

To the Graduate Council:

I am submitting herewith a thesis written by Mark Steven King entitled "A MAP to Data Highway Implementation using Manufacturing Message Specification." 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 O. Pace

William J. McLean

Accepted for the Council:

Cowminkel

Vice Provost
and Dean of The Graduate School

**A MAP To Data Highway
Implementation Using
Manufacturing Message
Specification**

A Thesis
Presented for the
Master of Science
Degree
The University of Tennessee, Knoxville

Mark Steven King
December 1990

ACKNOWLEDGMENTS

I would like to thank Denise Patrishkoff at Systems Integration Specialists Company for her invaluable assistance with both MMS-EASE and F/MAP-DGW. Furthermore, I would also like to thank Allen-Bradley and John Torrance for providing Data Highway hardware and advice.

I especially appreciate the effort made by my graduate committee chairman, Dr. Asa O. Bishop, Jr., Professor of Electrical and Computer Engineering, to help me complete this thesis on a tight schedule. I also thank Dr. William McClain and Dr. Marshall Pace for serving on my committee.

Finally, I would like to thank three people for the assistance they provided at the UTK MAP Laboratory: James Barrett for sharing his vast knowledge of the C programming language; Rick Igou of the Martin Marietta Corporation for helping solve some of the mysteries of MMS; and Patrick McHugh for keeping the PLC and model factory in running condition. The research reported in this thesis was sponsored in part by the Martin Marietta Corporation UTK-101C Task 500 Agreement.

ABSTRACT

As users migrate to MAP-based factory automation solutions, the problem of communicating with devices on proprietary networks must be addressed. A service for device message passing, Manufacturing Message Specification (MMS), was used to develop an application program. MMS is a tool that resides in the Application Layer of the MAP seven-layer protocol stack. MMS was created to allow user application programs running on a MAP node to communicate with programmable devices such as programmable logic controllers (PLC's), robots and computer numerically controlled machine tools. This thesis addresses the specific problem of connecting a MAP network to Data Highway, Allen-Bradley's proprietary network, using a software gateway. The gateway, which runs on an IBM-PC or compatible, communicates over the MAP network using a Concord MAP interface board, and over the Data Highway proprietary network using an Allen-Bradley interface module. The resultant MMS application program allows a user to operate the PLC-controlled model factory from a node on the MAP network. The details of MAP, MMS and Data Highway are transparent to the user.

TABLE OF CONTENTS

CHAPTER	PAGE
1. INTRODUCTION	1
1.1 Computer Communication Protocols	1
1.2 Manufacturing Message Specification	4
1.3 Connecting Existing Networks to MAP	7
2. THE ALLEN-BRADLEY PLC-5/15 AND DATA HIGHWAY	10
2.1 The UTK MAP Laboratory	10
2.2 The ASRS Warehouse PLC Program	11
2.3 The Data Highway and Data Highway Plus Networks.	15
2.4 The 1785-KE Interface Module	16
3. TWO TYPES OF MAP NETWORK NODES	17
3.1 The Concord 1200 Series Node	17
3.2 The Industrial Networking Incorporated MP-400 Node	19
3.3 Communication Between the INI and Concord Nodes	21
3.4 The SISCO F/MAP-DGW Data Highway Gateway.	24

CHAPTER	PAGE
4. CONTROL OF THE MODEL FACTORY OVER THE MAP NETWORK	.26
4.1 Using the MMS-EASE Demonstration Program to Operate the Model Factory	2 6
4.2 Using a Custom MMS Program to Operate the Model Factory	3 3
5. RESULTS AND CONCLUSION	.40
5.1 Results	4 0
5.2 Conclusion	4 4
5.3 Future Work	4 4
LIST OF REFERENCES	.46
APPENDICES	.49
APPENDIX A PLC-5/15 PROGRAM	5 0
APPENDIX B DATA HIGHWAY INTERFACE	5 6
APPENDIX C MAP NODE SUPPORT	5 9
APPENDIX D MAP V2.1 MMS-EASE SUPPORT	6 6
APPENDIX E GATEWAY SUPPORT	7 0
APPENDIX F MAP V2.1 MMS-EASE DEMONSTRATION PROGRAM	8 0

CHAPTER	PAGE
APPENDIX G ASRSSEQ MMS PROGRAM.....	89
APPENDIX H UPGRADE TO MAP V3.0	129
VITA.....	141

LIST OF FIGURES

FIGURE	PAGE
1-1 ISO/OSI Seven Layer Reference Model.....	3
1-2 MMS Client/Server Relationship	5
1-3 Project Work Flow	8
1-4 System Block Diagram.....	9
2-1 View of the Model Factory ASRS Warehouse.	12
4-1 ASRSSEQ Program Flowchart	35
4-2 ASRSSEQ Program State Diagram.....	36
B-1 1785-KE Interface Module Cable Diagram.....	57
C-1 Concord MAP Protocol Stack Loading Procedure.	60
C-2 INI Global DIB Setup Procedure.....	64
D-1 Procedure for Initiating and Concluding an Association.....	67
D-2 MMS-EASE Demo Variable Write Procedure	69
E-1 Gateway Startup Procedure	79
F-1 Adding a New Variable to the MMS-EASE Demo.	85

FIGURE	PAGE
F-2 Operating the Factory with the MMS-EASE Demo Program.	86
G-1 Operating the Factory with the ASRSSEQ Program.....	128

ABBREVIATIONS

AP Title	Application Process Title. The user-assigned name for a process executing on a MAP node.
ASRS	Automated Storage and Retrieval System
CONCDIREXEXE	SUIC_DIB creation utility
CSMA/CD	Carrier-Sense, Multiple Access with Collision Detection
DDGW21.EXE	Debug gateway program for MAP V2.1
DDGW30.EXE	Debug gateway program for MAP V3.0
DGW21.EXE	Gateway program for MAP V2.1
DGW30.EXE	Gateway program for MAP V3.0
DIB	Directory Information Base. This file contains the network address of all processes accessible on the MAP network.
F/MAP-DGW	A software gateway available from Systems Integration Specialists Company used for connecting Allen-Bradley's Data Highway network to a Manufacturing Automation Protocol network.
GATEWAY.CFG	An ASCII-text based configuration file for gateway parameters.
IEEE	Institute of Electrical and Electronic Engineers
INI	Industrial Networking Incorporated
ISO	International Standards Organization

LAN	Local Area Network
MAP	Manufacturing Automation Protocol
MMS	Manufacturing Message Specification
MMS-EASE	Manufacturing Message Specification - Embedded Application Service Element. A product available from Systems Integration Specialists Company used to write MMS applications.
MMS_EASE.CFG	An ASCII-text based configuration file for gateway MMS parameters.
NMC	Network Management Console
OSI	Open Systems Interconnect
PLC	Programmable Logic Controller
PRE_GW.EXE	A program used to check the GATEWAY.CFG file.
SUIC.CFG	An ASCII-text based configuration file that lists the AP Titles that are to be activated and registered on selected channels.
SUIC.DIB	An ASCII-text base file containing the Directory Information Base information for a Concord 1200 Series MAP V3.0 node.
SUIC_DIB	A binary configuration file containing the Directory Information Base information for a Concord 1200 Series MAP V2.1 node.
TBC	Token Bus Controller
TOP	Technical and Office Protocol

CHAPTER 1

INTRODUCTION

1.1 Computer Communication Protocols

Until recently, computers were primarily interconnected by proprietary networks. These networks were usually developed and sold by the same vendor that sold the computer itself. As a result, hardware vendors could provide a completely integrated turn-key package to a user. A problem with this philosophy arises when the user chooses to use equipment from different vendors. Since each vendor uses a different proprietary network, a dilemma known as "Islands of Automation" results in that equipment from one vendor cannot communicate with equipment from other vendors. Large computer users, such as corporations and government, can rarely have all their needs met by a single vendor, thus indicating a need for a common communication protocol. Manufacturing Automation Protocol (MAP), a network for the factory floor, is based on the

International Standards Organization (ISO) Open Systems

Interconnect (OSI) Reference Model shown in Figure 1-1 [1]. IEEE 802.4 specifies a token bus physical network architecture for MAP. Technical and Office Protocol (TOP) is the companion specification designed for the office environment. IEEE 802.3 specifies a carrier-sense, multiple access with collision detection (CSMA/CD) physical network architecture for TOP [2].

At present, MAP V3.0 is the most recent revision of the specification, however many MAP V2.1 networks are still in use. The most significant difference between MAP V2.1 and MAP V3.0 is the network addressing scheme. Since the Presentation Layer is not implemented in the MAP V2.1 protocol stack, the final OSI address is called the Session Address. MAP V3.0 has a Presentation Layer, so the final OSI address is called the Presentation Address [F/MAP Manual]. Under MAP V2.1, an Instance Number was used to specify the particular device with which communication was desired. MAP V3.0 does not use instance numbers. The particular device is identified by setting variables in the node address, not by calling a function as was required under MAP V2.1 [3].

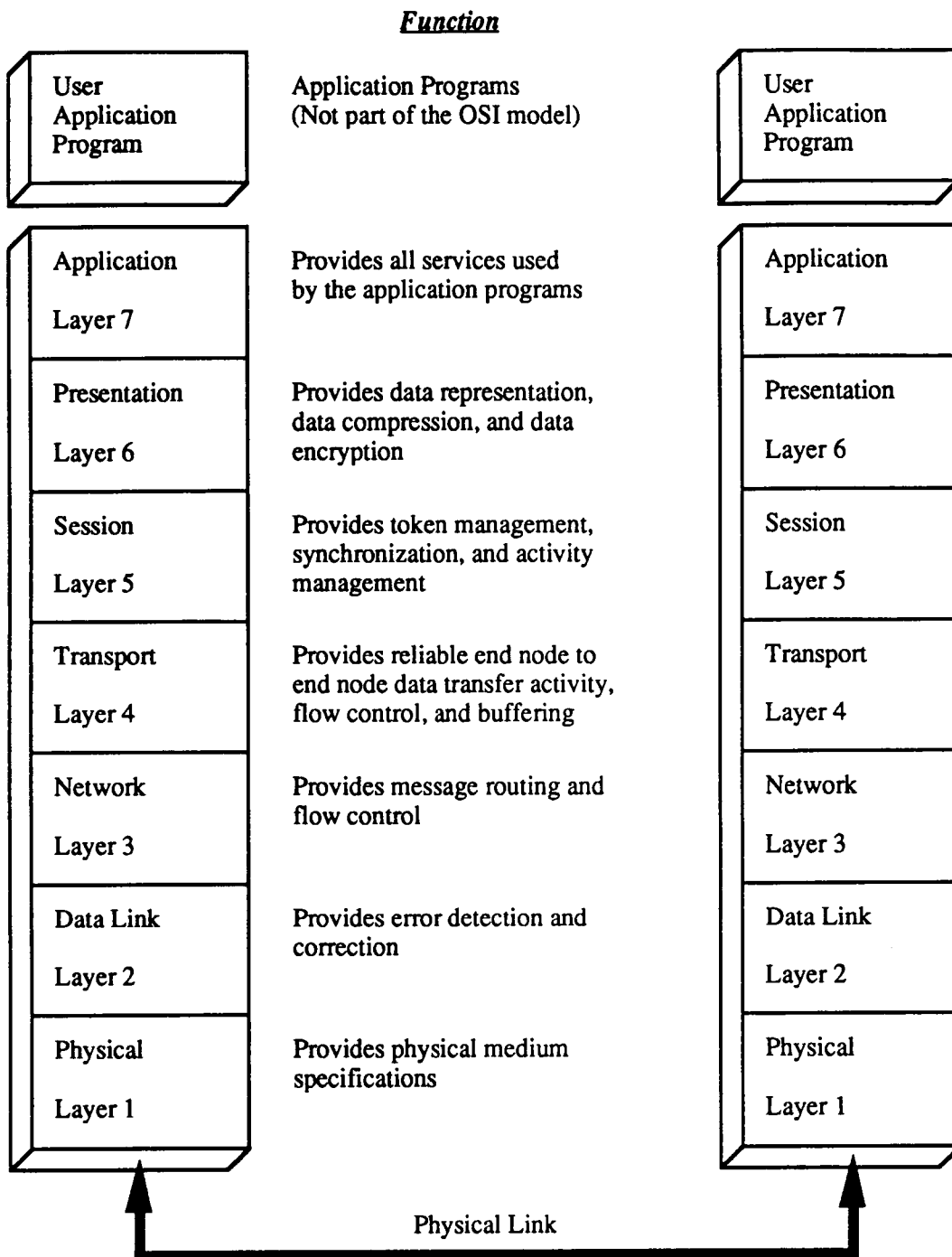


Figure 1-1. ISO/OSI Seven Layer Reference Model¹.

¹MAP Specification Versions 2.1A & 2.2.
(Michigan: General Motors Technical Center, 1986), p. 3-27.

1.2 Manufacturing Message Specification

Manufacturing Message Specification (MMS) is a communications application that resides in the application layer of the ISO/OSI Reference Model. The protocol is designed for passing messages between machines, not between humans. Without MMS, custom software would have to be written above the MAP application layer to represent a task in the proprietary command language of the programmable device. MMS eliminates the need for the customization and conversion of proprietary command languages. Vendors must adhere to the companion standard for a particular type of device, such as a Programmable Logic Controller (PLC) or Robot, to ensure that all similar devices from all vendors interpret the same commands in the same manner. Since this thesis involves communication with a PLC, the services given in the Programmable Controller Message Specification IEC/SC65A/WG6/TF companion standard were used [4]. MMS allows a programmer to concentrate on application programming instead of communication programming [5].

MMS services are inherently asymmetric in their behavior. One user plays the "client" role and requests another device, the

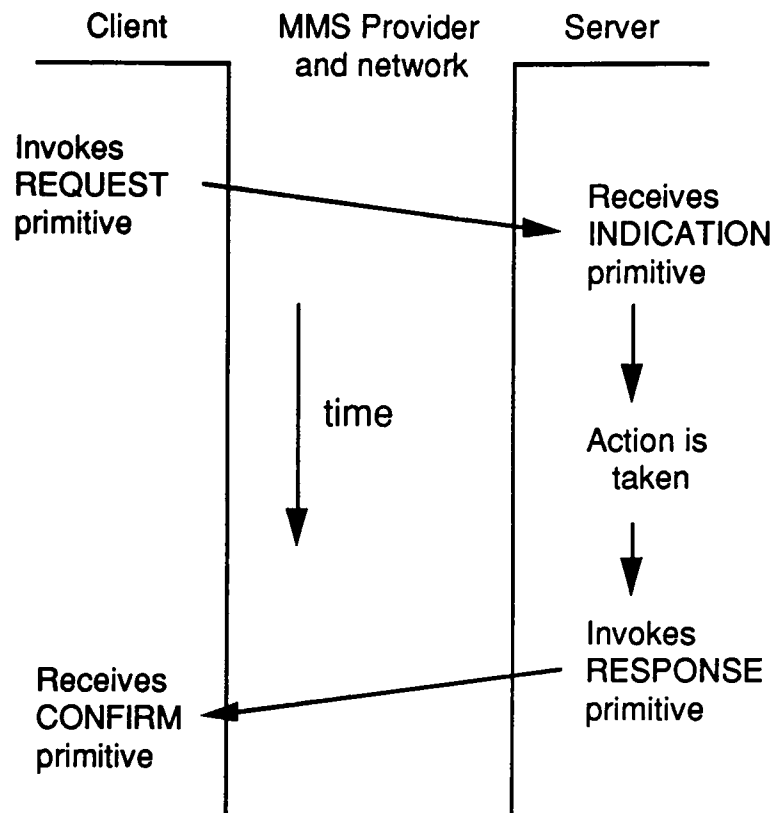


Figure 1-2. MMS Client/Server Relationship

"server", to perform an operation. Figure 1-2 shows a typical MMS transaction. MMS-EASE (Manufacturing Message Specification - Embedded Application Service Element) is a product from Systems Integration Specialists Company based on Revision 7 of the MMS Draft International Standard ISO DIS 9506. The four MMS services used in this project are:

Initiate Used to establish an application association (a connection at the application layer) with another MMS user and to negotiate the attributes of that association, such as maximum message size.

Note: This service must be requested and responded to before any other service is allowed.

Conclude Used to end an association without any loss of data.

Read Used to read the value of a variable from a remote MMS user.

Write Used to write a new value of a variable to a remote MMS user.

1.3 Connecting Existing Networks to MAP

Although MAP will eventually allow vendor independence for users, a problem still exists with the interconnection of proprietary networks. This thesis addresses the specific problem of interfacing Allen-Bradley's Data Highway proprietary network to MAP using products that are commercially available at the present time. The sequence of events that were followed to arrive at a workable solution is shown in Figure 1-3. Figure 1-4 is a block diagram of the system.

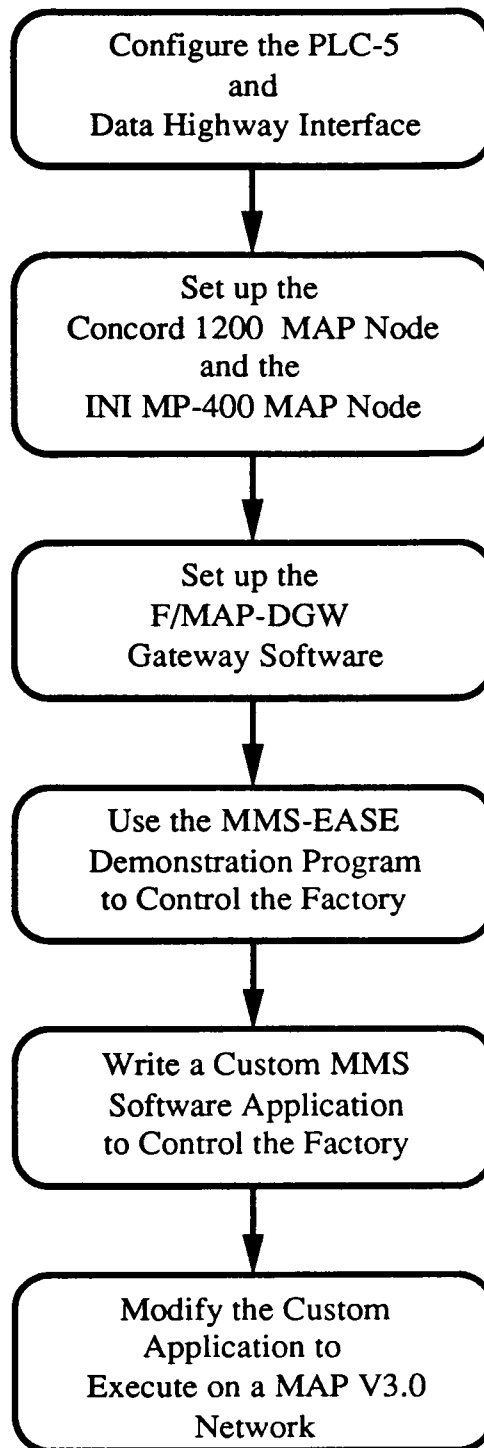


Figure 1-3. Project Work Flow.

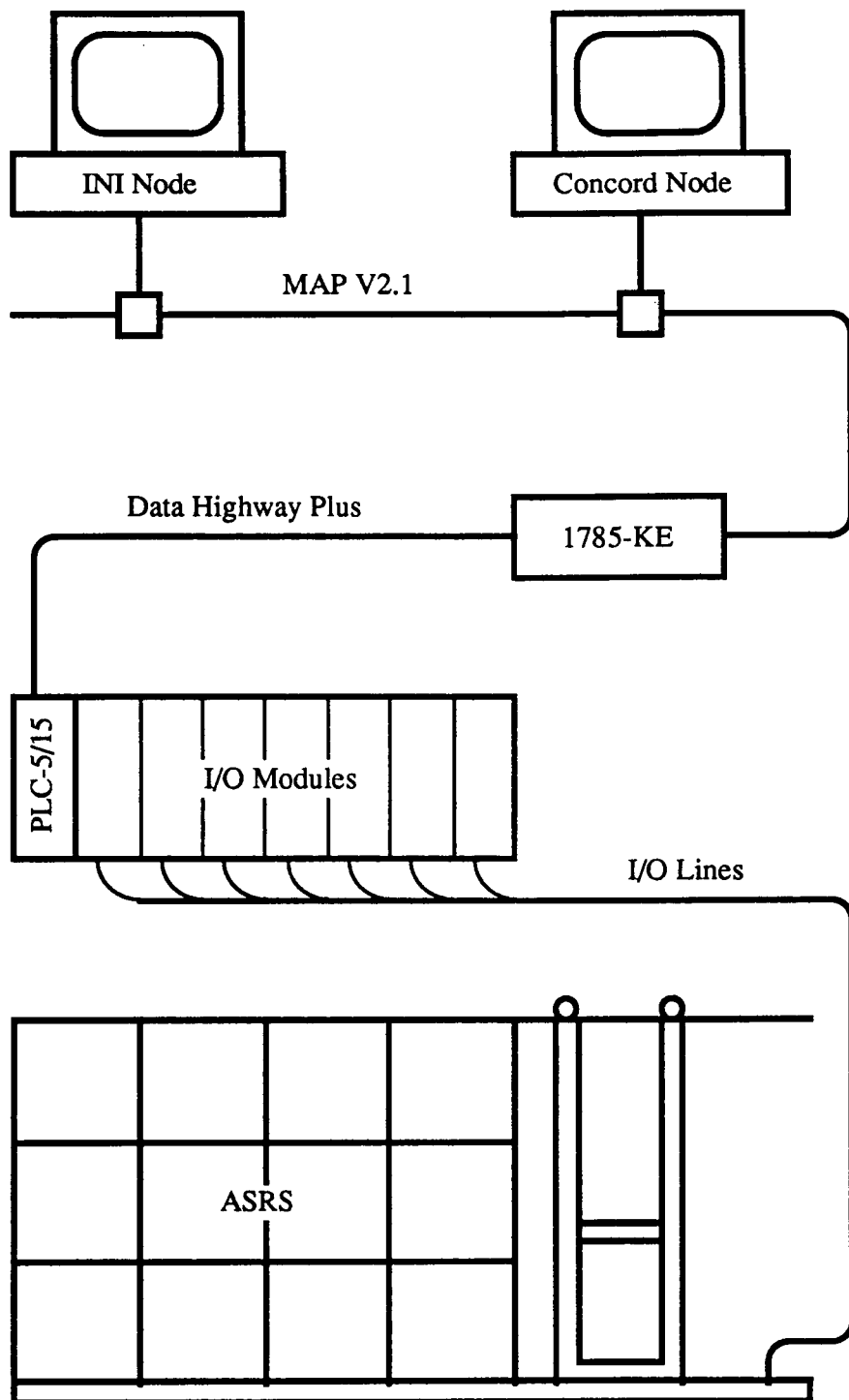


Figure 1-4. System Block Diagram.

CHAPTER 2

THE ALLEN-BRADLEY PLC-5/15 AND DATA HIGHWAY

2.1 The UTK MAP Laboratory

The University of Tennessee, Knoxville, MAP Laboratory includes an Allen-Bradley PLC-5/15 Programmable Logic Controller which operates a model factory. PLC input and output modules sense and control the factory's 49 limiting switches, 5 proximity switches, 6 pairs of infrared emitting detecting sensors, and 34 DC motors. The Allen-Bradley PLC configuration includes:

- ** T45 Portable Terminal with 6200 Programming Software.
- ** I/O chassis with power supply.
- ** 1771-IG TTL input modules.
- ** 1771-OW Switchable Contact output modules.

The Fisher Technik model factory is comprised of a warehouse section, conveyor belts, and various machine tools. Ladder logic programs to operate the various parts of the factory were written for a previous thesis [6]. The programs are downloaded from the T45 Programming terminal to the PLC-5/15 [7]. For example, the program for the Automated Storage and Retrieval System (ASRS) warehouse, which is a part of the factory that is controlled by the Allen-Bradley PLC, can be loaded to retrieve or replace pallets in the warehouse cells. Figure 2-1 gives a view of the ASRS warehouse.

2.2 The ASRS Warehouse PLC Program

A short program named "Homeboy" (Appendix A) was written to move the elevator car to a known position before executing the ASRS program. The purpose of this program was to move the elevator car to the home position, retract the beginning ram, and lower the lift conveyor if it happened to be elevated. If the 'transfer table retracted' limit switch was not activated, indicating the transfer table was extended into a cell, the elevator car would not move until

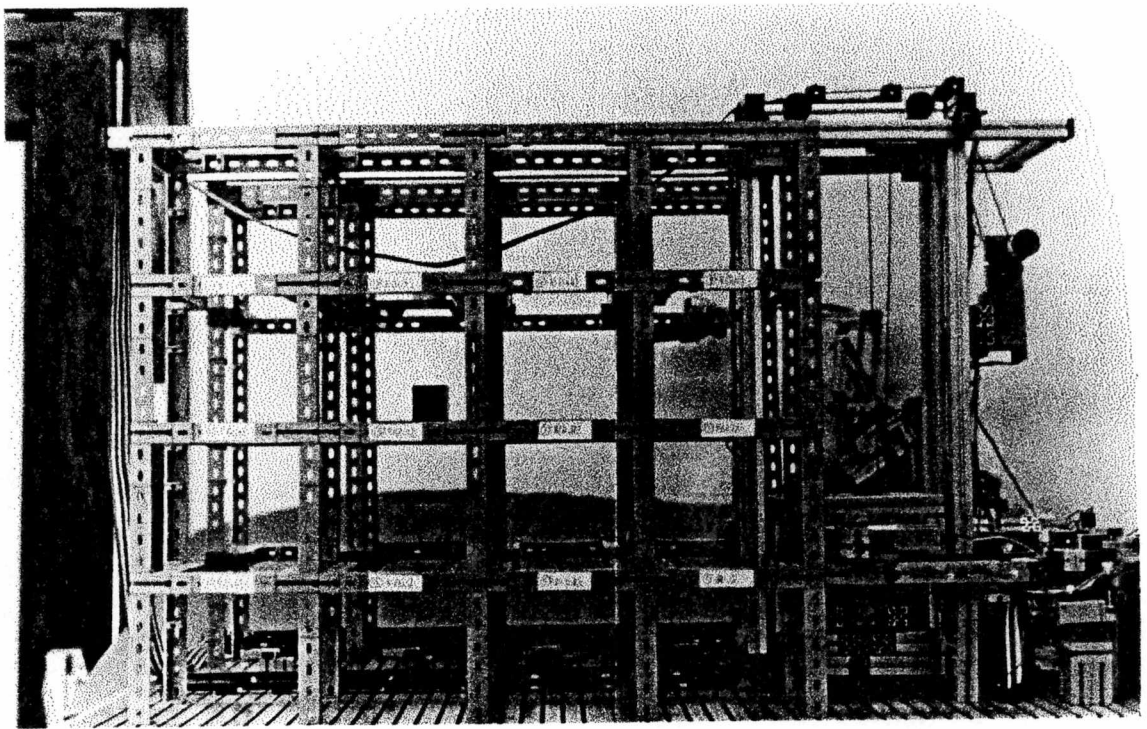


Figure 2-1. View of the model factory ASRS warehouse.

the operator, using the T45 Programming Terminal, manually set a flag indicating the direction in which the transfer table was extended. After the flag was set, the transfer table would retract, and the elevator car would move to the home position.

The ASRS was equipped with a warehouse that was composed of two storage structures, an elevator and its carriage, a telescopic transfer table, and the loading/unloading stations. Each half of the warehouse had 12 storage cells, forming a 3-row by 4-column matrix. Each cell used a pallet to provide a movable base for storing material.

The elevator, which was located between the two storage structures, used a telescopic transfer table to load/unload pallets onto the elevator car and onto the loading/unloading stations.

To operate the ASRS program, the user was required to write data to the following addresses using the T45 Programming Terminal:

input_reg Data Table Address N7:0. This is the warehouse cell to use for the take-out or put-back operation. The two-digit value consists of a row and column number with the row number being the most significant.

- b_side** Data Table Address N7:7. This is a flag to indicate the "beginning" side of the warehouse. The single-digit value is set to one if the beginning side is used and zero otherwise.
- e_side** Data Table Address N7:8. This is a flag to indicate the "ending" side of the warehouse. The single-digit value is set to one if the ending side is used and zero otherwise.
- take_out** Data Table Address N7:5. This is a flag to indicate the operation to be performed is "take-out". The single-digit value is set to one if the operation is take-out and zero otherwise.
- put_back** Data Table Address N7:6. This is a flag to indicate the operation to be performed is "put-back". The single-digit value is set to one if the operation is put-back and zero otherwise.

Note: The Data Table Address is given in the Allen Bradley PLC-5/15 format. For example, N7:0 means word zero in integer table seven [7].

The data must be written in a particular order: (1) input_reg; (2) b_side or e_side; and (3) take_out or put_back [6].

2.3 The Data Highway and Data Highway Plus Networks

At the present time, Allen-Bradley does not support the MAP network. All communication between Allen-Bradley products takes place over the Data Highway network and, in the case of the PLC-5/15, over the Data Highway Plus network. Data Highway and Data Highway Plus are local area networks (LAN's) that operates in half-duplex mode at a rate of 57.6 kilobaud. A Data Highway Link consists of a twisted, shielded pair trunkline up to 10,000 feet long and droplines that can be up to 100 feet from node to device. The Data Highway LAN can support up to 64 nodes. A Data Highway Plus LAN is similar to the Data Highway LAN, however it is designed for fewer PLC's [8].

Since there is such a large installed base of Data Highway networks that will probably not be replaced in the near-term, Data Highway users could benefit greatly from a software gateway, such as Systems Integration Specialists Company's (SISCO) F/MAP-DGW, that would make Data Highway accessible as a MAP application. Once the hardware MAP interface does become available, no changes

should be required in the MMS application programs already written as long as the MMS services are properly implemented.

2.4 The 1785-KE Interface Module

Since an IBM-PC or compatible cannot communicate directly onto the Data Highway network, an interface module is required. The 1785-KE, which is supported by SISCO's F/MAP-DGW, interfaces directly in the Allen-Bradley I/O chassis [9]. Since the module does not use the standard 25-pin D shell connector for the RS-232 serial port, it was necessary to construct a custom cable to connect the computer serial port to the 1785-KE module (Appendix B, Figure B-1). The switch settings for the module are also given in Appendix B.

CHAPTER 3

TWO TYPES OF MAP NETWORK NODES

3.1 The Concord 1200 Series Node

The University of Tennessee MAP Laboratory presently has two Concord 1200 nodes. Each node consists of a Token Bus Controller (TBC) card that interfaces to the bus of the IBM-PC (in this case, an Everex Step 286 compatible) and an external broadband modem transmitting on the 6' - FM1' (83.75MHz - 89.75Mhz) MAP frequency pair [10]. Either MAP 2.1 or MAP 3.0 can be run on the node, depending on which protocol software stack is downloaded to the TBC by the user program or Mapware Utility, a Concord-provided program. Note, however, that switching between MAP 2.1 and 3.0 can cause erratic operation of the Concord node because the non-volatile RAM memory is not completely erased by a hardware reset. A method of recovering from this condition is to use the procedure developed by the author and given in Appendix C, Figure C-1.

The MMS-EASE software includes a batch file that will compile a demonstration program. Before compiling the program, the Microsoft C Compiler and MMS-EASE C libraries must be installed on the PC. This program allows operation of all services implemented by MMS-EASE through a menu system. Two files must be present on the node before starting the program (Appendix C):

SUIC.CFG This is a configuration file that lists the Application Process (AP) Titles that will be activated and registered on selected channels. The AP Title is the name of the process entity on a MAP node.

SUIC_DIB the Directory Information Base for this particular MAP node.

These two files are in addition to the stack file, which must be downloaded into the TBC board before executing the demonstration program.

3.2 The Industrial Networking Incorporated MP-400 Node

The University of Tennessee MAP Laboratory presently has six INI MP-400 nodes on line. Each node consists of a Token Bus Controller (TBC) board and a broad-band, fixed-frequency modem transmitting on the 6' - FM' MAP frequency pair. The two-board set uses two slots in the IBM-PC or compatible expansion bus. Of the six nodes, only four are available for development work. The other two are used as the:

Network Monitor	This is used to display logical ring membership in a graphical or text format.
	The network monitor can also display statistical data about network activity.

Network Management Console

This node is used to download Directory Information Base and other network information to the other MP-400 nodes.

Note: INI uses a central DIB and Concord uses a local DIB. This means that the INI network management node must transfer the MAP protocol software stack and DIB to each remote node over the MAP network, while the Concord TBC obtains this information from the PC itself.

Before running the MMS-EASE demonstration program on the INI MP- 400 nodes, the SUIC.CFG and Network Management Console (NMC) must be configured to allow communication between the MAP nodes. There is no SUIC_DIB file for the MP-400 nodes, since the directory information base information is stored at a central point, the NMC, and downloaded to the remote nodes. The Network Management Console is used to add the Concord node to the INI Global DIB. Even though the NMC does not have to download any files to the Concord node, the INI nodes need to know that it exists [11]. The procedure is outlined in Appendix C, Figure C-2. The SUIC.CFG file performs the same function as the SUIC.CFG file written for the Concord node. Only the field entries are different (Appendix C).

3.3 Communication Between the INI and Concord Nodes

The MMS-EASE demonstration program provided an ideal platform to check interoperability of various vendor's MAP nodes. Since the F/MAP-DGW gateway would only run on the Concord node, communication between the INI and Concord nodes was required for the integration of the Data Highway network onto the MAP network. The two MMS function groups used were context management and variable access. If these two functions performed as documented, then the most important functions of the gateway would be operable.

Context management is a necessity for any MMS application, since an Application Process must have an active association with another Application Process before any of the other MMS services are used. The two services most frequently used are:

Initiate	Used to establish an association between one Application Process Title and another on a selected channel.
----------	---

Conclude Used to "gracefully" end an association between two Application Processes. Note that no data will be lost when ending an association with *conclude* because a conclude must be agreed to by both nodes, whereas an abort can lose data because one node can release the association without the consent of the other node.

As an example, the steps to follow to establish an association between the INI node AP Title "Wilma" and the Concord node AP Title "concord" is given in Appendix D, Figure D-1 [5]. Note that the AP Titles are case-sensitive.

Variable access is used to read or write a variable or group of variables to/from a local node from/to a remote node. Both nodes must have all the information pertaining to a particular operation. A variable operation requires both nodes to have the definitions of the variable name and type before attempting to execute a read or write. MMS-EASE specifies fifteen standard data types, including integers, floating-point, and character strings (Appendix D). The two most commonly used variable access services are:

- Read** Allows a local node to read the value of a variable from a remote node. Both the variable name and type must be specified when accessing the variable.
- Write** Allows a local node to write a value to a variable located at a remote node. In addition to the value, the variable name and type must be specified when accessing the variable.

The MMS-EASE demonstration program is a menu-driven application that allows a user to access the MMS services supported by MMS-EASE. Several pre-defined variables and all the standard data types are present in the demonstration program. For example, the procedure to write a value to the "Air_Pressure" variable which is present in the demonstration program is given in Appendix D, Figure D-2 [5]. Note that an association must be established before attempting to use any of the variable access services.

3.4 The SISCO F/MAP-DGW Data Highway Gateway

SISCO's F/MAP-DGW product is a software package that runs on an IBM-PC or compatible meeting the following requirements:

IBM-PC/AT or compatible

MS-DOS V3.0 or higher

640K RAM minimum

360K 5.25" floppy diskette drive

The MAP node hardware must be either a Concord 1200 Series or a Computrol LP-25 and must be specified when ordering the product [3]. The gateway software is delivered as an executable module; no source code is provided by SISCO. The programs provided are:

CONCDIR.EXE - SUIC_DIB creation utility

DGW21.EXE - Gateway for MAP V2.1

DDGW21.EXE - Debug Gateway for MAP V2.1

DGW30.EXE - Gateway for MAP V3.0

DDGW30.EXE - Debug Gateway for MAP V3.0

GATEWAY.CFG - Sample file

MMS_EASE.CFG - Sample file

PRE_GW.EXE - Program to check GATEWAY.CFG file

SUIC_DIB - Sample file

SUIC.DIB - Sample file

The user may choose to run either MAP V2.1 or MAP V3.0, depending on the software stack loaded in the MAP node. For this research, the initial program development was accomplished using the MAP V2.1 network. Conversion to MAP V3.0 was made and will be addressed in Chapter 5. Before actually executing the gateway program, three support files must be set up: SUIC_DIB (Appendix C), MMS_EASE.CFG, and GATEWAY.CFG (Appendix E). Once the support files have been properly configured, the gateway program can be initiated (Appendix E, Figure E-1). Minimal operator interaction is required during startup since the parameters have been set in the MMS_EASE.CFG file. The only question asked is whether to reset and reload the TBC.

CHAPTER 4

CONTROL OF THE MODEL FACTORY OVER THE MAP NETWORK

4.1 Using the MMS-EASE Demonstration Program to Operate the Model Factory

Once the gateway is running on the Concord node and the MMS-EASE demonstration program is running on an INI node, the first logical step is to establish an association. The demonstration program is connected to the gateway process. However, to read or write variables in the PLC, the demonstration program requires a connection to the PLC. Under MAP V2.1, a variable named Instance Number must be set to specify a particular task or device. Since the MMS-EASE demonstration program does not implement Instance Numbers, the association is established to the gateway by default in that it is set in the GATEWAY.CFG file. To connect to the PLC-5/15, the MMS-EASE library function

"mllp_set_rem_inst(chan,enable,instance)" must be used. This function is used to specify the Instance Number that is to be used for outgoing associate requests. Note that the channel should already have the AP Title activated and registered before calling this function. The three inputs to the mllp_set_rem_inst() function are:

SHORT chan This is the channel number to which the Instance Number is assigned.

SHORT enable This parameter indicates whether to enable or disable the Instance Number:

= 0 do not use the Instance Number

<> 0 use the Instance Number

LONG instance This is the Instance Number to be used.

The function has a SHORT return value that is equal to zero if there is no error, it is not equal to zero if there is an error [5]. To check the operation of the function, the following code segment was added to the MMSACON.C file (and the modified file was named MSKACON.C)

just after the channel was registered with the "mlp_reg_ar_name()" function (Appendix G):

```
LONG inst_nmbr;
inst_nmbr = 500;    /* this is the PLC-5 instance number */
printf ("\nInstance Number = %d", inst_nmbr);
if (!mlp_set_rem_inst(chan,1,inst_nmbr))
{
    printf ("\nInstance Number FAILED!");
    wait_msg ("");    /* wait for user to press a key */
}
```

Since the batch file MAKEMMS.BAT provided with the MMS-EASE recompiles all modules, not just the ones that have been modified, it was decided that a MAKE file was needed. MAKE is a utility provided with the Microsoft C Compiler that performs the same task as the MAKEMMS.BAT file, however it first checks to see if the source file has a more recent time and date stamp than the object file [12]. If the source file is more current (or if the object file does not exist), then the module is recompiled. For programs like the MMS-EASE demonstration, which has 22 source modules, the time savings can be considerable. The MAKE file for the demonstration program is named MARKDEMO.MAK and is listed in Appendix F. After the demonstration program was recompiled using the Microsoft

C Compiler, it was started up on the INI node. When the modified demonstration program connected to the gateway, the gateway state was:

"Channel 0 (CASE): Connected to device 0x105 (privileged)

Channel 1 (CASE): IDLE"

This indicated that the INI node was connected to the device on Data Highway Plus Network 1 Station 05, which was the PLC-5/15. The next step was to perform a variable read or write to determine if the INI node was actually connected to the PLC-5/15.

To enable the MMS-EASE demonstration program to access a variable in the PLC, it was necessary to add a named variable to the GATEWAY.CFG file. The ASRS warehouse program variable "input_reg", which had an Allen-Bradley symbolic address of N7:0, was added to the "Variable definitions" field of the GATEWAY.CFG file. The new variable was:

input_reg = /type_1/N7:0/0/

where the first field, type_1, is the variable type name (which was previously assigned as Short), the second field, N7:0, is the symbolic address, and the third field, 0, is the protection (0 = read/write, 1 = read, 2 = write). Note that the field "Number of VMD specific variables for this Device" must also be increased by one to reflect the addition of the "input_reg" variable.

The 1785-KE Data Highway interface module must be reconfigured with the following parameters to be compatible with the gateway software:

- Half-duplex
- BCC
- No parity
- No embedded responses
- Accept all messages
- Use handshaking signals
- Do not execute diagnostic commands

The corresponding switch settings on the 1785-KE are given in Appendix F.

Since the variable must also be present on the client node, the procedure outlined in Appendix F, Figure F-1 must be followed to add the "input_reg" variable type and name to the MMS-EASE demonstration program running on the INI node.

With the "input_reg" variable present on the INI MMS-EASE node and the Concord F/MAP-DGW gateway node, a variable read was executed in exactly the same manner as the MMS-EASE to MMS-EASE variable read discussed previously.

Now that variable reads and writes could be performed, the next step was to control the ASRS warehouse program of the model factory from a remote node. Up to this point, the only way to control the program was by using the 6200 Programming Software on the T45 Portable Terminal to write variables to the data table of the ASRS program loaded in PLC-5/15.

Before the ASRS program could be operated over the MAP network, the five variables input_reg, b_side, e_side, take_out, and put_back had to be added to both the gateway program and the MMS-EASE demonstration program so that they could be accessed by the MMS services.

To add the variables to the gateway program, the GATEWAY.CFG file "Variable definitions" field was modified to include the following:

```
input_reg = /type_1/N7:0/0/  
take_out  = /type_1/N7:5/0/  
put_back  = /type_1/N7:6/0/  
b_side    = /type_1/N7:7/0/  
e_side    = /type_1/N7:8/0/
```

Also the "Number of VMD specific variables for this Device" field was modified to reflect the addition of the variables.

Adding the variables to the MMS-EASE demonstration program was more involved. First, the new MMS variable type "type_1" was added to the "set_user_types()" function in the demonstration program source module U_DATA.C and the modified module was renamed MSK_DATA.C (Appendix G). Next, the MMS named variables input_reg, take_out, put_back, b_side, and e_side were added to the "set_user_vars()" function in the U_DATA.C (now named MSK_DATA.C) source module (Appendix G). With the appropriate name changes made in the MAKE file and link list, which were renamed MARKDEMO.MAK and MARKLINK, respectively, the demonstration program was recompiled.

The modified MMS-EASE demonstration program could now be used to execute the ASRS program over the MAP network. The demonstration program can be executed at any INI node that has

been registered (on the Network Management Console) to communicate with the AP Title "gateway" on the Concord node. An example procedure is given in Appendix F, Figure F-2 to perform the take-out operation on a pallet in warehouse cell twenty-two (row two, column two) on the beginning conveyor side of the warehouse.

4.2 Using a Custom MMS Program to Operate the Model Factory

Now that evidence had been obtained that the model factory could be controlled over the MAP network using MMS, a custom program was written to isolate the operator from the tedious task of entering the data with the MMS-EASE demonstration program. The program has a logical flow that prompts the user for required information and then executes the desired operation with no further operator interaction required unless an error is encountered. The program sequence of events is as shown in Figure 4-1. The program source module, named ASRSSEQ.C (Appendix G), was implemented as a software state machine. After the initialization activities were

completed, the states were executed as shown in Figure 4-2. The state assignments are as follows:

State 1 = getting sequence of operations from user

State 2 = connecting to gateway

State 3 = writing 'input_reg' data

State 4 = writing 'b_side' data

State 5 = writing 'e_side' data

State 6 = writing 'take_out' data

State 7 = writing 'put_back' data

State 8 = waiting for PLC to finish program execution

State 9 = checking to see if there are any more ops pending

State 10 = disconnecting gateway

This source module was substituted for the MMSAPP.C module in the SISCO-provided demonstration program. Before recompiling the program, modifications were made to three source code modules to adapt them to the needs of the custom program as follows:

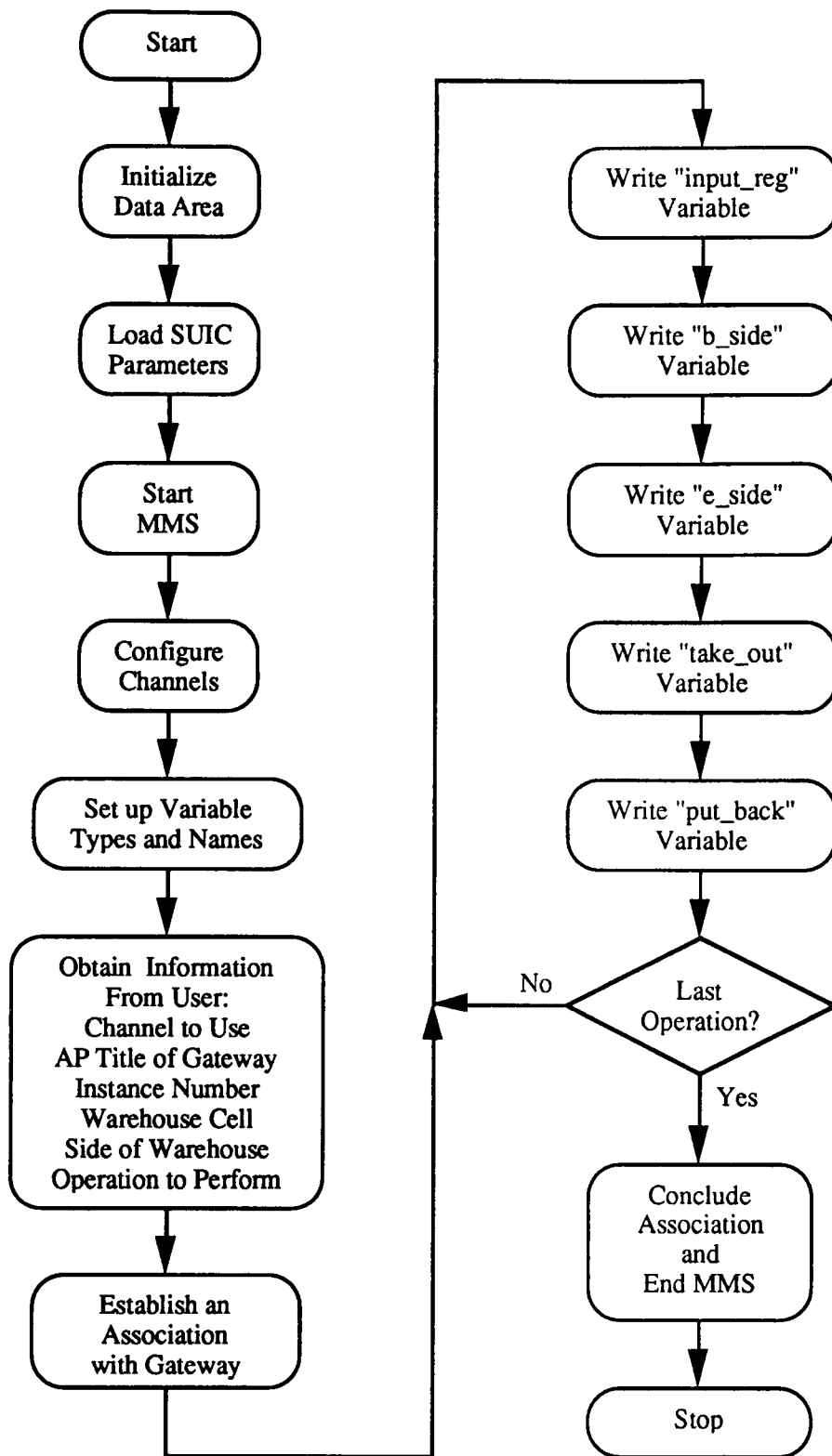


Figure 4-1. ASRSSEQ Program Flowchart.

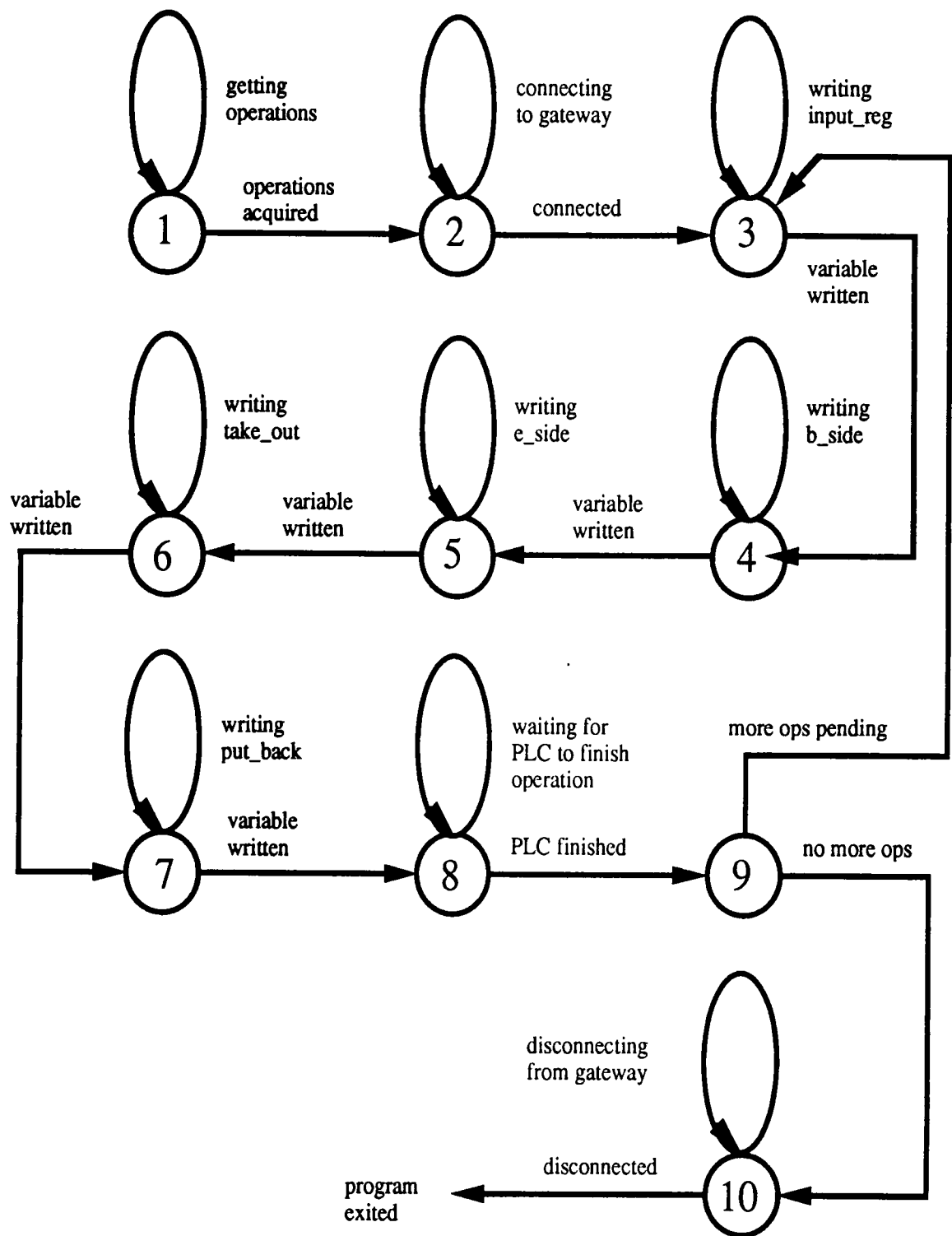


Figure 4-2. ASRSSEQ Program State Diagram.

The following three functions were deleted from the U_CONFIRM.C module, with the resulting module named MSK_CONF.C (Appendix G), and added to the ASRSSEQ.C source module:

`u_mv_init_conf()` This function is called when a confirm to a `mv_init()` is received. If an error was returned, the user is so informed, else the state variable is incremented.

`u_mv_write_conf()` This function is called when a confirm to a `mv_write()` is received. If an error was returned, the user is so informed, else the state variable is incremented.

`u_mp_conclude_conf()`

 This function is called when a confirm to a `mp_conclude()` is received. If an error was returned, the user is so informed, else the state variable is incremented.

The following three functions were deleted from the U_DATA.C module, with the resulting module named MSK_DATA.C (Appendix G), and added to the ASRSSEQ.C source module:

set_user_data()	Allocates memory for the source and destination buffers.
set_user_types()	Adds all the MMS-EASE standard variable types to a specific domain or VMD.
set_user_vars()	Adds the specific variable names, such as "input_reg" to a specific domain or VMD.

The following function was deleted from the MMSACON.C module, with the resulting module named MSKACON.C (Appendix G), and added to the ASRSSEQ.C source module:

load_local_titles()	This function reads the information from the SUIC.CFG file. The AR names from the file are activated and registered on the channels assigned.
---------------------	---

Note: load_local_titles() is renamed configure_channels() in the ASRSSEQ.C module.

A new MAKE file was written to compile the ASRSSEQ.EXE file (Appendix G). The ASRSSEQ.EXE program can be executed at any INI node that has been registered (on the Network Management Console) to communicate with the AP Title "gateway" on the Concord node.

An example procedure to perform the take-out operation on a pallet in warehouse cell twenty-two (row two, column two) on the beginning conveyor side is given in Appendix G, Figure G-1.

CHAPTER 5

RESULTS AND CONCLUSION

5.1 Results

The PLC-controlled model factory, which previously was operated manually using the programming terminal, can now be operated remotely and autonomously from a MAP network node.

The factory can be operated by three different methods:

- 1) Over the MAP V2.1 network using the modified MMS-EASE demonstration program. The modified MMS-EASE demonstration program, while tedious to use, does have the advantage of allowing the operator to see each of the individual MMS transactions required to operate the model factory.
- 2) Over the MAP V2.1 network using the custom MMS

application ASRSSEQ.EXE. The custom application presents the operator with a user-friendly function key menu system. The only two network details that the operator must be concerned with is the AP Title of the gateway process and the channel to use for the association. After this information is entered, a menu is presented.

- 3) Over the MAP V3.0 network using the custom MMS application ASRSMAP3.EXE. This custom application is functionally the same as ASRSSEQ.EXE, but a MAP V3.0 network is required.

If the user wants to observe each of the MMS transactions required to operate the model factory, then the modified MMS-EASE demonstration program should be used. If an automated execution is desired, then either ASRSSEQ or ASRSMAP3 should be used, depending on the MAP network version.

The menu presented by ASRSSEQ and ASRSMAP3 presents the user with the following choices:

<u>Function Key</u>	<u>Function</u>
F1	Reset sequence counter
F2	Enter an operation
F3	Save sequence to a file
F4	Retrieve sequence from a file
F5	Execute the sequence

F1 is used to reset the step counter if an error is made entering a sequence of operations to perform. F2 is used to enter the ASRS information required such as the warehouse row and column, which side of the warehouse, and whether to take a pallet out or put one back. F3 and F4 are used to save or retrieve respectively, a set of operations to a disk file. F5 is used to start execution of the sequence. Once F5 is depressed, assuming that the PLC is in RUN mode, the program should begin executing.

Migration from MAP V2.1 to MAP V3.0

The only modification required in the ASRSSEQ.C source module was deletion of the Instance Number code segment and the re-

compilation with the Concord MAP V3.0 libraries. The modified module was named ASRSMAP3.C (Appendix H). Additionally, MAP V3.0 uses an ASCII text-based SUIC.DIB file instead of the binary SUIC_DIB file used under MAP V2.1 (Appendix H).

The only changes required on the gateway node were: (1) creating a SUIC.DIB file, (2) modifying the MMS_EASE.CFG file to point to the MAP V3.0 stack download file, and (3) executing DGW30.EXE instead of DGW21.EXE. The SUIC.DIB must match the parameters set in the GATEWAY.CFG file to enable the remote MMS application to connect to a particular device. Since the Concord MAP nodes can run either V2.1 or V3.0, depending on which protocol stack is downloaded to the hardware, a field in the MMS_EASE.CFG file was used to set the stack file name. Once the required changes were made, the application executed exactly as it did under MAP V2.1.

5.2 Conclusion

This thesis proved that a software gateway is an effective solution to the problem of communication between a MAP network and Allen-Bradley's Data Highway Plus proprietary network. Also, the Manufacturing Message Specification (MMS) services effectively isolate the application programmer from the effects of MAP network upgrades, such as the MAP V2.1 to MAP V3.0 conversion performed during this project.

5.3 Future Work

This thesis will lead to future projects. Once a hardware MAP interface is available for the Allen-Bradley PLC-5/15, a comparison should be made. As long as the hardware interface implements the MMS services according to the PLC companion standard, the change from a software gateway to a hardware interface should be transparent to a custom MMS application running on a remote MAP

node. Also, substituting another vendor's PLC and MAP interface should have no effect on the remote MMS application, as long as the same symbolic names are used for the accessed variables.

LIST OF REFERENCES

LIST OF REFERENCES

- [1] MAP Specification Versions 2.1A, 2.2 & 3.0, General Motors Technical Center, Michigan, 1986, 1988.
- [2] Tanenbaum, Andrew S., Computer Networks, Prentice-Hall, Englewood Cliffs, N.J., 1988.
- [3] F/MAP-DGW Device Gateway User's Guide, Revision 2, Systems Integration Specialists Company, Sterling Heights, Michigan, 1989.
- [4] Programmable Controller Message Specification, International Electrotechnical Commission, Murnau, West Germany, 1989.
- [5] MMS-EASE Reference Manual, Revision 9, Systems Integration Specialists Company, Sterling Heights, Michigan, 1989.
- [6] Chung, Simon S., "Standardized Computer Communication: An Effective Alternative for the Manufacturing Environment," University of Tennessee Thesis, May 1990.
- [7] PLC-5 Family Programmable Controllers Processor Manual, Allen-Bradley Company, Cleveland, Ohio, 1987.

- [8] Data Highway/Data Highway Plus Protocol and Command Set Reference Manual, Allen-Bradley Company, Cleveland, Ohio, 1988.
- [9] 1785-KE Data Highway Plus Communications Interface Module User's Manual, Allen-Bradley Company, Cleveland, Ohio, May 1989.
- [10] ANSI/IEEE Std 802.4-1985, Institute of Electrical and Electronic Engineering, New York, New York, 1985.
- [11] NMC-40 Network Management User's Manual, Industrial Networking Incorporated, Santa Clara, California, May 1987.
- [12] Codeview and Utilities Software Development Tools, Microsoft Corporation, Redmond, Washington, 1987.

APPENDICES

APPENDIX A

PLC-5/15 PROGRAM

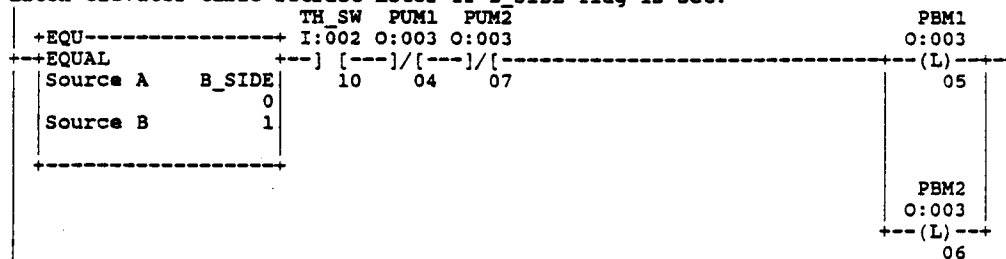
Allen-Bradley Co.
6200 Series Software
PLC-5 Programming Terminal Software
Release 2.2
Program Listing

HOMEBOY

Processor File: HOMEBOY.ACH
2 November 1990 - 10:51

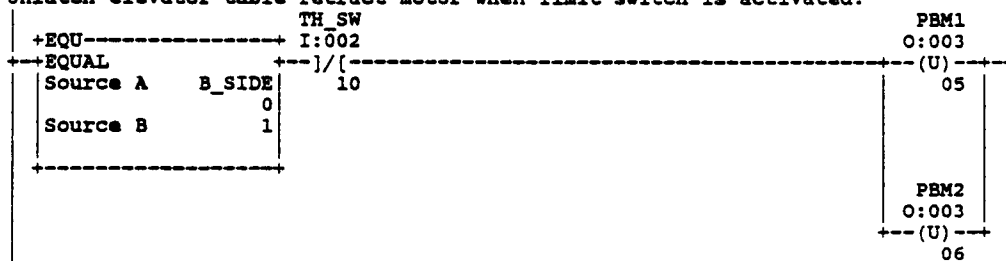
Rung 2:0

Latch elevator table retract motor if B_SIDE flag is set.



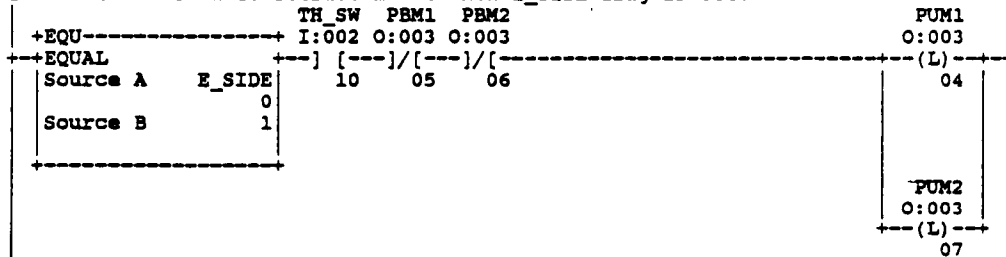
Rung 2:1

Unlatch elevator table retract motor when limit switch is activated.

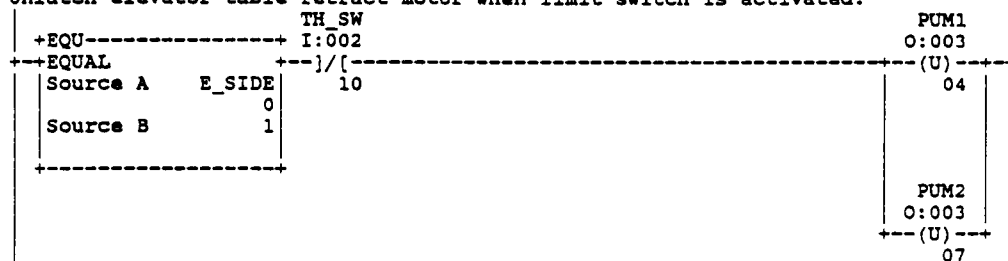


Rung 2:2

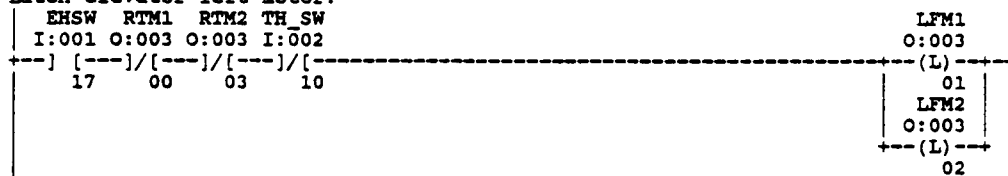
Latch elevator table retract motor when E_SIDE flag is set.



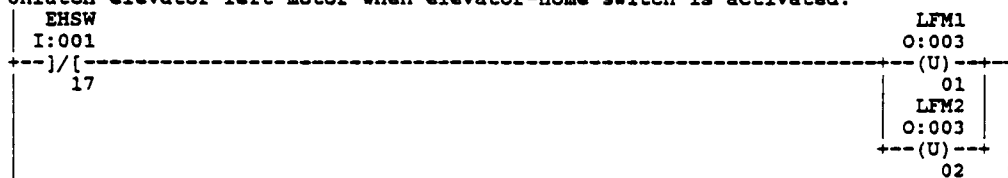
Unlatch elevator table retract motor when limit switch is activated.



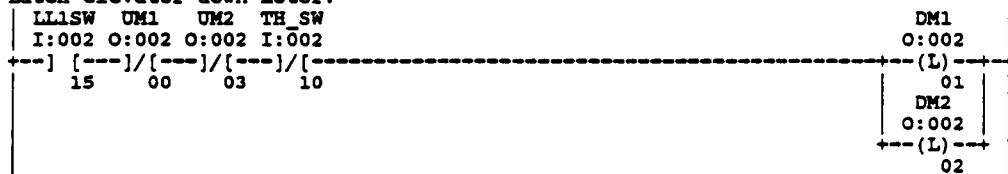
Latch elevator-left motor.



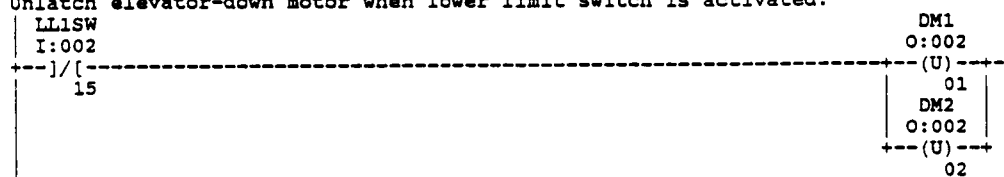
Unlatch elevator-left motor when elevator-home switch is activated.



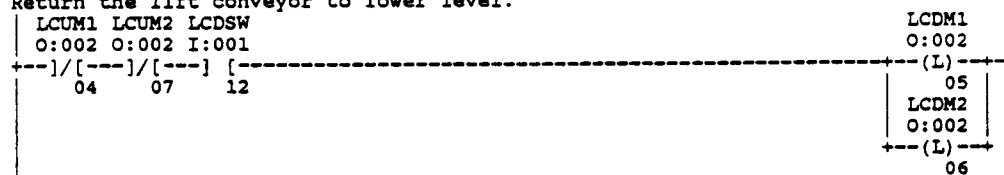
Latch elevator-down motor.



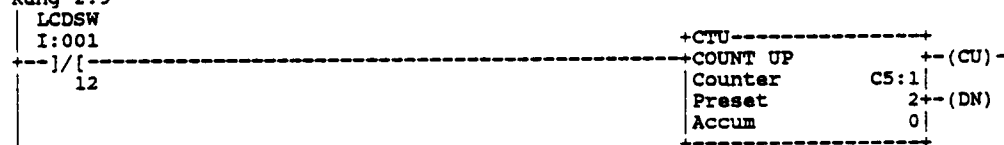
Rung 2:7
Unlatch elevator-down motor when lower limit switch is activated.



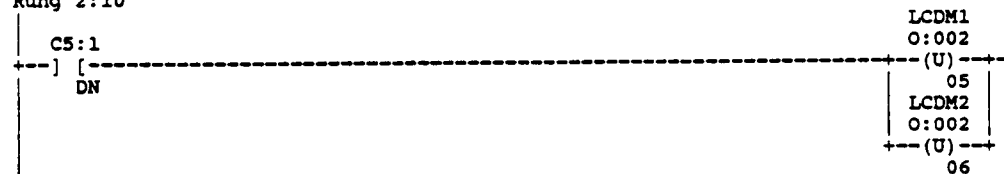
Rung 2:8
Return the lift conveyor to lower level.



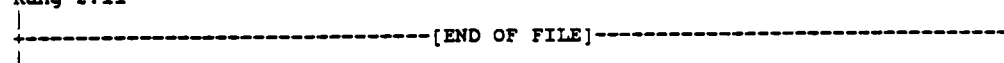
Rung 2:9



Rung 2:10



Rung 2:11



HOMEBOY
Ladder Listing

Processor File: HOMEBOY.ACH

2 November 1990

Page 4
Summary

REPORT OPTIONS SUMMARY

Page Width:	80
Page Length:	66
Graphics Capabilities:	NO
Right Power Rail:	YES
Address Comments:	YES
Symbols:	YES
Rung Comments:	YES
Cross References:	NO
Starting Rung:	2:0
Ending Rung:	2:32767

APPENDIX B

DATA HIGHWAY INTERFACE

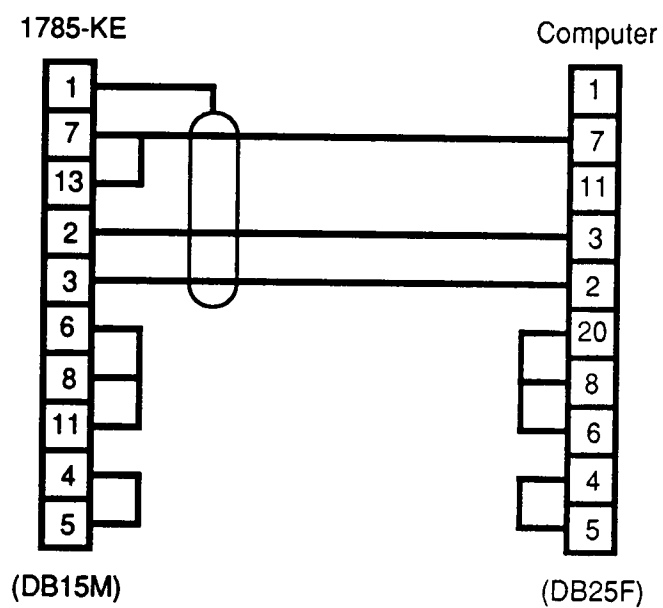


Figure B-1. 1785-KE Interface Module Cable Diagram.

The switch settings for the 1785-KE module when running the Allen-Bradley 6200 Programming Software are as follows:

<u>SW-1</u>	<u>Setting</u>	<u>Purpose</u>
1	up	full-duplex, CRC, no parity
2	dwn	"
3	dwn	"
4	up	accept all messages
5	up	ignore handshaking signals
6	dwn	execute diagnostic cmds

SW-2

1	dwn	DH+ node address (10 octal)
2	dwn	"
3	dwn	"
4	dwn	"
5	up	"
6	dwn	"
7	dwn	"
8	dwn	"

SW-3

1	dwn	DH+ data rate (57.6 kbaud)
2	dwn	"
3	up	RS-232 data rate (9600 bd)
4	dwn	"
5	dwn	"
6	dwn	local mode

SW-4

1	up	reserved for future use
2	up	"
3	up	"
4	up	"

APPENDIX C

MAP NODE SUPPORT

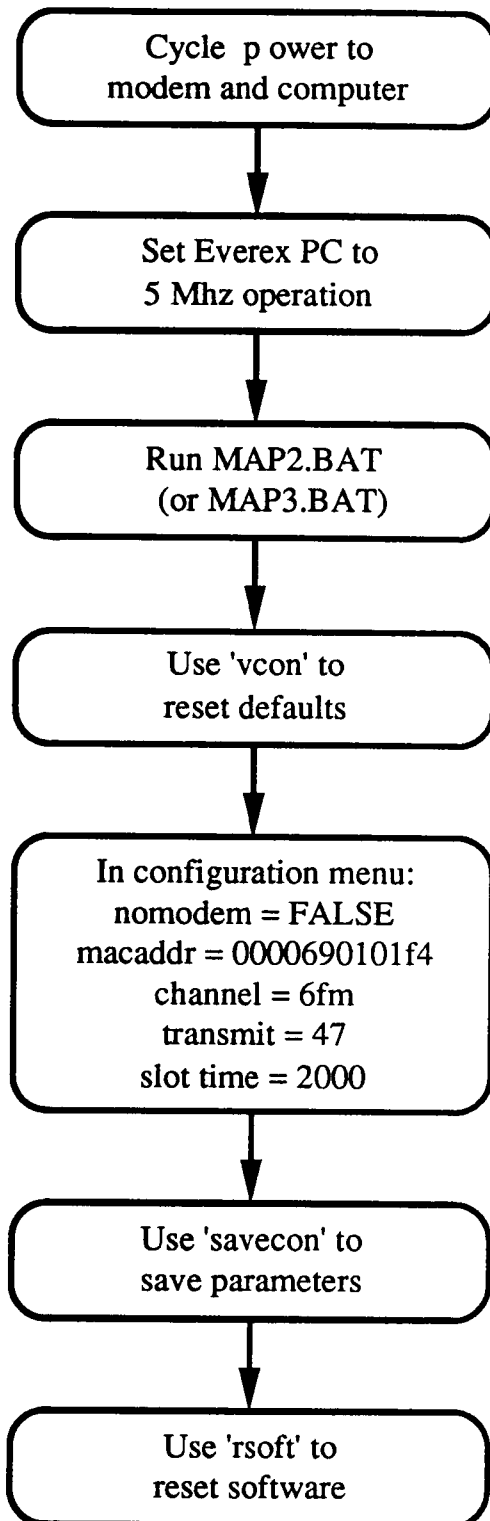


Figure C-1. Concord MAP Protocol Stack Loading Procedure.

SUIC.CFG FOR CONCORD MMS-EASE MAP V2.1 DEMO

#1 AR Names to be activated

concord

gateway

#2 Names to be register on channels

concord

gateway

#3 Channels to be listening

concord

gateway

#4 End of file

SUIC_DIB For Concord MMS-EASE MAP V2.1 Demo

Note: SUIC_DIB is a binary file that must be generated with a utility named CONCDIR.EXE provided with the F/MAP-DGW gateway software. The following four names are entered:

AP Name: 2axis
SSAP: 34
TSAP: 0001
AFI: 49
MAC ADDR: 080060002760
NID: 00
NSAP: 00
LSAP: FE
PSI: 01

AP Name: 3axis
SSAP: 36
TSAP: 0001
AFI: 49
MAC ADDR: 080060001A03
NID: 00
NSAP: 00
LSAP: FE
PSI: 01

AP Name: concord
SSAP: 38
TSAP: 0001
AFI: 49
MAC ADDR: 0000690101F4
NID: 00
NSAP: 00
LSAP: FE
PSI: 01

AP Name:	gateway
SSAP:	42
TSAP:	0001
AFI:	49
MAC ADDR:	0000690101F4
NID:	00
NSAP:	00
LSAP:	FE
PSI:	01

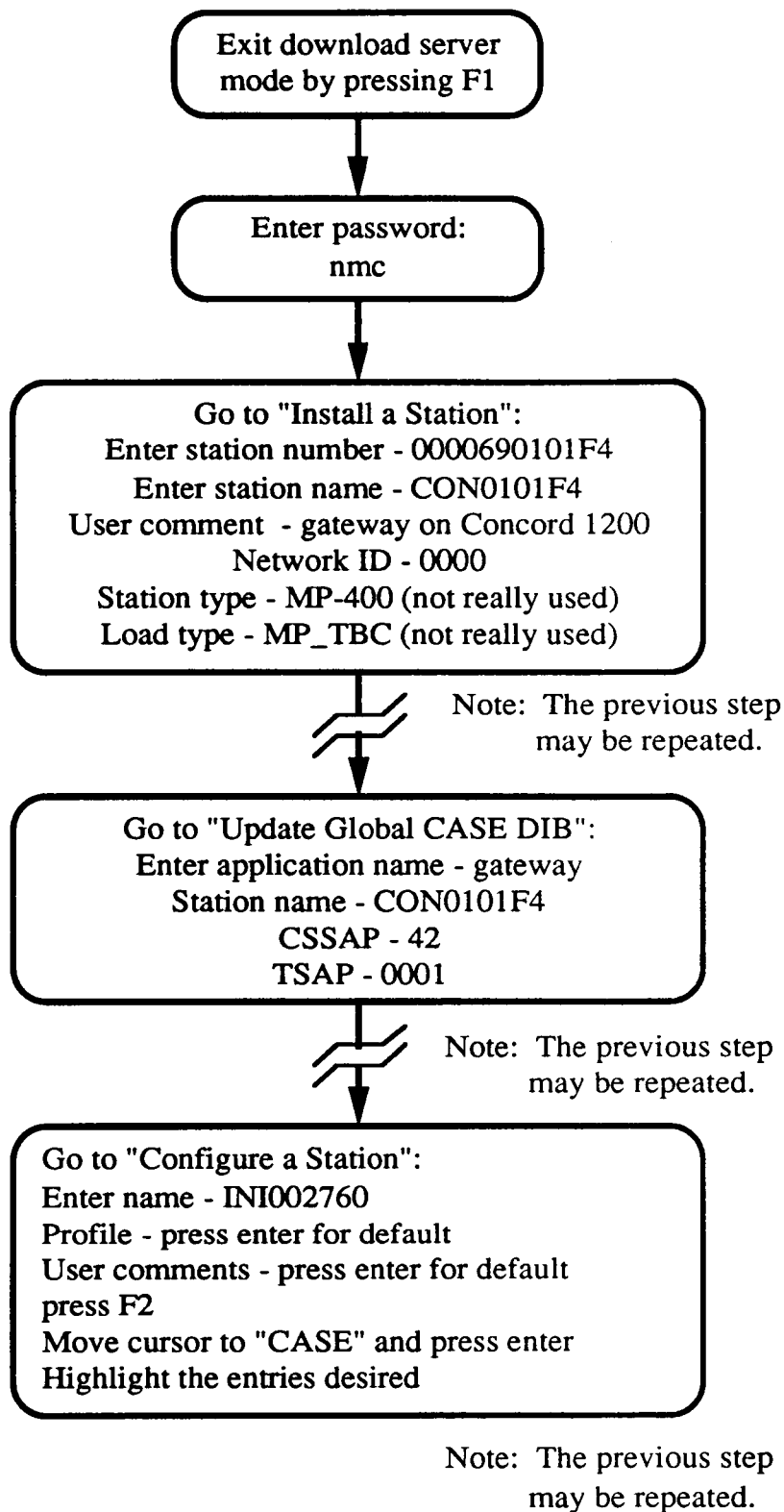


Figure C-2. INI Global DIB Setup Procedure.

SUIC.CFG For INI MP-400 MMS-EASE MAP V2.1 Demo

#1 AR Names to be activated

Wilma

#2 Names to be register on channels

Wilma

#3 Channels to be listening

#4 End of file

APPENDIX D

MAP V2.1 MMS-EASE SUPPORT

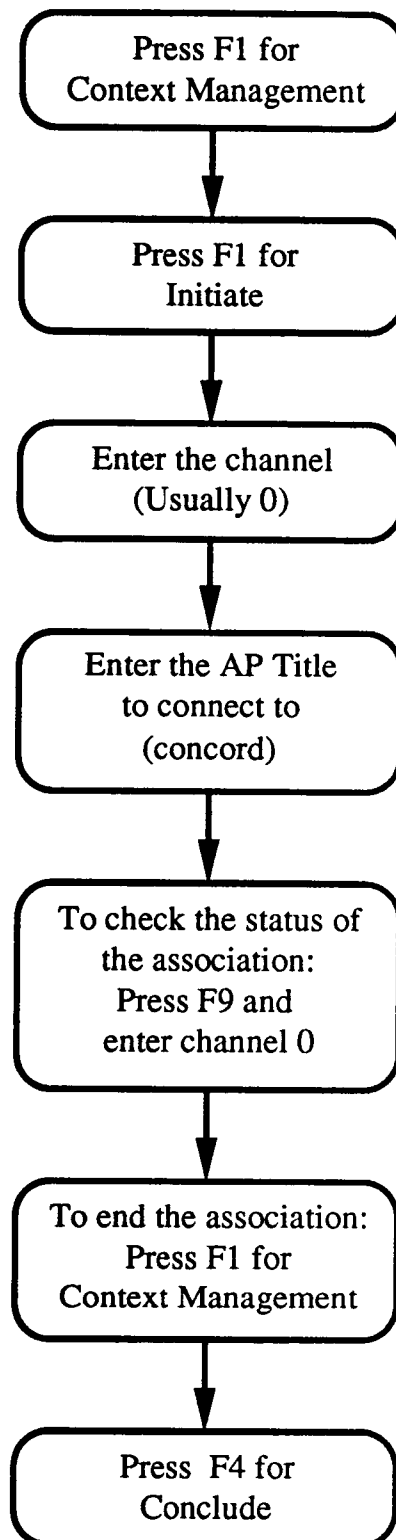


Figure D-1. Procedure for Initiating and Concluding an Association.

MMS-EASE Standard Types

Integer8	Signed 8-bit integer
Integer16	Signed 16-bit integer
Integer32	Signed 32-bit integer
Unsigned8	Unsigned 8-bit integer
Unsigned16	Unsigned 16-bit integer
Unsigned32	Unsigned 32-bit integer
Boolean	Unsigned 8-bit True or False
Float	Machine dependent floating point
Double	Machine dependent double precision
String8	9 sequential bytes
String16	17 sequential bytes
String32	33 sequential bytes
String64	65 sequential bytes
String128	129 sequential bytes
String256	257 sequential bytes

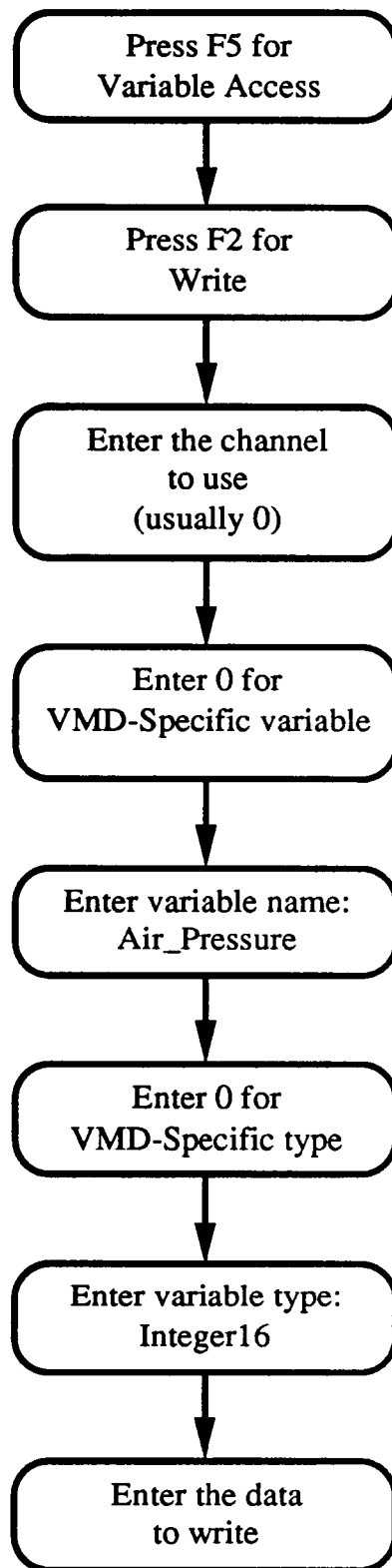


Figure D-2. MMS-EASE Demo Variable Write Procedure.

APPENDIX E

GATEWAY SUPPORT

MMS EASE.CFG for MAP V2.1 Gateway

BootTalkEnable	1
HWreset	1
LoadFile	C:\CONCORD\MAP2X.DLD
DIBfile	C:\CONCORD\SUIC_DIB
NumberOfChannels	2
BaseAddress	300 (TBC)
DMAchannel	3 (TBC)
IRQlevel	3 (TBC)

GATEWAY.CFG for Gateway

```
#####  
# Gateway configuration  
#####
```

```
#####  
$ Keyboard Option  
#      0 = Disabled  
#      1 = Enabled
```

1

```
#####  
$ Security Mode  
#      0 = Protected  
#      1 = Non-protected
```

1

```
#####  
$ Connection option  
# This option determines whether the Gateway will automatically  
# connect the client to the Gateway if an unrecognized AR name is  
# received on a Associate Request or whether it will send a Reject.  
#  
#      0 = Reject  
#      1 = Gateway Connection
```

1

```
#####  
$ Gateway Listen AR name  
#####
```

gateway

```
#####  
$ Gateway Privileged AR Name  
# Should be used if privileged services are required
```

/1 2 3 4 5//1//

```
#####
```

\$ Gateway Non-Privileged AR Name
Should be used if no privileged services are required

/1 2 3 4 5//2//

\$ Gateway Privileged instance number
Should be used if privileged services are required

100

\$ Gateway Non-Privileged instance number
Should be used if no privileged services are required

200

\$ Gateway Password
This is used in protected mode

gateway

\$ Number of Data Highways supported
#####

1

\$ Standard Types Mask
#####

7fff

\$ Asynchronous Board Type

0 = Standard COM1/COM2
1 = DigiBoard

0

\$ Asynchronous Board Specific Parameters

#####

```

# Data Highway configurations
#####

#####

# Data Highway 1
#####

#####
$ TYPE:
#       0 = Allen-Bradley Data Highway One
#       1 = Allen-Bradley Data Highway Plus
#       2 = MODBUS

1

#####
$ Port Configuration
# Format:
#   COM# baud parity byte_size stop_bits
# If 0 is used for any of this fields, a default value will be
# used.

1 7 0 1 1

#####
$ Network-Specific Configuration
#
# This section is used to configure certain parameters used by the
# network driver.
#
# For Data Highway, the following items are defined:
#
#   Addressing Mode      (0 = Local, 1 = Remote)
#   Network Adapter Address (Mandatory if Addressing Mode = 0)
#   Message Retry Count  (Optional default = 3)
#
# Format:
#       /addr_mode/adaptor_addr/retry_count/
#
# Note! "adaptor_addr" entry MUST BE DECIMAL (not octal)

/0/8/100/

#####
$ Gateway source address (1 - 254) for this Data Highway
# Data Highway station number

```

8

\$ Number of devices supported

1

Device configuration #
#####

Device #1
#####

\$ Device type:
0 = PLC-2
1 = PLC-3
2 = PLC-5

2

\$ Device Destination Address (1 - 254)
Data Highway Station Number

5

\$ Device Privileged AR Name
Should be used if privileged services are required

/1 2 3 4 8//1//

\$ Device Non-Privileged AR Name
Should be used if no privileged services are required

/1 2 3 4 8//2//

\$ Device Privileged instance number
Should be used if privileged services are required

500

\$ Device Non-Privileged instance number
Should be used if no privileged services are required

600

\$ Device Password
This is used in protected mode

plc5

\$ MMS Control Block address (Not Implemented)
If section is empty, MMS control block is assumed to be
non-existent, i.e. no services described in Chapter 8
may be performed.

\$ How often to check MMS Control Block for PLC commands (in sec)
(Not Implemented)

12

\$ Number of non-MMS services supported for this device (Not Implemented
See Chapter 9 for PLC-5)

6

\$ Non-MMS services supported for this device
PLC-2:
0 - Diagnostic counters reset
1 - Diagnostic loop
2 - Diagnostic read
3 - Set ENQ
4 - Set NAK
5 - Set timeout
6 - Set Variables
7 - Protected bit write
8 - Unprotected bit write

1

2

4

5

7
8

\$ Number of PLC-specific variable types

4

\$ PLC-specific variable types

type_1 = Short
type_2 = [16:Short]
type_3 = Float
type_4 = [2:Float]

\$ Number of Device Domains (Not Used)

2

\$ Domain definitions

Format:

/domain_name/capability_1/capability_2/./

#

where capabilities are optional

Domain 1

/plc5_image_dom/IMAGE/

Domain 2

/plc5_range_dom/RANGE 100:300/

\$ Number of VMD specific variables for this Device

8

\$ Variable definitions

Format:

var_name = /var_type_name/PLC_addr/float/protection/

#

Float information is only required for PLC-5

#

protection:

0 = READ/WRITE

1 = READ

2 = WRITE

plc5_var1 = /type_1/N7:3/0/

plc5_var2 = /type_3/F8:0/0/

input_reg = /type_1/N7:0/0/

dat_1 = /type_1/N7:1/0/

take_out = /type_1/N7:5/0/

put_back = /type_1/N7:6/0/

b_side = /type_1/N7:7/0/

e_side = /type_1/N7:8/0/

#####

\$ Number of variable lists for this device (Not Used)

0

#####

\$ Variable lists

Format: var_list_name = /var_name1/var_name2/...

#####

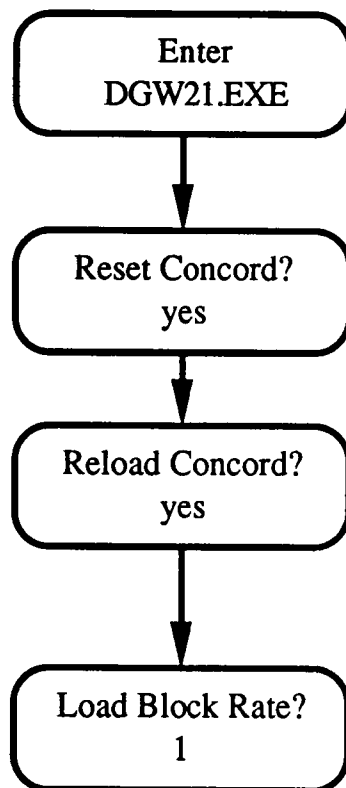


Figure E-1. Gateway Startup Procedure.

APPENDIX F

MAP V2.1 MMS-EASE DEMONSTRATION PROGRAM

MARKDEMO.MAK

mskapp.obj : mskapp.c mmsop_en.h

cl /c /Osi /G2 /DMP_400 /DDEBUG_MMS /DLINT_ARGS /DMAP21_CASE /W2 /AL
mskapp.c ;

mmsacase.obj : mmsacase.c mmsop_en.h

cl /c /Osi /G2 /DMP_400 /DDEBUG_MMS /DLINT_ARGS /W2 /AL mmsacase.c ;

mmsamisc.obj : mmsamisc.c mmsop_en.h

cl /c /Osi /G2 /DDEBUG_MMS /DLINT_ARGS /DMAP21_CASE /W2 /AL mmsamisc.c ;

mmsallp.obj : mmsallp.c mmsop_en.h

cl /c /Osi /G2 /DDEBUG_MMS /DLINT_ARGS /W2 /AL mmsallp.c ;

mskacon.obj : mskacon.c mmsop_en.h

cl /c /Osi /G2 /DDEBUG_MMS /DLINT_ARGS /W2 /AL mskacon.c ;

mmsavmd.obj : mmsavmd.c mmsop_en.h

cl /c /Osi /G2 /DDEBUG_MMS /DLINT_ARGS /W2 /AL mmsavmd.c ;

mmsadom.obj : mmsadom.c mmsop_en.h

cl /c /Osi /G2 /DDEBUG_MMS /DLINT_ARGS /W2 /AL mmsadom.c ;

mmsaprg.obj : mmsaprg.c mmsop_en.h

cl /c /Osi /G2 /DDEBUG_MMS /DLINT_ARGS /W2 /AL mmsaprg.c ;

mmsavar.obj : mmsavar.c mmsop_en.h

cl /c /Osi /G2 /DDEBUG_MMS /DLINT_ARGS /W2 /AL mmsavar.c ;

mmsasem.obj : mmsasem.c mmsop_en.h

cl /c /Osi /G2 /DDEBUG_MMS /DLINT_ARGS /W2 /AL mmsasem.c ;

mmsaocs.obj : mmsaocs.c mmsop_en.h

cl /c /Osi /G2 /DDEBUG_MMS /DLINT_ARGS /W2 /AL mmsaocs.c ;

mmsaevn.obj : mmsaevn.c mmsop_en.h

cl /c /Osi /G2 /DDEBUG_MMS /DLINT_ARGS /W2 /AL mmsaevn.c ;

mmsajou.obj : mmsajou.c mmsop_en.h

cl /c /Osi /G2 /DDEBUG_MMS /DLINT_ARGS /W2 /AL mmsajou.c ;

mmsafil.obj : mmsafil.c mmsop_en.h

cl /c /Osi /G2 /DDEBUG_MMS /DLINT_ARGS /W2 /AL mmsafil.c ;

```

uservar.obj : uservar.c mmsop_en.h
    cl /c /Osi /G2 /DDEBUG_MMS /DLINT_ARGS /W2 /AL uservar.c ;

msk_data.obj : msk_data.c mmsop_en.h
    cl /c /Osi /G2 /DDEBUG_MMS /DLINT_ARGS /W2 /AL msk_data.c ;

u_case.obj : u_case.c
    cl /c /Osi /G2 /DDEBUG_MMS /DLINT_ARGS /W2 /AL u_case.c ;

u_indic.obj : u_indic.c mmsop_en.h
    cl /c /Osi /G2 /DDEBUG_MMS /DLINT_ARGS /W2 /AL u_indic.c ;

u_confrm.obj : u_confrm.c mmsop_en.h
    cl /c /Osi /G2 /DDEBUG_MMS /DLINT_ARGS /W2 /AL u_confrm.c ;

mmsop_en.obj : mmsop_en.c mmsop_en.h
    cl /c /Osi /G2 /DDEBUG_MMS /DLINT_ARGS /W2 /AL mmsop_en.c ;

fkey.obj : fkey.c
    cl /c /Osi /G2 /DDEBUG_MMS /DLINT_ARGS /W2 /AL fkey.c ;

getval5.obj : getval5.c
    cl /c /Osi /G2 /DDEBUG_MMS /DLINT_ARGS /W2 /AL getval5.c ;

markdemo.exe : mskapp.obj mskacon.obj mmsavmd.obj mmsadom.obj mmsaprg.obj \
    mmsavar.obj mmsasem.obj mmsaocs.obj mmsaevn.obj mmsajou.obj \
    mmsafil.obj mmsallp.obj mmsacase.obj u_case.obj mmsamisc.obj \
    uservar.obj msk_data.obj u_indic.obj u_confrm.obj mmsop_en.obj \
    getval5.obj fkey.obj
    link @marklink

```

MARKLINK Link File for MARKDEMO.MAK

mskapp +
mskacon +
mmsavmd +
mmsadom +
mmsaprg +
mmsavar +
mmsasem +
mmsaocs +
mmsaevn +
mmsajou +
mmsafil +
mmsallp +
mmsacase +
u_case +
mmsamisc +
uservar +
msk_data +
u_indic +
u_confrm +
mmsop_en +
getval5 +
fkey
markdemo
/MAP:60000 /E /ST:8196 /SEGMENTS:521 /NOD /NOE /NOI /CO
asn1d mmsd1 mmsd2 mmscased suiccase +
llibce

1785-KE Switch Settings for Gateway

<u>SW-1</u>	<u>Setting</u>	<u>Purpose</u>
1	up	half-duplex, BCC, no parity
2	dwn	"
3	dwn	"
4	up	accept all messages
5	dwn	use handshaking signals
6	up	pass thru diagnostic cmds
 <u>SW-2</u>		
1	dwn	DH+ node address (10 octal)
2	dwn	"
3	dwn	"
4	dwn	"
5	up	"
6	dwn	"
7	dwn	"
8	dwn	"
 <u>SW-3</u>		
1	dwn	DH+ data rate (57.6 kbaud)
2	dwn	"
3	up	RS-232 data rate (9600 bd)
4	dwn	"
5	dwn	"
6	dwn	local mode
 <u>SW-4</u>		
1	up	reserved for future use
2	up	"
3	up	"
4	up	"

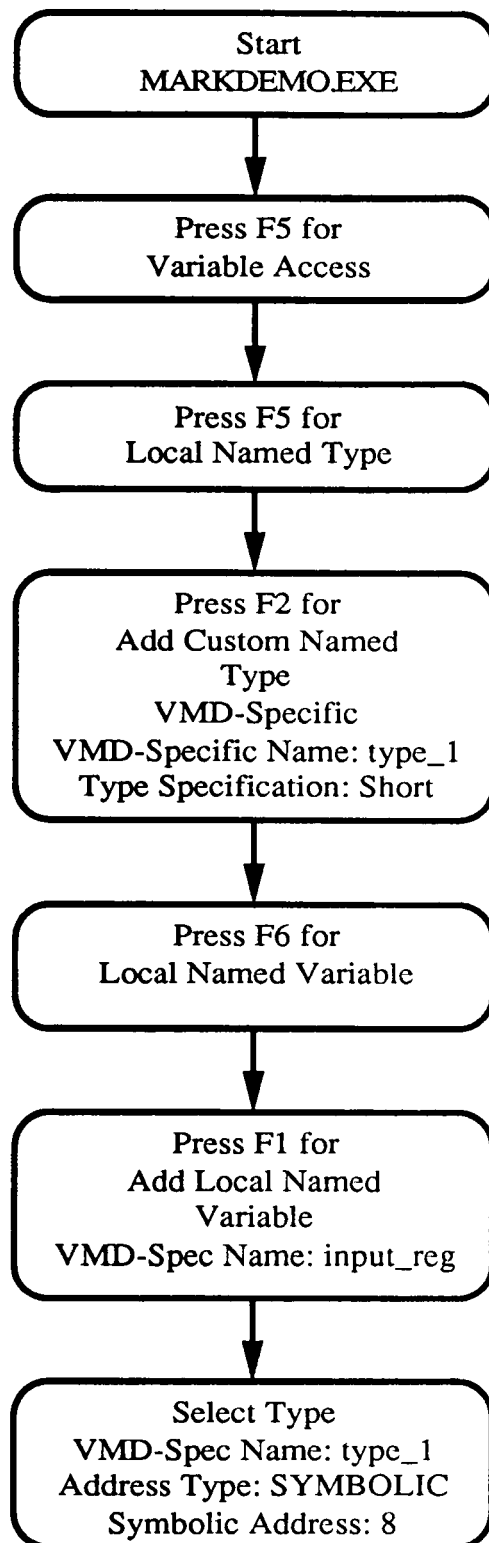


Figure F-1. Adding a New Variable to the MMS-EASE Demo.

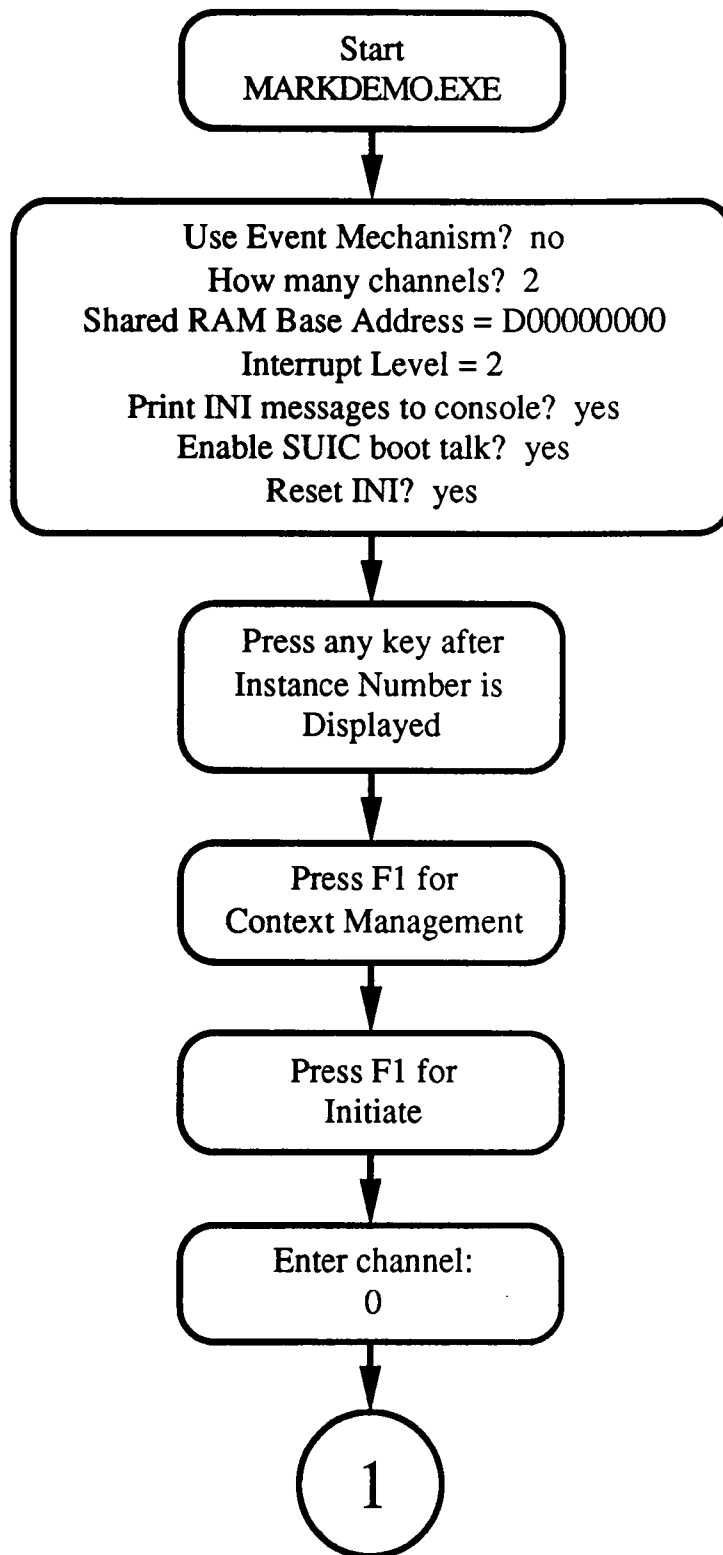


Figure F-2. Operating the Factory with the MMS-EASE Demo Program.

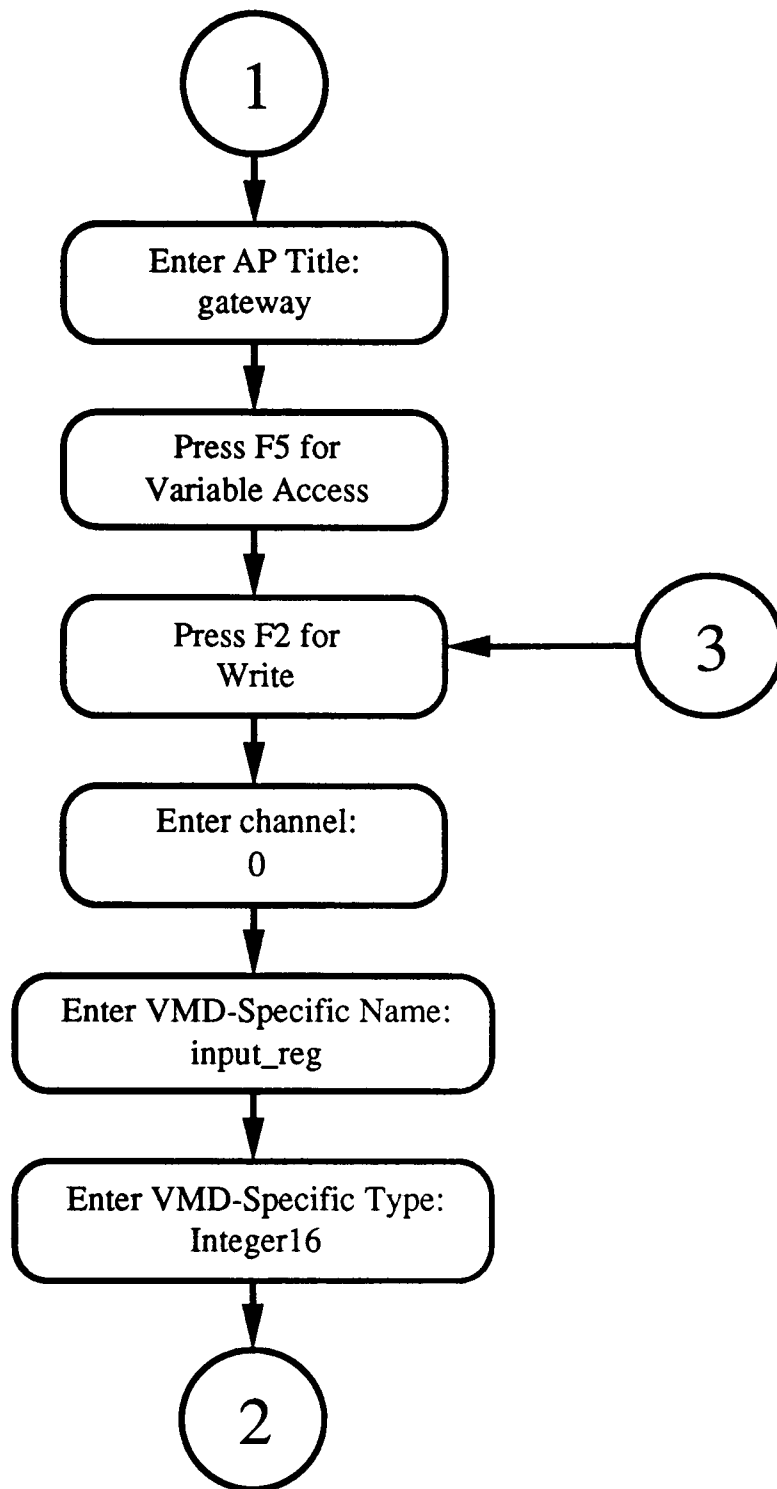


Figure F-2. Continued.

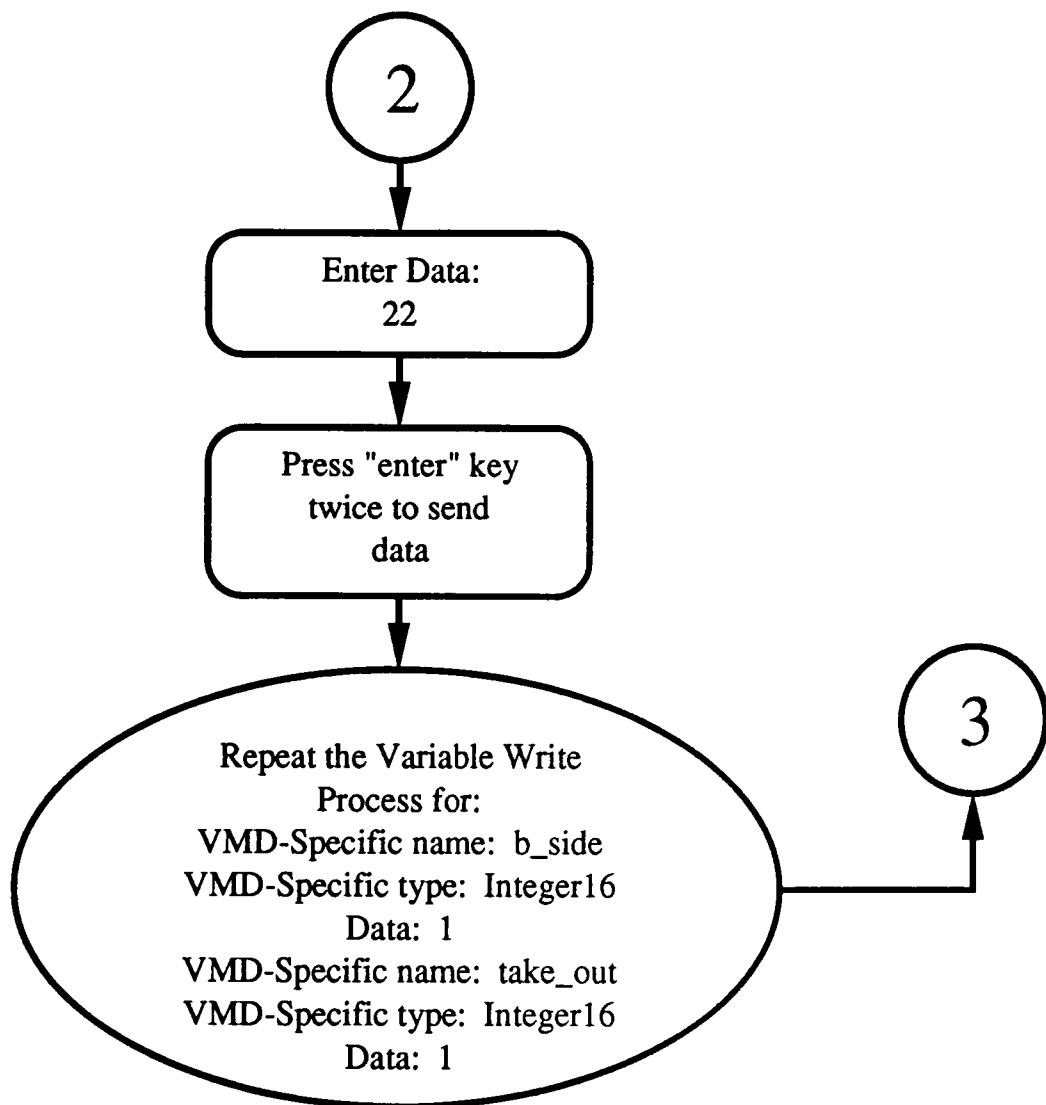


Figure F-2. Continued.

APPENDIX G

ASRSSEQ MMS-EASE PROGRAM

ASRSSEQ.C Source Code Module

```

/*****
/*  UTK MAP LAB SOFTWARE MODULE *****/
/*****
*/
/* MODULE NAME: asrsseq.c */
/* WRITTEN BY: Mark S. King */
/*
*/
/* MODULE DESCRIPTION: */
/* This module contains the 'main' function. The purpose of the */
/* module is to control the ASRS ladder logic program in the */
/* Allen-Bradley PLC-5/15. This module is a modification of the */
/* MMSAPP.C module and allows a sequence of operations to be */
/* learned, stored in a file, retrieved from a file, or executed. */

/***** Include files *****/

#include "glbtypes.h"
#include "sysincs.h"
#include "mms_user.h" /* to access MMS functions, variables*/
#include "mmsop_en.h"
#include "fkeydefs.h" /* function key handling */
#include "asn1defs.h" /* to access ASN.1 variables */
#include "userdefs.h"
#include "gvaldefs.h"
#include "scrndefs.h"
#include <dos.h>
#include <stdio.h>
#include <process.h>
#include <conio.h>
#include <stdlib.h>

/***** Constants *****/

#define F1 59
#define F2 60
#define F3 61
#define F4 62
#define F10 68

/***** Function Prototypes *****/

VOID set_user_data ( VOID );
VOID set_suic_parameters ( VOID );
```

```

VOID configure_channels ( VOID );
VOID set_user_types ( VOID );
VOID set_user_vars ( VOID );
VOID print_demo ( VOID );
VOID print_stats ( SHORT, SHORT, SHORT );
VOID get_user_info ( VOID );
VOID repeat ( VOID );

SHORT connect_gateway ( SHORT, BYTE * );
SHORT write_plc5 ( BYTE *, SHORT );
SHORT disconnect_gateway ( SHORT );

VOID u_mv_init_conf ( MMSREQ_PEND * );
VOID u_mv_write_conf ( MMSREQ_PEND * );
VOID u_mp_conclude_conf ( MMSREQ_PEND * );

/* the following functions are necessary for sequencing */

VOID get_operations ( VOID );
VOID get_chan_info ( VOID );
VOID print_menu ( VOID );
VOID get_warehouse_info ( SHORT );
VOID save_to_disk ( SHORT );
SHORT load_from_disk ( VOID );
SHORT read_plc5 ( BYTE * );
VOID u_mv_read_conf ( MMSREQ_PEND * );

/***** Global Variables *****/

SHORT state,          /* present state of the software state machine */
      working,        /* set to 1 if a fnct is working, 0 otherwise */
      chan,           /* Channel to use */
      input_reg[25],
      b_side[25],
      e_side[25],
      take_out[25],
      put_back[25],
      step,
      nmbr_steps,
      dat_1;

BYTE  title[180],
      var_name[32];

LONG  inst_nmbr;

```

```

/*****                               Main                               *****/

DEFAULT main ()
{
    SHORT RetCode,
        connect_attempts,
        write_attempts,
        disconnect_attempts;

    CLEARSCR;
    /* in scrndefs.h */
    printf ("\n...Reserving user data area");
    set_user_data();
    /* in this module */

    printf ("\n...Setting SUIC parameters");
    set_suic_parameters();
    /* in this module */

    strt_MMS();
    /* in LIB */

    printf ("\nMMS has been started...configuring channels");
    configure_channels();
    /* in this module */

    printf ("\n...Setting up user types and variables");
    set_user_types();
    /* in this module */
    set_user_vars();
    /* in this module */

    /* Initialization is now complete */

    /* The following states are used during program execution: */
    /* State 1 = getting sequence of operations from user */
    /* State 2 = connecting to gateway */
    /* State 3 = writing 'input_reg' data */
    /* State 4 = writing 'b_side' data */
    /* State 5 = writing 'e_side' data */
    /* State 6 = writing 'take_out' data */
    /* State 7 = writing 'put_back' data */
    /* State 8 = waiting for PLC to finish program execution */
    /* State 9 = checking to see if there are any more ops pending */
    /* State 10 = disconnecting gateway */

```

```

state = 1;           /* initialize the variables */
working = 0;         /* 0 = not working, 1 = working */
connect_attempts = 0;
write_attempts = 0;
disconnect_attempts = 0;
step = 0;           /* initially there are no steps defined */

while (state != 11)    /* a software state machine */
{
    switch (state)
    {
        case 1:
            get_operations();
            break;

        case 2:
            if (working == 0)
            {
                working = 1;
                connect_attempts++;
                RetCode = connect_gateway (chan,title);
                if (RetCode != 0)
                {
                    working = 0;
                }
            }
            break;

        case 3:
            if (working == 0)
            {
                working = 1;
                write_attempts++;
                RetCode = write_plc5 ( "input_reg",input_reg[step] );
                if (RetCode != 0)
                {
                    working = 0;
                }
            }
            break;

        case 4:
            if (working == 0)
            {
                working = 1;
                write_attempts++;
                RetCode = write_plc5 ( "b_side",b_side[step] );
            }
    }
}

```

```

        if (RetCode != 0)
        {
            working = 0;
        }
    }
    break;

case 5:
    if (working == 0)
    {
        working = 1;
        write_attempts++;
        RetCode = write_plc5 ( "e_side",e_side[step] );
        if (RetCode != 0)
        {
            working = 0;
        }
    }
    break;

case 6:
    if (working == 0)
    {
        working = 1;
        write_attempts++;
        RetCode = write_plc5 ( "take_out",take_out[step] );
        if (RetCode != 0)
        {
            working = 0;
        }
    }
    break;

case 7:
    if (working == 0)
    {
        working = 1;
        write_attempts++;
        RetCode = write_plc5 ( "put_back",put_back[step] );
        if (RetCode != 0)
        {
            working = 0;
        }
    }
    break;

case 8:      /* waiting for PLC to finish executing program */

```

```

    if (working == 0)
    {
        working = 1;
        RetCode = read_plc5 ("dat_1");
        if (RetCode != 0)
        {
            printf ("\nread_plc5() was NOT successfully called...press a key.");
            getch();
            working = 0;
        }
    }
    break;

case 9:      /* checking to see if any more ops are pending */
    step++;
    if ( step < nmbr_steps )
    {
        working = 0;
        state = 3;
    }
    else
    {
        working = 0;
        state++;      /* go to disconnect state */
    }
    break;

case 10:
    if (working == 0)
    {
        working = 1;
        disconnect_attempts++;
        RetCode = disconnect_gateway (chan);
        if (RetCode != 0)
        {
            working = 0;
        }
    }
    break;

default:
    printf ("\n\nAn Undefined state = %d has been entered.", state);
    printf ("\nPress any key to restart at State 1.");
    getch();
    working = 0;
    state = 1;
    break;

```

```

    }
    comm_serve(); /* Do any MMS communication service needed */
} /* End of the while loop */
end_MMS();
print_stats(connect_attempts,write_attempts,disconnect_attempts);
return (0);
} /* end of main */

/*****
/***** connect_gateway() *****/
/*****/

```

```

SHORT connect_gateway ( SHORT chan, BYTE *title )
{
    if (mms_chan_info[chan].ctxt.chan_state == M_ACTIVE)
    {
        state++;
        working = 0;
        return (0); /* channel is already connected! */
    }
    else
    {
        if (lmv_init (chan, title))
        {
            printf ("\n\nERROR: init request not sent, error code = %d", mms_op_err);
            printf ("\nPress a key to continue...");
            getch();
            return (1);
        }
        else
        {
            return (0); /* successful connection request send */
        }
    }
} /* end of connect_gateway() function */

```

```

/***** u_mv_init_conf() *****/

```

```

/* This function is called by comm_serve() whenever a confirmation */
/* is received. */

```

```

VOID u_mv_init_conf ( req_ptr )

```

```

MMSREQ_PEND *req_ptr;
{
    if ( req_ptr -> resp_err == RESP_OK )
    {
        working = 0;
        state++;
    }
    else
    {
        printf ("\nError code %d returned in mv_init() CONFIRM!", req_ptr->resp_err);
        printf ("\nPress any key to continue...");
        getch();
        working = 0;
    }
    ms_clr_mvreq ( req_ptr );
}
/* end of u_mv_init_conf() function */

```

```

/*****
/*****      write_plc5()      *****/
/*****
/* returns 0 if mv_write() was successfully sent.      */
/* returns 1 if mv_write() was not successfully sent.  */
/*****

```

```

SHORT write_plc5 ( BYTE *var_name, SHORT data )
{
    struct object_name    variable_name,
                          type_name;

    /* Note: SHORT 'chan' is a global variable!      */

    variable_name.object_tag = VMD_SPEC;              /* VMD-specific variable name */
    type_name.object_tag = VMD_SPEC; /* VMD-specific variable type */

    strcpy (variable_name.obj_name.vmd_spec, var_name); /* select var name */
    strcpy (type_name.obj_name.vmd_spec, "Integer16"); /* select var type */

    if ( mv_write(chan, &variable_name, &type_name, &data) )
    {
        return (0);
    }
    else
    {

```

```

        printf ("\nError: mv_write() request not sent, error code = %d.", mms_op_err);
        printf ("\nPress any key to continue...");
        getch();
        return (1);
    }
} /* end of write_plc5() function */

```

```

/*****                                u_mv_write_conf()                                *****/

```

```

VOID u_mv_write_conf (req_ptr)

```

```

MMSREQ_PEND *req_ptr;
{
    if (req_ptr->resp_err == RESP_OK)
    {
        working = 0;
        state++;
    }
    else
    {
        printf ("\nError code %d returned in mv_write() CONFIRM!", req_ptr->resp_err);
        printf ("\nPress any key to continue...");
        getch();
        working = 0;
    }
    ms_clr_mvreq (req_ptr); /* free the data area used */
} /* end of u_mv_write_conf() function */

```

```

/*****                                disconnect_gateway()                                *****/
/*****                                disconnect_gateway()                                *****/
/*****                                disconnect_gateway()                                *****/

```

```

SHORT disconnect_gateway ( SHORT chan )

```

```

{
    CLEARSCR;
    if (mms_chan_info[chan].ctxt.chan_state != M_ACTIVE)
    {
        printf ("\nChannel is not ACTIVE...press a key to continue.");
        getch();
    }
}

```

```

        state++;
        return (0);
    }
    if ( mp_conclude(chan))
    {
        return (0);
    }
    else
    {
        printf ("\nError: conclude request not sent, error code = %d.",mms_op_err);
        printf ("\nPress any key...");
        getch();
        return (1);
    }
}          /* end of disconnect_gateway() fct */


/*****          u_mp_conclude_conf()          *****/

VOID u_mp_conclude_conf ( req_ptr )

MMSREQ_PEND *req_ptr;
{
    if ( req_ptr->resp_err == RESP_OK )
    {
        working = 0;
        state++;
    }
    else
    {
        printf ("\nError code %d returned in mp_conclude() CONFIRM!",
req_ptr->resp_err);
        printf ("\nPress any key to continue...");
        getch();
        working = 0;
    }
}          /* end of u_mp_conclude_conf() function */


/* The following miscellaneous support functions are required          */
/* by the main module.          */
*/

/*****          set_user_data()          *****/
/*****          set_user_data()          *****/
/*****          set_user_data()          *****/

VOID set_user_data ( VOID )

```

```

{
    dest_buffer = ( BYTE * )malloc (10000);/* allocate 'dest_buffer'      */
    source_buf = ( BYTE * )malloc (1020); /* allocate 'source_buf'      */
    logical_mem = ( BYTE * )malloc (0x3FF);/* create logical memory space */

    log_mem_start = logical_mem;
    log_mem_len = 0x3FF;
}

/*****
/*****      set_suic_parameters()      *****/
*****/

VOID set_suic_parameters ( VOID )
{
    extern SHORT  SUIC_int_no,
                printen,
                boot_talk_en;

    extern LONG  board_base_address;

    CLEARSCR;
    printf ("\nThe following SUIC parameters are being used:\n");
    printf ("\tNumber of channels: 2");
    printf ("\tShared RAM base address (hex): D0000000");
    printf ("\tInterrupt level: 2");
    printf ("\tPrint INI messages: Yes");
    printf ("\tEnable SUIC boot talk: Yes");
    printf ("\n\nPress any key to continue...");
    getch();

    max_mms_chan = 2;
    listen_sel = (SHORT *) calloc(max_mms_chan,sizeof(short));
    board_base_address = 0xD0000000L;
    SUIC_int_no = 2;
    printen = 1;
    boot_talk_en = 1;
}

/* end of set_suic_parameters() */

/*****
/*****      configure_channels()      *****/
*****/
/*
/* Note: this is the load_local_titles() function from MMSACON.C
*****/

VOID configure_channels ( VOID )

```

```

{
    register i;
    FILE *file_ptr;
    BYTE line[200];
    SHORT ret,done;
    SHORT err;
    SHORT section, new_section;
    SHORT chan;

    for (i = 0; i < max_mms_chan; ++i)
        listen_sel[i] = 0;

    printf ("\n\n LOADING PARAMETERS FROM CONFIGURATION FILE");

    if (! (file_ptr = fopen ("suic.cfg","r")))
    {
        wait_msg ("SUIC PARAMETER FILE ACCESS ERROR");
        return;
    }

    done = 0;
    err = 0;
    while (!done && !err)
    {
        if (! (fgets (line,199,file_ptr)))
        {
            done = 1;
            break;
        }
        if (line[0] == '#')
        {
            line[2] = 0;
            section = atoi (&line[1]);
            new_section = 1;
        }
        else
        {
            for (i = 0 ; i < 64; ++i)
                if (line[i] == '\n')
                    break;
            if (line[i] == '\n')
                line[i] = 0;
        }
        else
        {
            err = 1;
            break;
        }
    }
}

```

```

switch (section)
{
case 1 :          /* AR names to activate */
    printf ("\n ACTIVATE : %s",line);
    if (ret = mllp_act_ar_name (line))
    {
        printf ("\n ACTIVATE ERROR : %d",ret);
        wait_msg ("");
    }
    break;

case 2 :          /* AR names to be registered */
    if (new_section)
    {
        chan = 0;
        new_section = 0;
    }
    if (chan < max_mms_chan)
    {
        printf ("\n REGISTER CHAN %d: %s",chan,line);
        if (ret = mllp_reg_ar_name (chan,line))
        {
            printf ("\n REGISTER ERROR : %d",ret);
            wait_msg ("");
        }
        chan++;
    }
    else
        err = 1;
    break;

case 3 :          /* called channels*/
    chan = atoi (line);
    if (chan >= 0 && chan < max_mms_chan)
    {
        listen_sel[chan] = 1;          /* set flag */
        printf ("\n Listen Chan %d",chan);
        if (ret = mllp_ass_listen (chan))
        {
            printf ("\n Channel %d Listen Error : %d",chan,ret);
            wait_msg ("");
        }
        chan++;
    }
    else
        err = 1;
}

```

```

        break;

        default :
        break;
    }
}
}
if (err)
    wait_msg ("File Error Detected");
else
    key_err ("Load Complete");
fclose (file_ptr);
}                                     /* end of configure_channels() */


/*****
*****      set_user_types()      *****/
*****/

VOID set_user_types ( VOID )
{
    SHORT RetCode;

    RetCode = ms_add_std_types (&m_vmd_select->vmd_wide);
    if ( RetCode != 0 )
    {
        printf ("\nAdd Standard Types failed!  Error is %#X", RetCode);
        printf ("\nPress a key to continue...");
        getch();
    }
}                                     /* end of set_user_types() */


/*****
*****      set_user_vars()      *****/
*****/

VOID set_user_vars ( VOID )
{
    struct var_acc_addr adr;      /* contains the address of the variable */
    struct domain_objs *dom;
    struct object_name type;      /* contains the name of object to use */
    /***** struct named_dom_ctrl *ndom; *****/

    /* All types are VMD wide */
    /* All are symbolic Addresses */

```

```

type.object_tag = 0;          /* select VMD scope type      */
adr.addr_tag = 1;            /* select symbolic address */

/* First, for VMD wide variables, select the domain obj's structure */

dom = &m_vmd_select->vmd_wide; /* add all as VMD-specific */
                               /* using VMD-specific std types*/

adr.addr.sym_addr = "104";    /* byte offset */
strcpy (type.obj_name.vmd_spec,"Integer16"); /* select type */
if (!ms_add_named_var (dom,"input_reg",&type,&adr,0))
    wait_msg ("Local Variable Name ADD FAILURE");
else
    *((SHORT *) (log_mem_start + atoi(adr.addr.sym_addr))) = 72;

adr.addr.sym_addr = "106";    /* byte offset */
strcpy (type.obj_name.vmd_spec,"Integer16"); /* select type */
if (!ms_add_named_var (dom,"b_side",&type,&adr,0))
    wait_msg ("Local Variable Name ADD FAILURE");
else
    *((SHORT *) (log_mem_start + atoi(adr.addr.sym_addr))) = 72;

adr.addr.sym_addr = "108";    /* byte offset */
strcpy (type.obj_name.vmd_spec,"Integer16"); /* select type */
if (!ms_add_named_var (dom,"e_side",&type,&adr,0))
    wait_msg ("Local Variable Name ADD FAILURE");
else
    *((SHORT *) (log_mem_start + atoi(adr.addr.sym_addr))) = 72;

adr.addr.sym_addr = "110";    /* byte offset */
strcpy (type.obj_name.vmd_spec,"Integer16"); /* select type */
if (!ms_add_named_var (dom,"take_out",&type,&adr,0))
    wait_msg ("Local Variable Name ADD FAILURE");
else
    *((SHORT *) (log_mem_start + atoi(adr.addr.sym_addr))) = 72;

adr.addr.sym_addr = "112";    /* byte offset */
strcpy (type.obj_name.vmd_spec,"Integer16"); /* select type */
if (!ms_add_named_var (dom,"put_back",&type,&adr,0))
    wait_msg ("Local Variable Name ADD FAILURE");
else
    *((SHORT *) (log_mem_start + atoi(adr.addr.sym_addr))) = 72;

adr.addr.sym_addr = "114";    /* byte offset */
strcpy (type.obj_name.vmd_spec,"Integer16"); /* select type */
if (!ms_add_named_var (dom,"dat_1",&type,&adr,0))

```

```

        wait_msg ("Local Variable Name ADD FAILURE");
    else
        *((SHORT *) (log_mem_start + atoi(adr.addr.sym_addr))) = 72;

}          /* end of set_user_vars() */


/*****
*****      print_demo()      *****/
*****/

VOID print_demo()
{
    printf ("*****\n");
    printf ("\n");
    printf ("    University of Tennessee, Knoxville MAP Laboratory    \n");
    printf ("\n");
    printf ("*****");
    printf ("\n\n");
}


/*****
*****      print_stats()      *****/
*****/

VOID print_stats (SHORT connects, SHORT writes, SHORT disconnects)
{
    CLEARSCR;
    print_demo();
    printf ("\n\nThe program has executed successfully.");
    printf ("\n\n\tNumber of connect attempts: %d",connects);
    printf ("\n\n\tNumber of write attempts: %d",writes);
    printf ("\n\n\tNumber of disconnect attempts: %d",disconnects);
    printf ("\n\n");
}          /* end of print_stats() function */


/* The following functions were added to implement the sequencing.      */

/*****
*****      get_operations()      *****/
*****/

```

```

VOID get_operations ( VOID )
{
    SHORT row,
        column,
        key,
        ext;

    CLEARSCR;
    print_demo();
    get_chan_info();
    CLEARSCR;
    print_menu();

    step = 0;                                /* initialize variables */
    nmbr_steps = 0;
    while (state == 1)
    {
        if ((key = getch()) == 0)
        {
            key = getch();
            ext = 1;
        }
        else
        {
            printf ("\n\nIllegal key!...press an assigned function key: ");
            ext = 0;
        }
        if (ext)
        {
            switch (key)
            {
                case F1:                        /* reset learned sequence */
                    step = 0;
                    CLEARSCR;
                    print_menu();
                    break;

                case F2:                        /* learn an operation */
                    get_warehouse_info(step++);
                    nmbr_steps = step;
                    CLEARSCR;
                    print_menu();
                    break;

                case F3:                        /* save sequence to a file */
                    save_to_disk(nmbr_steps);
                    CLEARSCR;
            }
        }
    }
}

```

```

        print_menu();
        break;

    case F4:                                /* retrieve sequence from file */
        nmbr_steps = load_from_disk();
        CLEARSCR;
        print_menu();
        break;

    case F10:                               /* execute the sequence */
        CLEARSCR;
        print_demo();
        state = 2;
        working = 0;
        step = 0;
        break;

    default:
        break;

}                                           /* end of switch */
}                                           /* end of if */
}                                           /* end of while */
}                                           /* end of get_operations() function */

/***** get_chan_info() *****/

VOID get_chan_info ( VOID )
{
    SHORT key;

    while (1)
    {
        CLEARSCR;
        print_demo();
        printf ("\nEnter the Channel number to use: ");
        while ( lintget(&chan) || chan >= max_mms_chan || chan < 0 )
        {
            printf ("\nIllegal Channel number...");
            printf ("\nTry again: ");
        }

        printf ("\nEnter the Application Process Title to connect to: ");
        while (lstrget(title))
        {
            printf ("\nIllegal AP Title...");
            printf ("\nTry again: ");
        }
    }
}

```

```

}

/** Now comes the instance number stuff ****/

inst_nmbr = 500;
printf ("\nInstance number = %d", inst_nmbr);
if (ask ("\nModify? (N) ", 0 ))
{
    printf ("\nEnter new Instance Number: ");
    longget (&inst_nmbr);
}

if (mlip_set_rem_inst ( chan, 1, inst_nmbr ))
{
    printf ("\nInstance Number CROAKED!...press any key.");
    getch();
}

CLEARSCR;
printf ("\nThe following information has been entered: ");
printf ("\n\tChannel: %d", chan);
printf ("\tRemote AP Title: %s", title);
printf ("\tInstance Number: %d", inst_nmbr);
printf ("\nIs this information correct ('y' or 'n')? ");

key = getche();
if (key == 'y')
    break;
}
/* end of while */
}
/* end of get_chan_info() function */

/***** print_menu() *****/

VOID print_menu ( VOID )
{
    puts ("\nKey Assignments\n-----\n");
    puts ("\tF1\tReset sequence counter");
    puts ("\tF2\tEnter an operation");
    puts ("\tF3\tSave sequence to a file");
    puts ("\tF4\tRetrieve sequence from a file");
    puts ("\tF10\tExecute the sequence");
    printf ("\nEnter a choice: ");
}

/***** get_warehouse_info() *****/

```

```

VOID get_warehouse_info ( SHORT step )
{
    SHORT row,
        column,
        key;

    while (1)
    {
        CLEARSCR;
        print_demo();
        printf ("\nYou are presently entering step %d", step+1 );
            printf  ("\n-----\n");
        printf ("\nEnter warehouse row number: ");
        scanf ( "%d", &row );
        printf ("\nEnter warehouse column number: ");
        scanf ( "%d", &column );

        input_reg[step] = 10 * row + column; /* Calculate the input_reg value */

        b_side[step] = 0;                /* initialize variables */
        e_side[step] = 0;
        take_out[step] = 0;
        put_back[step] = 0;

        printf ("\nEnter 'b' for beginning or 'e' for ending side: ");
        key = (SHORT) getche();
        if ( key == 'b' )
            b_side[step] = 1;
        if ( key == 'e' )
            e_side[step] = 1;

        printf ("\nEnter 't' for take-out or 'p' for put back: ");
        key = (SHORT) getche();
        if ( key == 't' )
            take_out[step] = 1;
        if ( key == 'p' )
            put_back[step] = 1;

        CLEARSCR;
        printf ("\nThe following information has been entered: ");
        printf ("\tWarehouse Row: %d", row);
        printf ("\tWarehouse Column: %d", column);
        if (b_side[step])
            printf ("\tBeginning Side");
        if (e_side[step])
            printf ("\tEnding Side");
    }
}

```

```

    if (take_out[step])
        printf ("\n\tOperation: take-out");
    if (put_back[step])
        printf ("\n\tOperation: put-back");
    printf ("\n\nIs this information correct ('y' or 'n')? ");

    key = getche();
    if (key == 'y')
    {
        CLEARSCR;
        break;
    }
}
/* end of while */
}
/* end of get_warehouse_info() function */

/*****
***** save_to_disk() *****/
*****/

VOID save_to_disk ( SHORT nmbr_steps )
{
    BYTE fname[15];

    FILE *fptr;

    SHORT i;

    while (1)
    {
        CLEARSCR;
        printf ("\n\nEnter file name (up to 8 characters, no ext): ");
        gets (fname);
        if (strlen(fname) < 1)
        {
            printf ("\nFilename not entered - transaction aborted!");
            printf ("\nPress any key to continue...");
            getch();
            break;
        }

        /* open file for reading - check for duplicate */

        if (( fptr = fopen ( strcat(fname, ".SEQ"), "r") ) != NULL )
        {
            fclose (fptr);
            printf ("\nFile %s already exists. Overwrite ('y' or 'n')?", fname);

```

```

        if ( getche() != 'y' )
            continue;          /* start back at beginning of while loop */
    }

    /* open file for writing */

    printf ("\nFile to be opened is %s.", fname );
    if (( fptr = fopen(fname, "w")) == NULL )
    {
        printf ("\nFile open error!");
        printf ("\nPress a key to continue...");
        getch();
        continue;          /* start back at beginning of while loop */
    }

    /* write the data to the file */

    fprintf ( fptr, "%d\n", nmbr_steps );  /* put nmbr steps at top of file */

    for ( i=0 ; i < nmbr_steps ; i++ )
        fprintf(fptr,"%d %d %d %d %d\n",input_reg[i],b_side[i],e_side[i],take_out[i],put_back[i]);

    fclose (fptr);          /* close the file */

    printf ("\n\n%d operations saved in file %s.",nmbr_steps, fname);
    printf ("\nPress any key to continue...");
    getch();
    break;
}                          /* end of while loop */
}                          /* end of save_to_disk() function */

/*****
/*****          load_from_disk()          *****/
/*****/

SHORT load_from_disk ( VOID )
{
    BYTE fname[15];

    FILE *fptr;

    SHORT n,
        step;

    CLEARSCR;

```

```

printf ("\nSequence files available: \n\n");

if ( system("dir *.seq"))
{
    printf ("\nNo file found with .seq extension.");
    printf ("\nPress any key to continue...");
    getch();
    return (0);
}
printf ("\n\nEnter file name (no extension): ");
gets (fname);

if ( strlen(fname) < 1 )
{
    printf ("\nFilename not entered - transaction aborted!");
    return (0);
}

/* open file for reading */

if ( ( fptr = fopen ( strcat(fname, ".SEQ"), "ra" )) == NULL )
{
    printf ("\nFile open failed!");
    printf ("Press any key to continue...");
    getch();
    return (0);
}
printf ("\n\nFile opened...");
fscanf (fptr, "%d", &nbr_steps);    /* get the number of steps */

printf ("\nnbr_steps = %d...press a key", nbr_steps);
getch();

for ( n=0 ; n < nbr_steps ; n++ )    /* get the steps */
{
    fscanf(fptr,"%d %d %d %d
%d",&input_reg[n],&b_side[n],&e_side[n],&take_out[n],&put_back[n]);
    printf ("\ninput_reg[%d] = %d", n, input_reg[n] );
    printf ("\nb_side[%d] = %d", n, b_side[n] );
    printf ("\ne_side[%d] = %d", n, e_side[n] );
    printf ("\ntake_out[%d] = %d", n, take_out[n] );
    printf ("\nput_back[%d] = %d", n, put_back[n] );
}
printf ("\n\n...sequence is loaded.");
printf ("\nPress any key to continue...");
getch();

```

```

fclose (fptr);          /* close the file */

return (nmbr_steps);
}                        /* end of load_from_disk() function */

/*****
/*****      read_plc5()      *****/
/*****

SHORT read_plc5 ( BYTE *var_name )
{
    SHORT RetCode;          /* returns 0 if request successfully sent */

    struct object_name      variable_name,
                           type_name;

    variable_name.object_tag = VMD_SPEC;      /* VMD-specific variable name */
    type_name.object_tag = VMD_SPEC;          /* VMD-specific variable type */

    strcpy (variable_name.obj_name.vmd_spec, var_name);
    strcpy (type_name.obj_name.vmd_spec, "Integer16");

    if ( mv_read(chan, &variable_name, &type_name, &dat_1 ))
    {
        return (0);
    }
    else
    {
        printf ("\nError: mv_read() request not sent, error code = %d.", mms_op_err);
        printf ("\nPress any key to continue...");
        getch();
        return (1);
    }
}                        /* end of read_plc5() function */

/*****      u_mv_read_conf()      *****/

VOID u_mv_read_conf ( req_ptr )

MMSREQ_PEND *req_ptr;
{
    if ( req_ptr->resp_err != RESP_OK )
    {
        printf ("\nmv_read() COFIRM failed! Error = %#X", req_ptr->resp_err);
        printf ("\nPress any key to re-send the mv_read() REQUEST...");
    }
}

```

```

    getch();
    working = 0;
    state = 8;                                /* re-send the variable read request */
}
else
{
    if ( dat_1 == 0 )        /* PLC program has finished executing */
    {

        working = 0;
        state ++;
    }
    else                    /* PLC program has not finished,*/
    {                        /* so check the variable again.*/
        working = 0;
        state = 8;          /* re-send the variable read request */
    }
}
ms_clr_mvreq (req_ptr);
}                                     /* end of u_mv_read_conf() function */

```

MSK_CONF.C Source Code Module

```
/* ***** */
/* UNIVERSITY OF TENNESSEE MAP LAB ***** */
/* ***** */
/*
/* MODULE NAME : msk_conf.c
/* PRODUCT(S) : MMSEASE
/*
/* MODULE DESCRIPTION :
/* This module contains user confirm functions. These functions
/* are invoked from 'mms_req_serve' when the response to an
/* outstanding request is received. These functions should check
/* the resp_err member of the request control structure before
/* assuming returned data.
/*
/*
/* NOTES ON REQUEST INFORMATION :
/* In the user application, request information structures may be
/* committed when the request is made (e.g. for error recovery).
/* If so, these information buffers could be freed in the confirm
/* functions. In the sample application, all request information
/* structures are automatic or handled when the request is issued.
/*
/*
/* GLOBAL FUNCTIONS DEFINED IN THIS MODULE : */
/* VOID list_err_info (req_ptr) */
/* VOID u_mp_xxxx_conf (req_ptr) */
/* VOID u_mv_xxxx_conf (req_ptr)
/*
/* 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 "mms_user.h"
#include "userdefs.h"
#include "mmsop_en.h"
#include "gvaldefs.h"
#include "scrndefs.h"
```

```

/*****
/*****
/*          CONTEXT MANAGEMENT          */
/*****
/*****
/* MV INITIATE                          */
/* NOTE : user MUST make use of VM function for initiate, so no 'u_mp_      */
/* init_conf' function need be provided                                     */
/*****

/*****          u_mv_init_conf()          *****/

/* Note: this function is now in ASRSSEQ.C module.  see U_CONFRM.C to      */
/*   replace!                                                                    */

/*****
/* MP CONCLUDE                          */
/*****

/*****          u_mp_conclude_conf()          *****/

/* Note: this function is now in ASRSRMT.C module.  To replace see      */
/*   U_CONFRM.C                                                                    */

.....

/*****
/* MV WRITE                          */
/*****

#if MMS_WRITE_EN & REQ_EN
/*****          u_mv_write_conf()          *****/

/* Note: this function is now in ASRSSEQ.C.  See U_CONF.C for replace */
/*****

.....

```

MSK_DATA.C Source Code Module

```
/* ***** */
/* UNIVERSITY OF TENNESSEE MAP LAB ***** */
/* ***** */
/*
/* MODULE NAME : msk_data.c
/* PRODUCT(S) : MMSEASE
/*
/* MODULE DESCRIPTION :
/*   User defined data handling functions
/*
/* GLOBAL FUNCTIONS DEFINED IN THIS MODULE :
/*   BYTE *u_get_named_addr (vardef)
/*   VOID set_user_doms ()
/*   VOID set_user_types ()
/*   VOID set_user_vars ()
/*   VOID set_user_data ()
/*
/* MODIFICATION LOG :
/* Date   Who   Rev   Comments
/* -----
/* 09/06/88 MDE   4.01   Removed listen_sel allocation
/* 08/15/88     4.00   MMSEASE 4.0 release. See MODL40.DOC for
/*                      history.
/* ***** */

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

/* ***** */
/* Functions used internally for user interface
/* ***** */
/*
/*           set_user_data
/* Function to allocate large user data items, to conserve static data
/* space (for Lattice, especially).
/* ***** */

/* ***** set_user_data ***** */
/* Note: this function is now in the .C module. To replace
/*       see U_DATA.C file!!
```

```

/*****
/* Allow exclusion of all variable access MMSEASE code if not needed */
/*****
#if MMS_VA_EN /* if any variable access stuff enabled */
/*****
/*
/* u_get_named_addr */
/* User function to translate variable def structure to physical addr */
/* Return 0 for fail, otherwise return a pointer to data area. This is */
/* called by the virtual machine when servicing read or write requests */
/* (mv_read_resp, mv_write_resp). */
/*
/* In this sample application, a 'logical' address space has been */
/* defined (in uservar.c, userdefs.h). Treat symbolic address items as */
/* character string that represents offset into this space. */
/* Numeric addresses are treated as pointers. */
/* Unconstrained addresses not really treated here, just return base */
/* address of the logical buffer. */
/*****

```

```

BYTE *u_get_named_addr (vardef)

```

```

struct named_var *vardef;

```

```

{
SHORT offset;

```

```

switch (vardef->addr.addr_tag)

```

```

{
case 0 : /* numeric address */
/* assume this is a long pointer*/
/* this is machine specific

```

```

operation*/

```

```

return ((BYTE *) vardef->addr.addr.num_addr);

```

```

break;

```

```

case 1 : /* symbolic address */
/* assume BYTE string = offset

```

```

into*/

```

```

offset = atoi (vardef->addr.addr.sym_addr); /* user defined space */

```

```

if (offset > log_mem_len || offset < 0) /* check for bad offset */

```

```

return ((BYTE *) 0);

```

```

if (offset > log_mem_len) /* make sure not too long */

```

```

return ((BYTE *) 0);

```

```

return (log_mem_start + offset);

```

```

break;

```

```

case 2 : /* unconstrained address*/

```

```

        return (log_mem_start);
    break;

    default :
        return (NULL);
    break;
}
}

/*****
/*                      set_user_doms                      */
/* Function to create named domains for the sample application. */
*****/

VOID set_user_doms ()
{
    if (lms_add_named_dom ("Flex_Gage",(BYTE)0xff))
        wait_msg ("NAMED DOMAIN ADD FAILURE");
    if (lms_add_named_dom ("A_Application",(BYTE)0xff))
        wait_msg ("NAMED DOMAIN ADD FAILURE");
    if (lms_add_named_dom ("Z_Application",(BYTE)0xff))
        wait_msg ("NAMED DOMAIN ADD FAILURE");
    if (lms_add_named_dom ("Robot",(BYTE)0xff))
        wait_msg ("NAMED DOMAIN ADD FAILURE");
    if (lms_add_named_dom ("plc5",(BYTE)0xff))    /* Add PLC-5 domain */
        wait_msg ("NAMED DOMAIN ADD FAILURE");
}

/*****
/*                      set_user_types                      */
/* Function to set up the type definitions required for the application */
/* just do VMD specific types for now. */
*****/

/*****                      set_user_types()                      *****/

/* Note: this function is now in the .C module. To replace */
/* see U_DATA.C file!! */

/*****
/*                      set_user_vars                      */
/* Function to add sample application variable names to name table. */
/* Before these names can be added, the type names used must have been */
/* registered. */
*****/

```

```

/*****          set_user_vars()          *****/

/* Note: this function is now in ASRSSEQ.C module. To replace see      */
/*      U_DATA.C module!!                                              */

```

```

/*****          add_complex_vars          *****/
/*          add_complex_vars          */
/* Add test case variables          */
/*****          *****/

```

```

VOID add_complex_vars ()
{
    /**** NOTE: COMPLEX VARIABLES REMOVED BY MSK 6/25/90      ****/
}

```

```

/*****          *****/
#endif      /* end of #ifdef COMPLEX_TYPES          */

```

```

/*****          *****/
#else      /* MMS_VA_EN          */
/*****          *****/
/*          u_get_named_addr          */
/* This function needs to exist for any object handling          */

```

```

BYTE *u_get_named_addr (vardef)
struct named_var *vardef;
{
    return (NULL);
}

```

```

/*****          *****/
#endif      /* end of #if MMS_VA_EN          */
/*****          *****/

```

MSKACON.C Source Code Module

```
/* ***** */
/* UNIVERSITY OF TENNESSEE MAP LAB ***** */
/* ***** */
/*
/* MODULE NAME : mskacon.c
/*
/*
/* MODULE DESCRIPTION :
/*   This module contains functions to initiate the MMS context
/*   management services. These functions are called from the
/*   mainloop (check_key).
/*
/*
/* GLOBAL FUNCTIONS DEFINED IN THIS MODULE :
/*
/*   VOID do_init_rel ()
/*   VOID do_init ()
/*   VOID do_init ()
/*   VOID do_concl ()
/*   VOID do_abort ()
/*   VOID do_cancel ()
/*   VOID do_mchan ()
/*   VOID do_listen ()
/*   VOID do_stop_listen ()
/*
/*
/* 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"
#if ( SYSTEM_SEL & ( MSOFT | OS2 ) )
#include <ctype.h>
#include <conio.h>
#endif

#if ( SYSTEM_SEL & ( XENIX | SYS_5 | BSD | VMS ) )
#include <ctype.h>
#endif

#include "mms_user.h"
#include "mmsop_en.h"
#include "userdefs.h"
```

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

```
#ifdef LINT_ARGS
extern SHORT req_unlink_conf (MMSREQ_PEND *);
extern VOID req_put_pend (MMSREQ_PEND *);
#else
extern SHORT req_unlink_conf ();
extern VOID req_put_pend ();
#endif
```

```
.....

/*****
/*          load_local_titles  MODIFIED BY MARK KING!          */
*****/
```

```
VOID load_local_titles ()
```

```
{
register i;
FILE *file_ptr;
BYTE line[200];
SHORT ret,done;
SHORT err;
SHORT section, new_section;
SHORT chan;
```

```
for (i = 0; i < max_mms_chan; ++i)
    listen_sel[i] = 0;
```

```
printf ("\n\n LOADING PARAMETERS FROM CONFIGURATION FILE");
```

```
if (! (file_ptr = fopen ("suic.cfg","r")))
{
wait_msg ("SUIC PARAMETER FILE ACCESS ERROR");
return;
}
```

```
done = 0;
err = 0;
while (!done && !err)
```

```

{
    if (!fgets (line,199,file_ptr))
    {
        done = 1;
        break;
    }
    if (line[0] == '#')
    {
        line[2] = 0;
        section = atoi (&line[1]);
        new_section = 1;
    }
else
    {
        for (i = 0 ; i < 64; ++i)
            if (line[i] == '\n')
                break;
            if (line[i] == '\n')
                line[i] = 0;
else
        {
            err = 1;
            break;
        }

switch (section)
{
case 1 :                /* AR names to activate      */
    printf ("\n ACTIVATE : %s",line);
    if (ret = mllp_act_ar_name (line))
    {
        printf ("\n ACTIVATE ERROR : %d",ret);
        wait_msg ("");
    }
    break;

case 2 :                /* AR names to be registered */
    if (new_section)
    {
        chan = 0;
        new_section = 0;
    }
    if (chan < max_mms_chan)
    {
        printf ("\n REGISTER CHAN %d: %s",chan,line);
        if (ret = mllp_reg_ar_name (chan,line))
        {

```

```

    printf ("\n REGISTER ERROR : %d",ret);
    wait_msg ("");
}

    inst_nmbr = 500; /* this is the PLC-5 inst number */
    printf ("\nInstance Number = %d", inst_nmbr);
    if (!mlp_set_rem_inst(chan,1,inst_nmbr))
    {
        printf ("\nInstance Number FAILED!");
        wait_msg (""); /* wait for user to press a key */
    }

    chan++;
}
else
    err = 1;
break;

case 3 : /* called channels*/
    chan = atoi (line);
    if (chan >= 0 && chan < max_mms_chan)
    {
        listen_sel[chan] = 1; /* set flag */
        printf ("\n Listen Chan %d",chan);
        if (ret = mlp_ass_listen (chan))
        {
            printf ("\n Channel %d Listen Error : %d",chan,ret);
            wait_msg ("");
        }
        chan++;
    }
    else
        err = 1;
break;

default :
break;
}
}
}
if (err)
    wait_msg ("File Error Detected");
else
    key_err ("Load Complete");
fclose (file_ptr);
}

```

ASRSSEQ.MAK Make File

asrsseq.obj : asrsseq.c mmsop_en.h

cl /c /Osi /G2 /DMP_400 /DDEBUG_MMS /DLINT_ARGS /DMAP21_CASE /W2 /AL
/Fs asrsseq.c ;

mmsacase.obj : mmsacase.c mmsop_en.h

cl /c /Osi /G2 /DMP_400 /DDEBUG_MMS /DLINT_ARGS /W2 /AL mmsacase.c ;

mmsamisc.obj : mmsamisc.c mmsop_en.h

cl /c /Osi /G2 /DDEBUG_MMS /DLINT_ARGS /DMAP21_CASE /W2 /AL mmsamisc.c ;

mmsallp.obj : mmsallp.c mmsop_en.h

cl /c /Osi /G2 /DDEBUG_MMS /DLINT_ARGS /W2 /AL mmsallp.c ;

mskacon.obj : mskacon.c mmsop_en.h

cl /c /Osi /G2 /DDEBUG_MMS /DLINT_ARGS /W2 /AL mskacon.c ;

mmsavmd.obj : mmsavmd.c mmsop_en.h

cl /c /Osi /G2 /DDEBUG_MMS /DLINT_ARGS /W2 /AL mmsavmd.c ;

mmsadom.obj : mmsadom.c mmsop_en.h

cl /c /Osi /G2 /DDEBUG_MMS /DLINT_ARGS /W2 /AL mmsadom.c ;

mmsaprg.obj : mmsaprg.c mmsop_en.h

cl /c /Osi /G2 /DDEBUG_MMS /DLINT_ARGS /W2 /AL mmsaprg.c ;

mmsavar.obj : mmsavar.c mmsop_en.h

cl /c /Osi /G2 /DDEBUG_MMS /DLINT_ARGS /W2 /AL mmsavar.c ;

mmsasem.obj : mmsasem.c mmsop_en.h

cl /c /Osi /G2 /DDEBUG_MMS /DLINT_ARGS /W2 /AL mmsasem.c ;

mmsaocs.obj : mmsaocs.c mmsop_en.h

cl /c /Osi /G2 /DDEBUG_MMS /DLINT_ARGS /W2 /AL mmsaocs.c ;

mmsaevn.obj : mmsaevn.c mmsop_en.h

cl /c /Osi /G2 /DDEBUG_MMS /DLINT_ARGS /W2 /AL mmsaevn.c ;

mmsajou.obj : mmsajou.c mmsop_en.h

cl /c /Osi /G2 /DDEBUG_MMS /DLINT_ARGS /W2 /AL mmsajou.c ;

mmsafil.obj : mmsafil.c mmsop_en.h

cl /c /Osi /G2 /DDEBUG_MMS /DLINT_ARGS /W2 /AL mmsafil.c ;

```

uservar.obj : uservar.c mmsop_en.h
    cl /c /Osi /G2 /DDEBUG_MMS /DLINT_ARGS /W2 /AL uservar.c ;

msk_data.obj : msk_data.c mmsop_en.h
    cl /c /Osi /G2 /DDEBUG_MMS /DLINT_ARGS /W2 /AL msk_data.c ;

u_case.obj : u_case.c
    cl /c /Osi /G2 /DDEBUG_MMS /DLINT_ARGS /W2 /AL u_case.c ;

u_indic.obj : u_indic.c mmsop_en.h
    cl /c /Osi /G2 /DDEBUG_MMS /DLINT_ARGS /W2 /AL u_indic.c ;

msk_conf.obj : msk_conf.c mmsop_en.h
    cl /c /Osi /G2 /DDEBUG_MMS /DLINT_ARGS /W2 /AL msk_conf.c ;

mmsop_en.obj : mmsop_en.c mmsop_en.h
    cl /c /Osi /G2 /DDEBUG_MMS /DLINT_ARGS /W2 /AL mmsop_en.c ;

fkey.obj : fkey.c
    cl /c /Osi /G2 /DDEBUG_MMS /DLINT_ARGS /W2 /AL fkey.c ;

getval5.obj : getval5.c
    cl /c /Osi /G2 /DDEBUG_MMS /DLINT_ARGS /W2 /AL getval5.c ;

asrsseq.exe : asrsseq.obj mskacon.obj mmsavmd.obj mmsadom.obj mmsaprg.obj \
    mmsavar.obj mmsasem.obj mmsaocs.obj mmsaevn.obj mmsajou.obj \
    mmsafil.obj mmsallp.obj mmsacase.obj u_case.obj mmsamisc.obj \
    uservar.obj msk_data.obj u_indic.obj msk_conf.obj mmsop_en.obj \
    getval5.obj fkey.obj
    link @seqlink

```

SEOLINK Link File for ASRSSEQ.MAK

asrsseq +
mskacon +
mmsavmd +
mmsadom +
mmsaprg +
mmsavar +
mmsasem +
mmsaocs +
mmsaevn +
mmsajou +
mmsafil +
mmsallp +
mmsacase +
u_case +
mmsamisc +
uservar +
msk_data +
u_indic +
msk_conf +
mmsop_en +
getval5 +
fkey
asrsseq
/MAP:60000 /E /ST:8196 /SEGMENTS:521 /NOD /NOE /NOI /CO
asn1d mmsd1 mmsd2 mmscased suiccase +
llibce

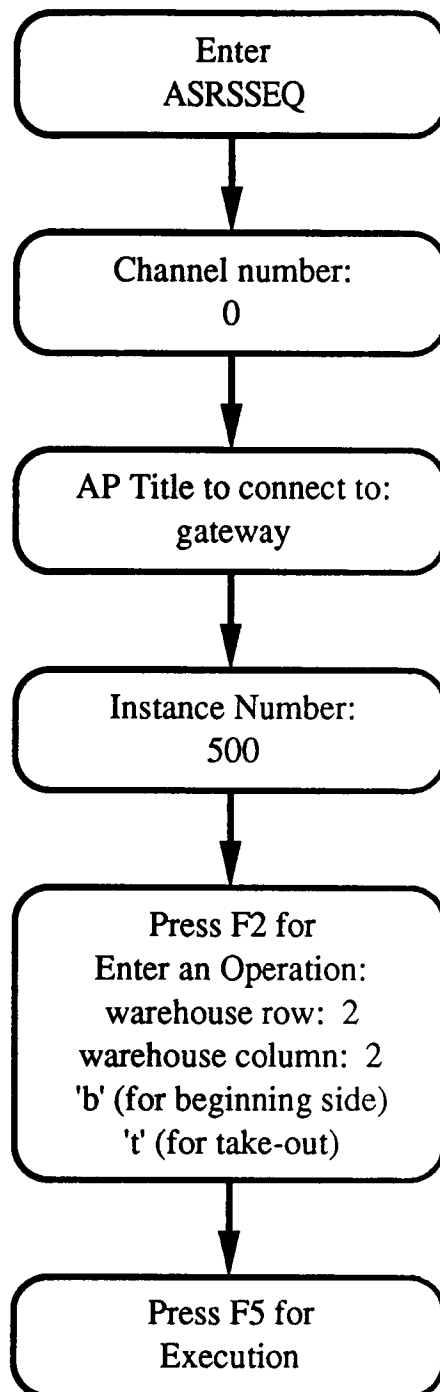


Figure G-1. Operating the Factory with the ASRSSEQ Program.

APPENDIX H

UPGRADE TO MAP V3.0

ASRSMAP3.C Source Code Modifications

```
/* *****
/*  UTK MAP LAB SOFTWARE MODULE *****
/* *****
/*
/*  MODULE NAME: asrsm3.c
/*  WRITTEN BY:  Mark S. King
/*
/*  MODULE DESCRIPTION:
/*      This module contains the 'main' function.  The purpose of the
/*      module is to control the ASRS ladder logic program in the
/*      Allen-Bradley PLC-5/15.  This module is a modification of the
/*      ASRSSEQ.C module and allows a sequence of operations to be
/*      learned, stored in a file, retrieved from a file, or executed.
/*
```

```
*****
***** disconnect_gateway() *****
*****
```

```
SHORT disconnect_gateway ( SHORT chan )
{
    CLEARSCR;
    if (mms_chan_info[chan].ctxt.chan_state != M_ACTIVE)
    {
        printf ("\nChannel is not ACTIVE...press a key to continue.");
        getch();
        state++;
        return (0);
    }
    if ( mp_conclude(chan))
    {
        return (0);
    }
    else
    {
        printf ("\nError: conclude request not sent, error code = %d.",mms_op_err);
    }
}
```

```

        printf ("\nPress any key...");
        getch();
        return (1);
    }
} /* end of disconnect_gateway() fct */

/*****          u_mp_conclude_conf()          *****/

VOID u_mp_conclude_conf ( req_ptr )

MMSREQ_PEND *req_ptr;
{
    if ( req_ptr->resp_err == RESP_OK )
    {
        /** working = 0; **/
        /** state++; **/
    }
    else
    {
        printf ("\nError code %d returned in mp_conclude() CONFIRM!",
req_ptr->resp_err);
        printf ("\nPress any key to continue...");
        getch();
        working = 0;
    }
} /* end of u_mp_conclude_conf() function */

/*****
/ *          u_mp_conclude_done          * /
/ * This function is invoked when a release confirm is received OR when * /
/ * ACSE refuses the request. This signals the real termination of this * /
/ * connection. * /
/ *****/

VOID u_mp_conclude_done (chan,ret_code)
SHORT chan;
USHORT ret_code;
{
#ifdef DEBUG_MMS
    if (mms_debug_sel & MMSCONF_PRINT)
    {
        clear_bottom ();
        printf ("\n\n Conclude Complete for channel %d returned",chan);
        printf ("\n Return code = %d",ret_code);
    }
#endif
}

```

```

    if (!ret_code)
    {
/* Take care of resource de-allocation for this channel.      * /

        ms_clr_ind_que (chan,1);      /* don't want any more indications from */
                                      /* the disconnected channel.          */
        ms_clr_rem_fctrl (chan);      /* release all remote file control for   */
                                      /* this channel                          */
        ms_clr_loc_fctrl (chan);      /* release all local file control for   */
                                      /* this channel, and close files.      */
        if (!listen_sel[chan])        /* if this is a called channel        */
            mllp_ass_listen (chan); /* post another listen                  */
        state++;
    }
}

```

.....

```

/*****          get_chan_info()          *****/

```

```

VOID get_chan_info ( VOID )

```

```

{
    SHORT key;

    while (1)
    {
        CLEARSCR;
        print_demo();
        printf ("\nEnter the Channel number to use: ");
        while ( !intget(&chan) || chan >= max_mms_chan || chan < 0 )
        {
            printf ("\nIllegal Channel number...");
            printf ("\nTry again: ");
        }

        printf ("\n\nEnter the Application Process Title to connect to: ");
        while (!strget(title))
        {
            printf ("\nIllegal AP Title...");

```

```

    printf ("\nTry again: ");
}

CLEARSCR;
printf ("\nThe following information has been entered: ");
printf ("\n\n\tChannel: %d", chan);
printf ("\n\tRemote AP Title: %s", title);
printf ("\n\tInstance Number: %d", inst_nmbr);
printf ("\n\nIs this information correct ('y' or 'n')? ");

key = getche();
if (key == 'y')
    break;
}                                     /* end of while */
}                                     /* end of get_chan_info() function */

```

Note: Code shown is the modification to the ASRSSEQ.C source code.

ASRSMAP3.MAK Make File

```
asrsm3.obj : asrsm3.c mmsop_en.h
cl /c /Osi /G2 /DCONCORD /DDEBUG_MMS /DLINT_ARGS /DMAP30_ACSE /W2 /AL
/Fs asrsm3.c ;

mmsaacse.obj : mmsaacse.c mmsop_en.h
cl /c /Osi /G2 /DCONCORD /DDEBUG_MMS /DLINT_ARGS /W2 /AL mmsaacse.c ;

mmsamisc.obj : mmsamisc.c mmsop_en.h
cl /c /Osi /G2 /DDEBUG_MMS /DLINT_ARGS /DMAP30_ACSE /W2 /AL mmsamisc.c ;

mmsallp.obj : mmsallp.c mmsop_en.h
cl /c /Osi /G2 /DDEBUG_MMS /DLINT_ARGS /W2 /AL mmsallp.c ;

mskacon.obj : mskacon.c mmsop_en.h
cl /c /Osi /G2 /DDEBUG_MMS /DLINT_ARGS /W2 /AL mskacon.c ;

mmsavmd.obj : mmsavmd.c mmsop_en.h
cl /c /Osi /G2 /DDEBUG_MMS /DLINT_ARGS /W2 /AL mmsavmd.c ;

mmsadom.obj : mmsadom.c mmsop_en.h
cl /c /Osi /G2 /DDEBUG_MMS /DLINT_ARGS /W2 /AL mmsadom.c ;

mmsaprg.obj : mmsaprg.c mmsop_en.h
cl /c /Osi /G2 /DDEBUG_MMS /DLINT_ARGS /W2 /AL mmsaprg.c ;

mmsavar.obj : mmsavar.c mmsop_en.h
cl /c /Osi /G2 /DDEBUG_MMS /DLINT_ARGS /W2 /AL mmsavar.c ;

mmsasem.obj : mmsasem.c mmsop_en.h
cl /c /Osi /G2 /DDEBUG_MMS /DLINT_ARGS /W2 /AL mmsasem.c ;

mmsaocs.obj : mmsaocs.c mmsop_en.h
cl /c /Osi /G2 /DDEBUG_MMS /DLINT_ARGS /W2 /AL mmsaocs.c ;

mmsaevn.obj : mmsaevn.c mmsop_en.h
cl /c /Osi /G2 /DDEBUG_MMS /DLINT_ARGS /W2 /AL mmsaevn.c ;

mmsajou.obj : mmsajou.c mmsop_en.h
cl /c /Osi /G2 /DDEBUG_MMS /DLINT_ARGS /W2 /AL mmsajou.c ;

mmsafil.obj : mmsafil.c mmsop_en.h
cl /c /Osi /G2 /DDEBUG_MMS /DLINT_ARGS /W2 /AL mmsafil.c ;
```

```

uservar.obj : uservar.c mmsop_en.h
cl /c /Osi /G2 /DDEBUG_MMS /DLINT_ARGS /W2 /AL uservar.c ;

msk_data.obj : msk_data.c mmsop_en.h
cl /c /Osi /G2 /DDEBUG_MMS /DLINT_ARGS /W2 /AL msk_data.c ;

u_acse.obj : u_acse.c
cl /c /Osi /G2 /DDEBUG_MMS /DLINT_ARGS /W2 /AL u_acse.c ;

u_indic.obj : u_indic.c mmsop_en.h
cl /c /Osi /G2 /DDEBUG_MMS /DLINT_ARGS /W2 /AL u_indic.c ;

msk_conf.obj : msk_conf.c mmsop_en.h
cl /c /Osi /G2 /DDEBUG_MMS /DLINT_ARGS /W2 /AL msk_conf.c ;

mmsop_en.obj : mmsop_en.c mmsop_en.h
cl /c /Osi /G2 /DDEBUG_MMS /DLINT_ARGS /W2 /AL mmsop_en.c ;

fkey.obj : fkey.c
cl /c /Osi /G2 /DDEBUG_MMS /DLINT_ARGS /W2 /AL fkey.c ;

getval5.obj : getval5.c
cl /c /Osi /G2 /DDEBUG_MMS /DLINT_ARGS /W2 /AL getval5.c ;

asrsm3.exe : asrsm3.obj mskacon.obj mmsavmd.obj mmsadom.obj mmsaprg.obj \
mmsavar.obj mmsasem.obj mmsaocs.obj mmsaevn.obj mmsajou.obj \
mmsafil.obj mmsallp.obj mmsaacse.obj u_acse.obj mmsamisc.obj \
uservar.obj msk_data.obj u_indic.obj msk_conf.obj mmsop_en.obj \
getval5.obj fkey.obj
link @map3link

```

MAP3LINK Link File for ASRSMAP3.MAK

asrsm3 +
mskacon +
mmsavmd +
mmsadom +
mmsaprg +
mmsavar +
mmsasem +
mmsaocs +
mmsaevn +
mmsajou +
mmsafil +
mmsallp +
mmsaacse +
u_acse +
mmsamisc +
uservar +
msk_data +
u_indic +
msk_conf +
mmsop_en +
getval5 +
fkey
asrsm3
/MAP:60000 /E /ST:8196 /SEGMENTS:521 /NOD /NOE /NOI /CO
asn1d mmsd1 mmsd2 mmsacsed suicacse +
llibce

SUIC.DIB for ASRSMAP3.EXE

\$ 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/

#

/master/////00000005/0001/0001/

/taskb/////00000002/0001/0001/

\$ Remote AP Names

#

Format:

#

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

#

/gateway/1 2 3 4 8//1//

00000005/0003/0001/39840f454e450000690101f4000100/

SUIC.DIB for MAP V3.0 Gateway

#This file contains data required by SUIC to function properly.

#

The '\$' marks the beginning of each section.

#

You can use the '#' for comments.

\$ Abstract Syntax Names

#

Format :

Valid OBJECT IDENTIFIER

#

\$ Transfer Syntax Names

#

Format :

Valid OBJECT IDENTIFIER

#

\$ Application Context Names

Format :

Valid OBJECT IDENTIFIER

#

\$ Local Network Address

#

/39840f454e450000690101f4000100/

\$ Local AP Names

#

Format:

/name/AP_title/AP_invoke_id/AE_qual/AE_invoke_id/p_sel/s_sel/t_sel/

#

where AP_title is a valid OBJECT IDENTIFIER;

AP_invoke_id, AE_qual, AE_invoke_id are integers.

If any of these 4 parameters are not to be used, they may be omitted.

For example:

/name/AP_title//AE_qual//p_sel/s_sel/t_sel/

#

/gateway/1 2 3 4 8//1//00000005/0003/0001/

\$ Remote AP Names

#

Format:
/name/AP_Title/AP_invoke_id/AE_qual/AE_invoke_id/p_sel/s_sel/t_sel/n_sel/

where AP_title is a valid OBJECT IDENTIFIER;
AP_invoke_id, AE_qual, AE_invoke_id are integers.
If any of these 4 parameteres are not to be used, they may be omitted.
For example:
/name/AP_Title//AE_qual//p_sel/s_sel/t_sel/n_sel/

/master/////00000005/0001/0001/39840f454e450000690101f4000200/

MMS EASE.CFG for MAP V3.0 Gateway

BootTalkEnable	1
HWreset	1
LoadFile	C:\CONCORD\DLDPROD.CNV
DIBfile	C:\CONCORD\SUIC.DIB
NumberOfChannels	2
BaseAddress	300 (TBC)
DMAchannel	3 (TBC)
IRQlevel	3 (TBC)

VITA

VITA

Mark Steven King was born in St. Petersburg, Florida on December 1, 1962. He attended elementary and secondary schools in Oak Ridge, Tennessee and he graduated from Oak Ridge High School in 1981. The following September he entered the University of Tennessee, Knoxville. Mr. King entered the Co-operative Engineering Program his sophomore year where he gained employment with the National Aeronautics and Space Administration at Marshall Space Flight Center in Huntsville, Alabama. In June 1986 he received a Bachelor of Science degree in Electrical Engineering.

Following graduation, Mr. King accepted employment with NASA at Marshall Space Flight Center, Alabama. In the fall of 1989 he was awarded a one-year fellowship to pursue a Master of Science degree at the University of Tennessee. Following graduation, he will return to work for NASA. Mr. King is a member of Eta Kappa Nu Electrical Engineering Honor Society.