_not_supp
#endif
#if (MMS_REPECS_EN & RESP_EN)
#define MMS_REPECS_REQ_DFUN mms_repecs_req
#define MMS_REPECS_IND_SFUN u_repecs_ind
#define REPECS_SSI 0x20
#else
#define MMS_REPECS_REQ_DFUN mms_req_not_supp
#define MMS_REPECS_IND_SFUN u_ind_not_supp
#define REPECS_SSI 0x00
#endif

```

```

#if (MMS_ALTECM_EN & REQ_EN)
#define MMS_ALTECM_RSP_DFUNK mms_rsp_not_supp
#define MMS_ALTECM_CONF_SFUNK u_conf_not_supp
#else

#define MMS_ALTECM_RSP_DFUNK mms_rsp_not_supp
#define MMS_ALTECM_CONF_SFUNK u_conf_not_supp
#endif
#if (MMS_ALTECM_EN & RESP_EN)
#define MMS_ALTECM_REQ_DFUNK mms_req_not_supp
#define MMS_ALTECM_IND_SFUNK u_ind_not_supp
#define ALTECM_SSI 0x10
#else
#define MMS_ALTECM_REQ_DFUNK mms_req_not_supp
#define MMS_ALTECM_IND_SFUNK u_ind_not_supp
#define ALTECM_SSI 0x00
#endif

#if (MMS_TRIGE_EN & REQ_EN)
#define MMS_TRIGE_RSP_DFUNK mms_rsp_not_supp
#define MMS_TRIGE_CONF_SFUNK u_conf_not_supp
#else
#define MMS_TRIGE_RSP_DFUNK mms_rsp_not_supp
#define MMS_TRIGE_CONF_SFUNK u_conf_not_supp
#endif
#if (MMS_TRIGE_EN & RESP_EN)
#define MMS_TRIGE_REQ_DFUNK mms_req_not_supp
#define MMS_TRIGE_IND_SFUNK u_ind_not_supp
#define TRIGE_SSI 0x08
#else
#define MMS_TRIGE_REQ_DFUNK mms_req_not_supp
#define MMS_TRIGE_IND_SFUNK u_ind_not_supp
#define TRIGE_SSI 0x00
#endif

#if (MMS_DEFEA_EN & REQ_EN)
#define MMS_DEFEA_RSP_DFUNK mms_rsp_not_supp
#define MMS_DEFEA_CONF_SFUNK u_conf_not_supp
#else
#define MMS_DEFEA_RSP_DFUNK mms_rsp_not_supp
#define MMS_DEFEA_CONF_SFUNK u_conf_not_supp
#endif
#if (MMS_DEFEA_EN & RESP_EN)
#define MMS_DEFEA_REQ_DFUNK mms_req_not_supp
#define MMS_DEFEA_IND_SFUNK u_ind_not_supp
#define DEFEA_SSI 0x04
#else
#define MMS_DEFEA_REQ_DFUNK mms_req_not_supp
#define MMS_DEFEA_IND_SFUNK u_ind_not_supp
#define DEFEA_SSI 0x00
#endif

#if (MMS_DELEA_EN & REQ_EN)
#define MMS_DELEA_RSP_DFUNK mms_rsp_not_supp
#define MMS_DELEA_CONF_SFUNK u_conf_not_supp
#else
#define MMS_DELEA_RSP_DFUNK mms_rsp_not_supp
#define MMS_DELEA_CONF_SFUNK u_conf_not_supp
#endif
#if (MMS_DELEA_EN & RESP_EN)
#define MMS_DELEA_REQ_DFUNK mms_req_not_supp
#define MMS_DELEA_IND_SFUNK u_ind_not_supp
#define DELEA_SSI 0x02
#else
#define MMS_DELEA_REQ_DFUNK mms_req_not_supp
#define MMS_DELEA_IND_SFUNK u_ind_not_supp
#define DELEA_SSI 0x00
#endif

```

```

#if (MMS_GETEAA_EN & REQ_EN)
#define MMS_GETEAA_RSP_DFUN mms_rsp_not_supp
#define MMS_GETEAA_CONF_SFUN u_conf_not_supp
#else
#define MMS_GETEAA_RSP_DFUN mms_rsp_not_supp
#define MMS_GETEAA_CONF_SFUN u_conf_not_supp
#endif
#if (MMS_GETEAA_EN & RESP_EN)
#define MMS_GETEAA_REQ_DFUN mms_req_not_supp
#define MMS_GETEAA_IND_SFUN u_ind_not_supp
#define GETEAA_SSI 0x01
#else
#define MMS_GETEAA_REQ_DFUN mms_req_not_supp
#define MMS_GETEAA_IND_SFUN u_ind_not_supp
#define GETEAA_SSI 0x00
#endif

#if (MMS_REPEAS_EN & REQ_EN)
#define MMS_REPEAS_RSP_DFUN mms_rsp_not_supp
#define MMS_REPEAS_CONF_SFUN u_conf_not_supp
#else
#define MMS_REPEAS_RSP_DFUN mms_rsp_not_supp
#define MMS_REPEAS_CONF_SFUN u_conf_not_supp
#endif
#if (MMS_REPEAS_EN & RESP_EN)
#define MMS_REPEAS_REQ_DFUN mms_req_not_supp
#define MMS_REPEAS_IND_SFUN u_ind_not_supp
#define REPEAS_SSI 0x80
#else
#define MMS_REPEAS_REQ_DFUN mms_req_not_supp
#define MMS_REPEAS_IND_SFUN u_ind_not_supp
#define REPEAS_SSI 0x00
#endif

#if (MMS_DEFEE_EN & REQ_EN)
#define MMS_DEFEE_RSP_DFUN mms_rsp_not_supp
#define MMS_DEFEE_CONF_SFUN u_conf_not_supp
#else
#define MMS_DEFEE_RSP_DFUN mms_rsp_not_supp
#define MMS_DEFEE_CONF_SFUN u_conf_not_supp
#endif
#if (MMS_DEFEE_EN & RESP_EN)
#define MMS_DEFEE_REQ_DFUN mms_req_not_supp
#define MMS_DEFEE_IND_SFUN u_ind_not_supp
#define DEFEE_SSI 0x40
#else
#define MMS_DEFEE_REQ_DFUN mms_req_not_supp
#define MMS_DEFEE_IND_SFUN u_ind_not_supp
#define DEFEE_SSI 0x00
#endif

#if (MMS_DELEE_EN & REQ_EN)
#define MMS_DELEE_RSP_DFUN mms_rsp_not_supp
#define MMS_DELEE_CONF_SFUN u_conf_not_supp
#else
#define MMS_DELEE_RSP_DFUN mms_rsp_not_supp
#define MMS_DELEE_CONF_SFUN u_conf_not_supp
#endif
#if (MMS_DELEE_EN & RESP_EN)
#define MMS_DELEE_REQ_DFUN mms_req_not_supp
#define MMS_DELEE_IND_SFUN u_ind_not_supp
#define DELEE_SSI 0x20
#else
#define MMS_DELEE_REQ_DFUN mms_req_not_supp
#define MMS_DELEE_IND_SFUN u_ind_not_supp
#define DELEE_SSI 0x00
#endif

```

```

#if (MMS_ALTEE_EN & REQ_EN)
#define MMS_ALTEE_RSP_DFUN mms_rsp_not_supp
#define MMS_ALTEE_CONF_SFUN u_conf_not_supp
#else

#define MMS_ALTEE_RSP_DFUN mms_rsp_not_supp
#define MMS_ALTEE_CONF_SFUN u_conf_not_supp
#endif
#if (MMS_ALTEE_EN & RESP_EN)
#define MMS_ALTEE_REQ_DFUN mms_req_not_supp
#define MMS_ALTEE_IND_SFUN u_ind_not_supp
#define ALTEE_SSI 0x10
#else
#define MMS_ALTEE_REQ_DFUN mms_req_not_supp
#define MMS_ALTEE_IND_SFUN u_ind_not_supp
#define ALTEE_SSI 0x00
#endif

#if (MMS_REPEES_EN & REQ_EN)
#define MMS_REPEES_RSP_DFUN mms_rsp_not_supp
#define MMS_REPEES_CONF_SFUN u_conf_not_supp
#else
#define MMS_REPEES_RSP_DFUN mms_rsp_not_supp
#define MMS_REPEES_CONF_SFUN u_conf_not_supp
#endif
#if (MMS_REPEES_EN & RESP_EN)
#define MMS_REPEES_REQ_DFUN mms_req_not_supp
#define MMS_REPEES_IND_SFUN u_ind_not_supp
#define REPEES_SSI 0x08
#else
#define MMS_REPEES_REQ_DFUN mms_req_not_supp
#define MMS_REPEES_IND_SFUN u_ind_not_supp
#define REPEES_SSI 0x00
#endif

#if (MMS_GETEEA_EN & REQ_EN)
#define MMS_GETEEA_RSP_DFUN mms_rsp_not_supp
#define MMS_GETEEA_CONF_SFUN u_conf_not_supp
#else
#define MMS_GETEEA_RSP_DFUN mms_rsp_not_supp
#define MMS_GETEEA_CONF_SFUN u_conf_not_supp
#endif
#if (MMS_GETEEA_EN & RESP_EN)
#define MMS_GETEEA_REQ_DFUN mms_req_not_supp
#define MMS_GETEEA_IND_SFUN u_ind_not_supp
#define GETEEA_SSI 0x04
#else
#define MMS_GETEEA_REQ_DFUN mms_req_not_supp
#define MMS_GETEEA_IND_SFUN u_ind_not_supp
#define GETEEA_SSI 0x00
#endif

#if (MMS_ACKEVNOT_EN & REQ_EN)
#define MMS_ACKEVNOT_RSP_DFUN mms_null_pdu_dec
#define MMS_ACKEVNOT_CONF_SFUN u_mp_ackevnot_conf
#else
#define MMS_ACKEVNOT_RSP_DFUN mms_rsp_not_supp
#define MMS_ACKEVNOT_CONF_SFUN u_conf_not_supp
#endif
#if (MMS_ACKEVNOT_EN & RESP_EN)
#define MMS_ACKEVNOT_REQ_DFUN mms_ackevnot_req
#define MMS_ACKEVNOT_IND_SFUN u_ackevnot_ind
#define ACKEVNOT_SSI 0x02
#else
#define MMS_ACKEVNOT_REQ_DFUN mms_req_not_supp
#define MMS_ACKEVNOT_IND_SFUN u_ind_not_supp
#define ACKEVNOT_SSI 0x00
#endif

```

```

#if (MMS_GETAS_EN & REQ_EN)
#define MMS_GETAS_RSP_DFUN mms_getas_rsp
#define MMS_GETAS_CONF_SFUN u_mp_getas_conf
#else
#define MMS_GETAS_RSP_DFUN mms_rsp_not_supp
#define MMS_GETAS_CONF_SFUN u_conf_not_supp
#endif
#if (MMS_GETAS_EN & RESP_EN)
#define MMS_GETAS_REQ_DFUN mms_getas_req
#define MMS_GETAS_IND_SFUN u_getas_ind
#define GETAS_SSI 0x01
#else
#define MMS_GETAS_REQ_DFUN mms_req_not_supp
#define MMS_GETAS_IND_SFUN u_ind_not_supp
#define GETAS_SSI 0x00
#endif

#if (MMS_GETAES_EN & REQ_EN)
#define MMS_GETAES_RSP_DFUN mms_rsp_not_supp
#define MMS_GETAES_CONF_SFUN u_conf_not_supp
#else
#define MMS_GETAES_RSP_DFUN mms_rsp_not_supp
#define MMS_GETAES_CONF_SFUN u_conf_not_supp
#endif
#if (MMS_GETAES_EN & RESP_EN)
#define MMS_GETAES_REQ_DFUN mms_req_not_supp
#define MMS_GETAES_IND_SFUN u_ind_not_supp
#define GETAES_SSI 0x80
#else
#define MMS_GETAES_REQ_DFUN mms_req_not_supp
#define MMS_GETAES_IND_SFUN u_ind_not_supp
#define GETAES_SSI 0x00
#endif

#if (MMS_JREAD_EN & REQ_EN)
#define MMS_JREAD_RSP_DFUN mms_jread_rsp
#define MMS_JREAD_CONF_SFUN u_mp_jread_conf
#else
#define MMS_JREAD_RSP_DFUN mms_rsp_not_supp
#define MMS_JREAD_CONF_SFUN u_conf_not_supp
#endif
#if (MMS_JREAD_EN & RESP_EN)
#define MMS_JREAD_REQ_DFUN mms_jread_req
#define MMS_JREAD_IND_SFUN u_jread_ind
#define JREAD_SSI 0x40
#else
#define MMS_JREAD_REQ_DFUN mms_req_not_supp
#define MMS_JREAD_IND_SFUN u_ind_not_supp
#define JREAD_SSI 0x00
#endif

#if (MMS_JWRITE_EN & REQ_EN)
#define MMS_JWRITE_RSP_DFUN mms_null_pdu_dec
#define MMS_JWRITE_CONF_SFUN u_mp_jwrite_conf
#else
#define MMS_JWRITE_RSP_DFUN mms_rsp_not_supp
#define MMS_JWRITE_CONF_SFUN u_conf_not_supp
#endif
#if (MMS_JWRITE_EN & RESP_EN)
#define MMS_JWRITE_REQ_DFUN mms_jwrite_req
#define MMS_JWRITE_IND_SFUN u_jwrite_ind
#define JWRITE_SSI 0x20
#else
#define MMS_JWRITE_REQ_DFUN mms_req_not_supp
#define MMS_JWRITE_IND_SFUN u_ind_not_supp
#define JWRITE_SSI 0x00
#endif

```

```

#if (MMS_JINIT_EN & REQ_EN)
#define MMS_JINIT_RSP_DFUN mms_jinit_rsp
#define MMS_JINIT_CONF_SFUN u_mp_jinit_conf
#else
#define MMS_JINIT_RSP_DFUN mms_rsp_not_supp
#define MMS_JINIT_CONF_SFUN u_conf_not_supp
#endif
#if (MMS_JINIT_EN & RESP_EN)
#define MMS_JINIT_REQ_DFUN mms_jinit_req
#define MMS_JINIT_IND_SFUN u_jinit_ind
#define JINIT_SSI 0x10
#else
#define MMS_JINIT_REQ_DFUN mms_req_not_supp
#define MMS_JINIT_IND_SFUN u_ind_not_supp
#define JINIT_SSI 0x00
#endif

#if (MMS_JSTAT_EN & REQ_EN)
#define MMS_JSTAT_RSP_DFUN mms_jstat_rsp
#define MMS_JSTAT_CONF_SFUN u_mp_jstat_conf
#else
#define MMS_JSTAT_RSP_DFUN mms_rsp_not_supp
#define MMS_JSTAT_CONF_SFUN u_conf_not_supp
#endif
#if (MMS_JSTAT_EN & RESP_EN)
#define MMS_JSTAT_REQ_DFUN mms_jstat_req
#define MMS_JSTAT_IND_SFUN u_jstat_ind
#define JSTAT_SSI 0x08
#else
#define MMS_JSTAT_REQ_DFUN mms_req_not_supp
#define MMS_JSTAT_IND_SFUN u_ind_not_supp
#define JSTAT_SSI 0x00
#endif

#if (MMS_JCREATE_EN & REQ_EN)
#define MMS_JCREATE_RSP_DFUN mms_rsp_not_supp
#define MMS_JCREATE_CONF_SFUN u_conf_not_supp
#else
#define MMS_JCREATE_RSP_DFUN mms_rsp_not_supp
#define MMS_JCREATE_CONF_SFUN u_conf_not_supp
#endif
#if (MMS_JCREATE_EN & RESP_EN)
#define MMS_JCREATE_REQ_DFUN mms_req_not_supp
#define MMS_JCREATE_IND_SFUN u_ind_not_supp
#define JCREATE_SSI 0x04
#else
#define MMS_JCREATE_REQ_DFUN mms_req_not_supp
#define MMS_JCREATE_IND_SFUN u_ind_not_supp
#define JCREATE_SSI 0x00
#endif

#if (MMS_JDELETE_EN & REQ_EN)
#define MMS_JDELETE_RSP_DFUN mms_rsp_not_supp
#define MMS_JDELETE_CONF_SFUN u_conf_not_supp
#else
#define MMS_JDELETE_RSP_DFUN mms_rsp_not_supp
#define MMS_JDELETE_CONF_SFUN u_conf_not_supp
#endif
#if (MMS_JDELETE_EN & RESP_EN)
#define MMS_JDELETE_REQ_DFUN mms_req_not_supp
#define MMS_JDELETE_IND_SFUN u_ind_not_supp
#define JDELETE_SSI 0x02
#else
#define MMS_JDELETE_REQ_DFUN mms_req_not_supp
#define MMS_JDELETE_IND_SFUN u_ind_not_supp
#define JDELETE_SSI 0x00
#endif

```

```

#if (MMS_GETCL_EN & REQ_EN)
#define MMS_GETCL_RSP_DFUN mms_getcl_rsp
#define MMS_GETCL_CONF_SFUN u_mp_getcl_conf
#else
#define MMS_GETCL_RSP_DFUN mms_rsp_not_supp
#define MMS_GETCL_CONF_SFUN u_conf_not_supp
#endif
#if (MMS_GETCL_EN & RESP_EN)
#define MMS_GETCL_REQ_DFUN mms_getcl_req
#define MMS_GETCL_IND_SFUN u_getcl_ind
#define GETCL_SSI 0x01
#else
#define MMS_GETCL_REQ_DFUN mms_req_not_supp
#define MMS_GETCL_IND_SFUN u_ind_not_supp
#define GETCL_SSI 0x00
#endif

#if (MMS_FOPEN_EN & REQ_EN)
#define MMS_FOPEN_RSP_DFUN mms_file_open_rsp
#define MMS_FOPEN_CONF_SFUN u_mp_fopen_conf
#define MMS_MV_FOPEN_CONF_SFUN u_mv_fopen_conf
#else
#define MMS_FOPEN_RSP_DFUN mms_rsp_not_supp
#define MMS_FOPEN_CONF_SFUN u_conf_not_supp
#define MMS_MV_FOPEN_CONF_SFUN u_conf_not_supp
#endif
#if (MMS_FOPEN_EN & RESP_EN)
#define MMS_FOPEN_REQ_DFUN mms_file_open_req
#define MMS_FOPEN_IND_SFUN u_fopen_ind
#define FOPEN_SSI 0x80
#else
#define MMS_FOPEN_REQ_DFUN mms_req_not_supp
#define MMS_FOPEN_IND_SFUN u_ind_not_supp
#define FOPEN_SSI 0x00
#endif

#if (MMS_FREAD_EN & REQ_EN)
#define MMS_FREAD_RSP_DFUN mms_file_read_rsp
#define MMS_FREAD_CONF_SFUN u_mp_fread_conf
#define MMS_MV_FREAD_CONF_SFUN u_mv_fread_conf
#else
#define MMS_FREAD_RSP_DFUN mms_rsp_not_supp
#define MMS_FREAD_CONF_SFUN u_conf_not_supp
#define MMS_MV_FREAD_CONF_SFUN u_conf_not_supp
#endif
#if (MMS_FREAD_EN & RESP_EN)
#define MMS_FREAD_REQ_DFUN mms_file_read_req
#define MMS_FREAD_IND_SFUN u_fread_ind
#define FREAD_SSI 0x40
#else
#define MMS_FREAD_REQ_DFUN mms_req_not_supp
#define MMS_FREAD_IND_SFUN u_ind_not_supp
#define FREAD_SSI 0x00
#endif

```

```

#if (MMS_FCLOSE_EN & REQ_EN)
#define MMS_FCLOSE_RSP_DFUN mms_null_pdu_dec
#define MMS_FCLOSE_CONF_SFUN u_mp_fclose_conf
#define MMS_MV_FCLOSE_CONF_SFUN u_mv_fclose_conf
#else
#define MMS_FCLOSE_RSP_DFUN mms_rsp_not_supp
#define MMS_FCLOSE_CONF_SFUN u_conf_not_supp
#define MMS_MV_FCLOSE_CONF_SFUN u_conf_not_supp
#endif
#if (MMS_FCLOSE_EN & RESP_EN)
#define MMS_FCLOSE_REQ_DFUN mms_file_close_req
#define MMS_FCLOSE_IND_SFUN u_fclose_ind
#define FCLOSE_SSI 0x20
#else
#define MMS_FCLOSE_REQ_DFUN mms_req_not_supp
#define MMS_FCLOSE_IND_SFUN u_ind_not_supp
#define FCLOSE_SSI 0x00
#endif

#if ( (MMS_FOPEN_EN & MMS_FREAD_EN & MMS_FCLOSE_EN) & REQ_EN)
#define MMS_MV_FCOPY_CONF_SFUN u_mv_fcopy_conf
#else
#define MMS_MV_FCOPY_CONF_SFUN u_conf_not_supp
#endif

#if (MMS_FRENAME_EN & REQ_EN)
#define MMS_FRENAME_RSP_DFUN mms_null_pdu_dec
#define MMS_FRENAME_CONF_SFUN u_mp_frename_conf
#else
#define MMS_FRENAME_RSP_DFUN mms_rsp_not_supp
#define MMS_FRENAME_CONF_SFUN u_conf_not_supp
#endif
#if (MMS_FRENAME_EN & RESP_EN)
#define MMS_FRENAME_REQ_DFUN mms_file_rename_req
#define MMS_FRENAME_IND_SFUN u_frename_ind
#define FRENAME_SSI 0x10
#else
#define MMS_FRENAME_REQ_DFUN mms_req_not_supp
#define MMS_FRENAME_IND_SFUN u_ind_not_supp
#define FRENAME_SSI 0x00
#endif

#if (MMS_FDELETE_EN & REQ_EN)
#define MMS_FDELETE_RSP_DFUN mms_null_pdu_dec
#define MMS_FDELETE_CONF_SFUN u_mp_fdelete_conf
#else
#define MMS_FDELETE_RSP_DFUN mms_rsp_not_supp
#define MMS_FDELETE_CONF_SFUN u_conf_not_supp
#endif
#if (MMS_FDELETE_EN & RESP_EN)
#define MMS_FDELETE_REQ_DFUN mms_file_delete_req
#define MMS_FDELETE_IND_SFUN u_fdelete_ind
#define FDELETE_SSI 0x08
#else
#define MMS_FDELETE_REQ_DFUN mms_req_not_supp
#define MMS_FDELETE_IND_SFUN u_ind_not_supp
#define FDELETE_SSI 0x00
#endif

```

```

#if (MMS_FDIR_EN & REQ_EN)
#define MMS_FDIR_RSP_DFUN mms_file_dir_rsp
#define MMS_FDIR_CONF_SFUN u_mp_fdir_conf
#else
#define MMS_FDIR_RSP_DFUN mms_rsp_not_supp
#define MMS_FDIR_CONF_SFUN u_conf_not_supp
#endif
#if (MMS_FDIR_EN & RESP_EN)
#define MMS_FDIR_REQ_DFUN mms_file_dir_req
#define MMS_FDIR_IND_SFUN u_fdir_ind
#define FDIR_SSI 0x04
#else
#define MMS_FDIR_REQ_DFUN mms_req_not_supp
#define MMS_FDIR_IND_SFUN u_ind_not_supp
#define FDIR_SSI 0x00
#endif

/*****
/* UNCONFIRMED SERVICES */
*****/

#if (MMS_INFO_EN & RESP_EN)
#define MMS_INFO_REQ_DFUN mms_info_rpt_req
#define MMS_INFO_IND_SFUN u_info_ind
#define INFO_SSI 0x80
#else
#define MMS_INFO_REQ_DFUN mms_req_not_supp
#define MMS_INFO_IND_SFUN u_ind_not_supp
#define INFO_SSI 0x00
#endif

#if (MMS_USTATUS_EN & RESP_EN)
#define MMS_USTATUS_REQ_DFUN mms_ustatus_req
#define MMS_USTATUS_IND_SFUN u_ustatus_ind
#define USTATUS_SSI 0x01
#else
#define MMS_USTATUS_REQ_DFUN mms_req_not_supp
#define MMS_USTATUS_IND_SFUN u_ind_not_supp
#define USTATUS_SSI 0x00
#endif

#if (MMS_EVNOT_EN & RESP_EN)
#define MMS_EVNOT_REQ_DFUN mms_evnot_req
#define MMS_EVNOT_IND_SFUN u_evnot_ind
#define EVNOT_SSI 0x40
#else
#define MMS_EVNOT_REQ_DFUN mms_req_not_supp
#define MMS_EVNOT_IND_SFUN u_ind_not_supp
#define EVNOT_SSI 0x00
#endif

BYTE *mms_op_string [] = /* opcode is index into table */
(
"STATUS", /* 00 STATUS */
"GET_NAMLIST", /* 01 GET_NAMLIST */
"IDENTIFY", /* 02 IDENTIFY */
"RENAME", /* 03 RENAME */
"READ", /* 04 READ */
"WRITE", /* 05 WRITE */
"GET_VARDEF", /* 06 GET_VARDEF */
"DEF_VARNAM", /* 07 DEF_VARNAM */
"DEF_SCATTERED", /* 08 DEF_SCATTERED */
"GET_SCATTERED", /* 09 GET_SCATTERED */
"DEL_VARNAM", /* 10 DEL_VARNAM */
"DEF_VARLIST", /* 11 DEF_VARLIST */
"GET_VARLIST", /* 12 GET_VARLIST */
"DEL_VARLIST", /* 13 DEL_VARLIST */
"DEF_TYPENAM", /* 14 DEF_TYPENAM */

```

"GET_TYPEDEF",	/* 15 GET_TYPEDEF	*/
"DEL_TYPENAM",	/* 16 DEL_TYPENAM	*/
"INPUT",	/* 17 INPUT	*/
"OUTPUT",	/* 18 OUTPUT	*/
"TAKE_CONTROL",	/* 19 TAKE_CONTROL	*/
"REL_CONTROL",	/* 20 REL_CONTROL	*/
"DEF_SEMAPHORE",	/* 21 DEF_SEMAPHORE	*/
"DEL_SEMAPHORE",	/* 22 DEL_SEMAPHORE	*/
"REP_SEM_STATUS",	/* 23 REP_SEM_STATUS	*/
"REP_SEM_POOL_STATUS",	/* 24 REP_SEM_POOL_STATUS	*/
"REP_SEM_ENTRY_STATUS",	/* 25 REP_SEM_ENTRY_STATUS	*/
"INIT_DOWNLOAD",	/* 26 INIT_DOWNLOAD	*/
"DOWN_LOAD",	/* 27 DOWN_LOAD	*/
"TERM_DOWNLOAD",	/* 28 TERM_DOWNLOAD	*/
"INIT_UPLOAD",	/* 29 INIT_UPLOAD	*/
"UP_LOAD",	/* 30 UP_LOAD	*/
"TERM_UPLOAD",	/* 31 TERM_UPLOAD	*/
"REQ_DOWNLOAD",	/* 32 REQ_DOWNLOAD	*/
"REQ_UPLOAD",	/* 33 REQ_UPLOAD	*/
"LOAD_DOMAIN",	/* 34 LOAD_DOMAIN	*/
"STORE_DOMAIN",	/* 35 STORE_DOMAIN	*/
"DELETE_DOMAIN",	/* 36 DELETE_DOMAIN	*/
"GET_DOMAIN",	/* 37 GET_DOMAIN	*/
"CREATE_PROGRAM_INVOCATION",	/* 38 CREATE_PROGRAM_INVOCATION	*/
"DEL_PROGRAM_INVOCATION",	/* 39 DEL_PROGRAM_INVOCATION	*/
"START",	/* 40 START	*/
"STOP",	/* 41 STOP	*/
"RESUME",	/* 42 RESUME	*/
"RESET",	/* 43 RESET	*/
"KILL",	/* 44 KILL	*/
"GET_PROGRAM_INVOCATION",	/* 45 GET_PROGRAM_INVOCATION	*/
"OBTAIN_FILE",	/* 46 OBTAIN_FILE	*/
"MMS_EVMGMT_SERVICE",	/* 47 MMS_EVMGMT_SERVICE	*/
"MMS_EVMGMT_SERVICE",	/* 48 MMS_EVMGMT_SERVICE	*/
"GET_EVENT_CONDITION_ATTR",	/* 49 GET_EVENT_CONDITION_ATTR	*/
"REPORT_EVENT_COND_STATUS",	/* 50 REPORT_EVENT_COND_STATUS	*/
"MMS_EVMGMT_SERVICE",	/* 51 MMS_EVMGMT_SERVICE	*/
"MMS_EVMGMT_SERVICE",	/* 52 MMS_EVMGMT_SERVICE	*/
"MMS_EVMGMT_SERVICE",	/* 53 MMS_EVMGMT_SERVICE	*/
"MMS_EVMGMT_SERVICE",	/* 54 MMS_EVMGMT_SERVICE	*/
"MMS_EVMGMT_SERVICE",	/* 55 MMS_EVMGMT_SERVICE	*/
"MMS_EVMGMT_SERVICE",	/* 56 MMS_EVMGMT_SERVICE	*/
"MMS_EVMGMT_SERVICE",	/* 57 MMS_EVMGMT_SERVICE	*/
"MMS_EVMGMT_SERVICE",	/* 58 MMS_EVMGMT_SERVICE	*/
"MMS_EVMGMT_SERVICE",	/* 59 MMS_EVMGMT_SERVICE	*/
"MMS_EVMGMT_SERVICE",	/* 60 MMS_EVMGMT_SERVICE	*/
"MMS_EVMGMT_SERVICE",	/* 61 MMS_EVMGMT_SERVICE	*/
"ACK_EVENT_NOTIFICATION",	/* 62 ACK_EVENT_NOTIFICATION	*/
"GET_ALRAM_SUMMARY",	/* 63 GET_ALRAM_SUMMARY	*/
"MMS_EVMGMT_SERVICE",	/* 64 MMS_EVMGMT_SERVICE	*/
"READ_JOURNAL",	/* 65 READ_JOURNAL	*/
"WRITE_JOURNAL",	/* 66 WRITE_JOURNAL	*/
"INITIALIZE_JOURNAL",	/* 67 INITIALIZE_JOURNAL	*/
"REPORT_JOURNAL_STATUS",	/* 68 REPORT_JOURNAL_STATUS	*/
"CREATE_JOURNAL",	/* 69 CREATE_JOURNAL	*/
"DELETE_JOURNAL",	/* 70 DELETE_JOURNAL	*/
"GET_CAPABILITY_LIST",	/* 71 GET_CAPABILITY_LIST	*/
"FILE_OPEN",	/* 72 FILE_OPEN	*/
"FILE_READ",	/* 73 FILE_READ	*/
"FILE_CLOSE",	/* 74 FILE_CLOSE	*/
"FILE_RENAME",	/* 75 FILE_RENAME	*/
"FILE_DELETE",	/* 76 FILE_DELETE	*/
"FILE_DIR",	/* 77 FILE_DIR	*/
"ADDITIONAL_SERVICE",	/* 78 ADDITIONAL_SERVICE	*/
"INFO_RPT",	/* 79 INFO_RPT	*/
"UNSOLICITED_STATUS",	/* 80 UNSOLICITED_STATUS	*/
"EVENT_NOTIFICATION",	/* 81 EVENT_NOTIFICATION	*/
"NOT_A_MMS_OPCODE",	/* 82 NOT_A_MMS_OPCODE	*/

```

    "INITIATE",          /* 83 INITIATE          */
    "CONCLUDE",         /* 84 CONCLUDE          */
    "CANCEL"            /* 85 CANCEL            */
};

/*****
/* MODIFIERS
*****/
#define MODAEC_SSI 0x00
#define MODAS_SSI 0x00

/*****
/* ENVIRONMENT & GENERAL MANAGEMENT
*****/
#if (MMS_CONCLUDE_EN & REQ_EN)
#define MMS_CONCLUDE_RSP_DFUN mms_null_pdu_dec
#define MMS_CONCLUDE_CONF_SFUN u_mp_conclude_conf
#else
#define MMS_CONCLUDE_RSP_DFUN mms_rsp_not_supp
#define MMS_CONCLUDE_CONF_SFUN u_conf_not_supp
#endif
#if (MMS_CONCLUDE_EN & RESP_EN)
#define MMS_CONCLUDE_REQ_DFUN mms_null_pdu_dec
#define MMS_CONCLUDE_IND_SFUN u_conclude_ind
#define CONCLUDE_SSI 0x10
#else
#define MMS_CONCLUDE_REQ_DFUN mms_req_not_supp
#define MMS_CONCLUDE_IND_SFUN u_ind_not_supp
#define CONCLUDE_SSI 0x00
#endif

#if (MMS_CANCEL_EN & REQ_EN)
#define MMS_CANCEL_RSP_DFUN mms_cancel_rsp
#define MMS_CANCEL_CONF_SFUN u_mp_cancel_conf
#else
#define MMS_CANCEL_RSP_DFUN mms_rsp_not_supp
#define MMS_CANCEL_CONF_SFUN u_conf_not_supp
#endif
/* Cancel responder always enabled */
#define MMS_CANCEL_REQ_DFUN mms_cancel_req
#define MMS_CANCEL_IND_SFUN u_cancel_ind
#define CANCEL_SSI 0x10

#if (MMS_INIT_EN & REQ_EN)
#define MMS_INIT_RSP_DFUN mms_init_dec
#define MMS_INIT_CONF_SFUN u_conf_not_supp
#define MMS_MV_INIT_CONF_SFUN u_mv_init_conf
#else
#define MMS_INIT_RSP_DFUN mms_rsp_not_supp
#define MMS_INIT_CONF_SFUN u_conf_not_supp
#define MMS_MV_INIT_CONF_SFUN u_conf_not_supp
#endif
#if (MMS_INIT_EN & RESP_EN)
#define MMS_INIT_REQ_DFUN mms_init_dec
#define MMS_INIT_IND_SFUN u_init_ind
#define INIT_SSI 0x10
#else
#define MMS_INIT_REQ_DFUN mms_req_not_supp
#define MMS_INIT_IND_SFUN u_ind_not_supp
#define INIT_SSI 0x00
#endif

```

```

/*****
/*****
/* initialize the REQUEST DECODE table */
/*****
VOID (*mms_req_decode_fun [86]) () = /* opcode is index into table */
{
MMS_STATUS_REQ_DFUN,      /* 00 STATUS */
MMS_GETNAMES_REQ_DFUN,    /* 01 GET_NAMLIST */
MMS_IDENT_REQ_DFUN,       /* 02 IDENTIFY */
MMS_RENAME_REQ_DFUN,      /* 03 RENAME */
MMS_READ_REQ_DFUN,        /* 04 READ */
MMS_WRITE_REQ_DFUN,       /* 05 WRITE */
MMS_GETVAR_REQ_DFUN,      /* 06 GET_VARDEF */
MMS_DEFVAR_REQ_DFUN,      /* 07 DEF_VARNAM */
MMS_DEFSCAT_REQ_DFUN,     /* 08 DEF_SCATTERED */
MMS_GETSCAT_REQ_DFUN,     /* 09 GET_SCATTERED */
MMS_DELVAR_REQ_DFUN,      /* 10 DEL_VARNAM */
MMS_DEFLIST_REQ_DFUN,     /* 11 DEF_VARLIST */
MMS_GETVLIST_REQ_DFUN,    /* 12 GET_VARLIST */
MMS_DELVLIST_REQ_DFUN,    /* 13 DEL_VARLIST */
MMS_DEFTYPE_REQ_DFUN,     /* 14 DEF_TYPENAM */
MMS_GETTYPE_REQ_DFUN,     /* 15 GET_TYPEDEF */
MMS_DELTTYPE_REQ_DFUN,    /* 16 DEL_TYPENAM */
MMS_INPUT_REQ_DFUN,       /* 17 INPUT */
MMS_OUTPUT_REQ_DFUN,      /* 18 OUTPUT */
MMS_TAKECTRL_REQ_DFUN,    /* 19 TAKE_CONTROL */
MMS_RELCTRL_REQ_DFUN,     /* 20 REL_CONTROL */
MMS_DEFINE_SEM_REQ_DFUN,  /* 21 DEF_SEMAPHORE */
MMS_DELETE_SEM_REQ_DFUN,  /* 22 DEL_SEMAPHORE */
MMS_REP_SEMSTAT_REQ_DFUN, /* 23 REP_SEM_STATUS */
MMS_REP_SEMPOOL_REQ_DFUN, /* 24 REP_SEM_POOL_STATUS */
MMS_REP_SEMENTRY_REQ_DFUN, /* 25 REP_SEM_ENTRY_STATUS */
MMS_INIT_DWN_REQ_DFUN,    /* 26 INIT_DOWNLOAD */
MMS_DWN_LOAD_REQ_DFUN,    /* 27 DOWN_LOAD */
MMS_TERM_DWN_REQ_DFUN,    /* 28 TERM_DOWNLOAD */
MMS_INIT_UPL_REQ_DFUN,    /* 29 INIT_UPLOAD */
MMS_UP_LOAD_REQ_DFUN,     /* 30 UP_LOAD */
MMS_TERM_UPL_REQ_DFUN,    /* 31 TERM_UPLOAD */
MMS_RDDWN_REQ_DFUN,       /* 32 REQ_DOWNLOAD */
MMS_RDUPL_REQ_DFUN,       /* 33 REQ_UPLOAD */
MMS_LOAD_DOM_REQ_DFUN,    /* 34 LOAD_DOMAIN */
MMS_STR_DOM_REQ_DFUN,     /* 35 STORE_DOMAIN */
MMS_DEL_DOM_REQ_DFUN,     /* 36 DELETE_DOMAIN */
MMS_GET_DOM_REQ_DFUN,     /* 37 GET_DOMAIN */
MMS_CRE_PI_REQ_DFUN,      /* 38 CREATE_PROGRAM_INVOCATION */
MMS_DEL_PI_REQ_DFUN,      /* 39 DEL_PROGRAM_INVOCATION */
MMS_START_REQ_DFUN,       /* 40 START */
MMS_STOP_REQ_DFUN,        /* 41 STOP */
MMS_RESUME_REQ_DFUN,       /* 42 RESUME */
MMS_RESET_REQ_DFUN,       /* 43 RESET */
MMS_KILL_REQ_DFUN,        /* 44 KILL */
MMS_GET_PI_REQ_DFUN,      /* 45 GET_PROGRAM_INVOCATION */
MMS_OBTAINFILE_REQ_DFUN,  /* 46 OBTAIN_FILE */
MMS_DEFEC_REQ_DFUN,       /* 47 DEFINE_EVENT_CONDITION */
MMS_DELEC_REQ_DFUN,       /* 48 DELETE_EVENT_CONDITION */
MMS_GETECA_REQ_DFUN,      /* 49 GET_EVENT_CONDITION_ATTR */
MMS_REPECS_REQ_DFUN,      /* 50 REPORT_EVENT_COND_STATUS */
MMS_ALTECH_REQ_DFUN,      /* 51 ALTER_EV_COND_MONITORING */
MMS_TRIGE_REQ_DFUN,       /* 52 TRIGGER_EVENT */
MMS_DEFEEA_REQ_DFUN,      /* 53 DEFINE_EVENT_ACTION */
MMS_DELEA_REQ_DFUN,       /* 54 DELETE_EVENT_ACTION */
MMS_GETEAA_REQ_DFUN,      /* 55 GET_EV_ACTION_ATTRIBUTES */
MMS_REPEAS_REQ_DFUN,      /* 56 REPORT_EV_ACTION_STATUS */
MMS_DEFEE_REQ_DFUN,       /* 57 DEFINE_EVENT_ENROLLMENT */
MMS_DELEE_REQ_DFUN,       /* 58 DELETE_EVENT_ENROLLMENT */
MMS_ALTEE_REQ_DFUN,       /* 59 ALTER_EVENT_ENROLLMENT */
MMS_REPEES_REQ_DFUN,      /* 60 REPORT_ENROLL_ATTRIBUTES */
MMS_GETEEA_REQ_DFUN,      /* 61 GET_ENROLLMENT_STATUS */

```

```

MMS_ACKEVNOT_REQ_DFUN, /* 62 ACK_EVENT_NOTIFICATION */
MMS_GETAS_REQ_DFUN, /* 63 GET_ALARM_SUMMARY */
MMS_GETAES_REQ_DFUN, /* 64 GET_ALARM_ENROLLMENT_SUM */
MMS_JREAD_REQ_DFUN, /* 65 READ_JOURNAL */
MMS_JWRITE_REQ_DFUN, /* 66 WRITE_JOURNAL */
MMS_JINIT_REQ_DFUN, /* 67 INITIALIZE_JOURNAL */
MMS_JSTAT_REQ_DFUN, /* 68 REPORT_JOURNAL_STATUS */
MMS_JCREATE_REQ_DFUN, /* 69 CREATE_JOURNAL */
MMS_JDELETE_REQ_DFUN, /* 70 DELETE_JOURNAL */
MMS_GETCL_REQ_DFUN, /* 71 GET_CAPABILITY_LIST */
MMS_FOPEN_REQ_DFUN, /* 72 FILE_OPEN */
MMS_FREAD_REQ_DFUN, /* 73 FILE_READ */
MMS_FCLOSE_REQ_DFUN, /* 74 FILE_CLOSE */
MMS_FRENAME_REQ_DFUN, /* 75 FILE_RENAME */
MMS_FDELETE_REQ_DFUN, /* 76 FILE_DELETE */
MMS_FDIR_REQ_DFUN, /* 77 FILE_DIR */
mms_req_not_supp, /* 78 ADDITIONAL_SERVICE */
MMS_INFO_REQ_DFUN, /* 79 INFO_RPT */
MMS_USTATUS_REQ_DFUN, /* 80 UNSOLICITED_STATUS */
MMS_EVNOT_REQ_DFUN, /* 81 EVENT_NOTIFICATION */
mms_req_not_supp, /* 82 NOT_A_MMS_OPERATION */
MMS_INIT_REQ_DFUN, /* 83 INITIATE */
MMS_CONCLUDE_REQ_DFUN, /* 84 CONCLUDE */
MMS_CANCEL_REQ_DFUN, /* 85 CANCEL */
};

/*****
/* initialize the RESPONSE DECODE table */
*****/
VOID (*mms_rsp_decode_fun [86]) () = /* opcode is index into table */
{
MMS_STATUS_RSP_DFUN, /* 00 STATUS */
MMS_GETNAMES_RSP_DFUN, /* 01 GET_NAMLIST */
MMS_IDENT_RSP_DFUN, /* 02 IDENTIFY */
MMS_RENAME_RSP_DFUN, /* 03 RENAME */
MMS_READ_RSP_DFUN, /* 04 READ */
MMS_WRITE_RSP_DFUN, /* 05 WRITE */
MMS_GETVAR_RSP_DFUN, /* 06 GET_VARDEF */
MMS_DEFVAR_RSP_DFUN, /* 07 DEF_VARNAM */
MMS_DEFSCAT_RSP_DFUN, /* 08 DEF_SCATTERED */
MMS_GETSCAT_RSP_DFUN, /* 09 GET_SCATTERED */
MMS_DELVAR_RSP_DFUN, /* 10 DEL_VARNAM */
MMS_DEFLVLIST_RSP_DFUN, /* 11 DEF_VARLIST */
MMS_GETVLIST_RSP_DFUN, /* 12 GET_VARLIST */
MMS_DELVLIST_RSP_DFUN, /* 13 DEL_VARLIST */
MMS_DEFTYPE_RSP_DFUN, /* 14 DEF_TYPENAM */
MMS_GETTYPE_RSP_DFUN, /* 15 GET_TYPEDEF */
MMS_DELTYPE_RSP_DFUN, /* 16 DEL_TYPENAM */
MMS_INPUT_RSP_DFUN, /* 17 INPUT */
MMS_OUTPUT_RSP_DFUN, /* 18 OUTPUT */
MMS_TAKECTRL_RSP_DFUN, /* 19 TAKE_CONTROL */
MMS_RELCTRL_RSP_DFUN, /* 20 REL_CONTROL */
MMS_DEFINE_SEM_RSP_DFUN, /* 21 DEF_SEMAPHORE */
MMS_DELETE_SEM_RSP_DFUN, /* 22 DEL_SEMAPHORE */
MMS_REP_SEMSTAT_RSP_DFUN, /* 23 REP_SEM_STATUS */
MMS_REP_SEMPOOL_RSP_DFUN, /* 24 REP_SEM_POOL_STATUS */
MMS_REP_SEMENTRY_RSP_DFUN, /* 25 REP_SEM_ENTRY_STATUS */
MMS_INIT_DWN_RSP_DFUN, /* 26 INIT_DOWNLOAD */
MMS_DWN_LOAD_RSP_DFUN, /* 27 DOWN_LOAD */
MMS_TERM_DWN_RSP_DFUN, /* 28 TERM_DOWNLOAD */
MMS_INIT_UPL_RSP_DFUN, /* 29 INIT_UPLOAD */
MMS_UP_LOAD_RSP_DFUN, /* 30 UP_LOAD */
MMS_TERM_UPL_RSP_DFUN, /* 31 TERM_UPLOAD */
MMS_RDDWN_RSP_DFUN, /* 32 REQ_DOWNLOAD */
MMS_RDUPL_RSP_DFUN, /* 33 REQ_UPLOAD */
MMS_LOAD_DOM_RSP_DFUN, /* 34 LOAD_DOMAIN */
MMS_STR_DOM_RSP_DFUN, /* 35 STORE_DOMAIN */
MMS_DEL_DOM_RSP_DFUN, /* 36 DELETE_DOMAIN */

```

```

MMS_GET_DOM_RSP_DFUN, /* 37 GET_DOMAIN */
MMS_CRE_PI_RSP_DFUN, /* 38 CREATE_PROGRAM_INVOCATION */
MMS_DEL_PI_RSP_DFUN, /* 39 DEL_PROGRAM_INVOCATION */
MMS_START_RSP_DFUN, /* 40 START */
MMS_STOP_RSP_DFUN, /* 41 STOP */
MMS_RESUME_RSP_DFUN, /* 42 RESUME */
MMS_RESET_RSP_DFUN, /* 43 RESET */
MMS_KILL_RSP_DFUN, /* 44 KILL */
MMS_GET_PI_RSP_DFUN, /* 45 GET_PROGRAM_INVOCATION */
MMS_OBTAINFILE_RSP_DFUN, /* 46 OBTAIN_FILE */
MMS_DEFEC_RSP_DFUN, /* 47 DEFINE EVENT CONDITION */
MMS_DELEC_RSP_DFUN, /* 48 DELETE EVENT CONDITION */
MMS_GETECA_RSP_DFUN, /* 49 GET EVENT CONDITION ATTR */
MMS_REPECS_RSP_DFUN, /* 50 REPORT EVENT COND STATUS */
MMS_ALTECH_RSP_DFUN, /* 51 ALTER EV COND MONITORING */
MMS_TRIGE_RSP_DFUN, /* 52 TRIGGER EVENT */
MMS_DEFEA_RSP_DFUN, /* 53 DEFINE EVENT ACTION */
MMS_DELEA_RSP_DFUN, /* 54 DELETE EVENT ACTION */
MMS_GETEAA_RSP_DFUN, /* 55 GET EV ACTION ATTRIBUTES */
MMS_REPEAS_RSP_DFUN, /* 56 REPORT EV ACTION STATUS */
MMS_DEFEE_RSP_DFUN, /* 57 DEFINE EVENT ENROLLMENT */
MMS_DELEE_RSP_DFUN, /* 58 DELETE EVENT ENROLLMENT */
MMS_ALTEE_RSP_DFUN, /* 59 ALTER EVENT ENROLLMENT */
MMS_REPEES_RSP_DFUN, /* 60 REPORT ENROLL ATTRIBUTES */
MMS_GETEEA_RSP_DFUN, /* 61 GET ENROLLMENT STATUS */
MMS_ACKEVNOT_RSP_DFUN, /* 62 ACK_EVENT_NOTIFICATION */
MMS_GETAS_RSP_DFUN, /* 63 GET_ALARM_SUMMARY */
MMS_GETAES_RSP_DFUN, /* 64 GET_ALARM_ENROLLMENT_SUM */
MMS_JREAD_RSP_DFUN, /* 65 READ_JOURNAL */
MMS_JWRITE_RSP_DFUN, /* 66 WRITE_JOURNAL */
MMS_JINIT_RSP_DFUN, /* 67 INITIALIZE_JOURNAL */
MMS_JSTAT_RSP_DFUN, /* 68 REPORT_JOURNAL_STATUS */
MMS_JCREATE_RSP_DFUN, /* 69 CREATE_JOURNAL */
MMS_JDELETE_RSP_DFUN, /* 70 DELETE_JOURNAL */
MMS_GETCL_RSP_DFUN, /* 71 GET_CAPABILITY_LIST */
MMS_FOPEN_RSP_DFUN, /* 72 FILE_OPEN */
MMS_FREAD_RSP_DFUN, /* 73 FILE_READ */
MMS_FCLOSE_RSP_DFUN, /* 74 FILE_CLOSE */
MMS_FRENAME_RSP_DFUN, /* 75 FILE_RENAME */
MMS_FDELETE_RSP_DFUN, /* 76 FILE_DELETE */
MMS_FDIR_RSP_DFUN, /* 77 FILE_DIR */
mms_rsp_not_supp, /* 78 ADDITIONAL_SERVICE */
mms_rsp_not_supp, /* 79 INFO_RPT */
mms_rsp_not_supp, /* 80 UNSOLICITED_STATUS */
mms_rsp_not_supp, /* 81 EVENT_NOTIFICATION */
mms_rsp_not_supp, /* 82 NOT_A_MMS_OPERATION */
MMS_INIT_RSP_DFUN, /* 83 INITIATE */
MMS_CONCLUDE_RSP_DFUN, /* 84 CONCLUDE */
MMS_CANCEL_RSP_DFUN /* 85 CANCEL */
);

```

```

/*****
/* initialize the INDICATION SERVICE table */
/*****
VOID (*mms_ind_serve_fun [86]) () = /* opcode is index into table */
{
  MMS_STATUS_IND_SFUN, /* 00 STATUS */
  MMS_GETNAMES_IND_SFUN, /* 01 GET_NAMLIST */
  MMS_IDENT_IND_SFUN, /* 02 IDENTIFY */
  MMS_RENAME_IND_SFUN, /* 03 RENAME */
  MMS_READ_IND_SFUN, /* 04 READ */
  MMS_WRITE_IND_SFUN, /* 05 WRITE */
  MMS_GETVAR_IND_SFUN, /* 06 GET_VARDEF */
  MMS_DEFVAR_IND_SFUN, /* 07 DEF_VARNAM */
  MMS_DEFSCAT_IND_SFUN, /* 08 DEF_SCATTERED */
  MMS_GETSCAT_IND_SFUN, /* 09 GET_SCATTERED */
  MMS_DELVAR_IND_SFUN, /* 10 DEL_VARNAM */
  MMS_DEFVLIST_IND_SFUN, /* 11 DEF_VARLIST */

```

MMS_GETVLIST_IND_SFUN,	/* 12 GET_VARLIST	*/
MMS_DELVLIST_IND_SFUN,	/* 13 DEL_VARLIST	*/
MMS_DEFTYPE_IND_SFUN,	/* 14 DEF_TYPENAM	*/
MMS_GETTYPE_IND_SFUN,	/* 15 GET_TYPEDEF	*/
MMS_DELTYPE_IND_SFUN,	/* 16 DEL_TYPENAM	*/
MMS_INPUT_IND_SFUN,	/* 17 INPUT	*/
MMS_OUTPUT_IND_SFUN,	/* 18 OUTPUT	*/
MMS_TAKECTRL_IND_SFUN,	/* 19 TAKE_CONTROL	*/
MMS_RELCTRL_IND_SFUN,	/* 20 REL_CONTROL	*/
MMS_DEFINE_SEM_IND_SFUN,	/* 21 DEF_SEMAPHORE	*/
MMS_DELETE_SEM_IND_SFUN,	/* 22 DEL_SEMAPHORE	*/
MMS_REP_SEMSTAT_IND_SFUN,	/* 23 REP_SEM_STATUS	*/
MMS_REP_SEMPOOL_IND_SFUN,	/* 24 REP_SEM_POOL_STATUS	*/
MMS_REP_SEMENTRY_IND_SFUN,	/* 25 REP_SEM_ENTRY_STATUS	*/
MMS_INIT_DWN_IND_SFUN,	/* 26 INIT_DOWNLOAD	*/
MMS_DWN_LOAD_IND_SFUN,	/* 27 DWN_LOAD	*/
MMS_TERM_DWN_IND_SFUN,	/* 28 TERM_DOWNLOAD	*/
MMS_INIT_UPL_IND_SFUN,	/* 29 INIT_UPLOAD	*/
MMS_UP_LOAD_IND_SFUN,	/* 30 UP_LOAD	*/
MMS_TERM_UPL_IND_SFUN,	/* 31 TERM_UPLOAD	*/
MMS_RDDWN_IND_SFUN,	/* 32 REQ_DOWNLOAD	*/
MMS_RDUPL_IND_SFUN,	/* 33 REQ_UPLOAD	*/
MMS_LOAD_DOM_IND_SFUN,	/* 34 LOAD_DOMAIN	*/
MMS_STR_DOM_IND_SFUN,	/* 35 STORE_DOMAIN	*/
MMS_DEL_DOM_IND_SFUN,	/* 36 DELETE_DOMAIN	*/
MMS_GET_DOM_IND_SFUN,	/* 37 GET_DOMAIN	*/
MMS_CRE_PI_IND_SFUN,	/* 38 CREATE_PROGRAM_INVOCATION	*/
MMS_DEL_PI_IND_SFUN,	/* 39 DEL_PROGRAM_INVOCATION	*/
MMS_START_IND_SFUN,	/* 40 START	*/
MMS_STOP_IND_SFUN,	/* 41 STOP	*/
MMS_RESUME_IND_SFUN,	/* 42 RESUME	*/
MMS_RESET_IND_SFUN,	/* 43 RESET	*/
MMS_KILL_IND_SFUN,	/* 44 KILL	*/
MMS_GET_PI_IND_SFUN,	/* 45 GET_PROGRAM_INVOCATION	*/
MMS_OBTAINFILE_IND_SFUN,	/* 46 OBTAIN_FILE	*/
MMS_DEFEC_IND_SFUN,	/* 47 DEFINE_EVENT_CONDITION	*/
MMS_DELEC_IND_SFUN,	/* 48 DELETE_EVENT_CONDITION	*/
MMS_GETECA_IND_SFUN,	/* 49 GET_EVENT_CONDITION_ATTR	*/
MMS_REPECS_IND_SFUN,	/* 50 REPORT_EVENT_COND_STATUS	*/
MMS_ALTECM_IND_SFUN,	/* 51 ALTER_EV_COND_MONITORING	*/
MMS_TRIGE_IND_SFUN,	/* 52 TRIGGER_EVENT	*/
MMS_DEFEA_IND_SFUN,	/* 53 DEFINE_EVENT_ACTION	*/
MMS_DELEA_IND_SFUN,	/* 54 DELETE_EVENT_ACTION	*/
MMS_GETEAA_IND_SFUN,	/* 55 GET_EV_ACTION_ATTRIBUTES	*/
MMS_REPEAS_IND_SFUN,	/* 56 REPORT_EV_ACTION_STATUS	*/
MMS_DEFEE_IND_SFUN,	/* 57 DEFINE_EVENT_ENROLLMENT	*/
MMS_DELEE_IND_SFUN,	/* 58 DELETE_EVENT_ENROLLMENT	*/
MMS_ALTEE_IND_SFUN,	/* 59 ALTER_EVENT_ENROLLMENT	*/
MMS_REPEES_IND_SFUN,	/* 60 REPORT_ENROLL_ATTRIBUTES	*/
MMS_GETEEA_IND_SFUN,	/* 61 GET_ENROLLMENT_STATUS	*/
MMS_ACKEVNOT_IND_SFUN,	/* 62 ACK_EVENT_NOTIFICATION	*/
MMS_GETAS_IND_SFUN,	/* 63 GET_ALARM_SUMMARY	*/
MMS_GETAES_IND_SFUN,	/* 64 GET_ALARM_ENROLLMENT_SUM	*/
MMS_JREAD_IND_SFUN,	/* 65 READ_JOURNAL	*/
MMS_JWRITE_IND_SFUN,	/* 66 WRITE_JOURNAL	*/
MMS_JINIT_IND_SFUN,	/* 67 INITIALIZE_JOURNAL	*/
MMS_JSTAT_IND_SFUN,	/* 68 REPORT_JOURNAL_STATUS	*/
MMS_JCREATE_IND_SFUN,	/* 69 CREATE_JOURNAL	*/
MMS_JDELETE_IND_SFUN,	/* 70 DELETE_JOURNAL	*/
MMS_GETCL_IND_SFUN,	/* 71 GET_CAPABILITY_LIST	*/
MMS_FOPEN_IND_SFUN,	/* 72 FILE_OPEN	*/
MMS_FREAD_IND_SFUN,	/* 73 FILE_READ	*/
MMS_FCLOSE_IND_SFUN,	/* 74 FILE_CLOSE	*/
MMS_FRENAME_IND_SFUN,	/* 75 FILE_RENAME	*/
MMS_FDELETE_IND_SFUN,	/* 76 FILE_DELETE	*/
MMS_FDIR_IND_SFUN,	/* 77 FILE_DIR	*/
u_ind_not_supp,	/* 78 ADDITIONAL_SERVICE	*/
MMS_INFO_IND_SFUN,	/* 79 INFO_RPT	*/

```

MMS_USTATUS_IND_SFUN, /* 80 UNSOLICITED STATUS */
MMS_EVNOT_IND_SFUN, /* 81 EVENT NOTIFICATION */
u_ind_not_supp, /* 82 NOT A MMS OPERATION */
MMS_INIT_IND_SFUN, /* 83 INITIATE */
MMS_CONCLUDE_IND_SFUN, /* 84 CONCLUDE */
MMS_CANCEL_IND_SFUN /* 85 CANCEL */
);

```

```

/*****
/* initialize the CONFIRMATION SERVICE table */
/*****
VOID (*mms_conf_serve_fun [106]) () = /* opcode is index into table */
{
MMS_STATUS_CONF_SFUN, /* 00 STATUS */
MMS_GETNAMES_CONF_SFUN, /* 01 GET_NAMLIST */
MMS_IDENT_CONF_SFUN, /* 02 IDENTIFY */
MMS_RENAME_CONF_SFUN, /* 03 RENAME */
MMS_READ_CONF_SFUN, /* 04 READ */
MMS_WRITE_CONF_SFUN, /* 05 WRITE */
MMS_GETVAR_CONF_SFUN, /* 06 GET_VARDEF */
MMS_DEFVAR_CONF_SFUN, /* 07 DEF_VARNAME */
MMS_DEFSCAT_CONF_SFUN, /* 08 DEF_SCATTERED */
MMS_GETSCAT_CONF_SFUN, /* 09 GET_SCATTERED */
MMS_DELVAR_CONF_SFUN, /* 10 DEL_VARNAME */
MMS_DEFLIST_CONF_SFUN, /* 11 DEF_VARLIST */
MMS_GETVLIST_CONF_SFUN, /* 12 GET_VARLIST */
MMS_DELVLIST_CONF_SFUN, /* 13 DEL_VARLIST */
MMS_DEFTYPE_CONF_SFUN, /* 14 DEF_TYPENAME */
MMS_GETTYPE_CONF_SFUN, /* 15 GET_TYPEDEF */
MMS_DELTYPE_CONF_SFUN, /* 16 DEL_TYPENAME */
MMS_INPUT_CONF_SFUN, /* 17 INPUT */
MMS_OUTPUT_CONF_SFUN, /* 18 OUTPUT */
MMS_TAKECTRL_CONF_SFUN, /* 19 TAKE_CONTROL */
MMS_RELCTRL_CONF_SFUN, /* 20 REL_CONTROL */
MMS_DEFINE_SEM_CONF_SFUN, /* 21 DEF_SEMAPHORE */
MMS_DELETE_SEM_CONF_SFUN, /* 22 DEL_SEMAPHORE */
MMS_REP_SEMSTAT_CONF_SFUN, /* 23 REP_SEM_STATUS */
MMS_REP_SEMPOOL_CONF_SFUN, /* 24 REP_SEM_POOL_STATUS */
MMS_REP_SEMENTRY_CONF_SFUN, /* 25 REP_SEM_ENTRY_STATUS */
MMS_INIT_DWN_CONF_SFUN, /* 26 INIT_DOWNLOAD */
MMS_DWN_LOAD_CONF_SFUN, /* 27 DOWN_LOAD */
MMS_TERM_DWN_CONF_SFUN, /* 28 TERM_DOWNLOAD */
MMS_INIT_UPL_CONF_SFUN, /* 29 INIT_UPLOAD */
MMS_UP_LOAD_CONF_SFUN, /* 30 UP_LOAD */
MMS_TERM_UPL_CONF_SFUN, /* 31 TERM_UPLOAD */
MMS_RDOWN_CONF_SFUN, /* 32 REQ_DOWNLOAD */
MMS_RDUPL_CONF_SFUN, /* 33 REQ_UPLOAD */
MMS_LOAD_DOM_CONF_SFUN, /* 34 LOAD_DOMAIN */
MMS_STR_DOM_CONF_SFUN, /* 35 STORE_DOMAIN */
MMS_DEL_DOM_CONF_SFUN, /* 36 DELETE_DOMAIN */
MMS_GET_DOM_CONF_SFUN, /* 37 GET_DOMAIN */
MMS_CRE_PI_CONF_SFUN, /* 38 CREATE_PROGRAM_INVOCATION */
MMS_DEL_PI_CONF_SFUN, /* 39 DEL_PROGRAM_INVOCATION */
MMS_START_CONF_SFUN, /* 40 START */
MMS_STOP_CONF_SFUN, /* 41 STOP */
MMS_RESUME_CONF_SFUN, /* 42 RESUME */
MMS_RESET_CONF_SFUN, /* 43 RESET */
MMS_KILL_CONF_SFUN, /* 44 KILL */
MMS_GET_PI_CONF_SFUN, /* 45 GET_PROGRAM_INVOCATION */
MMS_OBTAINFILE_CONF_SFUN, /* 46 OBTAIN_FILE */
MMS_DEFEC_CONF_SFUN, /* 47 DEFINE_EVENT_CONDITION */
MMS_DELEC_CONF_SFUN, /* 48 DELETE_EVENT_CONDITION */
MMS_GETECA_CONF_SFUN, /* 49 GET_EVENT_CONDITION_ATTR */
MMS_REPECS_CONF_SFUN, /* 50 REPORT_EVENT_COND_STATUS */
MMS_ALTECH_CONF_SFUN, /* 51 ALTER_EV_COND_MONITORING */
MMS_TRIGE_CONF_SFUN, /* 52 TRIGGER_EVENT */
MMS_DEFEA_CONF_SFUN, /* 53 DEFINE_EVENT_ACTION */
MMS_DELEA_CONF_SFUN, /* 54 DELETE_EVENT_ACTION */

```

```

MMS_GETEAA_CONF_SFUN, /* 55 GET EV ACTION ATTRIBUTES */
MMS_REPEAS_CONF_SFUN, /* 56 REPORT EV ACTION STATUS */
MMS_DEFEE_CONF_SFUN, /* 57 DEFINE EVENT ENROLLMENT */
MMS_DELEE_CONF_SFUN, /* 58 DELETE EVENT ENROLLMENT */
MMS_ALTEE_CONF_SFUN, /* 59 ALTER EVENT ENROLLMENT */
MMS_REPEES_CONF_SFUN, /* 60 REPORT ENROLL ATTRIBUTES */
MMS_GETEEA_CONF_SFUN, /* 61 GET ENROLLMENT STATUS */
MMS_ACKEVNOT_CONF_SFUN, /* 62 ACK_EVENT_NOTIFICATION */
MMS_GETAS_CONF_SFUN, /* 63 GET_ALARM_SUMMARY */
MMS_GETAES_CONF_SFUN, /* 64 GET_ALARM_ENROLLMENT_SUM */
MMS_JREAD_CONF_SFUN, /* 65 READ_JOURNAL */
MMS_JWRITE_CONF_SFUN, /* 66 WRITE_JOURNAL */
MMS_JINIT_CONF_SFUN, /* 67 INITIALIZE_JOURNAL */
MMS_JSTAT_CONF_SFUN, /* 68 REPORT_JOURNAL_STATUS */
MMS_JCREATE_CONF_SFUN, /* 69 CREATE JOURNAL */
MMS_JDELETE_CONF_SFUN, /* 70 DELETE JOURNAL */
MMS_GETCL_CONF_SFUN, /* 71 GET CAPABILITY LIST */
MMS_FOPEN_CONF_SFUN, /* 72 FILE_OPEN */
MMS_FREAD_CONF_SFUN, /* 73 FILE_READ */
MMS_FCLOSE_CONF_SFUN, /* 74 FILE_CLOSE */
MMS_FRENAME_CONF_SFUN, /* 75 FILE_RENAME */
MMS_FDELETE_CONF_SFUN, /* 76 FILE_DELETE */
MMS_FDIR_CONF_SFUN, /* 77 FILE_DIR */
u_conf_not_supp, /* 78 ADDITIONAL SERVICE */
u_conf_not_supp, /* 79 INFO_RPT */
u_conf_not_supp, /* 80 UNSOLICITED_STATUS */
u_conf_not_supp, /* 81 EVENT NOTIFICATION */
u_conf_not_supp, /* 82 NOT A MMS OPERATION */
MMS_INIT_CONF_SFUN, /* 83 INITIATE */
MMS_CONCLUDE_CONF_SFUN, /* 84 CONCLUDE */
MMS_CANCEL_CONF_SFUN, /* 85 CANCEL */
u_conf_not_supp, /* 86 NOT A MMS OPERATION */
u_conf_not_supp, /* 87 NOT A MMS OPERATION */
u_conf_not_supp, /* 88 NOT A MMS OPERATION */
u_conf_not_supp, /* 89 NOT A MMS OPERATION */
MMS_MV_READ_CONF_SFUN, /* 90 NAMED READ */
MMS_MV_WRITE_CONF_SFUN, /* 91 NAMED WRITE */
u_conf_not_supp, /* 92 NOT A MMS OPERATION */
MMS_MV_FOPEN_CONF_SFUN, /* 93 REMOTE FILE OPEN */
MMS_MV_FREAD_CONF_SFUN, /* 94 REMOTE FILE READ */
MMS_MV_FCLOSE_CONF_SFUN, /* 95 REMOTE FILE CLOSE */
MMS_MV_INIT_CONF_SFUN, /* 96 INITIATE */
MMS_MV_DEFTYPE_CONF_SFUN, /* 97 TYPE DEFINITION */
MMS_MV_FCOPY_CONF_SFUN, /* 98 REMOTE FILE COPY */
u_conf_not_supp, /* 99 NOT A MMS OPERATION */
u_conf_not_supp, /* 100 NOT A MMS OPERATION */
u_conf_not_supp, /* 101 NOT A MMS OPERATION */
u_conf_not_supp, /* 102 NOT A MMS OPERATION */
u_conf_not_supp, /* 103 NOT A MMS OPERATION */
u_conf_not_supp, /* 104 NOT A MMS OPERATION */
u_conf_not_supp, /* 105 NOT A MMS OPERATION */
);

```

```

/*****
/*****
/* initialize the supported parameters variable for the INITIATE */
/* service. */
/*****
#define NOTUSED_SSI 0x00

#define STR1 0x80 /* structures - supported */
#define STR2 0x40 /* arrays - supported */
#define VNAM 0x20 /* named variables - supported */
#define VALT 0x00 /* alternate access - NOT supported */
#define VADR 0x08 /* address formed variables - supported */
#define VSCA 0x00 /* scattered access - NOT supported */
#define TYP 0x00 /* third party - NOT supported */

```

```
#define PARAM STR1 | STR2 | VNAME | VALT | VADR | VSCA | TYP | NOTUSED_SSI
```

```
UBYTE m_param = PARAM; /* parameter support */
```

```

/*****
/*****
/***** initialize the supported services variable for the INITIATE service. */
/*****

```

```

/*
0x80      0x40      0x20      0x10
0x08      0x04      0x02      0x01 */
#define SERV0 STATUS_SSI  GETNAMES_SSI  IDENT_SSI  RENAME_SSI  \
  READ_SSI    WRITE_SSI  GETVAR_SSI  DEFVAR_SSI

#define SERV1 DEFSCAT_SSI  GETSCAT_SSI  DELVAR_SSI  DEFVLIST_SSI  \
  GETVLIST_SSI  DELVLIST_SSI  DEFTYPE_SSI  GETTYPE_SSI

#define SERV2 DELTYPE_SSI  INPUT_SSI  OUTPUT_SSI  TAKECTRL_SSI  \
  RELCTRL_SSI  DEFSEM_SSI  DELSEM_SSI  RSSTAT_SSI

#define SERV3 RSPOOL_SSI  RSENTRY_SSI  INITDWN_SSI  DOWNLOAD_SSI  \
  TERMDOWN_SSI  INITUPL_SSI  UPLOAD_SSI  TERMUPL_SSI

#define SERV4 RDDWN_SSI  RDUPL_SSI  LOADDOM_SSI  STRDOM_SSI  \
  DELDOM_SSI  GETDOM_SSI  CREPI_SSI  DELPI_SSI

#define SERV5 START_SSI  STOP_SSI  RESUME_SSI  RESET_SSI  \
  KILL_SSI  GETPI_SSI  OBTFILE_SSI  DEFEC_SSI

#define SERV6 DELEC_SSI  GETECA_SSI  REPECS_SSI  ALTECM_SSI  \
  TRIGE_SSI  DEFEA_SSI  DELEA_SSI  GETEAA_SSI

#define SERV7 REPEAS_SSI  DEFEE_SSI  DELEE_SSI  ALTEE_SSI  \
  REPEES_SSI  GETEEA_SSI  ACKEVNOT_SSI  GETAS_SSI

#define SERV8 GETAES_SSI  JREAD_SSI  JWRITE_SSI  JINIT_SSI  \
  JSTAT_SSI  JCREATE_SSI  JDELETE_SSI  GETCL_SSI

#define SERV9 FOPEN_SSI  FREAD_SSI  FCLOSE_SSI  FRENAME_SSI  \
  FDELETE_SSI  FDIR_SSI  USTATUS_SSI  INFO_SSI

#define SERV10 EVNOT_SSI  MODAEC_SSI  MODAS_SSI  CONCLUDE_SSI  \
  CANCEL_SSI  NOTUSED_SSI  NOTUSED_SSI  NOTUSED_SSI

```

```

/* services supported as responder */
UBYTE m_service_resp[11] = { SERV0, SERV1, SERV2, SERV3, SERV4, SERV5, SERV6,
  SERV7, SERV8, SERV9, SERV10 };

```

Server Connection Oriented Functions "SENDCONN.C"

```
/*=====*\
                        SENDCONN.C
/*=====*\
Purpose: The purpose of this module is to provide high level
         connection management functions to the calling modules.
         Error checking is included inside these functions.
/*=====*\

/*=====| Include Files |=====*\
/*=====|

#include <stdio.h>
#include "glbtypes.h"
#include "mms_user.h"

/*=====| Global Variables |=====*\
/*=====|

short DoneFlag;    /* Let polling mechanism know when network done */

/*=====| Function Declarations |=====*\
/*=====|

short Activate      ( char * );
short Deactivate    ( char * );
short RegisterName  ( short, char * );
short PostListen    ( short );
short StopListen    ( short );
short Abort         ( short );
short CheckChannel  ( short, long );

/*=====| Activate |=====*\
Purpose: This function activates the name passed to it in name. If an
         error occurs it is passed back to the function.
Inputs:  Name - A pointer to the name to activate.
Outputs: A zero on success or an MMS-EASE error code on failure.
/*=====|

short Activate( char *Name )
{
    return ( mllp_act_ar_name( Name ) );
}

/*=====| Deactivate |=====*\
Purpose: This function deactivates the name passed to it in name. If
         an error occurs it is passed back to the function.
Inputs:  Name - A pointer to the name to deactivate.
Outputs: A zero on success or an MMS-EASE error code on failure.
/*=====|

short Deactivate( char *Name )
{
    return ( mllp_deact_ar_name( Name ) );
}
```

```

/*=====| RegisterName |=====*/
Purpose: This function registers the name passed to it on the channel
         requested
-----
Inputs:  Name - A pointer to the name to register.
         Chan - The channel on which to register the name.
-----
Outputs: A zero on success or an MMS-EASE error code on failure.
/*=====*/

```

```

short RegisterName( short Chan, char *Name )
{
    if ( !CheckChannel ( Chan, M_IDLE ) )
        return 1;
    return ( mllp_reg_ar_name( Chan, Name ) );
}

```

```

/*=====| PostListen |=====*/
Purpose: This function posts a listen on the channel requested
-----
Inputs:  Chan - The channel on which to post a listen.
-----
Outputs: A zero on success or an MMS-EASE error code on failure.
/*=====*/

```

```

short PostListen( short Chan )
{
    return ( mllp_ass_listen( Chan ) );
}

```

```

/*=====| StopListen |=====*/
Purpose: This function stops listening on the channel requested
-----
Inputs:  Chan - The channel on which to stop listening.
-----
Outputs: A zero on success or an MMS-EASE error code on failure.
/*=====*/

```

```

short StopListen( short chan )
{
    return ( mllp_stop_ass_listen( chan ) );
}

```

```

/*=====| u_init_ind |=====*/
Purpose: This function is a user defined but MMS-EASE implemented
         function that is called on receipt of a INIT.indication from
         the network. It is used to send an INIT.response back.
-----
Inputs:  ReqPtr - MMS-EASE data structure.
-----
Outputs: None.
/*=====*/

```

```

VOID u_init_ind( MMSREQ_IND *ReqPtr )
{
    short RetCode;

    printf ( "\nINIT.IND received ... sending response!\n" );

    if ( ( RetCode = mv_init_resp ( ReqPtr ) ) != 0 )
        printf ( "Error Code %0X returned from mv_init_resp\n", RetCode );
    else
        printf ( "mv_init_resp sent OK\n" );
}

```

```

/*=====| u_conclude_ind |=====*/
Purpose: This function is a user defined but MMS-EASE implemented
         function that is called on receipt of a CONCLUDE.indication
         from the network. It is used to send a CONCLUDE.response back
-----
Inputs:  ReqPtr - MMS-EASE data structure.
-----
Outputs: None.
/*=====*/

```

```

VOID u_conclude_ind(req_ptr)
MMSREQ_IND *req_ptr;
{
    short RetCode;

    printf ( "\nReceived CONCLUDE.IND\n" );

    if ( ( RetCode = mp_conclude_resp ( req_ptr ) ) != 0 )
        printf ( "Error Code %0X returned from mv_conclude_resp\n", RetCode );
    else
        printf ( "mv_conclude_resp sent OK\n" );
}

```

```

/*=====| u_mp_conclude_done |=====*/
Purpose: This function is a user defined but MMS-EASE implemented
         function that is called when a channel that has just concluded
         can be used again.
-----
Inputs:  chan      - Channel available again.
         ret_code  - Return code from process cleanup
-----
Outputs: None.
/*=====*/

```

```

VOID u_mp_conclude_done(chan,ret_code)
SHORT chan;
USHORT ret_code;
{
    printf ( "\nReceived U_CONCLUDE.DONE\n" );

    if ( ret_code )
        printf ( "\nError code %d returned in CONCLUDE.DONE!\n", ret_code );
    else
        printf ( "\nConclude Confirmed!\n" );

    DoneFlag = 1;
}

```

```

/*=====| Abort |=====*/
Purpose: This function aborts a connection on the channel requested
-----
Inputs:  Chan - The channel on which to abort a connection.
-----
Outputs: A zero on success or an MMS-EASE error code on failure.
/*=====*/

```

```

short Abort( short Chan )
{
    if ( !CheckChannel ( Chan, M_ACTIVE ) )
        return 1;
    return ( mp_abort ( Chan, 0 ) );
}

```

```

/*=====| u_abort_ind |=====*/
Purpose: This function is called when an ABORT.indication is received
         from the network. It cleans up an aborted channel.
-----
Inputs:  chan   - The channel on which the abort occurred.
         reason  - Reason code for abort.
         au_flag - Flag to indicate user or provider abort.
-----
Outputs: None.
/*=====| u_abort_ind |=====*/

```

```

VOID u_abort_ind (chan,reason,au_flag)
SHORT chan;
SHORT reason;
SHORT au_flag;
{
    printf("\nAbort_ind received, chan = %d.",chan);
    printf("\nReason given = %d.",reason);

    if(!au_flag)
        printf("\nUser initiated ");
    else
        printf("\nProvider initiated");

    ms_clr_ind_que(chan,1);
    ms_clr_locl_fctrl(chan);
    ms_clr_rem_fctrl(chan);
}

```

```

/*=====| u_mllp_a_assoc_ind |=====*/
Purpose: This function is a user defined but MMS-EASE implemented
         function that is called when a channel receives a lower level
         provider association indication from ACSE.
-----
Inputs:  chan      - Channel association request occurred on.
         info_ptr  - Pointer to information structure.
-----
Outputs: short     - if zero allow association to proceed.
/*=====| u_mllp_a_assoc_ind |=====*/

```

```

SHORT u_mllp_a_assoc_ind (chan,info_ptr)
SHORT chan;
struct acse_assinfo *info_ptr;
{
    /*
    printf ( "\n ACSE connection established ... \n" );

    printf ( "Local Name is %s\n", info_ptr->ar_info->ar_name );
    printf ( "Remote Name is %s\n", info_ptr->ar_info->part_ar_name );

    */
    return 0;
}

```

```

/*=====| u_mllp_a_assoc_conf |=====*/
Purpose: This function is a user defined but MMS-EASE implemented
function that is called when a channel receives a lower level
provider association confirm from ACSE
-----
Inputs:  chan      - Channel association request occurred on.
         info_ptr  - Pointer to information structure.
-----
Outputs: short     - if 0 then allow association to proceed.
/*=====*/

```

```

SHORT u_mllp_a_assoc_conf(chan, info_ptr)
SHORT chan;
struct acse_assinfo *info_ptr;
{
    printf ( "\nACSE association confirm received!\n" );
    return 0;
}

```

```

/*=====| u_mllp_c_assoc_ind |=====*/
Purpose: This function is a user defined but MMS-EASE implemented
function that is called when a channel receives a lower level
provider association indication from CASE.
-----
Inputs:  chan      - Channel association request occurred on.
         info_ptr  - Pointer to information structure.
         matched   - Matching title found locally
-----
Outputs: short     - if zero allow association to proceed.
/*=====*/

```

```

SHORT u_mllp_c_assoc_ind (chan, info_ptr, matched)
SHORT chan;
struct case_assinfo *info_ptr;
UBYTE matched;
{
    return 0;
}

```

```

/*=====| u_release_ind |=====*/
Purpose: This function is called when a RELEASE.indication is received
from the network.
-----
Inputs:  chan      - The channel on which the release occurred.
-----
Outputs: None.
/*=====*/

```

```

VOID u_release_ind (chan)
SHORT chan;
{
    printf ( "\nU_RELEASE_IND received!\n" );
    ms_clr_ind_que ( -1, 0 );
    PostListen (0);
}

```

```

/*=====| u_llp_error_ind |=====*/
Purpose: This function is called when a lower layer provider error
occurs.
-----
Inputs:  chan   - The channel on which the error occurred.
        ecode   - The error code.
-----
Outputs: None.
/*=====*/

```

```

VOID u_llp_error_ind(chan, ecode)
SHORT chan;
LONG ecode;
{
    printf ("\\LLP error: chan = %d, error = %d", chan, ecode);
}

```

```

/*=====| u_ind_not_supp |=====*/
Purpose: This function is called when an indication which is not
supported is received.
-----
Inputs:  None.
-----
Outputs: None.
/*=====*/

```

```

VOID u_ind_not_supp()
{
    printf("\\nu_ind_not_supp: ERROR for getting here\\n");
}

```

```

/*=====| u_conf_not_supp |=====*/
Purpose: This function is called when a confirm which is not
supported is received.
-----
Inputs:  None.
-----
Outputs: None.
/*=====*/

```

```

VOID u_conf_not_supp()
{
    printf("\\nu_conf_not_supp: ERROR for getting here\\n");
}

```

```

/*=====| u_mmsexcept_ind |=====*/
Purpose: This function is called when an internal MMS-EASE problem is
detected.
-----
Inputs:  chan   - Channel for which the problem occurred.
        ecode   - Error code.
-----
Outputs: None.
/*=====*/

```

```

VOID u_mmsexcept_ind (chan, ecode)
SHORT chan;
SHORT ecode;
{
    printf ("\\nMMS exception (board problem): chan = %d, error = %d",
        chan, ecode);
}

```

```

/*=====| CheckChannel |=====*/
Purpose: This function is used to determine if the passed channel is
the indicated state.
-----
Inputs:  Chan - Channel whose state is being checked.
        State - State to check for.
-----
Outputs: One if "Chan" is in "State", zero otherwise.
/*=====*/

```

```

short CheckChannel ( short Chan, long State )
{
    if ( mms_chan_info[Chan].ctxt.chan_state != State )
        return 0;
    return 1;
}

```

Client Connection Oriented Functions "RCVCONN.C"

```
/*=====*\
                                     RCVCONN.C
-----*\
Purpose: The purpose of this module is to provide high level
         connection management functions to the calling modules.
         Error checking is included inside these functions.
/*=====*\

/*=====| Include Files |=====*\
/*=====*\

#include <stdio.h>

#include "glbtypes.h"
#include "mms_user.h"

/*=====| Global Variables |=====*\
/*=====*\

short DoneFlag;    /* Let polling mechanism know when network done */

/*=====| Function Declarations |=====*\
/*=====*\

short Activate      ( char * );
short Deactivate    ( char * );
short RegisterName  ( short, char * );
short PostListen    ( short );
short StopListen    ( short );
short Initiate      ( short, char * );
short Conclude      ( short );
short Abort         ( short );
short CheckChannel  ( short, long );

/*=====| Activate |=====*\
Purpose: This function activates the name passed to it in name. If an
         error occurs it is passed back to the function.
-----*\
Inputs:  Name - A pointer to the name to activate.
-----*\
Outputs: A zero on success or an MMS-EASE error code on failure.
/*=====*\

short Activate( char *Name )
{
    return ( mllp_act_ar_name( Name ) );
}
```

```

/*=====| Deactivate |=====*/
Purpose: This function deactivates the name passed to it in name. If
an error occurs it is passed back to the function.
-----
Inputs: Name - A pointer to the name to deactivate.
-----
Outputs: A zero on success or an MMS-EASE error code on failure.
/*=====*/

```

```

short Deactivate( char *Name )
{
    return ( mllp_deact_ar_name( Name ) );
}

```

```

/*=====| RegisterName |=====*/
Purpose: This function registers the name passed to it on the channel
requested
-----
Inputs: Name - A pointer to the name to register.
Chan - The channel on which to register the name.
-----
Outputs: A zero on success or an MMS-EASE error code on failure.
/*=====*/

```

```

short RegisterName( short Chan, char *Name )
{
    if ( !CheckChannel ( Chan, M_IDLE ) )
        return 1;
    return ( mllp_reg_ar_name( Chan, Name ) );
}

```

```

/*=====| PostListen |=====*/
Purpose: This function posts a listen on the channel requested
-----
Inputs: Chan - The channel on which to post a listen.
-----
Outputs: A zero on success or an MMS-EASE error code on failure.
/*=====*/

```

```

short PostListen( short Chan )
{
    return ( mllp_ass_listen( Chan ) );
}

```

```

/*=====| StopListen |=====*/
Purpose: This function stops listening on the channel requested
-----
Inputs: Chan - The channel on which to stop listening.
-----
Outputs: A zero on success or an MMS-EASE error code on failure.
/*=====*/

```

```

short StopListen( short chan )
{
    return ( mllp_stop_ass_listen( chan ) );
}

```

```

/*=====| Initiate |=====*/
Purpose: This function initiates a connection to a remote node.
-----
Inputs:  Chan      - The channel on which to initiate.
        RemoteName - A pointer to the name of the node to connect to.
-----
Outputs: A zero on success or an MMS-EASE error code on failure.
/*=====*/

```

```

short Initiate( short Chan, char *RemoteName )
(
    if ( !CheckChannel ( Chan, M_IDLE ) )
        return 1;

    if ( mv_init ( Chan, RemoteName ) )
        return 0;

    return ( mms_op_err );
)

```

```

/*=====| u_mv_init_conf |=====*/
Purpose: This function is a user defined but MMS-EASE implemented
        function that is called on receipt of a INIT.confirm from
        the network.
-----
Inputs:  ReqPtr - MMS-EASE data structure.
-----
Outputs: None.
/*=====*/

```

```

VOID u_mv_init_conf( MMSREQ_PEND *ReqPtr )
(
    printf ( "\n!!!! Received INIT.CONFIRM\n" );

    if ( ReqPtr->resp_err == RESP_OK )
        printf ( "\nInitiation Confirmed!\n" );
    else
        printf ( "\nError code %d returned in CONFIRM!\n", ReqPtr->resp_err );

    ms_clr_mvreq ( ReqPtr );

    DoneFlag = 1;
}

```

```

/*=====| Conclude |=====*/
Purpose: This function concludes a connection on the channel requested
-----
Inputs:  Chan - The channel on which to conclude a connection.
-----
Outputs: A zero on success or an MMS-EASE error code on failure.
/*=====*/

```

```

short Conclude( short Chan )
(
    if ( !CheckChannel ( Chan, M_ACTIVE ) )
        return 1;
    if ( mp_conclude ( Chan ) )
        return 0;
    else
        return 1;
}

```

```

/*=====| u_mp_conclude_conf |=====*/
Purpose: This function is a user defined but MMS-EASE implemented
         function that is called on receipt of a CONCLUDE.confirm from
         the network.
-----
Inputs:  ReqPtr - MMS-EASE data structure.
-----
Outputs: None.
/*=====*/

```

```

VOID u_mp_conclude_conf(req_ptr)
MMSREQ_PEND *req_ptr;
{
    printf ( "\nReceived U_CONCLUDE.CONFIRM\n" );

    if ( req_ptr->resp_err == RESP_OK )
        printf ( "\nConclude Confirmed!\n" );
    else
        printf ( "\nError code %d returned in CONCLUDE.CONFIRM!\n", req_ptr->resp_err );
}

```

```

/*=====| u_mp_conclude_done |=====*/
Purpose: This function is a user defined but MMS-EASE implemented
         function that is called when a channel that has just concluded
         can be used again.
-----
Inputs:  chan      - Channel available again.
         ret_code   - Return code from process cleanup
-----
Outputs: None.
/*=====*/

```

```

VOID u_mp_conclude_done(chan,ret_code)
SHORT chan;
USHORT ret_code;
{
    printf ( "\nReceived U_CONCLUDE.DONE\n" );

    if ( ret_code )
        printf ( "\nError code %d returned in CONCLUDE.DONE!\n", ret_code );
    else
        printf ( "\nConclude Confirmed!\n" );

    DoneFlag = 1;
}

```

```

/*=====| Abort |=====*/
Purpose: This function aborts a connection on the channel requested
-----
Inputs:  Chan - The channel on which to abort a connection.
-----
Outputs: A zero on success or an MMS-EASE error code on failure.
/*=====*/

```

```

short Abort( short Chan )
{
    if ( !CheckChannel ( Chan, M_ACTIVE ) )
        return 1;
    return ( mp_abort ( Chan, 0 ) );
}

```

```

/*=====| u_abort_ind |=====*/
Purpose: This function is called when an ABORT.indication is received
         from the network. It cleans up an aborted channel.
-----
Inputs:  chan   - The channel on which the abort occurred.
         reason  - Reason code for abort.
         au_flag - Flag to indicate user or provider abort.
-----
Outputs: None.
/*=====*/

```

```

VOID u_abort_ind (chan,reason,au_flag)
SHORT chan;
SHORT reason;
SHORT au_flag;
{
    printf("\nAbort_ind received, chan = %d.",chan);
    printf("\nReason given = %d.",reason);

    if(!au_flag)
        printf("\nUser initiated ");
    else
        printf("\nProvider initiated");

    ms_clr_ind_que(chan,1);
    ms_clr_locl_fctrl(chan);
    ms_clr_rem_fctrl(chan);
}

```

```

/*=====| u_mllp_a_assoc_ind |=====*/
Purpose: This function is a user defined but MMS-EASE implemented
         function that is called when a channel receives a lower level
         provider association indication from ACSE.
-----
Inputs:  chan      - Channel association request occurred on.
         info_ptr  - Pointer to information structure.
-----
Outputs: short     - if zero allow association to proceed.
/*=====*/

```

```

SHORT u_mllp_a_assoc_ind (chan,info_ptr)
SHORT chan;
struct acse_assinfo *info_ptr;
{
    return 0;
}

```

```

/*=====| u_mllp_a_assoc_conf |=====*/
Purpose: This function is a user defined but MMS-EASE implemented
         function that is called when a channel receives a lower level
         provider association confirm from ACSE
-----
Inputs:  chan      - Channel association request occurred on.
         info_ptr  - Pointer to information structure.
-----
Outputs: short     - if 0 then allow association to proceed.
/*=====*/

```

```

SHORT u_mllp_a_assoc_conf(chan, info_ptr)
SHORT chan;
struct acse_assinfo *info_ptr;
{
    printf ( "\nACSE association confirm received!\n" );
    return 0;
}

```

```

/*=====| u_mllp_c_assoc_ind |=====*/
Purpose: This function is a user defined but MMS-EASE implemented
function that is called when a channel receives a lower level
provider association indication from CASE.
-----
Inputs:  chan      - Channel association request occurred on.
         info_ptr   - Pointer to information structure.
         matched    - Matching title found locally
-----
Outputs: short      - if zero allow association to proceed.
/*=====*/

```

```

SHORT u_mllp_c_assoc_ind (chan, info_ptr, matched)
SHORT chan;
struct case_assinfo *info_ptr;
UBYTE matched;
{
    return 0;
}

```

```

/*=====| u_release_ind |=====*/
Purpose: This function is called when a RELEASE.indication is received
from the network.
-----
Inputs:  chan      - The channel on which the release occurred.
-----
Outputs: None.
/*=====*/

```

```

VOID u_release_ind (chan)
SHORT chan;
{
    ms_clr_ind_que ( -1, 0 );
}

```

```

/*=====| u_llp_error_ind |=====*/
Purpose: This function is called when a lower layer provider error
occurs.
-----
Inputs:  chan      - The channel on which the error occurred.
         ecode     - The error code.
-----
Outputs: None.
/*=====*/

```

```

VOID u_llp_error_ind(chan, ecode)
SHORT chan;
LONG ecode;
{
    printf ("\LLP error: chan = %d, error = %d", chan, ecode);
}

```

```

/*=====| u_ind_not_supp |=====*/
Purpose: This function is called when an indication which is not
supported is received.
-----
Inputs:  None.
-----
Outputs: None.
-----
/*=====*/

```

```

VOID u_ind_not_supp()
{
    printf("\nu_ind_not_supp: ERROR for getting here\n");
}

```

```

/*=====| u_conf_not_supp |=====*/
Purpose: This function is called when a confirm which is not
supported is received.
-----
Inputs:  None.
-----
Outputs: None.
-----
/*=====*/

```

```

VOID u_conf_not_supp()
{
    printf("\nu_conf_not_supp: ERROR for getting here\n");
}

```

```

/*=====| u_mmsexcept_ind |=====*/
Purpose: This function is called when an internal MMS-EASE problem is
detected.
-----
Inputs:  chan - Channel for which the problem occurred.
         ecode - Error code.
-----
Outputs: None.
-----
/*=====*/

```

```

VOID u_mmsexcept_ind (chan, ecode)
SHORT chan;
SHORT ecode;
{
    printf ( "\nMMS exception (board problem): chan = %d, error = %d",
            chan, ecode);
}

```

```

/*=====| CheckChannel |=====*/
Purpose: This function is used to determine if the passed channel is
the indicated state.
-----
Inputs:  Chan - Channel whose state is being checked.
         State - State to check for.
-----
Outputs: One if "Chan" is in "State", zero otherwise.
-----
/*=====*/

```

```

short CheckChannel ( short Chan, long State )
{
    if ( mms_chan_info[Chan].ctxt.chan_state != State )
        return 0;
    return 1;
}

```

Data Compression Header File "PACKBITS.H"

```
/*=====*\n                                     PACKBITS.H\n/*=====*\nPurpose:  This header file supplies bit Packing and Unpacking routines\n          function definitions.\n/*=====*/
```

```
/*=====| Function Declaration |=====*\n/*=====*/
```

```
long PackLBits ( unsigned char far *, unsigned char far *, long );\nlong PackUBits ( unsigned char far *, unsigned char far *, long );\nlong UnpackBits ( unsigned char far *, unsigned char far *, long );
```

Data Compression Module "PACKBITS.C"

```
/*=====
                                PACKBITS.C
=====*/
Purpose: This file supplies bit Packing and Unpacking routines that
         take a buffer of bytes whose values do not exceed 63 and pack
         the significant bits of every four bytes into three bytes.
         Its sole purpose is to reduce the size of the buffer that must
         be sent over the map network and will work ONLY for buffers
         that have 6 significant bits ... this is NOT a general purpose
         compression.
/*=====
```

```
/*=====| PackLBits |=====*/
/*=====|
```

```
long PackLBits ( unsigned char far *DestBuf, unsigned char far *SrcBuf,
                 long SrcLen )
```

```
{
    long DestLen = 0;

    while ( SrcLen >= 4 ) {
        *DestBuf      = (unsigned char) ( *SrcBuf << 2 );
        *DestBuf |= (unsigned char) ( *(++SrcBuf) >> 4 );
        *(++DestBuf) = (unsigned char) ( *SrcBuf << 4 );
        *DestBuf |= (unsigned char) ( *(++SrcBuf) >> 2 );
        *(++DestBuf) = (unsigned char) ( *SrcBuf << 6 );
        *DestBuf |= (unsigned char) ( *(++SrcBuf) );
        DestBuf++;
        SrcBuf++;
        SrcLen -= 4;
        DestLen += 3;
    }
}
```

```
    return DestLen;
```

```
}
```

```
/*=====| PackUBits |=====*/
/*=====|
```

```
long PackUBits ( unsigned char far *DestBuf, unsigned char far *SrcBuf,
                 long SrcLen )
```

```
{
    long DestLen = 0;

    while ( SrcLen >= 4 ) {
        *DestBuf      = (unsigned char) ( *SrcBuf & 0xFC );
        *DestBuf |= (unsigned char) ( *(++SrcBuf) >> 6 );
        *(++DestBuf) = (unsigned char) ( ( *SrcBuf << 2 ) & 0xF0 );
        *DestBuf |= (unsigned char) ( *(++SrcBuf) >> 4 );
        *(++DestBuf) = (unsigned char) ( ( *SrcBuf << 4 ) & 0xC0 );
        *DestBuf |= (unsigned char) ( *(++SrcBuf) >> 2 );
        DestBuf++;
        SrcBuf++;
        SrcLen -= 4;
        DestLen += 3;
    }
}
```

```
    return DestLen;
```

```
}
```

```

/*=====| UnpackBits |=====*\
\=====|

long UnpackBits ( unsigned char far *DestBuf, unsigned char far *SrcBuf,
                  long SrcLen )
{
    long DestLen = 0;

    while ( SrcLen >= 3 ) {
        *DestBuf      = (unsigned char) ( *SrcBuf >> 2 );
        *(++DestBuf) = (unsigned char) ( (unsigned char)(*SrcBuf << 6) >> 2 );
        *DestBuf      |= (unsigned char) ( (unsigned char)(*++SrcBuf) >> 4 );
        *(++DestBuf) = (unsigned char) ( (unsigned char)(*SrcBuf << 4) >> 2 );
        *DestBuf      |= (unsigned char) ( *(++SrcBuf) >> 6 );
        *(++DestBuf) = (unsigned char) ( (unsigned char)(*SrcBuf << 2) >> 2 );
        DestBuf++;
        SrcBuf++;
        SrcLen -= 3;
        DestLen += 4;
    }

    return DestLen;
}

```

Everex VGA Assembly Routines "EV678.ASM"

```

;=====
;                               EV678.ASM
;=====
; Programmed by: James Barrett                      December 1990
;=====
; Purpose: The purpose of this module is to provide a 'C' interface
;          to the Everex EV678 Viewpoint VGA card.
;=====
; Implementation: These routines are written in Microsoft Macro
;                  Assembler version 5.1 to be called from Microsoft 'C'
;                  version 5.1. The module is assembled by typeing
;
;                  masm /Mx /Zi ev678.asm
;
;                  at the DOS command line.
;=====

```

```

.MODEL LARGE          ; large model
.CODE                 ; make functions visible to outside modules
PUBLIC _SetExtendedPage, _SetExtendedMode, _SetNormalMode

```

```

;=====| SetExtendedPage |=====
; Purpose: This function is used to set the particular 64K page that is
;          mapped in the window at address A0000.
;=====
; 'C' Usage: void SetExtendedPage ( unsigned char Page )
;=====

```

```

_SetExtendedPage  PROC

    push    bp
    mov     bp, sp
    push    si
    push    di
    push    ds
    mov     ax, cs
    mov     ds, ax

    jmp     GetFunc

Func1st  DW    ?
Func2nd  DW    ?

GetFunc:  mov     bx, 0004h
          mov     ax, 7000h
          int     10h

          mov     Func1st, di      ; Put address in variable
          mov     Func2nd, es

          mov     bx, OFFSET Func1st
          mov     dl, [bp+6]      ; Put page number in DL register
          call    DWORD PTR [bx]  ; Call paging function

          pop     ds
          pop     di
          pop     si

          pop     bp
          ret

_SetExtendedPage  ENDP

```

```

;=====| SetExtendedMode |=====;
; Purpose: This function is used to set the non-standard video modes
;           for the EV678.
;-----;
; 'C' Usage: void SetExtendedMode ( unsigned char Mode )
;=====;

```

```

_SetExtendedMode PROC

    push    bp
    mov     bp, sp
    push    si
    push    di
    push    ds

    mov     ax, 0070h    ; Everex extended mode set select
    mov     bl, [bp+6]   ; Get Mode
    int     10h          ; Call interrupt to set mode

    pop     ds
    pop     di
    pop     si

    pop     bp
    ret

_SetExtendedMode ENDP

```

```

;=====| SetNormalMode |=====;
; Purpose: This function is used to set the standard VGA modes for the
;           Everex EV678 VGA card.
;-----;
; 'C' Usage: void SetNormalMode ( unsigned char Mode )
;=====;

```

```

_SetNormalMode PROC

    push    bp
    mov     bp, sp
    push    si
    push    di
    push    ds

    mov     ah, 00h      ; Normal mode selection function
    mov     al, [bp+6]   ; Get mode number
    int     10h          ; Call interrupt for mode set

    pop     ds
    pop     di
    pop     si

    pop     bp
    ret

_SetNormalMode ENDP

```

END

Server Directory File "SUIC.DIB"

\$ Abstract Syntax Names

\$ Transfer Syntax Names

\$ Application Context Names

\$ Local Network Address

/39840f454e450000690101f4000200/

\$ Local AR Names

#

Format:

#

AP_Title thru AE_invoke_id are used in multi-processing environment

/ARname/AP_Title/AP_invoke_id/AE_qual/AE_invoke_id/p_sel/s_sel/t_sel/

#

/net1/////00000001/0001/0001/

\$ Remote AP Names

#

Format:

#

/name/AP_Title/AP_invokeid/AE_qual/AE_invokeid/p_sel/s_sel/t_sel/n_sel/

#

/net2/////00000001/0001/0001/39840f454e450000690101f4000100/

Client Directory File "SUIC.DIB"

\$ Abstract Syntax Names

\$ Transfer Syntax Names

\$ Application Context Names

\$ Local Network Address

/39840f454e450000690101f4000100/

\$ Local AR Names

#

Format:

#

AP_Title thru AE_invoke_id are used in multi-processing environment

/ARname/AP_Title/AP_invoke_id/AE_qual/AE_invoke_id/p_sel/s_sel/t_sel/

#

/net2/////00000001/0001/0001/

\$ Remote AP Names

#

Format:

#

/name/AP_Title/AP_invokeid/AE_qual/AE_invokeid/p_sel/s_sel/t_sel/n_sel/

#

/net1/////00000001/0001/0001/39840f454e450000690101f4000200/

APPENDIX H

FILE TRANSFER SOURCE CODE

H.1 MAP 2.X Server Code

Server Make File "FTS_2X.MAK"

```
OPS = /c /AL /W2 /DLINT_ARGS

fts.obj      : fts.c
cl $(OPS) /DMAP_2X fts.c ;

sendconn.obj : sendconn.c
cl $(OPS) sendconn.c ;

fts_file.obj : fts_file.c
cl $(OPS) fts_file.c ;

getval5.obj  : getval5.c
cl $(OPS) getval5.c ;

fkey.obj     : fkey.c
cl $(OPS) fkey.c ;

fts_stub.obj : fts_stub.c
cl $(OPS) fts_stub.c ;

mmsop_en.obj : mmsop_en.c fts_enab.h
cl $(OPS) /DMAP21_CASE /DFILE_SEND mmsop_en.c ;

packbits.obj : packbits.c
cl /c /AL packbits.c ;

fts_2x.exe : fts.obj fts_file.obj sendconn.obj packbits.obj \
getval5.obj fkey.obj fts_stub.obj mmsop_en.obj
link @fts_2x.lnk
```

Server Linker Response File "FTS_2X.LNK"

```
fts +
fts_file +
sendconn +
fts_stub +
mmsop_en +
getval5 +
packbits +
fkey
fts_2x
fts_2x.map /LI /MAP:60000 /ST:8196 /SEGMENTS:200 /NOD /NOE /NOI
asn1d mmsd1 mmsd2 mmscased succase ltarga.lib llibce
```

Server Main File "FTS.C"

```
/*=====
                                     FTS.C
=====
Programmed by: James Barrett                      December 1990
=====
Purpose:      The purpose of this program is to File Transfer Send a
               video image from a TARGA-M8 video capture board to a
               requesting node on the network.
=====
Implementation: The interface for the data is through the standard
                 file transfer services provided under MMS. By taking over
                 the file services at the primitive level, any MMS program
                 that can receive a file can receive a video image that can be
                 manipulated by specialized programs. Additionally, programs
                 specifically written to take advantage of the interface, like
                 the companion program of FTS (FTR), may interactively display
                 video from remote locations.
=====
*/

/*=====| Defines |=====
\*=====
\

#define DEBUG_SET
#define OVER_HEAD 100

/*=====| Include Files |=====
\*=====
\

#include <stdlib.h>          /* Include standard 'C' header files */
#include <stdio.h>
#include <conio.h>

#include "glbtypes.h"       /* Include necessary MMS-EASE header files */
#include "mms_user.h"
#include "asn1defs.h"
#include "mms_ms.h"

#ifdef MAP_30                /* If under MAP 3.0 include ACSE */
#define CONCORD
#include "suicacse.h"
#endif

#ifdef MAP_2X                /* If under MAP 2.X include CASE */
#define MP_400
#include "suiccase.h"
#endif

/*=====| Function Declarations |=====
\*=====
\

VOID      msg_ready   ( BYTE * );
VOID      do_debugset ( void );
extern short Activate   ( char * );
extern short RegisterName ( short, char * );
extern short PostListen ( short );
extern short strget     ( BYTE * );
```

```

/*=====| main |=====*/
Purpose: Standard 'C' program entry. "main" starts up the MMS-EASE
system and handle user input.
/*=====*/

int main()
{
char MenuChoice; /* Stores user menu choice */
char Loc[65]; /* Local process name */
char Rem[65]; /* Remote process name */
int BlockSize; /* Data block size */
char Buffer[80]; /* Buffer for input */

printf ( "\n\nEnter Block Size: " ); /* Get data block size from user */
gets ( Buffer );
BlockSize = atoi ( Buffer );

mms_debug_sel = 0; /* Set debug parameters */
asn1_debug_sel = 0;
m_file_blk_size = BlockSize; /* Set data block size */

#ifdef MAP_30 /* If under MAP 3.0 include these lines */
m_segsize = BlockSize + OVER_HEAD; /* Set MMS vars up for block size */
s_msgsize = BlockSize + OVER_HEAD;

max_mms_chan = 1; /* Only one channel available */
s_max_chan = 1;

s_dma_en = TRUE; /* Enable DMA */
s_dma_chan = 3; /* On channel 3 */
s_int_en = TRUE; /* Enable interrupts */
s_int_no = 3; /* Use interrupt 3 */
s_boot_talk_en = TRUE; /* Ask user for reset */
#endif

#ifdef MAP_2X /* If under MAP 2.X include these lines */
m_segsize = 1020; /* Set MMS vars up for block size */
case_size = 1020;
max_mms_chan = 1; /* One channel only */
max_suic_chan = 1;

SUIC_int_no = 2; /* Use interrupt 2 */
boot_talk_en = 0; /* Reset always */
suic_hw_reset = 1;
prnten = 1; /* Print any board errors */
#endif

do_debugset (); /* Prompt user for debug levels */

strt_MMS(); /* Start up MMS-EASE */

printf ( "Enter the local Name: " ); /* Get local name from user */
gets ( Loc );

Activate( Loc ); /* Activate name */
RegisterName( 0, Loc ); /* Register name on channel 0 */
PostListen ( 0 ); /* Listen on channel 0 */

do { /* print out menu */
printf ( "\n\n\n\n\n\n\n\t\t\tTarga Image Server\n\n\n" );
printf ( "x) Exit Program\n\n" );
printf ( "Enter Number of Choice: " );

while ( !kbhit() ) /* While no keyboard input service */
comm_serve(); /* the network */

MenuChoice = getch(); /* Get keyboard input */
}

```

```

        switch ( MenuChoice ) {                /* Compare to choices */
            case 'x':
            case 'X': end_MMS();                /* Exit program */
                      exit (0);
                      break;
            default: break;                    /* Unknown ... keep going */
        }

    } while ( 1 );

}

/*=====| msg_ready |=====*/
Purpose: This function is required by MMS-EASE
-----
Inputs:  msgptr - Pointer to message.
-----
Outputs: None.
/*=====| msg_ready |=====*/

VOID msg_ready (msgptr)
BYTE *msgptr;
{
    register i;

    printf ("\33[1;1H");
    for (i = 0; i < 3; ++i)
    {
        printf ("\n");
    }
    printf ("\33[1;1H");
    printf ("%s",msgptr);
}

#ifdef DEBUG_SET

/*=====| do_debugset |=====*/
Purpose: This function sets the debug level for MMS-EASE. It was
        taken from the MMS-EASE sample program.
-----
Inputs:  None.
-----
Outputs: None.
/*=====| do_debugset |=====*/

VOID do_debugset ()
{
    BYTE q[100];
    FILE *temp_file;
    if (mms_debug_sel & MMSCASE_PRINT)
        printf ("\n\n Enable CASE INTERFACE Debug Print (now ON) ?");
    else
        printf ("\n\n Enable CASE INTERFACE Debug Print (now OFF) ?");
    if (strget (q))
    {
        if (q[0] == 'y' || q[0] == 'Y')
            mms_debug_sel |= MMSCASE_PRINT;
        else
            mms_debug_sel &= ~MMSCASE_PRINT;
    }
    if (mms_debug_sel & MMSENC_PRINT)
        printf (" Enable MMS ENCODE Debug Print (now ON) ?");
    else
        printf (" Enable MMS ENCODE Debug Print (now OFF) ?");
}

```

```

if (strget (q))
{
    if (q[0] == 'y' || q[0] == 'Y')
        mms_debug_sel |= MMSENC_PRINT;
    else
        mms_debug_sel &= ~MMSENC_PRINT;
}

if (mms_debug_sel & MMSDEC_PRINT)
    printf (" Enable MMS DECODE Debug Print (now ON) ? ");
else
    printf (" Enable MMS DECODE Debug Print (now OFF) ? ");
if (strget (q))
{
    if (q[0] == 'y' || q[0] == 'Y')
        mms_debug_sel |= MMSDEC_PRINT;
    else
        mms_debug_sel &= ~MMSDEC_PRINT;
}

if (asn1_debug_sel & ASN1ENC_PRINT)
    printf ("\n Enable ASN.1 ENCODE Debug Print (now ON) ?");
else
    printf ("\n Enable ASN.1 ENCODE Debug Print (now OFF) ? ");
if (strget (q))
{
    if (q[0] == 'y' || q[0] == 'Y')
        asn1_debug_sel |= ASN1ENC_PRINT;
    else
        asn1_debug_sel &= ~ASN1ENC_PRINT;
}

if (asn1_debug_sel & ASN1DEC_PRINT)
    printf (" Enable ASN.1 DECODE Debug Print (now ON) ?");
else
    printf (" Enable ASN.1 DECODE Debug Print (now OFF) ?");
if (strget (q))
{
    if (q[0] == 'y' || q[0] == 'Y')
        asn1_debug_sel |= ASN1DEC_PRINT;
    else
        asn1_debug_sel &= ~ASN1DEC_PRINT;
}

if (mms_debug_sel & MMSINDQUE_PRINT)
    printf ("\n Enable Indication Que Information Display (now ON) ?");
else
    printf ("\n Enable Indication Que Information Display (now OFF) ?");
if (strget (q))
{
    if (q[0] == 'y' || q[0] == 'Y')
        mms_debug_sel |= MMSINDQUE_PRINT;
    else
        mms_debug_sel &= ~MMSINDQUE_PRINT;
}

if (mms_debug_sel & MMSREQQUE_PRINT)
    printf (" Enable Request Que Information Display (now ON) ?");
else
    printf (" Enable Request Que Information Display (now OFF) ?");
if (strget (q))
{
    if (q[0] == 'y' || q[0] == 'Y')
        mms_debug_sel |= MMSREQQUE_PRINT;
    else
        mms_debug_sel &= ~MMSREQQUE_PRINT;
}

```

```

if (mms_debug_sel & MMSIND_PRINT)
    printf ("\n Enable MMS Indication Information (now ON) ?");
else
    printf ("\n Enable MMS Indication Information (now OFF) ?");
if (strget (q))
{
    if (q[0] == 'y' || q[0] == 'Y')
        mms_debug_sel |= MMSIND_PRINT;
    else
        mms_debug_sel &= ~MMSIND_PRINT;
}

if (mms_debug_sel & MMSCONF_PRINT)
    printf (" Enable MMS Confirm Information (now ON) ?");
else
    printf (" Enable MMS Confirm Information (now OFF) ?");
if (strget (q))
{
    if (q[0] == 'y' || q[0] == 'Y')
        mms_debug_sel |= MMSCONF_PRINT;
    else
        mms_debug_sel &= ~MMSCONF_PRINT;
}

}

#endif

```

Server File Function Routines "FTS_FILE.C"

```
/*=====*\
                                     FTS_FILE.C
/*=====*\

Purpose:      The purpose of this module is to take over the primitive
               file interface in order to send TARGA images in the guise of
               a file transfer.
/*=====*\

/*=====| Include Files |=====*\
/*=====*\

#include <stdio.h>          /* Standard 'C' include files */
#include <conio.h>

#include "glbtypes.h"      /* MMS-EASE include files    */
#include "mms_user.h"

#include "tardev.h"        /* Targa include files      */
#include "packbits.h"      /* Compression include files */

/*=====| Typedefs |=====*\
/*=====*\

typedef struct _IMAGE_DATA { /* Header for image */
short Version,
      XSize,
      YSize,
      Compressed;
long  Reserved1,
      Reserved2;
} IMAGE_DATA;

/*=====| Function Declarations |=====*\
/*=====*\

long FillBuffer ( short, short, short );
void SendHeader ( MMSREQ_IND * );
void SendData   ( MMSREQ_IND * );

/*=====| Global Variables |=====*\
/*=====*\

unsigned char far *BuffPtr;      /* Send buffer pointer */
unsigned char  Buff1[65536];    /* Compression Buffer    */
unsigned char  Buff2[65536];    /* Compression Buffer    */
long          Bytes;           /* Current bytes left   */
short        Page;            /* Current video page   */
IMAGE_DATA   image;           /* Image header         */
void         (*ReadIndFunc)(MMSREQ_IND *); /* Data function pointer */
```

```

/*=====| u_fopen_ind |=====*/
Purpose: This is an MMS-EASE defined, user implemented function that
         is called when MMS receives an fopen.IND from a remote file
         open request. This tells the program that a client on the
         network wants to download an image. The TARGA board and
         variables for encoding the state are initialized. An
         fopen.RESP is sent back and the first 64K video segment is
         aquired.
-----
Inputs:  req_ptr - A pointer to an MMS-EASE information structure
-----
Outputs: None.
/*=====*/

```

```

VOID u_fopen_ind(req_ptr)
MMSREQ_IND *req_ptr;
{
    char                Answer[30];
    short               RetCode,
                      XSize,
                      YSize;
    struct fopen_resp_info info;

    GraphInit ( TYPE_M8 );          /* Initialize Targa board */
    SetPageMode ( 1 );              /* Set to 512 X 512 mode */
    GetScrSize ( &XSize, &YSize ); /* Get actual page size */

    printf ( "\nu_fopen_ind received on channel %0X!\n", req_ptr->chan );
                                   /* Get compression type */
    printf ( "\nCompression Type? [0=None, 1=Bitpack]: " );
    gets ( Answer );
    if ( Answer[0] == '0' )
        image.Compressed = 0;
    else if ( Answer[0] == '1' )
        image.Compressed = 1;

    image.Version      = 1;          /* Fill in header */
    image.XSize        = ++XSize;
    image.YSize        = ++YSize;
    image.Reserved1     = 0;
    image.Reserved2     = 0;

    printf ( "\nFile Block Size is %d\n",
             mms_chan_info[req_ptr->chan].file_blk_size );

    printf ( "\nImage Size is %d X %d\n", XSize, YSize );

    info.frsmid = 1;                /* Fill in response header */
    info.ent.fsize = (long)XSize * (long)YSize + sizeof(IMAGE_DATA);
    info.ent.mtimpres = FALSE;

    ReadIndFunc = SendHeader;        /* Send header first */

    Page = 3;
    Bytes = FillBuffer ( YSize-1, 128-(512-YSize), image.Compressed );

    SetLiveMode();                  /* Grab image */
    printf ( "\nHit any key ... \n" );
    getch();
    GrabFrame();
    SetDispMode ();

                                   /* Send file open response back */
    if ( (RetCode = mp_fopen_resp ( req_ptr, &info )) )
        printf ( "\nmp_fopen_resp returned Error Code %0X\n", RetCode );
    else
        printf ( "\nfopen_resp sent ok ..." );
}

```

```

/*=====| u_fclose_ind |=====*/
Purpose: This is an MMS-EASE defined, user implemented function that
is called when MMS receives an fclose.IND from a remote file
close request. This tells the program that a client on the
network has finished loading an image. An fclose.RESP is
sent back.
-----
Inputs: req_ptr - A pointer to an MMS-EASE information structure
-----
Outputs: None.
/*=====| u_fclose_ind |=====*/

```

```

VOID u_fclose_ind ( MMSREQ_IND *req_ptr )
{
short RetCode;

```

```

    /* Send file close response */

```

```

    printf ( "\nu_fclose_ind received on channel %0X!\n", req_ptr->chan );

```

```

    if ( (RetCode = mp_fclose_resp ( req_ptr ) ) != 0 )
        printf ( "\nError %0X received while sending mp_fclose_resp.\n",
                RetCode );

```

```

    else
        printf ( "\nSending mp_fclose_resp.\n" );

```

```

}

```

```

/*=====| u_fread_ind |=====*/
Purpose: This is an MMS-EASE defined, user implemented function that
is called when MMS receives an fread.IND from a remote file
read request. In this funtion the client is requesting the
next block of data from the image. A function pointer is used
to control the state as an information header is sent before
the image data.
-----
Inputs: req_ptr - A pointer to an MMS-EASE information structure
-----
Outputs: None.
/*=====| u_fread_ind |=====*/

```

```

VOID u_fread_ind( MMSREQ_IND *req_ptr )
{

```

```

    /* Route indication to appropriate function */

```

```

    printf ( "\nBytes is %ld", Bytes );
    printf ( "\nPage is %d", Page );

```

```

    (*ReadIndFunc)( req_ptr );

```

```

}

```

```

/*=====| SendHeader |=====*/
Purpose: This function is called by the ReadIndFunc pointer to send
         the image information header. After the header is sent,
         the ReadIndFunc pointer is set to the SendData function to
         actually send the image data.
-----
Inputs:  req_ptr - A pointer to an MMS-EASE information structure
-----
Outputs: None.
/*=====*/

```

```

void SendHeader ( MMSREQ_IND *req_ptr )
{
    short          RetCode;
    struct fread_resp_info respinfo;

    /* Fill in response structure */

    respinfo.filedata    = (void *) &image;
    respinfo.fd_len      = sizeof(IMAGE_DATA);
    respinfo.more_follows = TRUE;

    /* Send image header to client */

    if ( (RetCode = mp_fread_resp ( req_ptr, &respinfo ) ) != 0 ) {
        printf ( "\nError %0X received while sending mp_fread_resp.\n",
                RetCode );
    } else {
        printf ( "\nHeader mp_fread_resp sent ok!\n" );
    }

    ReadIndFunc = SendData;    /* New indication function is SendData */
}

```

```

/*=====| SendData |=====*/
Purpose: This function is called by the ReadIndFunc pointer to send
the actual image data.
-----
Inputs:  req_ptr - A pointer to an MMS-EASE information structure
-----
Outputs: None.
/*=====*/

```

```

void SendData ( MMSREQ_IND *req_ptr )
{
    short          RetCode;
    struct fread_resp_info respinfo;

    /* If end of video page ... */
    if ( Bytes < mms_chan_info[req_ptr->chan].file_blk_size ) {
        respinfo.fd_len = Bytes; /* Get leftover bytes */
        respinfo.filedata = BuffPtr;
        if ( --Page < 0 ) { /* If no more video pages ... */
            respinfo.more_follows = FALSE; /* Then last block */
            /* Send block to client */
            if ( (RetCode = mp_fread_resp ( req_ptr, &respinfo ) ) != 0 ) {
                printf ( "\nError %0X received while sending mp_fread_resp.\n",
                    RetCode );
            } else {
                printf ( "\nPARTIAL NEG mp_fread_resp sent ok!\n" );
            }
        } else { /* Still more video pages ... */
            respinfo.more_follows = TRUE; /* Still more blocks */
            /* Send block to client */
            if ( (RetCode = mp_fread_resp ( req_ptr, &respinfo ) ) != 0 ) {
                printf ( "\nError %0X received while sending mp_fread_resp.\n",
                    RetCode );
            } else {
                printf ( "\nPARTIAL POS mp_fread_resp sent ok!\n" );
            }
        }
        /* Refill buffer from next page */
        Bytes = FillBuffer ( (Page+1)*128-1, 128, image.Compressed );
    } else { /* Still more full blocks on this video page ... */
        respinfo.fd_len = mms_chan_info[req_ptr->chan].file_blk_size;
        respinfo.filedata = BuffPtr;
        respinfo.more_follows = TRUE;
        /* Send block to client */
        if ( (RetCode = mp_fread_resp ( req_ptr, &respinfo ) ) != 0 ) {
            printf ( "\nError %0X received while sending mp_fread_resp.\n",
                RetCode );
        } else {
            printf ( "\nFULL mp_fread_resp sent ok!\n" );
        }
        /* Update buffer pointer */
        BuffPtr += mms_chan_info[req_ptr->chan].file_blk_size;
        Bytes -= mms_chan_info[req_ptr->chan].file_blk_size;
    }
}

```

```

/*=====| FillBuffer |=====*/
Purpose: This function is used to fill a buffer with a page (64K) of
image data at a time in the correct order.
-----
Inputs:  StartLine - the topmost scan line to start with.
         NumLines  - the number of scan lines to get.
         CompType  - the compression method for the data, if any.
-----
Outputs: The actual number of bytes in the buffer after processing.
/*=====*/

long FillBuffer ( short StartLine, short NumLines, short CompType )
{
    register short x;

    BuffPtr = Buff1;

    for ( x=0; x<NumLines; x++, BuffPtr+=512 )    /* Get image lines in */
        GetRow ( BuffPtr, 0, 511, StartLine-x, -1 ); /* buffer. */

    switch ( CompType ) {
        case 0: BuffPtr = Buff1;                    /* No compression */
                return ( 512L*(long)NumLines );
                break;
        case 1: BuffPtr = Buff2;                    /* Pack bits */
                return ( PackUBits ( Buff2, Buff1, 512L*(long)NumLines ) );
                break;
        default: return ( 0 );
                break;
    }
    return ( 0 );
}

```

Server Stub File "FTS_STUB.C"

```
/*=====*\
                        FTS_STUB.H
/*=====*\
Purpose:      The purpose of this module is to provide stubs for the
              MMS-EASE functions that must exist but do not do anything.
/*=====*\

/*=====| Include Files |=====*\
/*=====*\

#include "GLBTYPES.H"

/*****
/*      Variable Operation Stubs      */
/*****/

BYTE u_get_named_addr(){}

/*****
/*      Error Recovery Stubs      */
/*****/

VOID u_cancel_ind(){}
VOID u_reject_ind(){}
VOID u_mp_cancel_conf(){}

```

Server MMS-EASE Function Selection Header "FTS_ENAB.H"

```

/*=====
                                     FTS_ENAB.H
=====
Purpose:      The purpose of this module is to enable a subset of the
               MMS-EASE system. It is a version of the normal mmsop_en.h
               file used with all MMS-EASE programs.
=====
*/

#ifndef MMSOP_EN_INCLUDED
#define MMSOP_EN_INCLUDED

/*****
/*****
/* define the operation enable switches */
/*****
#define REQ_RESP_DIS 0x00 /* no support for req or resp */
#define REQ_EN       0x01 /* support for request */
#define RESP_EN      0x02 /* support for response */
#define REQ_RESP_EN  0x03 /* support for resp and req */

/*****
/* define the opcode enable switches */
/*****
#define MMS_INIT_EN          RESP_EN
#define MMS_CONCLUDE_EN     RESP_EN
#define MMS_CANCEL_EN       REQ_RESP_EN

#define MMS_STATUS_EN        REQ_RESP_DIS
#define MMS_USTATUS_EN      REQ_RESP_DIS
#define MMS_GETNAMES_EN     REQ_RESP_DIS
#define MMS_IDENT_EN        REQ_RESP_DIS
#define MMS_RENAME_EN       REQ_RESP_DIS
#define MMS_GETCL_EN        REQ_RESP_DIS
#define MMS_VMD_EN          1

#define MMS_MV_DWN_EN        REQ_RESP_DIS /* VM Upload */
#define MMS_INIT_DWN_EN     REQ_RESP_DIS
#define MMS_DWN_LOAD_EN     REQ_RESP_DIS
#define MMS_TERM_DWN_EN     REQ_RESP_DIS
#define MMS_MV_UPL_EN       REQ_RESP_DIS /* VM Download */
#define MMS_INIT_UPL_EN     REQ_RESP_DIS
#define MMS_UP_LOAD_EN      REQ_RESP_DIS
#define MMS_TERM_UPL_EN     REQ_RESP_DIS
#define MMS_RDDWN_EN        REQ_RESP_DIS
#define MMS_RDUPL_EN        REQ_RESP_DIS
#define MMS_LOAD_DOM_EN     REQ_RESP_DIS
#define MMS_STR_DOM_EN      REQ_RESP_DIS
#define MMS_DEL_DOM_EN      REQ_RESP_DIS
#define MMS_GET_DOM_EN      REQ_RESP_DIS
#define MMS_NAMED_DOM_EN    0

#define MMS_CRE_PI_EN        REQ_RESP_DIS
#define MMS_DEL_PI_EN        REQ_RESP_DIS
#define MMS_START_EN         REQ_RESP_DIS
#define MMS_STOP_EN          REQ_RESP_DIS
#define MMS_RESUME_EN        REQ_RESP_DIS
#define MMS_RESET_EN         REQ_RESP_DIS
#define MMS_KILL_EN          REQ_RESP_DIS
#define MMS_GET_PI_EN        REQ_RESP_DIS

#define MMS_VA_EN            1
#define MMS_MV_READ_EN       REQ_RESP_DIS /* VM Read variable(s) */
#define MMS_MV_RDVARS_EN     REQ_RESP_DIS /* General VM Rd var's */
#define MMS_READ_EN          REQ_RESP_DIS
#define MMS_MV_WRITE_EN      REQ_RESP_DIS /* VM Write variable(s) */

```

```

#define MMS_MV_WRVARS_EN      REQ_RESP_DIS /* General VM WR var's */
#define MMS_WRITE_EN          REQ_RESP_DIS
#define MMS_INFO_EN           REQ_RESP_DIS
#define MMS_GETVAR_EN         REQ_RESP_DIS
#define MMS_DETVAR_EN         REQ_RESP_DIS
#define MMS_DEFSCAT_EN        REQ_RESP_DIS
#define MMS_GETSCAT_EN        REQ_RESP_DIS
#define MMS_DELVAR_EN         REQ_RESP_DIS
#define MMS_DETVLIST_EN       REQ_RESP_DIS
#define MMS_GETVLIST_EN       REQ_RESP_DIS
#define MMS_DELVLIST_EN       REQ_RESP_DIS
#define MMS_MV_DEFTYPE_EN     REQ_RESP_DIS /* VM DefineType */
#define MMS_DEFTYPE_EN        REQ_RESP_DIS
#define MMS_GETTYPE_EN        REQ_RESP_DIS
#define MMS_DELTYPE_EN        REQ_RESP_DIS

#define MMS_TAKECTRL_EN        REQ_RESP_DIS
#define MMS_RELCTRL_EN         REQ_RESP_DIS
#define MMS_DEFINE_SEM_EN      REQ_RESP_DIS
#define MMS_DELETE_SEM_EN      REQ_RESP_DIS
#define MMS_REP_SEMSTAT_EN     REQ_RESP_DIS
#define MMS_REP_SEMPOOL_EN     REQ_RESP_DIS
#define MMS_REP_SEMENTRY_EN    REQ_RESP_DIS

#define MMS_JREAD_EN           REQ_RESP_DIS
#define MMS_JWRITE_EN          REQ_RESP_DIS
#define MMS_JINIT_EN           REQ_RESP_DIS
#define MMS_JSTAT_EN           REQ_RESP_DIS
#define MMS_JCREATE_EN         REQ_RESP_DIS
#define MMS_JDELETE_EN         REQ_RESP_DIS

#define MMS_DEFEC_EN           REQ_RESP_DIS
#define MMS_DELEC_EN           REQ_RESP_DIS
#define MMS_GETECA_EN          REQ_RESP_DIS
#define MMS_REPECS_EN          REQ_RESP_DIS
#define MMS_ALTECM_EN          REQ_RESP_DIS
#define MMS_TRIGE_EN           REQ_RESP_DIS
#define MMS_DEFEA_EN           REQ_RESP_DIS
#define MMS_DELEA_EN           REQ_RESP_DIS
#define MMS_GETEAA_EN          REQ_RESP_DIS
#define MMS_REPEAS_EN          REQ_RESP_DIS
#define MMS_DEFEE_EN           REQ_RESP_DIS
#define MMS_DELEE_EN           REQ_RESP_DIS
#define MMS_GETEEA_EN          REQ_RESP_DIS
#define MMS_REPEES_EN          REQ_RESP_DIS
#define MMS_ALTEE_EN           REQ_RESP_DIS
#define MMS_EVNOT_EN           REQ_RESP_DIS
#define MMS_ACKEVNOT_EN        REQ_RESP_DIS
#define MMS_GETAS_EN           REQ_RESP_DIS
#define MMS_GETAES_EN          REQ_RESP_DIS

#define MMS_INPUT_EN           REQ_RESP_DIS
#define MMS_OUTPUT_EN          REQ_RESP_DIS

#define MMS_MV_FCOPY_EN        REQ_RESP_DIS
#define MMS_OBTAINFILE_EN      REQ_RESP_DIS
#define MMS_MV_FOPEN_EN        REQ_RESP_DIS
#define MMS_FOPEN_EN           REQ_RESP_EN
#define MMS_MV_FREAD_EN        REQ_RESP_DIS
#define MMS_FREAD_EN           REQ_RESP_EN
#define MMS_MV_FCLOSE_EN       REQ_RESP_DIS
#define MMS_FCLOSE_EN          REQ_RESP_EN
#define MMS_FRENAME_EN         REQ_RESP_DIS
#define MMS_FDELETE_EN         REQ_RESP_DIS
#define MMS_FDIR_EN            REQ_RESP_DIS

#endif

```

H.2 MAP 2.X Client Code

Client Make File "FTR_2X.MAK"

```
OPS = /c /AL /W2 /DLINT_ARGS

ftr.obj      : ftr.c
cl $(OPS) /DMAP_2X ftr.c ;

ftr_file.obj : ftr_file.c
cl $(OPS) ftr_file.c ;

recvconn.obj : recvconn.c
cl $(OPS) recvconn.c ;

getval5.obj  : getval5.c
cl $(OPS) getval5.c ;

fkey.obj     : fkey.c
cl $(OPS) fkey.c ;

ftr_stub.obj : ftr_stub.c
cl $(OPS) ftr_stub.c ;

mmsop_en.obj : mmsop_en.c ftr_enab.h
cl $(OPS) /DMAP21_CASE /DFILE_RECV mmsop_en.c ;

packbits.obj : packbits.c
cl $(OPS) packbits.c ;

ev678.obj    : ev678.asm
tasm /Zd /Zi /Mx ev678

ftr_2x.exe : ftr.obj ftr_file.obj recvconn.obj packbits.obj getval5.obj \
             fkey.obj ftr_stub.obj mmsop_en.obj ev678.obj
link @ftr_2x.lnk
```

Client Linker Response File "FTR_2X.LNK"

```
ftr +
ftr_file +
recvconn +
ftr_stub +
mmsop_en +
getval5 +
fkey +
packbits +
ev678
ftr_2x
ftr_2x.map /LI /MAP:60000 /CO /ST:8196 /SEGMENTS:521 /NOD /NOE /NOI
asn1d mmsd1 mmsd2 mmscased succase llibce
```

Client Main File "FTR.C"

```
/*=====*\
                                     FTR.C
=====*\

Purpose: The purpose of this program is to get a video image from
a server on the network.

Implementation: The interface for the data is through the standard
file transfer services provided under MMS. By taking over
the file services at the primitive level, the program can
display the file data "on the fly" as a video image.
/*=====*\

/*=====| Defines |=====*\
/*=====*\

#define DEBUG_SET
#define OVER_HEAD 100

/*=====| Include Files |=====*\
/*=====*\

#include <stdio.h>           /* Include standard 'C' header files */
#include <stdlib.h>
#include <conio.h>

#include "glbtypes.h"        /* Include necessary MMS-EASE header files */
#include "mms_user.h"
#include "asn1defs.h"
#include "mms_ms.h"

#ifdef MAP_30                 /* If under MAP 3.0 include ACSE */
#define CONCORD
#include "suicacse.h"
#endif

#ifdef MAP_2X                 /* If under MAP 2.X include CASE */
#define MP_400
#include "suiccase.h"
#endif

/*=====| Function Declarations |=====*\
/*=====*\

VOID msg_ready ( BYTE * );
VOID GetImage ( char * );

extern void SetNormalMode ( unsigned char );
extern void SetExtendedMode ( unsigned char );
extern void SetExtendedPage ( unsigned char );
extern short Activate ( char * );
extern short RegisterName ( short, char * );
extern short Initiate ( short, char * );
extern short Conclude ( short );
extern VOID GetFile ( short, char * );

/*=====| Global Variables |=====*\
/*=====*\

extern short DoneFlag;      /* Network service done flag */
```

```

/*=====| main |=====*/
Standard 'C' program entry. "main" starts up the MMS-EASE system and
handles user input.
/*=====*/

VOID main()
(
char MenuChoice;      /* Stores user menu choice */
char Loc[65];          /* Local process name */
char Rem[65];          /* Remote process name */
int BlockSize;         /* Data block size */
char Buffer[80];        /* Buffer for input */

    printf ( "\n\nEnter Block Size: " );      /* Get data block size from user */
    gets ( Buffer );
    BlockSize = atoi ( Buffer );

    mms_debug_sel = 0;                          /* Set debug parameters */
    asnl_debug_sel = 0;
    m_file_blk_size = BlockSize;                /* Set data block size */

#ifdef MAP_30                                /* If under MAP 3.0 include these lines */
    s_msgsize = BlockSize + OVER_HEAD;          /* Set MMS vars up for block size */
    m_segsize = BlockSize + OVER_HEAD;
    max_mms_chan = 1;                          /* Only one channel available */
    s_max_chan = 1;

    s_dma_en = TRUE;                          /* Enable DMA */
    s_dma_chan = 3;                            /* On channel 3 */
    s_int_en = TRUE;                          /* Enable interrupts */
    s_int_no = 3;                             /* Use interrupt 3 */
    s_boot_talk_en = TRUE;                    /* Ask user for reset */
#endif

#ifdef MAP_2X                                /* If under MAP 2.X include these lines */
    /* Set MMS vars up for block size */
    m_segsize = 1020;
    case_size = 1020;
    max_mms_chan = 1;                          /* One channel only */
    max_suic_chan = 1;

    SUIC_int_no = 2;                          /* Use interrupt 2 */
    boot_talk_en = 0;                         /* Reset always */
    suic_hw_reset = 1;
    printen = 1;                              /* Print any board errors */
#endif

    do_debugset();                            /* Prompt user for debug levels */

    strt_MMS();                               /* Start up MMS-EASE */

    printf ( "Enter the local Name: " ); /* Get local name from user */
    gets ( Loc );

    Activate ( Loc );                          /* Activate name */
    RegisterName ( 0, Loc );                  /* Register name on channel 0 */

    do {                                       /* print out menu */
        printf ( "\n\n\n\n\n\n\n\t\t\tTarga Image Retriever\n\n\n" );
        printf ( "G) Get An Image\n" );
        printf ( "X) Exit Program\n\n" );
        printf ( "Enter Number of Choice: " );

        while ( !kbhit() )                  /* While no keyboard input service */
            comm_serve();                    /* the network */

        MenuChoice = getch();                /* Get keyboard input */
    }

```

```

switch ( MenuChoice ) {          /* Compare to choices */
    case 'g':                    /* Get image */
    case 'G': printf ( "\n\n\n\n\n\nEnter name of Image Server: " );
                gets( Rem );
                GetImage ( Rem );
                break;
    case 'x':                    /* Exit program */
    case 'X': break;
    default: break;
}

} while ( MenuChoice != 'x' && MenuChoice != 'X' );
}

/*=====| GetImage |=====*/
Purpose: This function makes a connection to an image server,
         downloads and displays an image, and disconnects from the
         server.
-----
Inputs:  Remote - Pointer to name of remote server.
-----
Outputs: None.
/*=====| GetImage |=====*/

VOID GetImage ( char *Remote )
{
    char FileName[256];

    printf ( "Enter File to Get: " );    /* Get "file name" to download */
    gets ( FileName );

    DoneFlag = 0;                        /* Init flag */
    Initiate ( 0, Remote );              /* Initiate connection */
    while ( !DoneFlag )                 /* Service network until init.conf */
        comm_serve();

    printf ( "\nGetting file ..." );

    DoneFlag = 0;                        /* Init flag */
    GetFile ( 0, FileName );             /* Download image */
    while ( !DoneFlag )                 /* Service network until download */
        comm_serve();                  /* is done. */

    DoneFlag = 0;                        /* Init flag */
    Conclude( 0 );                       /* Disconnect from server */
    while ( !DoneFlag )                 /* Service network until disconnected */
        comm_serve();

    getch();                            /* Keep image on screen until user */
    SetNormalMode ( 0x02 );              /* hits a key ... then restore text */

    return;
}

```

```

/*=====| msg_ready |=====*/
Purpose: This function is required by MMS-EASE
-----
Inputs:  msgptr - Pointer to message.
-----
Outputs: None.
/*=====|

```

```

VOID msg_ready (msgptr)
BYTE *msgptr;
{
register i;

printf ("\33[1;1H");
for (i = 0; i < 3; ++i)
{
printf ("\n");
}
printf ("\33[1;1H");
printf ("%s",msgptr);
}

```

```

#ifdef DEBUG_SET

```

```

/*=====| do_debugset |=====*/
Purpose: This function sets the debug level for MMS-EASE. It was
taken from the MMS-EASE sample program.
-----
Inputs:  None.
-----
Outputs: None.
/*=====|

```

```

VOID do_debugset ()
{
BYTE q[100];
FILE *temp_file;

if (mms_debug_sel & MMSCASE_PRINT)
printf ("\n\n\n Enable CASE INTERFACE Debug Print (now ON) ?");
else
printf ("\n\n\n Enable CASE INTERFACE Debug Print (now OFF) ?");
if (strget (q))
{
if (q[0] == 'y' || q[0] == 'Y')
mms_debug_sel |= MMSCASE_PRINT;
else
mms_debug_sel &= ~MMSCASE_PRINT;
}

if (mms_debug_sel & MMSENC_PRINT)
printf (" Enable MMS ENCODE Debug Print (now ON) ?");
else
printf (" Enable MMS ENCODE Debug Print (now OFF) ?");
if (strget (q))
{
if (q[0] == 'y' || q[0] == 'Y')
mms_debug_sel |= MMSENC_PRINT;
else
mms_debug_sel &= ~MMSENC_PRINT;
}
}

```

```

if (mms_debug_sel & MMSDEC_PRINT)
    printf (" Enable MMS DECODE Debug Print (now ON) ? ");
else
    printf (" Enable MMS DECODE Debug Print (now OFF) ? ");
if (strget (q))
{
    if (q[0] == 'y' || q[0] == 'Y')
        mms_debug_sel |= MMSDEC_PRINT;
    else
        mms_debug_sel &= ~MMSDEC_PRINT;
}

if (asn1_debug_sel & ASN1ENC_PRINT)
    printf ("\n Enable ASN.1 ENCODE Debug Print (now ON) ?");
else
    printf ("\n Enable ASN.1 ENCODE Debug Print (now OFF) ? ");
if (strget (q))
{
    if (q[0] == 'y' || q[0] == 'Y')
        asn1_debug_sel |= ASN1ENC_PRINT;
    else
        asn1_debug_sel &= ~ASN1ENC_PRINT;
}

if (asn1_debug_sel & ASN1DEC_PRINT)
    printf (" Enable ASN.1 DECODE Debug Print (now ON) ?");
else
    printf (" Enable ASN.1 DECODE Debug Print (now OFF) ?");
if (strget (q))
{
    if (q[0] == 'y' || q[0] == 'Y')
        asn1_debug_sel |= ASN1DEC_PRINT;
    else
        asn1_debug_sel &= ~ASN1DEC_PRINT;
}

if (mms_debug_sel & MMSINDQUE_PRINT)
    printf ("\n Enable Indication Que Information Display (now ON) ?");
else
    printf ("\n Enable Indication Que Information Display (now OFF) ?");
if (strget (q))
{
    if (q[0] == 'y' || q[0] == 'Y')
        mms_debug_sel |= MMSINDQUE_PRINT;
    else
        mms_debug_sel &= ~MMSINDQUE_PRINT;
}

if (mms_debug_sel & MMSREQQUE_PRINT)
    printf (" Enable Request Que Information Display (now ON) ?");
else
    printf (" Enable Request Que Information Display (now OFF) ?");
if (strget (q))
{
    if (q[0] == 'y' || q[0] == 'Y')
        mms_debug_sel |= MMSREQQUE_PRINT;
    else
        mms_debug_sel &= ~MMSREQQUE_PRINT;
}

if (mms_debug_sel & MMSIND_PRINT)
    printf ("\n Enable MMS Indication Information (now ON) ?");
else
    printf ("\n Enable MMS Indication Information (now OFF) ?");

```

```

if (strget (q))
{
    if (q[0] == 'y' || q[0] == 'Y')
        mms_debug_sel |= MMSIND_PRINT;
    else
        mms_debug_sel &= ~MMSIND_PRINT;
}

if (mms_debug_sel & MMSCONF_PRINT)
    printf (" Enable MMS Confirm Information (now ON) ?");
else
    printf (" Enable MMS Confirm Information (now OFF) ?");
if (strget (q))
{
    if (q[0] == 'y' || q[0] == 'Y')
        mms_debug_sel |= MMSCONF_PRINT;
    else
        mms_debug_sel &= ~MMSCONF_PRINT;
}
}

#endif

```

Client File Routines "FTR_FILE.C"

```
/*=====*\
                                     FTR_FILE.C
/*=====*\

Purpose: The purpose of this module is to take over the primitive
         file interface in order to receive TARGA images in the guise
         of a file transfer.

/*=====*\

/*=====| Include Files |=====*\
/*=====|

#include <stdio.h>          /* Standard 'C' include files */
#include <conio.h>
#include <stdlib.h>
#include <string.h>
#include <malloc.h>

#include "glbtypes.h"      /* MMS-EASE include files */
#include "mms_user.h"

#include "packbits.h"      /* Compression include file */

/*=====| Defines |=====*\
/*=====|

#define VID_BUFFER 0xA0000000 /* Video buffer address 0xA0000 */

/*=====| Function Declarations |=====*\
/*=====|

VOID GetFile      ( short, char * );
void LoadDacs256 ( void );
void LoadDacs64  ( void );
void ReadHeader   ( MMSREQ_PEND * );
void ReadData     ( MMSREQ_PEND * );
void ProcessBuffer ( unsigned char far *, short, short );

extern void SetNormalMode ( unsigned char );
extern void SetExtendedMode ( unsigned char );
extern void SetExtendedPage ( unsigned char );

/*=====| Typedefs |=====*\
/*=====|

typedef struct _IMAGE_DATA { /* Header for image */
short Version,
      XSize,
      YSize,
      Compressed;
long  Reserved1,
      Reserved2;
} IMAGE_DATA;
```

```

/*=====| Global Variables |=====*\
\*=====*\

long          BytesRead;          /* Total bytes read */
unsigned char far *VidBuf;        /* Video buffer ptr */
unsigned char Buff1[65536];       /* Decompression buffer */
short         Page;               /* Current video page */
void          (*ReadFunc) ( MMSREQ_PEND * ); /* Decoding pointer */
IMAGE_DATA    image;              /* image header struct */
struct fread_req_info fr_info;     /* File read struct */

extern short   DoneFlag;           /* Polling flag */

```

```

/*=====| GetFile |=====*\
| Purpose: This function reads a file from a remote image server and |
| displays the file as an image.                                     |
|-----|
| Inputs:  cham      - Channel to use.                               |
|          FileName - Pointer to name of file to download.          |
|-----|
| Outputs: None.                                                    |
\*=====*\

```

```

VOID GetFile ( short chan, char *FileName )
{
    short          RetCode;
    struct fopen_req_info *info;
    struct file_name *fname;

    printf ( "\nEntering GetFile ..." );

    /* Fill in open file request structure */
    info = (struct fopen_req_info *) calloc ( sizeof(struct fopen_req_info) +
        sizeof(struct file_name), 1 );
    printf ( "\nAllocated info ..." );

    fname = (struct file_name *) ( ( (char *)info ) +
        sizeof(struct fopen_req_info));

    printf ( "\nPointed fname ..." );

    info->num_of_fname = 1;
    info->init_pos      = 0;
    fname->fn_len = strlen ( FileName );
    fname->fname = FileName;

    printf ( "\nAbout to mp_fopen ..." );

    if ( mp_fopen ( chan, info ) != NULL ) /* Request file open */
        printf ( "\nmp_fopen sent ok...\n" );
    else
        printf ( "\nmp_fopen bombed with Error = %X\n", mms_op_err );

    free ( info );
}

```

```

/*=====| u_mp_fopen_conf |=====*/
Purpose: This is an MMS-EASE defined, user implemented function that
         is called when MMS receives an fopen.RESP from the server.
         This tells the program that the server on the network has
         agreed to download an image. The read function is set for
         reading the header as that will be the first block received.
-----
Inputs:  req_ptr - A pointer to an MMS-EASE information structure
-----
Outputs: None.
/*=====*/

VOID u_mp_fopen_conf( req_ptr )
MMSREQ_PEND *req_ptr;
{
    if ( req_ptr->resp_err == RESP_OK ) {
        printf ( "\nmp_fopen Confirmed!\n" );
    } else {
        printf ( "\nError code %d returned in u_mp_fopen_conf!\n", req_ptr->resp_err );
        return;
    }

    /* Get file read state machine ID */
    fr_info.frsmid=((struct fopen_resp_info*)(req_ptr->resp_info_ptr))->frsmid;

    if ( mp_fread ( req_ptr->chan, &fr_info ) ) /* Request header block */
        printf ( "\nFirst mp_fread sent ok ...\n" );
    else
        printf ( "\nmp_fread died with Error = %X\n", mms_op_err );

    VidBuf = (char far *) VID_BUFFER; /* Initialize video pointer */

    ReadFunc = ReadHeader; /* Set up read function pointer */
}

/*=====| u_mp_fread_conf |=====*/
Purpose: This is an MMS-EASE defined, user implemented function that
         is called when MMS receives an fread.CONF from a remote file
         read request. In this function the server has returned the
         next block of data from the image. A function pointer is used
         to control the state as an information header is sent before
         the image data.
-----
Inputs:  req_ptr - A pointer to an MMS-EASE information structure
-----
Outputs: None.
/*=====*/

VOID u_mp_fread_conf( req_ptr )
MMSREQ_PEND *req_ptr;
{
    if ( req_ptr->resp_err ) {
        SetNormalMode ( 0x02 );
        printf ( "\nError %0X recieved in u_mp_fread_conf\n", req_ptr->resp_err );
        printf ( "Hit any key ..." );
        getch();
    } else
        (*ReadFunc)( req_ptr );
}

```

```

/*=====| ReadHeader |=====*/
Purpose: This function is called by the ReadFunc pointer to acquire
         the image information header. After the header is received,
         the ReadFunc pointer is set to the ReadData function to
         actually receive the image data.
-----
Inputs:  req_ptr - A pointer to an MMS-EASE information structure
-----
Outputs: None.
/*=====*/

void ReadHeader ( MMSREQ_PEND *req_ptr )
{
    struct fread_resp_info *resp_info;

    /* Get pointer to data */
    resp_info = (struct fread_resp_info *)req_ptr->resp_info_ptr;

    /* Copy header into header structure */
    memcpy ( &image, resp_info->filedata, sizeof(IMAGE_DATA) );

    printf ( "\nImage header: X = %d, Y = %d, C = %d", image.XSize, image.YSize,
            image.Compressed );
    getch();

    /* Request next block of data */
    if ( !mp_fread ( req_ptr->chan, &fr_info ) ) {
        SetNormalMode ( 0x02 );
        printf ( "\nmp_fread died with Error = %X\n", mms_op_err );
        printf ( "\nHit any Key ... \n" );
        getch();
    }

    SetExtendedMode ( 0x15 ); /* Go into graphics mode */

    switch ( image.Compressed ) { /* Set up palette depending on compression */
        case 0: LoadDacs256 ();
            break;
        case 1: LoadDacs64 ();
            break;
    }

    SetExtendedPage ( 0 ); /* Set first video page */

    BytesRead = (512L - (long)image.YSize) * 512L;
    Page      = 0;
    memset ( Buff1, 0x00, (short) BytesRead );

    ReadFunc = ReadData; /* Now read data */
}

```

```

/*=====| ReadData |=====*/
Purpose: This function is called by the ReadFunc pointer to receive
the actual image data.
-----
Inputs:  req_ptr - A pointer to an MMS-EASE information structure
-----
Outputs: None.
/*=====|

void ReadData ( MMSREQ_PEND *req_ptr )
{
    struct fread_resp_info *resp_info;

    /* Get pointer to data */
    resp_info = (struct fread_resp_info *)req_ptr->resp_info_ptr;

    /* If more data to be read request next block */
    if ( resp_info->more_follows == TRUE ) {
        if ( lmp_fread ( req_ptr->chan, &fr_info ) ) {
            SetNormalMode ( 0x02 );
            printf ( "\nmp_fread died with Error = %X\n", mms_op_err );
            printf ( "\nHit any Key ...\n" );
            getch();
        }
    } else { /* otherwise the transfer is done */
        DoneFlag = 1;
    }

    /* Process the image data */
    ProcessBuffer ( resp_info->filedata, resp_info->fd_len, image.Compressed );
}

```

```

/*=====| ProcessBuffer |=====*/
Purpose: This function processes the received data blocks according to
the type of compression used.
-----
Inputs:  data      - A pointer to the image data
        len        - Length of the data buffer
        CompType   - Compression method used on data
-----
Outputs: None.
/*=====|

void ProcessBuffer ( unsigned char far *data, short len, short CompType )
{
    switch ( CompType ) {
        case 0: memcpy ( Buff1+BytesRead, data, len ); /* Not compressed */
                BytesRead += len;
                break;
        case 1: BytesRead += UnpackBits ( Buff1+BytesRead, data, (long)len );
                break;                                /* Decompress data */
    }

    if ( BytesRead > 65534L ) { /* When we have a full video page then */
        memcpy ( VidBuf, Buff1, (unsigned short)65535 ); /* display it */
        VidBuf[65535] = Buff1[65535];
        BytesRead = 0L;
        SetExtendedPage ( (unsigned char)(++Page) ); /* Go to next page */
    }
}

```

```

/*=====| LoadDacs256 |=====*/
Purpose: This function is used to set up the grey scale palette for the
VGA for the case of no compression.
-----
Inputs:  None.
-----
Outputs: None.
-----
/*=====|

```

```

void LoadDacs256 ( void )
{
    int      Loop;
    unsigned char Color;

    outp ( 0x3C8, 0x00 );

    for ( Color=0; Color<64; Color++ ) {
        for ( Loop=0; Loop<4; Loop++ ) {
            outp ( 0x3C9, Color );
            outp ( 0x3C9, Color );
            outp ( 0x3C9, Color );
        }
    }
}

```

```

/*=====| LoadDacs64 |=====*/
Purpose: This function is used to set up the grey scale palette for the
VGA for the case of compression.
-----
Inputs:  None.
-----
Outputs: None.
-----
/*=====|

```

```

void LoadDacs64 ( void )
{
    unsigned char Color;
    int Loop;

    outp ( 0x3C8, 0x00 );

    for ( Loop=0; Loop<4; Loop++ )
        for ( Color=0; Color<64; Color++ ) {
            outp ( 0x3C9, Color );
            outp ( 0x3C9, Color );
            outp ( 0x3C9, Color );
        }
}

```

Client Stub File "FTR_STUB.C"

```
/*=====*\
                                     FTR_STUB.H
-----*/
Purpose:   The purpose of this module is to provide stubs for the
           MMS-EASE functions that must exist but do not do anything.
/*=====*\

/*=====| Include Files |=====*\
/*=====*\

#include "GLBTYPES.H"

/*****/
/*      File Operation Stubs      */
/*****/

VOID u_mv_fcopy_conf(){}
VOID u_mv_fopen_conf(){}
VOID u_mv_fread_conf(){}
VOID u_mp_fclose_conf(){}
VOID u_mv_fclose_conf(){}

/*****/
/*      Variable Operation Stubs  */
/*****/

BYTE u_get_named_addr(){}

/*****/
/*      Error Recovery Stubs      */
/*****/

VOID u_cancel_ind(){}
VOID u_reject_ind(){}
VOID u_mp_cancel_conf(){}

```

Client MMS-EASE Function Selection Header "FTR_ENAB.H"

```

/*=====
                                     FTR_ENAB.H
=====*/

Purpose:    The purpose of this module is to enable a subset of the
            MMS-EASE system. It is a version of the normal mmsop_en.h
            file used with all MMS-EASE prgrams.

/*=====

#ifndef MMSOP_EN_INCLUDED
#define MMSOP_EN_INCLUDED

/*****
/*****
/* define the operation enable switches */
/*****
#define REQ_RESP_DIS 0x00 /* no support for req or resp */
#define REQ_EN      0x01 /* support for request */
#define RESP_EN     0x02 /* support for response */
#define REQ_RESP_EN 0x03 /* support for resp and req */

/*****
/* define the opcode enable switches */
/*****
#define MMS_INIT_EN      REQ_EN
#define MMS_CONCLUDE_EN  REQ_EN
#define MMS_CANCEL_EN    REQ_RESP_EN

#define MMS_STATUS_EN    REQ_RESP_DIS
#define MMS_USTATUS_EN   REQ_RESP_DIS
#define MMS_GETNAMES_EN  REQ_RESP_DIS
#define MMS_IDENT_EN     REQ_RESP_DIS
#define MMS_RENAME_EN    REQ_RESP_DIS
#define MMS_GETCL_EN     REQ_RESP_DIS
#define MMS_VMD_EN       1

#define MMS_MV_DWN_EN     REQ_RESP_DIS /* VM Upload */
#define MMS_INIT_DWN_EN   REQ_RESP_DIS
#define MMS_DWN_LOAD_EN   REQ_RESP_DIS
#define MMS_TERM_DWN_EN   REQ_RESP_DIS
#define MMS_MV_UPL_EN     REQ_RESP_DIS /* VM Download */
#define MMS_INIT_UPL_EN   REQ_RESP_DIS
#define MMS_UP_LOAD_EN    REQ_RESP_DIS
#define MMS_TERM_UPL_EN   REQ_RESP_DIS
#define MMS_RDDWN_EN      REQ_RESP_DIS
#define MMS_RDUPL_EN      REQ_RESP_DIS
#define MMS_LOAD_DOM_EN   REQ_RESP_DIS
#define MMS_STR_DOM_EN    REQ_RESP_DIS
#define MMS_DEL_DOM_EN    REQ_RESP_DIS
#define MMS_GET_DOM_EN    REQ_RESP_DIS
#define MMS_NAMED_DOM_EN  0

#define MMS_CRE_PI_EN     REQ_RESP_DIS
#define MMS_DEL_PI_EN     REQ_RESP_DIS
#define MMS_START_EN      REQ_RESP_DIS
#define MMS_STOP_EN       REQ_RESP_DIS
#define MMS_RESUME_EN     REQ_RESP_DIS
#define MMS_RESET_EN      REQ_RESP_DIS
#define MMS_KILL_EN       REQ_RESP_DIS
#define MMS_GET_PI_EN     REQ_RESP_DIS

#define MMS_VA_EN         1
#define MMS_MV_READ_EN    REQ_RESP_DIS /* VM Read variable(s) */
#define MMS_MV_RDVARS_EN  REQ_RESP_DIS /* General VM Rd var's */
#define MMS_READ_EN       REQ_RESP_DIS

```

```

#define MMS_MV_WRITE_EN      REQ_RESP_DIS /* VM Write variable(s) */
#define MMS_MV_WRVARS_EN     REQ_RESP_DIS /* General VM WR var's */
#define MMS_WRITE_EN         REQ_RESP_DIS
#define MMS_INFO_EN          REQ_RESP_DIS
#define MMS_GETVAR_EN        REQ_RESP_DIS
#define MMS_DEFVAR_EN        REQ_RESP_DIS
#define MMS_DEFSCAT_EN       REQ_RESP_DIS
#define MMS_GETSCAT_EN       REQ_RESP_DIS
#define MMS_DELVAR_EN        REQ_RESP_DIS
#define MMS_DEFLVLIST_EN     REQ_RESP_DIS
#define MMS_GETVLIST_EN      REQ_RESP_DIS
#define MMS_DELVLIST_EN      REQ_RESP_DIS
#define MMS_MV_DEFTYPE_EN     REQ_RESP_DIS /* VM DefineType */
#define MMS_DEFTYPE_EN       REQ_RESP_DIS
#define MMS_GETTYPE_EN       REQ_RESP_DIS
#define MMS_DELTYPE_EN       REQ_RESP_DIS

#define MMS_TAKECTRL_EN      REQ_RESP_DIS
#define MMS_RELCTRL_EN      REQ_RESP_DIS
#define MMS_DEFINE_SEM_EN    REQ_RESP_DIS
#define MMS_DELETE_SEM_EN    REQ_RESP_DIS
#define MMS_REP_SEMSTAT_EN   REQ_RESP_DIS
#define MMS_REP_SEMPOOL_EN   REQ_RESP_DIS
#define MMS_REP_SEMENTRY_EN  REQ_RESP_DIS

#define MMS_JREAD_EN         REQ_RESP_DIS
#define MMS_JWRITE_EN        REQ_RESP_DIS
#define MMS_JINIT_EN         REQ_RESP_DIS
#define MMS_JSTAT_EN         REQ_RESP_DIS
#define MMS_JCREATE_EN       REQ_RESP_DIS
#define MMS_JDELETE_EN       REQ_RESP_DIS

#define MMS_DEFEC_EN         REQ_RESP_DIS
#define MMS_DELEC_EN         REQ_RESP_DIS
#define MMS_GETECA_EN        REQ_RESP_DIS
#define MMS_REPECS_EN        REQ_RESP_DIS
#define MMS_ALTECM_EN        REQ_RESP_DIS
#define MMS_TRIGE_EN         REQ_RESP_DIS
#define MMS_DEFEA_EN         REQ_RESP_DIS
#define MMS_DELEA_EN         REQ_RESP_DIS
#define MMS_GETEAA_EN        REQ_RESP_DIS
#define MMS_REPEAS_EN        REQ_RESP_DIS
#define MMS_DEFEE_EN         REQ_RESP_DIS
#define MMS_DELEE_EN         REQ_RESP_DIS
#define MMS_GETEEA_EN        REQ_RESP_DIS
#define MMS_REPEES_EN        REQ_RESP_DIS
#define MMS_ALTEE_EN         REQ_RESP_DIS
#define MMS_EVNOT_EN         REQ_RESP_DIS
#define MMS_ACKEVNOT_EN      REQ_RESP_DIS
#define MMS_GETAS_EN         REQ_RESP_DIS
#define MMS_GETAES_EN        REQ_RESP_DIS

#define MMS_INPUT_EN         REQ_RESP_DIS
#define MMS_OUTPUT_EN        REQ_RESP_DIS

#define MMS_MV_FCOPY_EN      REQ_RESP_DIS
#define MMS_OBTAINFILE_EN    REQ_RESP_DIS
#define MMS_MV_FOPEN_EN      REQ_RESP_DIS
#define MMS_FOPEN_EN         REQ_EN
#define MMS_MV_FREAD_EN      REQ_RESP_DIS
#define MMS_FREAD_EN         REQ_EN
#define MMS_MV_FCLOSE_EN     REQ_RESP_DIS
#define MMS_FCLOSE_EN        REQ_EN
#define MMS_FRENAME_EN       REQ_RESP_DIS
#define MMS_FDELETE_EN       REQ_RESP_DIS
#define MMS_FDIR_EN          REQ_RESP_DIS

#endif

```

H.3 MAP 3.0 Server Code

The only difference between the 2.X and 3.0 code is the make files and linker response files, which are given below.

Server Make File "FTS_30.MAK"

```
OPS = /c /AL /W2 /DLINT_ARGS

fts.obj : fts.c
    cl $(OPS) /DMAP_30 fts.c ;

sendconn.obj : sendconn.c
    cl $(OPS) sendconn.c ;

fts_file.obj : fts_file.c
    cl $(OPS) fts_file.c ;

getval5.obj : getval5.c
    cl $(OPS) getval5.c ;

fkey.obj : fkey.c
    cl $(OPS) fkey.c ;

fts_stub.obj : fts_stub.c
    cl $(OPS) fts_stub.c ;

mmsop_en.obj : mmsop_en.c fts_enab.h
    cl $(OPS) /DMAP30_ACSE /DFILE_SEND mmsop_en.c ;

packbits.obj : packbits.c
    cl /c /AL packbits.c ;

fts_30.exe : fts.obj fts_file.obj sendconn.obj packbits.obj \
    getval5.obj fkey.obj fts_stub.obj mmsop_en.obj
    link @fts_30.lnk
```

Server Linker Response File "FTS_30.LNK"

```
fts +
fts_file +
sendconn +
fts_stub +
mmsop_en +
getval5 +
packbits +
fkey
fts_30
fts_30.map /LI /MAP:60000 /ST:8196 /SEGMENTS:200 /NOD /NOE /NOI
asn1d mmsd1 mmsd2 mmsacsed suicacse ltarga.lib llibce
```

H.4 MAP 3.0 Client Code

The only difference between the 2.X and 3.0 code is the make files and linker response files, which are given below.

Client Make File "FTR_30.MAK"

```
OPS = /c /AL /W2 /DLINT_ARGS

ftr.obj      : ftr.c
cl $(OPS) /DMAP_30 ftr.c ;

ftr_file.obj : ftr_file.c
cl $(OPS) ftr_file.c ;

recvconn.obj : recvconn.c
cl $(OPS) recvconn.c ;

getval5.obj  : getval5.c
cl $(OPS) getval5.c ;

fkey.obj     : fkey.c
cl $(OPS) fkey.c ;

ftr_stub.obj : ftr_stub.c
cl $(OPS) ftr_stub.c ;

mmsop_en.obj : mmsop_en.c ftr_enab.h
cl $(OPS) /DMAP30_ACSE /DFILE_RECV mmsop_en.c ;

packbits.obj : packbits.c
cl $(OPS) packbits.c ;

ev678.obj    : ev678.asm
tasm /Zd /Zi /Mx ev678

ftr_30.exe : ftr.obj ftr_file.obj recvconn.obj packbits.obj getval5.obj \
             fkey.obj ftr_stub.obj mmsop_en.obj ev678.obj
link @ftr_30.lnk
```

Client Linker Response File "FTR_30.LNK"

```
ftr +
ftr_file +
recvconn +
ftr_stub +
mmsop_en +
getval5 +
fkey +
packbits +
ev678
ftr_30
ftr_30.map /LI /MAP:60000 /CO /ST:8196 /SEGMENTS:521 /NOD /NOE /NOI
asn1d mmsd1 mmsd2 mmsacsd suicacse llibce
```

APPENDIX I

DOMAIN UPLOAD SOURCE CODE

I.1 MAP 2.X Server Code

Server Make File "DTS_2X.MAK"

```
OPS = /c /AL /W2 /DLINT_ARGS

dts.obj : dts.c
cl $(OPS) /DMAP_2X dts.c ;

sendconn.obj : sendconn.c
cl $(OPS) sendconn.c ;

dts_dom.obj : dts_dom.c
cl $(OPS) dts_dom.c ;

getval5.obj : getval5.c
cl $(OPS) getval5.c ;

fkey.obj : fkey.c
cl $(OPS) fkey.c ;

dts_stub.obj : dts_stub.c
cl $(OPS) dts_stub.c ;

mmsop_en.obj : mmsop_en.c dts_enab.h
cl $(OPS) /DMAP21_CASE /DDOM_SEND mmsop_en.c ;

packbits.obj : packbits.c
cl /c /AL packbits.c ;

dts_2x.exe : dts.obj dts_dom.obj sendconn.obj packbits.obj \
getval5.obj fkey.obj dts_stub.obj mmsop_en.obj
link @dts_2x.lnk
```

Server Linker Response File "FTS_2X.LNK"

```
dts +
dts_dom +
sendconn +
dts_stub +
mmsop_en +
getval5 +
packbits +
fkey
dts_2x
dts_2x.map /LI /MAP:60000 /ST:8196 /SEGMENTS:200 /NOD /NOE /NOI
asn1d mmsd1 mmsd2 mmscased suiccase ltarga.lib llibce
```

Server Main File "DTS.C"

```
/*=====*\
                                     DTS.C
/*=====*\

Programmed by: James Barrett           Date: December 1990
/*=====*\

Purpose:   The purpose of this program is to Domain Transfer Send a
           video image from a TARGA-M8 video capture board to a
           requesting node on the network.
/*=====*\

Implementation: The interface for the data is through the standard
                domain upload services provided under MMS. By taking over
                the domain services at the primitive level, another MMS m
                program that knows how to use the domain can receive a video
                image that can be displayed or stored.
/*=====*\

/*=====*\ Defines |=====*\
/*=====*\

#define DEBUG_SET
#define OVER_HEAD 100

/*=====*\ Include Files |=====*\
/*=====*\

#include <stdlib.h>           /* Include standard 'C' header files */
#include <stdio.h>
#include <conio.h>

#include "glbtypes.h"        /* Include necessary MMS-EASE header files */
#include "mms_user.h"
#include "asn1defs.h"
#include "mms_ms.h"

#ifdef MAP_30                 /* If under MAP 3.0 include ACSE */
#define CONCORD
#include "suicacse.h"
#endif

#ifdef MAP_2X                 /* If under MAP 2.X include CASE */
#define MP_400
#include "suiccase.h"
#endif

/*=====*\ Function Declarations |=====*\
/*=====*\

VOID      msg_ready   ( BYTE * );
VOID      do_debugset ( void );
extern short Activate  ( char * );
extern short RegisterName ( short, char * );
extern short PostListen ( short );
extern short strget    ( BYTE * );
```

```

/*=====| main |=====*/
Standard 'C' program entry. "main" starts up the MMS-EASE system and
handles user input.
/*=====|

int main()
{
char MenuChoice;    /* Stores user menu choice */
char Loc[65];       /* Local process name */
char Rem[65];       /* Remote process name */
int BlockSize;      /* Data block size */
char Buffer[80];     /* Buffer for input */

printf ( "\n\nEnter Block Size: " );    /* Get data block size from user */
gets ( Buffer );
BlockSize = atoi ( Buffer );

mms_debug_sel = 0;                      /* Set debug parameters */
asn1_debug_sel = 0;
m_file_blk_size = BlockSize;           /* Set data block size */

#ifdef MAP_30                          /* If under MAP 3.0 include these lines */
m_segsize = BlockSize + OVER_HEAD;    /* Set MMS vars up for block size */
s_msgsize = BlockSize + OVER_HEAD;

max_mms_chan = 1;                      /* Only one channel available */
s_max_chan = 1;

s_dma_en = TRUE;                       /* Enable DMA */
s_dma_chan = 3;                        /* On channel 3 */
s_int_en = TRUE;                       /* Enable interrupts */
s_int_no = 3;                          /* Use interrupt 3 */
s_boot_talk_en = TRUE;                 /* Ask user for reset */
#endif

#ifdef MAP_2X                          /* If under MAP 2.X include these lines */
m_segsize = 1020;                      /* Set MMS vars up for block size */
case_size = 1020;
max_mms_chan = 1;                      /* One channel only */
max_suic_chan = 1;

SUIC_int_no = 2;                       /* Use interrupt 2 */
boot_talk_en = 0;                     /* Reset always */
suic_hw_reset = 1;
prnten = 1;                           /* Print any board errors */
#endif

do_debugset ();                        /* Prompt user for debug levels */

strt_MMS();                            /* Start up MMS-EASE */

if ( lms_add_named_dom ( "TARGA-M8", 0x00 ) ) {
printf ( "\nError adding named domain! ... exiting." );
exit( 0 );
}

printf ( "Enter the local Name: " ); /* Get local name from user */
gets ( Loc );

Activate( Loc );                       /* Activate name */
RegisterName( 0, Loc );                /* Register name on channel 0 */
PostListen ( 0 );                      /* Listen on channel 0 */

do {                                  /* print out menu */
printf ( "\n\n\n\n\n\n\n\t\t\tTarga Image Server\n\n\n" );
printf ( "x) Exit Program\n\n" );
printf ( "Enter Number of Choice: " );

```

```

        while ( !kbhit() )                /* While no keyboard input service */
            comm_serve();                  /* the network */

        MenuChoice = getch();              /* Get keyboard input */

        switch ( MenuChoice ) {            /* Compare to choices */
            case 'x':
            case 'X': end_MMS();            /* Exit program */
                    exit (0);
                    break;
            default: break;                 /* Unknown ... keep going */
        }

    } while ( 1 );

}

/*=====| msg_ready |=====*/
Purpose: This function is required by MMS-EASE
-----
Inputs:  msgptr - Pointer to message.
-----
Outputs: None.
/*=====|

VOID msg_ready (msgptr)
BYTE *msgptr;
{
    register i;

    printf ("\33[1;1H");
    for (i = 0; i < 3; ++i)
    {
        printf ("\n");
    }
    printf ("\33[1;1H");
    printf ("%s",msgptr);
}

#ifdef DEBUG_SET

/*=====| do_debugset |=====*/
Purpose: This function sets the debug level for MMS-EASE. It was
        taken from the MMS-EASE sample program.
-----
Inputs:  None.
-----
Outputs: None.
/*=====|

VOID do_debugset ()
{
    BYTE q[100];
    FILE *temp_file;

    if (mms_debug_sel & MMSCASE_PRINT)
        printf ("\n\n\n Enable CASE INTERFACE Debug Print (now ON) ?");
    else
        printf ("\n\n\n Enable CASE INTERFACE Debug Print (now OFF) ?");
    if (strget (q))
    {
        if (q[0] == 'y' || q[0] == 'Y')
            mms_debug_sel |= MMSCASE_PRINT;
        else
            mms_debug_sel &= ~MMSCASE_PRINT;
    }
}

```

```

    }

    if (mms_debug_sel & MMSENC_PRINT)
        printf (" Enable MMS ENCODE Debug Print (now ON) ?");
    else
        printf (" Enable MMS ENCODE Debug Print (now OFF) ?");
    if (strget (q))
    {
        if (q[0] == 'y' || q[0] == 'Y')
            mms_debug_sel |= MMSENC_PRINT;
        else
            mms_debug_sel &= ~MMSENC_PRINT;
    }

    if (mms_debug_sel & MMSDEC_PRINT)
        printf (" Enable MMS DECODE Debug Print (now ON) ? ");
    else
        printf (" Enable MMS DECODE Debug Print (now OFF) ? ");
    if (strget (q))
    {
        if (q[0] == 'y' || q[0] == 'Y')
            mms_debug_sel |= MMSDEC_PRINT;
        else
            mms_debug_sel &= ~MMSDEC_PRINT;
    }

    if (asn1_debug_sel & ASN1ENC_PRINT)
        printf ("\n Enable ASN.1 ENCODE Debug Print (now ON) ?");
    else
        printf ("\n Enable ASN.1 ENCODE Debug Print (now OFF) ? ");
    if (strget (q))
    {
        if (q[0] == 'y' || q[0] == 'Y')
            asn1_debug_sel |= ASN1ENC_PRINT;
        else
            asn1_debug_sel &= ~ASN1ENC_PRINT;
    }

    if (asn1_debug_sel & ASN1DEC_PRINT)
        printf (" Enable ASN.1 DECODE Debug Print (now ON) ?");
    else
        printf (" Enable ASN.1 DECODE Debug Print (now OFF) ?");
    if (strget (q))
    {
        if (q[0] == 'y' || q[0] == 'Y')
            asn1_debug_sel |= ASN1DEC_PRINT;
        else
            asn1_debug_sel &= ~ASN1DEC_PRINT;
    }

    if (mms_debug_sel & MMSINDQUE_PRINT)
        printf ("\n Enable Indication Que Information Display (now ON) ?");
    else
        printf ("\n Enable Indication Que Information Display (now OFF) ?");
    if (strget (q))
    {
        if (q[0] == 'y' || q[0] == 'Y')
            mms_debug_sel |= MMSINDQUE_PRINT;
        else
            mms_debug_sel &= ~MMSINDQUE_PRINT;
    }

    if (mms_debug_sel & MMSREQQUE_PRINT)
        printf (" Enable Request Que Information Display (now ON) ?");
    else
        printf (" Enable Request Que Information Display (now OFF) ?");

```

```

if (strget (q))
{
    if (q[0] == 'y' || q[0] == 'Y')
        mms_debug_sel |= MMSREQUEUE_PRINT;
    else
        mms_debug_sel &= ~MMSREQUEUE_PRINT;
}

if (mms_debug_sel & MMSIND_PRINT)
    printf ("\n Enable MMS Indication Information (now ON) ?");
else
    printf ("\n Enable MMS Indication Information (now OFF) ?");
if (strget (q))
{
    if (q[0] == 'y' || q[0] == 'Y')
        mms_debug_sel |= MMSIND_PRINT;
    else
        mms_debug_sel &= ~MMSIND_PRINT;
}

if (mms_debug_sel & MMSCONF_PRINT)
    printf (" Enable MMS Confirm Information (now ON) ?");
else
    printf (" Enable MMS Confirm Information (now OFF) ?");
if (strget (q))
{
    if (q[0] == 'y' || q[0] == 'Y')
        mms_debug_sel |= MMSCONF_PRINT;
    else
        mms_debug_sel &= ~MMSCONF_PRINT;
}
}

#endif

```

Server Domain Function Routines "DTS_DOM.C"

```
/*=====
                                     DTS_DOM.C
=====
Purpose: The purpose of this module is to take over the primitive
         domain interface in order to send TARGA images in the guise
         of a domain upload.
=====*/

/*=====| Include Files |=====*/
/*=====|

#include <stdio.h>          /* Standard 'C' include files */
#include <conio.h>

#include "glbtypes.h"      /* MMS-EASE include files   */
#include "mms_user.h"

#include "tardev.h"         /* Targa include files     */
#include "packbits.h"      /* Compression include file */

/*=====| Typedefs |=====*/
/*=====|

typedef struct _IMAGE_DATA { /* Header for image */
short Version,
    XSize,
    YSize,
    Compressed;
long  Reserved1,
    Reserved2;
} IMAGE_DATA;

/*=====| Function Declarations |=====*/
/*=====|

long FillBuffer ( short, short, short );
void SendHeader ( MMSREQ_IND * );
void SendData   ( MMSREQ_IND * );

/*=====| Global Variables |=====*/
/*=====|

unsigned char far *BuffPtr;      /* Send buffer pointer */
unsigned char  Buff1[65536];    /* Compression Buffer */
unsigned char  Buff2[65536];    /* Compression Buffer */
long          Bytes;           /* Current bytes left */
short         Page;           /* Current video page */
IMAGE_DATA    image;          /* Image header */
void          (*ReadIndFunc)(MMSREQ_IND *); /* Data function pointer */
```

```

/*=====| u_initupl_ind |=====*/
Purpose: This is an MMS-EASE defined, user implemented function that
         is called when MMS receives a initupl.IND from a remote
         initiate upload request. This tells the program that a client
         on the network wants to upload an image domain. The TARGA
         board and variables for encoding the state are initialized. An
         initupl.RESP is sent back and the first 64K video segment is
         acquired.
-----
Inputs:  req_ptr - A pointer to an MMS-EASE information structure
-----
Outputs: None.
/*=====*/

```

```

VOID u_initupl_ind(req_ptr)
MMSREQ_IND *req_ptr;
{
    char                Answer[30];
    short               RetCode,
                      XSize,
                      YSize;
    struct initupl_resp_info info;

    GraphInit ( TYPE_M8 );          /* Initialize Targa board */
    SetPageMode ( 1 );              /* Set to 512 X 512 mode */
    GetScrSize ( &XSize, &YSize ); /* Get actual page size */

    printf ( "\nu_fopen_ind received on channel %0X!\n", req_ptr->chan );
                                   /* Get compression type */
    printf ( "\nCompression Type? [0=None, 1=Bitpack]: " );
    gets ( Answer );
    if ( Answer[0] == '0' )
        image.Compressed = 0;
    else if ( Answer[0] == '1' )
        image.Compressed = 1;

    image.Version      = 1;          /* Fill in header */
    image.XSize        = ++XSize;
    image.YSize        = ++YSize;
    image.Reserved1     = 0;
    image.Reserved2     = 0;

    printf ( "\nDomain Block Size is %d\n",
             mms_chan_info[req_ptr->chan].file_blk_size );

    printf ( "\nImage Size is %d X %d\n", XSize, YSize );

    info.ulsmid = 1;                 /* Fill in response header */
    info.ul_detail_pres = 0;
    info.num_of_capab = 0;

    ReadIndFunc = SendHeader;        /* Send header first */

    Page = 3;
    Bytes = FillBuffer ( YSize-1, 128-(512-YSize), image.Compressed );

    SetBLiveMode();                 /* Grab image */
    printf ( "\nHit any key ...\n" );
    getch();
    GrabFrame();
    SetDispMode ();

                                   /* Send initupl response back */
    if ( (RetCode = mp_initupl_resp ( req_ptr, &info )) )
        printf ( "\nmp_initupl_resp returned Error Code %d\n", RetCode );
    else
        printf ( "\ninitupl_resp sent ok ..." );
}

```

```

/*=====| u_termupl_ind |=====*/
Purpose: This is an MMS-EASE defined, user implemented function that
         is called when MMS receives an termupl.IND from a remote
         domain upload. This tells the program that a client on the
         network has finished loading an image. A termupl.RESP is
         sent back.
-----
Inputs:  req_ptr - A pointer to an MMS-EASE information structure
-----
Outputs: None.
/*=====*/

```

```

VOID u_termupl_ind ( MMSREQ_IND *req_ptr )
{
    short RetCode;

    /* Send termupl response */

    printf ( "\nu_termupl_ind received on channel %0X!\n", req_ptr->chan );

    if ( (RetCode = mp_termupl_resp ( req_ptr ) ) != 0 )
        printf ( "\nError %0X received while sending mp_termupl_resp.\n",
            RetCode );
    else
        printf ( "\nSending mp_termupl_resp.\n" );
}

```

```

/*=====| u_upload_ind |=====*/
Purpose: This is an MMS-EASE defined, user implemented function that
         is called when MMS receives an upload.IND from a remote domain
         upload request. In this funtion the client is requesting the
         next block of data from the image. A function pointer is used
         to control the state as an information header is sent before
         the image data.
-----
Inputs:  req_ptr - A pointer to an MMS-EASE information structure
-----
Outputs: None.
/*=====*/

```

```

VOID u_upload_ind( MMSREQ_IND *req_ptr )
{
    /* Route indication to appropriate function */

    printf ( "\nBytes is %ld", Bytes );
    printf ( "\nPage is %d", Page );

    (*ReadIndFunc)( req_ptr );
}

```

```

/*=====| SendHeader |=====*/
Purpose: This function is called by the ReadIndFunc pointer to send
the image information header. After the header is sent,
the ReadIndFunc pointer is set to the SendData function to
actually send the image data.
-----
Inputs:  req_ptr - A pointer to an MMS-EASE information structure
-----
Outputs: None.
/*=====*/

```

```

void SendHeader ( MMSREQ_IND *req_ptr )
{
    short          RetCode;
    struct upload_resp_info  respinfo;

    /* Fill in response structure */

    respinfo.load_data      = (void *) &image;
    respinfo.load_data_len  = sizeof(IMAGE_DATA);
    respinfo.more_follows   = TRUE;

    /* Send image header to client */

    if ( (RetCode = mp_upload_resp ( req_ptr, &respinfo ) ) != 0 ) {
        printf ( "\nError %0X received while sending mp_upload_resp.\n",
                RetCode );
    } else {
        printf ( "\nHeader mp_upload_resp sent ok!\n" );
    }

    ReadIndFunc = SendData;    /* New indication function is SendData */
}

```

```

/*=====| SendData |=====*/
Purpose: This function is called by the ReadIndFunc pointer to send
the actual image data.
-----
Inputs:  req_ptr - A pointer to an MMS-EASE information structure
-----
Outputs: None.
/*=====| SendData |=====*/

void SendData ( MMSREQ_IND *req_ptr )
{
    short          RetCode;
    struct upload_resp_info  respinfo;

    /* If end of video page ... */
    if ( Bytes < mms_chan_info[req_ptr->chan].file_blk_size ) {
        respinfo.load_data_len = Bytes;
        respinfo.load_data     = BuffPtr;
        if ( --Page < 0 ) {          /* If no more video pages ... */
            respinfo.more_follows = FALSE;          /* Then last block */
                                                    /* Send block to client */
            if ( (RetCode = mp_upload_resp ( req_ptr, &respinfo ) ) != 0 ) {
                printf ( "\nError %0X received while sending mp_upload_resp.\n",
                    RetCode );
            } else {
                printf ( "\nPARTIAL NEG mp_fread_resp sent ok!\n" );
            }
        } else {
            /* Still more video pages ... */
            respinfo.more_follows = TRUE;
            if ( (RetCode = mp_upload_resp ( req_ptr, &respinfo ) ) != 0 ) {
                printf ( "\nError %0X received while sending mp_upload_resp.\n",
                    RetCode );
            } else {
                printf ( "\nPARTIAL POS mp_upload_resp sent ok!\n" );
            }
            Bytes = FillBuffer ( (Page+1)*128-1, 128, image.Compressed );
        }
    } else {
        /* Still more full blocks on this video page ... */
        respinfo.load_data_len = mms_chan_info[req_ptr->chan].file_blk_size;
        respinfo.load_data     = BuffPtr;
        respinfo.more_follows = TRUE;
                                                    /* Send block to client */
        if ( (RetCode = mp_upload_resp ( req_ptr, &respinfo ) ) != 0 ) {
            printf ( "\nError %0X received while sending mp_upload_resp.\n",
                RetCode );
        } else {
            printf ( "\nFULL mp_upload_resp sent ok!\n" );
        }
    }
    /* Update buffer pointer */
    BuffPtr += mms_chan_info[req_ptr->chan].file_blk_size;
    Bytes   -= mms_chan_info[req_ptr->chan].file_blk_size;
}
}

```

```

/*=====| FillBuffer |=====*/
Purpose: This function is used to fill a buffer with a page (64K) of
         image data at a time in the correct order.
-----
Inputs:  StartLine - the topmost scan line to start with.
         NumLines   - the number of scan lines to get.
         CompType   - the compression method for the data, if any.
-----
Outputs: The actual number of bytes in the buffer after processing.
/*=====*/

long FillBuffer ( short StartLine, short NumLines, short CompType )
{
    register short    x;

    BuffPtr = Buff1;

    for ( x=0; x<NumLines; x++, BuffPtr+=512 )    /* Get image lines in */
        GetRow ( BuffPtr, 0, 511, StartLine-x, -1 ); /* buffer.          */

    switch ( CompType ) {
        case 0: BuffPtr = Buff1;                    /* No compression      */
                return ( 512L*(long)NumLines );
                break;
        case 1: BuffPtr = Buff2;                    /* Pack bits           */
                return ( PackUBits ( Buff2, Buff1, 512L*(long)NumLines ) );
                break;
        default: return ( 0 );
                break;
    }
    return ( 0 );
}

```

Server Stub File "DTS_STUB.C"

```
/*=====*\
                                     DTS_STUB.H
-----*/
Purpose:      The purpose of this module is to provide stubs for the
               MMS-EASE functions that must exist but do not do anything.
/*=====*\

/*=====| Include Files |=====*\
\*=====*\

#include "GLBTYPES.H"

/*****
/*          Variable Operation Stubs          */
*****/

BYTE u_get_named_addr(){}

/*****
/*          Error Recovery Stubs              */
*****/

VOID u_cancel_ind(){}
VOID u_reject_ind(){}
VOID u_mp_cancel_conf(){}

```

Server MMS-EASE Function Selection Header "DTS_ENAB.H"

```

/*=====
                                DTS_ENAB.H
-----
Purpose:   The purpose of this module is to enable a subset of the
            MMS-EASE system. It is a version of the normal mmsop_en.h
            file used with all MMS-EASE programs.
=====*/

#ifndef MMSOP_EN_INCLUDED
#define MMSOP_EN_INCLUDED

/*****
/*****
/* define the operation enable switches */
/*****
#define REQ_RESP_DIS 0x00 /* no support for req or resp */
#define REQ_EN       0x01 /* support for request */
#define RESP_EN      0x02 /* support for response */
#define REQ_RESP_EN  0x03 /* support for resp and req */

/*****
/* define the opcode enable switches */
/*****
#define MMS_INIT_EN          RESP_EN
#define MMS_CONCLUDE_EN      RESP_EN
#define MMS_CANCEL_EN        REQ_RESP_EN

#define MMS_STATUS_EN        REQ_RESP_DIS
#define MMS_USTATUS_EN       REQ_RESP_DIS
#define MMS_GETNAMES_EN      REQ_RESP_DIS
#define MMS_IDENT_EN         REQ_RESP_DIS
#define MMS_RENAME_EN        REQ_RESP_DIS
#define MMS_GETCL_EN         REQ_RESP_DIS
#define MMS_VMD_EN           1

#define MMS_MV_DWN_EN         REQ_RESP_DIS /* VM Upload */
#define MMS_INIT_DWN_EN       REQ_RESP_DIS
#define MMS_DWN_LOAD_EN       REQ_RESP_DIS
#define MMS_TERM_DWN_EN       REQ_RESP_DIS
#define MMS_MV_UPL_EN         REQ_RESP_DIS /* VM Download */
#define MMS_INIT_UPL_EN       RESP_EN
#define MMS_UP_LOAD_EN        RESP_EN
#define MMS_TERM_UPL_EN       RESP_EN
#define MMS_RDDWN_EN          REQ_RESP_DIS
#define MMS_RDUPL_EN          REQ_RESP_DIS
#define MMS_LOAD_DOM_EN       REQ_RESP_DIS
#define MMS_STR_DOM_EN        REQ_RESP_DIS
#define MMS_DEL_DOM_EN        REQ_RESP_DIS
#define MMS_GET_DOM_EN        REQ_RESP_DIS
#define MMS_NAMED_DOM_EN      0

#define MMS_CRE_PI_EN         REQ_RESP_DIS
#define MMS_DEL_PI_EN         REQ_RESP_DIS
#define MMS_START_EN          REQ_RESP_DIS
#define MMS_STOP_EN           REQ_RESP_DIS
#define MMS_RESUME_EN         REQ_RESP_DIS
#define MMS_RESET_EN          REQ_RESP_DIS
#define MMS_KILL_EN           REQ_RESP_DIS
#define MMS_GET_PI_EN         REQ_RESP_DIS

#define MMS_VA_EN             1
#define MMS_MV_READ_EN        REQ_RESP_DIS /* VM Read variable(s) */
#define MMS_MV_RDVARS_EN      REQ_RESP_DIS /* General VM Rd var's */
#define MMS_READ_EN           REQ_RESP_DIS

```

```

#define MMS_MV_WRITE_EN      REQ_RESP_DIS /* VM Write variable(s) */
#define MMS_MV_WRVARS_EN    REQ_RESP_DIS /* General VM WR var's */
#define MMS_WRITE_EN        REQ_RESP_DIS
#define MMS_INFO_EN         REQ_RESP_DIS
#define MMS_GETVAR_EN        REQ_RESP_DIS
#define MMS_DEFVAR_EN        REQ_RESP_DIS
#define MMS_DEFSCAT_EN       REQ_RESP_DIS
#define MMS_GETSCAT_EN       REQ_RESP_DIS
#define MMS_DELVAR_EN        REQ_RESP_DIS
#define MMS_DEFVLIST_EN      REQ_RESP_DIS
#define MMS_GETVLIST_EN      REQ_RESP_DIS
#define MMS_DELVLIST_EN      REQ_RESP_DIS
#define MMS_MV_DEFTYPE_EN    REQ_RESP_DIS /* VM DefineType */
#define MMS_DEFTYPE_EN       REQ_RESP_DIS
#define MMS_GETTYPE_EN       REQ_RESP_DIS
#define MMS_DELTYPE_EN       REQ_RESP_DIS

#define MMS_TAKECTRL_EN      REQ_RESP_DIS
#define MMS_RELCTRL_EN       REQ_RESP_DIS
#define MMS_DEFINE_SEM_EN    REQ_RESP_DIS
#define MMS_DELETE_SEM_EN    REQ_RESP_DIS
#define MMS_REP_SEMSTAT_EN   REQ_RESP_DIS
#define MMS_REP_SEMPOOL_EN   REQ_RESP_DIS
#define MMS_REP_SEMENTRY_EN  REQ_RESP_DIS

#define MMS_JREAD_EN         REQ_RESP_DIS
#define MMS_JWRITE_EN        REQ_RESP_DIS
#define MMS_JINIT_EN         REQ_RESP_DIS
#define MMS_JSTAT_EN         REQ_RESP_DIS
#define MMS_JCREATE_EN       REQ_RESP_DIS
#define MMS_JDELETE_EN       REQ_RESP_DIS

#define MMS_DEFEC_EN         REQ_RESP_DIS
#define MMS_DELEC_EN         REQ_RESP_DIS
#define MMS_GETECA_EN        REQ_RESP_DIS
#define MMS_REPECS_EN        REQ_RESP_DIS
#define MMS_ALTECM_EN        REQ_RESP_DIS
#define MMS_TRIGE_EN         REQ_RESP_DIS
#define MMS_DEFEA_EN         REQ_RESP_DIS
#define MMS_DELEA_EN         REQ_RESP_DIS
#define MMS_GETEAA_EN        REQ_RESP_DIS
#define MMS_REPEAS_EN        REQ_RESP_DIS
#define MMS_DEFEE_EN         REQ_RESP_DIS
#define MMS_DELEE_EN         REQ_RESP_DIS
#define MMS_GETEEA_EN        REQ_RESP_DIS
#define MMS_REPEES_EN        REQ_RESP_DIS
#define MMS_ALTEE_EN         REQ_RESP_DIS
#define MMS_EVNOT_EN         REQ_RESP_DIS
#define MMS_ACKEVNOT_EN      REQ_RESP_DIS
#define MMS_GETAS_EN         REQ_RESP_DIS
#define MMS_GETAES_EN        REQ_RESP_DIS

#define MMS_INPUT_EN         REQ_RESP_DIS
#define MMS_OUTPUT_EN        REQ_RESP_DIS

#define MMS_MV_FCOPY_EN      REQ_RESP_DIS
#define MMS_OBTAINFILE_EN    REQ_RESP_DIS
#define MMS_MV_FOPEN_EN      REQ_RESP_DIS
#define MMS_FOPEN_EN         REQ_RESP_DIS
#define MMS_MV_FREAD_EN      REQ_RESP_DIS
#define MMS_FREAD_EN         REQ_RESP_DIS
#define MMS_MV_FCLOSE_EN     REQ_RESP_DIS
#define MMS_FCLOSE_EN        REQ_RESP_DIS
#define MMS_FRENAME_EN       REQ_RESP_DIS
#define MMS_FDELETE_EN       REQ_RESP_DIS
#define MMS_FDIR_EN          REQ_RESP_DIS

#endif

```

I.2 MAP 2.X Client Code

Client Make File "DTR_2X.MAK"

```
OPS = /c /AL /W2 /DLINT_ARGS

dtr.obj      : dtr.c
cl $(OPS) /DMAP_2X dtr.c ;

dtr_dom.obj  : dtr_dom.c
cl $(OPS) dtr_dom.c ;

recvconn.obj : recvconn.c
cl $(OPS) recvconn.c ;

getval5.obj  : getval5.c
cl $(OPS) getval5.c ;

fkey.obj     : fkey.c
cl $(OPS) fkey.c ;

dtr_stub.obj : dtr_stub.c
cl $(OPS) dtr_stub.c ;

mmsop_en.obj : mmsop_en.c dtr_enab.h
cl $(OPS) /DMAP21_CASE /DDOM_RECV mmsop_en.c ;

packbits.obj : packbits.c
cl $(OPS) packbits.c ;

ev678.obj    : ev678.asm
tasm /Zd /Zi /Mx ev678

dtr_2x.exe : dtr.obj dtr_dom.obj recvconn.obj packbits.obj getval5.obj \
             fkey.obj dtr_stub.obj mmsop_en.obj ev678.obj
link @dtr_2x.lnk
```

Client Linker Response File "DTR_2X.LNK"

```
dtr +
dtr_dom +
recvconn +
dtr_stub +
mmsop_en +
getval5 +
fkey +
packbits +
ev678
dtr_2x
dtr_2x.map /LI /MAP:60000 /CO /ST:8196 /SEGMENTS:521 /NOD /NOE /NOI
asn1d mmsd1 mmsd2 mmscasc suiccase llibce
```

Client Main File "DTR.C"

```
/*=====
                                     DTR.C
=====
Purpose: The purpose of this program is to get a video image from
a server on the network.
-----
Implementation: The interface for the data is through the standard
domain upload services provided under MMS. By taking over
the domain services at the primitive level, this program
can receive a video image that can be displayed or stored.
=====*/

/*=====| Defines |=====*/
\*=====*/

#define DEBUG_SET
#define OVER_HEAD 100

/*=====| Include Files |=====*/
\*=====*/

#include <stdio.h>          /* Include standard 'C' header files */
#include <stdlib.h>
#include <conio.h>

#include "glbtypes.h"       /* Include necessary MMS-EASE header files */
#include "mms_user.h"
#include "asn1defs.h"
#include "mms_ms.h"

#ifdef MAP_30               /* If under MAP 3.0 include ACSE */
#define CONCORD
#include "suicacse.h"
#endif

#ifdef MAP_2X               /* If under MAP 2.X include CASE */
#define MP_400
#include "suiccase.h"
#endif

/*=====| Function Declarations |=====*/
\*=====*/

VOID msg_ready ( BYTE * );
VOID GetImage ( char * );

extern void SetNormalMode ( unsigned char );
extern void SetExtendedMode ( unsigned char );
extern void SetExtendedPage ( unsigned char );
extern short Activate ( char * );
extern short RegisterName ( short, char * );
extern short Initiate ( short, char * );
extern short Conclude ( short );
extern VOID GetDomain ( short, char * );

/*=====| Global Variables |=====*/
\*=====*/

extern short DoneFlag;      /* Network service done flag */
```

```

/*=====| main |=====*/
Standard 'C' program entry. "main" starts up the MMS-EASE system and
handles user input.
/*=====*/

VOID main()
{
char MenuChoice;      /* Stores user menu choice */
char Loc[65];         /* Local process name */
char Rem[65];         /* Remote process name */
int BlockSize;        /* Data block size */
char Buffer[80];       /* Buffer for input */

printf ( "\n\nEnter Block Size: " ); /* Get data block size from user */
gets ( Buffer );
BlockSize = atoi ( Buffer );

mms_debug_sel = 0; /* Set debug parameters */
asn1_debug_sel = 0;
m_file_blk_size = BlockSize; /* Set data block size */

#ifdef MAP_30 /* If under MAP 3.0 include these lines */
s_msgsize = BlockSize + OVER_HEAD; /* Set MMS vars up for block size */
m_segsize = BlockSize + OVER_HEAD;
max_mms_chan = 1; /* Only one channel available */
s_max_chan = 1;

s_dma_en = TRUE; /* Enable DMA */
s_dma_chan = 3; /* On channel 3 */
s_int_en = TRUE; /* Enable interrupts */
s_int_no = 3; /* Use interrupt 3 */
s_boot_talk_en = TRUE; /* Ask user for reset */
#endif

#ifdef MAP_2X /* If under MAP 2.X include these lines */
m_segsize = 1020; /* Set MMS vars up for block size */
case_size = 1020;
max_mms_chan = 1; /* One channel only */
max_suic_chan = 1;

SUIC_int_no = 2; /* Use interrupt 2 */
boot_talk_en = 0; /* Reset always */
suic_hw_reset = 1;
prntn = 1; /* Print any board errors */
#endif

do_debugset(); /* Prompt user for debug levels */

strt_MMS(); /* Start up MMS-EASE */

printf ( "Enter the local Name: " ); /* Get local name from user */
gets ( Loc );

Activate ( Loc ); /* Activate name */
RegisterName ( 0, Loc ); /* Register name on channel 0 */

do { /* print out menu */
printf ( "\n\n\n\n\n\n\n\n\t\tTarga Image Retriever\n\n\n" );
printf ( "G) Get An Image\n" );
printf ( "X) Exit Program\n\n" );
printf ( "Enter Number of Choice: " );

while ( !kbhit() ) /* While no keyboard input service */
comm_serve(); /* the network */

MenuChoice = getch(); /* Get keyboard input */
}

```

```

switch ( MenuChoice ) {          /* Compare to choices */
    case 'g':                    /* Get image */
    case 'G': printf ( "\n\n\n\n\n\nEnter name of Image Server: " );
                gets( Rem );
                GetImage ( Rem );
                break;
    case 'x':                    /* Exit program */
    case 'X': break;
    default: break;
}

} while ( MenuChoice != 'x' && MenuChoice != 'X' );

}

/*=====| GetImage |=====*/
Purpose: This function makes a connection to an image server,
         downloads and displays an image, and disconnects from the
         server.
-----
Inputs:  Remote - Pointer to name of remote server.
-----
Outputs: None.
/*=====| GetImage |=====*/

VOID GetImage ( char *Remote )
{
    char FileName[256];

    DoneFlag = 0;                /* Init flag */
    Initiate ( 0, Remote );      /* Initiate connection */
    while ( !DoneFlag )         /* Service network until init.conf */
        comm_serve();

    printf ( "\nGetting domain ..." );

    DoneFlag = 0;                /* Init flag */
    GetDomain ( 0, "TARGA-M8" ); /* Download image */
    while ( !DoneFlag )         /* Service network until download */
        comm_serve();          /* is done. */

    DoneFlag = 0;                /* Init flag */
    Conclude( 0 );              /* Disconnect from server */
    while ( !DoneFlag )         /* Service network until disconnected */
        comm_serve();

    getch();                    /* Keep image on screen until user */
    SetNormalMode ( 0x02 );     /* hits a key ... then restore text */

    return;
}

```

```

/*=====| msg_ready |=====*/
Purpose: This function is required by MMS-EASE
-----
Inputs:  msgptr - Pointer to message.
-----
Outputs: None.
-----
/*=====|

```

```

VOID msg_ready (msgptr)
BYTE *msgptr;
{
    register i;

    printf ("\33[1;1H");
    for (i = 0; i < 3; ++i)
    {
        printf ("\n");
    }
    printf ("\33[1;1H");
    printf ("%s",msgptr);
}

```

```

#ifdef DEBUG_SET

```

```

/*=====| do_debugset |=====*/
Purpose: This function sets the debug level for MMS-EASE. It was
         taken from the MMS-EASE sample program.
-----
Inputs:  None.
-----
Outputs: None.
-----
/*=====|

```

```

VOID do_debugset ()
{
    BYTE q[100];
    FILE *temp_file;

    if (mms_debug_sel & MMSCASE_PRINT)
        printf ("\n\n\n Enable CASE INTERFACE Debug Print (now ON) ?");
    else
        printf ("\n\n\n Enable CASE INTERFACE Debug Print (now OFF) ?");
    if (strget (q))
    {
        if (q[0] == 'y' || q[0] == 'Y')
            mms_debug_sel |= MMSCASE_PRINT;
        else
            mms_debug_sel &= ~MMSCASE_PRINT;
    }

    if (mms_debug_sel & MMSENC_PRINT)
        printf (" Enable MMS ENCODE Debug Print (now ON) ?");
    else
        printf (" Enable MMS ENCODE Debug Print (now OFF) ?");
    if (strget (q))
    {
        if (q[0] == 'y' || q[0] == 'Y')
            mms_debug_sel |= MMSENC_PRINT;
        else
            mms_debug_sel &= ~MMSENC_PRINT;
    }

    if (mms_debug_sel & MMSDEC_PRINT)
        printf (" Enable MMS DECODE Debug Print (now ON) ? ");
    else

```

```

    printf (" Enable MMS DECODE Debug Print (now OFF) ? ");
if (strget (q))
{
    if (q[0] == 'y' || q[0] == 'Y')
        mms_debug_sel |= MMSDEC_PRINT;
    else
        mms_debug_sel &= ~MMSDEC_PRINT;
}

if (asn1_debug_sel & ASN1ENC_PRINT)
    printf ("\n Enable ASN.1 ENCODE Debug Print (now ON) ?");
else
    printf ("\n Enable ASN.1 ENCODE Debug Print (now OFF) ? ");
if (strget (q))
{
    if (q[0] == 'y' || q[0] == 'Y')
        asn1_debug_sel |= ASN1ENC_PRINT;
    else
        asn1_debug_sel &= ~ASN1ENC_PRINT;
}

if (asn1_debug_sel & ASN1DEC_PRINT)
    printf (" Enable ASN.1 DECODE Debug Print (now ON) ?");
else
    printf (" Enable ASN.1 DECODE Debug Print (now OFF) ?");
if (strget (q))
{
    if (q[0] == 'y' || q[0] == 'Y')
        asn1_debug_sel |= ASN1DEC_PRINT;
    else
        asn1_debug_sel &= ~ASN1DEC_PRINT;
}

if (mms_debug_sel & MMSINDQUE_PRINT)
    printf ("\n Enable Indication Que Information Display (now ON) ?");
else
    printf ("\n Enable Indication Que Information Display (now OFF) ?");
if (strget (q))
{
    if (q[0] == 'y' || q[0] == 'Y')
        mms_debug_sel |= MMSINDQUE_PRINT;
    else
        mms_debug_sel &= ~MMSINDQUE_PRINT;
}

if (mms_debug_sel & MMSREQQUE_PRINT)
    printf (" Enable Request Que Information Display (now ON) ?");
else
    printf (" Enable Request Que Information Display (now OFF) ?");
if (strget (q))
{
    if (q[0] == 'y' || q[0] == 'Y')
        mms_debug_sel |= MMSREQQUE_PRINT;
    else
        mms_debug_sel &= ~MMSREQQUE_PRINT;
}

if (mms_debug_sel & MMSIND_PRINT)
    printf ("\n Enable MMS Indication Information (now ON) ?");
else
    printf ("\n Enable MMS Indication Information (now OFF) ?");
if (strget (q))
{
    if (q[0] == 'y' || q[0] == 'Y')
        mms_debug_sel |= MMSIND_PRINT;
    else
        mms_debug_sel &= ~MMSIND_PRINT;
}

```

```

if (mms_debug_sel & MMSCONF_PRINT)
    printf (" Enable MMS Confirm Information (now ON) ?");
else
    printf (" Enable MMS Confirm Information (now OFF) ?");
if (strget (q))
{
    if (q[0] == 'y' || q[0] == 'Y')
        mms_debug_sel |= MMSCONF_PRINT;
    else
        mms_debug_sel &= ~MMSCONF_PRINT;
}
}

#endif

```

Client Domain Routines "DTR_DOM.C"

```
/*=====
DTR_DOM.C
=====
Purpose: The purpose of this module is to take over the primitive
domain interface in order to receive TARGA images in the guise
of a domain upload.
=====*/

/*=====| Include Files |=====*/
/*=====*/

#include <stdio.h>          /* Standard 'C' include files */
#include <conio.h>
#include <stdlib.h>
#include <string.h>
#include <malloc.h>

#include "glbtypes.h"      /* MMS-EASE include files */
#include "mms_user.h"

#include "packbits.h"      /* Compression include file */

/*=====| Defines |=====*/
/*=====*/

#define VID_BUFFER 0xA0000000 /* Video buffer address 0xA0000 */

/*=====| Function Declarations |=====*/
/*=====*/

VOID GetDomain    ( short, char * );
void LoadDacs256 ( void );
void LoadDacs64  ( void );
void ReadHeader   ( MMSREQ_PEND * );
void ReadData     ( MMSREQ_PEND * );
void ProcessBuffer ( unsigned char far *, short, short );

extern void SetNormalMode ( unsigned char );
extern void SetExtendedMode ( unsigned char );
extern void SetExtendedPage ( unsigned char );

/*=====| Typedefs |=====*/
/*=====*/

typedef struct _IMAGE_DATA { /* Header for image */
short Version,
      XSize,
      YSize,
      Compressed;
long  Reserved1,
      Reserved2;
} IMAGE_DATA;
```

```

/*=====| Global Variables |=====*\
\*=====*\

long          BytesRead;          /* Total bytes read */
unsigned char far *VidBuf;        /* Video buffer ptr */
unsigned char Buff1[65536];      /* Decompression buffer */
short         Page;               /* Current video page */
void          (*ReadFunc) ( MMSREQ_PEND * ); /* Decoding pointer */
IMAGE_DATA    image;             /* image header struct */
struct upload_req_info ul_info;   /* Domain upload struct */

extern short   DoneFlag;          /* Polling flag */

```

```

/*=====| GetDomain |=====*\
| Purpose: This function uploads a domain from a remote image server and |
| displays the data as an image.                                         |
|-----|
| Inputs:  cham      - Channel to use.                                   |
|          DomName   - Pointer to name of domain to upload.             |
|-----|
| Outputs: None.                                                         |
\*=====*\

```

```

VOID GetDomain ( short chan, char *DomName )
{
    short          RetCode;
    struct initupl_req_info info;

    printf ( "\nEntering GetDomain ..." );

    /* Fill in initupl request structure */
    strcpy (info.dname, DomName );

    printf ( "\nAbout to mp_initupl ..." );

    if ( mp_initupl ( chan, &info ) != NULL )
        printf ( "\nmp_initupl sent ok...\n" );
    else
        printf ( "\nmp_initupl bombed with Error = %X\n", mms_op_err );
}

```

```

/*=====| u_mp_initupl_conf |=====*/
Purpose: This is an MMS-EASE defined, user implemented function that
         is called when MMS receives an initupl.RESP from the server.
         This tells the program that the server on the network has
         agreed to upload a domain. The read function is set for
         reading the header as that will be the first block received.
-----
Inputs:  req_ptr - A pointer to an MMS-EASE information structure
-----
Outputs: None.
/*=====*/

VOID u_mp_initupl_conf( req_ptr )
MMSREQ_PEND *req_ptr;
{

    if ( req_ptr->resp_err == RESP_OK ) {
        printf ( "\nmp_initupl Confirmed!\n" );
    } else {
        printf ( "\nError code %d returned in u_mp_initupl_conf!\n", req_ptr->resp_err );
        return;
    }

    /* Get domain upload state machine ID */
    ul_info.ulsmid=((struct upload_req_info *) (req_ptr->resp_info_ptr))->ulsmid;

    if ( mp_upload ( req_ptr->chan, &ul_info ) ) /* Request header block */
        printf ( "\nFirst mp_upload sent ok ...\n" );
    else
        printf ( "\nmp_upload died with Error = %X\n", mms_op_err );

    VidBuf = (char far *) VID_BUFFER;          /* Initialize video pointer */

    ReadFunc = ReadHeader;

}

/*=====| u_mp_upload_conf |=====*/
Purpose: This is an MMS-EASE defined, user implemented function that
         is called when MMS receives an upload.CONF from a remote
         domain upload request. In this function the server has
         returned the next block of data from the image. A function
         pointer is used to control the state as an information header
         is sent before the image data.
-----
Inputs:  req_ptr - A pointer to an MMS-EASE information structure
-----
Outputs: None.
/*=====*/

VOID u_mp_upload_conf( req_ptr )
MMSREQ_PEND *req_ptr;
{

    if ( req_ptr->resp_err ) {
        SetNormalMode ( 0x02 );
        printf ( "\nError %X%X recieved in u_mp_upload_conf\n", req_ptr->resp_err );
        printf ( "Hit any key ..." );
        getch();
    } else
        (*ReadFunc)( req_ptr );

}

```

```

/*=====| ReadHeader |=====*/
Purpose: This function is called by the ReadFunc pointer to acquire
         the image information header. After the header is received,
         the ReadFunc pointer is set to the ReadData function to
         actually receive the image data.
-----
Inputs:  req_ptr - A pointer to an MMS-EASE information structure
-----
Outputs: None.
/*=====*/

```

```

void ReadHeader ( MMSREQ_PEND *req_ptr )
{
    struct upload_resp_info *resp_info;

    /* Get pointer to data */
    resp_info = (struct upload_resp_info *)req_ptr->resp_info_ptr;

    /* Copy header into header structure */
    memcpy ( &image, resp_info->load_data, sizeof(IMAGE_DATA) );

    printf ( "\nImage header: X = %d, Y = %d, C = %d", image.XSize, image.YSize,
            image.Compressed );
    getch();

    /* Request next block of data */
    if ( !mp_upload ( req_ptr->chan, &ul_info ) ) {
        SetNormalMode ( 0x02 );
        printf ( "\nmp_upload died with Error = %X\n", mms_op_err );
        printf ( "\nHit any Key ...\n" );
        getch();
    }

    SetExtendedMode ( 0x15 ); /* Go into graphics mode */

    switch ( image.Compressed ) { /* Set up palette depending on compression */
        case 0: LoadDacs256 ();
            break;
        case 1: LoadDacs64 ();
            break;
    }

    SetExtendedPage ( 0 ); /* Set first video page */

    BytesRead = (512L - (long)image.YSize) * 512L;
    Page      = 0;
    memset ( Buff1, 0x00, (short) BytesRead );

    ReadFunc = ReadData; /* Now read data */
}

```

```

/*=====| ReadData |=====*/
Purpose: This function is called by the ReadFunc pointer to receive
         the actual image data.
-----
Inputs:  req_ptr - A pointer to an MMS-EASE information structure
-----
Outputs: None.
/*=====*/

void ReadData ( MMSREQ_PEND *req_ptr )
{
    struct upload_resp_info *resp_info;

    /* Get pointer to data */
    resp_info = (struct upload_resp_info *)req_ptr->resp_info_ptr;

    /* If more data to be read request next block */
    if ( resp_info->more_follows == TRUE ) {
        if ( !mp_upload ( req_ptr->chan, &ul_info ) ) {
            SetNormalMode ( 0x02 );
            printf ( "\nmp_upload died with Error = %X\n", mms_op_err );
            printf ( "\nHit any Key ...\n" );
            getch();
        }
    } else {
        DoneFlag = 1;
    }

    /* Process the image data */
    ProcessBuffer ( resp_info->filedata, resp_info->fd_len, image.Compressed );
}

```

```

/*=====| ProcessBuffer |=====*/
Purpose: This function processes the received data blocks according to
         the type of compression used.
-----
Inputs:  data      - A pointer to the image data
         len        - Length of the data buffer
         CompType   - Compression method used on data
-----
Outputs: None.
/*=====*/

void ProcessBuffer ( unsigned char far *data, short len, short CompType )
{
    switch ( CompType ) {
        case 0: memcpy ( Buff1+BytesRead, data, len ); /* Not compressed */
                BytesRead += len;
                break;
        case 1: BytesRead += UnpackBits ( Buff1+BytesRead, data, (long)len );
                break; /* Decompress data */
    }

    if ( BytesRead > 65534L ) { /* When we have a full video page then */
        memcpy ( VidBuf, Buff1, (unsigned short)65535 ); /* display it */
        VidBuf[65535] = Buff1[65535];
        BytesRead = 0L;
        SetExtendedPage ( (unsigned char)(++Page) ); /* Go to next page */
    }
}

```

```

/*=====| LoadDacs256 |=====*/
Purpose: This function is used to set up the grey scale palette for the
VGA for the case of no compression.
-----
Inputs:  None.
-----
Outputs: None.
/*=====*/

```

```

void LoadDacs256 ( void )
{
    int      Loop;
    unsigned char Color;

    outp ( 0x3C8, 0x00 );

    for ( Color=0; Color<64; Color++ ) {
        for ( Loop=0; Loop<4; Loop++ ) {
            outp ( 0x3C9, Color );
            outp ( 0x3C9, Color );
            outp ( 0x3C9, Color );
        }
    }
}

```

```

/*=====| LoadDacs64 |=====*/
Purpose: This function is used to set up the grey scale palette for the
VGA for the case of compression.
-----
Inputs:  None.
-----
Outputs: None.
/*=====*/

```

```

void LoadDacs64 ( void )
{
    unsigned char Color;
    int Loop;

    outp ( 0x3C8, 0x00 );

    for ( Loop=0; Loop<4; Loop++ )
        for ( Color=0; Color<64; Color++ ) {
            outp ( 0x3C9, Color );
            outp ( 0x3C9, Color );
            outp ( 0x3C9, Color );
        }
}

```

Client Stub File "DTR_STUB.C"

```
/*=====*\
                                     DTR_STUB.H
-----*/
Purpose:   The purpose of this module is to provide stubs for the
           MMS-EASE functions that must exist but do not do anything.
/*=====*\

/*=====| Include Files |=====*\
/*=====*\

#include "GLBTYPES.H"

/*****
/*      Variable Operation Stubs      */
*****/

BYTE u_get_named_addr(){}

/*****
/*      Error Recovery Stubs          */
*****/

VOID u_cancel_ind(){}
VOID u_reject_ind(){}
VOID u_mp_cancel_conf(){}

```

Client MMS-EASE Function Selection Header "DTR_ENAB.H"

```
/*=====*/
DTR_ENAB.H
/*=====*/
Purpose:      The purpose of this module is to enable a subset of the
               MMS-EASE system. It is a version of the normal mmsop_en.h
               file used with all MMS-EASE programs.
/*=====*/

#ifndef MMSOP_EN_INCLUDED
#define MMSOP_EN_INCLUDED

/*****
/*****
/* define the operation enable switches */
/*****
#define REQ_RESP_DIS 0x00 /* no support for req or resp */
#define REQ_EN       0x01 /* support for request */
#define RESP_EN      0x02 /* support for response */
#define REQ_RESP_EN  0x03 /* support for resp and req */

/*****
/* define the opcode enable switches */
/*****
#define MMS_INIT_EN          REQ_EN
#define MMS_CONCLUDE_EN      REQ_EN
#define MMS_CANCEL_EN        REQ_RESP_EN

#define MMS_STATUS_EN        REQ_RESP_DIS
#define MMS_USTATUS_EN       REQ_RESP_DIS
#define MMS_GETNAMES_EN      REQ_RESP_DIS
#define MMS_IDENT_EN         REQ_RESP_DIS
#define MMS_RENAME_EN        REQ_RESP_DIS
#define MMS_GETCL_EN         REQ_RESP_DIS
#define MMS_VMD_EN           1

#define MMS_MV_DWN_EN         REQ_RESP_DIS /* VM Upload */
#define MMS_INIT_DWN_EN       REQ_RESP_DIS
#define MMS_DWN_LOAD_EN       REQ_RESP_DIS
#define MMS_TERM_DWN_EN       REQ_RESP_DIS
#define MMS_MV_UPL_EN         REQ_RESP_DIS /* VM Download */
#define MMS_INIT_UPL_EN       REQ_EN
#define MMS_UP_LOAD_EN        REQ_EN
#define MMS_TERM_UPL_EN       REQ_EN
#define MMS_RDDWN_EN          REQ_RESP_DIS
#define MMS_RDUPL_EN          REQ_RESP_DIS
#define MMS_LOAD_DOM_EN       REQ_RESP_DIS
#define MMS_STR_DOM_EN        REQ_RESP_DIS
#define MMS_DEL_DOM_EN        REQ_RESP_DIS
#define MMS_GET_DOM_EN        REQ_RESP_DIS
#define MMS_NAMED_DOM_EN      0

#define MMS_CRE_PI_EN         REQ_RESP_DIS
#define MMS_DEL_PI_EN         REQ_RESP_DIS
#define MMS_START_EN          REQ_RESP_DIS
#define MMS_STOP_EN           REQ_RESP_DIS
#define MMS_RESUME_EN         REQ_RESP_DIS
#define MMS_RESET_EN          REQ_RESP_DIS
#define MMS_KILL_EN           REQ_RESP_DIS
#define MMS_GET_PI_EN         REQ_RESP_DIS

#define MMS_VA_EN             1
#define MMS_MV_READ_EN        REQ_RESP_DIS /* VM Read variable(s) */
#define MMS_MV_RDVARS_EN      REQ_RESP_DIS /* General VM Rd var's */
#define MMS_READ_EN           REQ_RESP_DIS
```

```

#define MMS_MV_WRITE_EN      REQ_RESP_DIS /* VM Write variable(s) */
#define MMS_MV_WRVARS_EN     REQ_RESP_DIS /* General VM WR var's */
#define MMS_WRITE_EN         REQ_RESP_DIS
#define MMS_INFO_EN          REQ_RESP_DIS
#define MMS_GETVAR_EN        REQ_RESP_DIS
#define MMS_DETVAR_EN        REQ_RESP_DIS
#define MMS_DEFSCAT_EN       REQ_RESP_DIS
#define MMS_GETSCAT_EN       REQ_RESP_DIS
#define MMS_DETVAR_EN        REQ_RESP_DIS
#define MMS_DETVLIST_EN      REQ_RESP_DIS
#define MMS_GETVLIST_EN      REQ_RESP_DIS
#define MMS_DETVLIST_EN      REQ_RESP_DIS
#define MMS_MV_DEFTYPE_EN    REQ_RESP_DIS /* VM DefineType */
#define MMS_DEFTYPE_EN       REQ_RESP_DIS
#define MMS_GETTYPE_EN       REQ_RESP_DIS
#define MMS_DELTYPE_EN       REQ_RESP_DIS

#define MMS_TAKECTRL_EN      REQ_RESP_DIS
#define MMS_RELCTRL_EN       REQ_RESP_DIS
#define MMS_DEFINE_SEM_EN    REQ_RESP_DIS
#define MMS_DELETE_SEM_EN    REQ_RESP_DIS
#define MMS_REP_SEMSTAT_EN   REQ_RESP_DIS
#define MMS_REP_SEMPOOL_EN   REQ_RESP_DIS
#define MMS_REP_SEMENTRY_EN  REQ_RESP_DIS

#define MMS_JREAD_EN         REQ_RESP_DIS
#define MMS_JWRITE_EN        REQ_RESP_DIS
#define MMS_JINIT_EN         REQ_RESP_DIS
#define MMS_JSTAT_EN         REQ_RESP_DIS
#define MMS_JCREATE_EN       REQ_RESP_DIS
#define MMS_JDELETE_EN       REQ_RESP_DIS

#define MMS_DEFEC_EN         REQ_RESP_DIS
#define MMS_DELEC_EN         REQ_RESP_DIS
#define MMS_GETECA_EN        REQ_RESP_DIS
#define MMS_REPCS_EN         REQ_RESP_DIS
#define MMS_ALTECM_EN        REQ_RESP_DIS
#define MMS_TRIGE_EN         REQ_RESP_DIS
#define MMS_DEFEA_EN         REQ_RESP_DIS
#define MMS_DELEA_EN         REQ_RESP_DIS
#define MMS_GETEAA_EN        REQ_RESP_DIS
#define MMS_REPEAS_EN        REQ_RESP_DIS
#define MMS_DEFEE_EN         REQ_RESP_DIS
#define MMS_DELEE_EN         REQ_RESP_DIS
#define MMS_GETEEA_EN        REQ_RESP_DIS
#define MMS_REPEES_EN        REQ_RESP_DIS
#define MMS_ALTEE_EN         REQ_RESP_DIS
#define MMS_EVNOT_EN         REQ_RESP_DIS
#define MMS_ACKEVNOT_EN      REQ_RESP_DIS
#define MMS_GETAS_EN         REQ_RESP_DIS
#define MMS_GETAES_EN        REQ_RESP_DIS

#define MMS_INPUT_EN         REQ_RESP_DIS
#define MMS_OUTPUT_EN        REQ_RESP_DIS

#define MMS_MV_FCOPY_EN      REQ_RESP_DIS
#define MMS_OBTAINFILE_EN    REQ_RESP_DIS
#define MMS_MV_FOPEN_EN      REQ_RESP_DIS
#define MMS_FOPEN_EN         REQ_RESP_DIS
#define MMS_MV_FREAD_EN      REQ_RESP_DIS
#define MMS_FREAD_EN         REQ_RESP_DIS
#define MMS_MV_FCLOSE_EN     REQ_RESP_DIS
#define MMS_FCLOSE_EN        REQ_RESP_DIS
#define MMS_FRENAME_EN       REQ_RESP_DIS
#define MMS_FDELETE_EN       REQ_RESP_DIS
#define MMS_FDIR_EN          REQ_RESP_DIS

#endif

```

I.3 MAP 3.0 Server Code

The only difference between the 2.X and 3.0 code is the make files and linker response files, which are given below.

Server Make File "DTS_30.MAK"

```
OPS = /c /AL /W2 /DLINT_ARGS

dts.obj : dts.c
cl $(OPS) /DMAP_30 dts.c ;

sendconn.obj : sendconn.c
cl $(OPS) sendconn.c ;

dts_dom.obj : dts_dom.c
cl $(OPS) dts_dom.c ;

getval5.obj : getval5.c
cl $(OPS) getval5.c ;

fkey.obj : fkey.c
cl $(OPS) fkey.c ;

dts_stub.obj : dts_stub.c
cl $(OPS) dts_stub.c ;

mmsop_en.obj : mmsop_en.c dts_enab.h
cl $(OPS) /DMAP30_ACSE /DDOM_SEND mmsop_en.c ;

packbits.obj : packbits.c
cl /c /AL packbits.c ;

dts_30.exe : dts.obj      dts_dom.obj  sendconn.obj  packbits.obj \
             getval5.obj  fkey.obj     dts_stub.obj  mmsop_en.obj
link @dts_30.lnk
```

Server Linker Response File "FTS_30.LNK"

```
dts +
dts_dom +
sendconn +
dts_stub +
mmsop_en +
getval5 +
packbits +
fkey
dts_30
dts_30.map /LI /MAP:60000 /ST:8196 /SEGMENTS:200 /NOD /NOE /NOI
asn1d mmsd1 mmsd2 mmsacsd suicacse ltarga.lib llibce
```

I.4 MAP 3.0 Client Code

The only difference between the 2.X and 3.0 code is the make files and linker response files, which are given below.

Client Make File "FTR_30.MAK"

```
OPS = /c /AL /W2 /DLINT_ARGS

dtr.obj      : dtr.c
cl $(OPS) /DMP30 dtr.c ;

dtr_dom.obj  : dtr_dom.c
cl $(OPS) dtr_dom.c ;

recvconn.obj : recvconn.c
cl $(OPS) recvconn.c ;

getval5.obj  : getval5.c
cl $(OPS) getval5.c ;

fkey.obj     : fkey.c
cl $(OPS) fkey.c ;

dtr_stub.obj : dtr_stub.c
cl $(OPS) dtr_stub.c ;

mmsop_en.obj : mmsop_en.c dtr_enab.h
cl $(OPS) /DMP30_ACSE /DDOM_RECV mmsop_en.c ;

packbits.obj : packbits.c
cl $(OPS) packbits.c ;

ev678.obj    : ev678.asm
tasm /Zd /Zi /Mx ev678

dtr_30.exe : dtr.obj dtr_dom.obj recvconn.obj packbits.obj getval5.obj \
             fkey.obj dtr_stub.obj mmsop_en.obj ev678.obj
link @dtr_30.lnk
```

Client Linker Response File "FTR_30.LNK"

```
dtr +
dtr_dom +
recvconn +
dtr_stub +
mmsop_en +
getval5 +
fkey +
packbits +
ev678
dtr_30
dtr_30.map /LI /MAP:60000 /CO /ST:8196 /SEGMENTS:521 /NOD /NOE /NOI
asn1d mmsd1 mmsd2 mmsacsd suicacse llibce
```

VITA

VITA

James Gordon Barrett was born in Maryville, Tennessee on December 4, 1966. He grew up in Brentwood, Tennessee and attended elementary school in that city. In June of 1985 he graduated from David Lipscomb High School in Nashville. The following September he entered Tennessee Technological University, and in June 1989 he received a Bachelor of Science degree in Electrical Engineering. In the fall of 1989 he accepted a teaching assistantship at the University of Tennessee, Knoxville and began study towards a Master's degree in electrical engineering. This degree was awarded in December of 1990.

While in graduate school at the University of Tennessee, he accepted a research assistantship at the Oak Ridge National Laboratories. The author is a member of Eta Kappa Nu and the National Honor Society.