

# MMD-2

## Tutorial User's Guide



MMD-2 TUTORIAL USER'S GUIDE

by John Bockelmann

© Copyright E&L Instruments Inc.  
July, 1981



### Acknowledgements

This book would not have been possible without the editorial assistance of Matt Veslocki, Keith Edmonds, David Levine and Susan Carbone. Thanks are due to Jose Gonzales for some of the drawings, and Marybeth Mikos for her help as we assembled the final document. Also appreciated were the numerous suggestions and comments made by others within E&L and in the field.

The Engineering Department,  
E&L Instruments Inc.

# WARNING

\*\*\*\*\*  
FEDERAL REGULATION (PART 15 OF FCC RULES) PROHIBITS THE USE OF  
COMPUTING EQUIPMENT WHICH CREATES RADIO OR TV INTERFERENCE  
\*\*\*\*\*

E & L Instruments specifically warns the user of this instrument that it is intended for use in a classroom or laboratory environment for the purpose of learning and experimentation. When building experimental circuits, it may emit interference that will effect radio and television reception and the user may be required to stop operaton until the interference problem is corrected. Home use of this equipment is discouraged since the likelihood of interference is increased by the close proximity of neighbors.

Corrective measures:

Interference can be reduced by the following practices.

- 1) Install a commercially built RFI power filter in the power line at the point where the cord enters the unit
- 2) Avoid long wires. They act as antenneas
- 3) If long wires must be used, use shielded cables or twisted pairs which are properly grounded and terminated

807-0018

## TABLE OF CONTENTS

<u>INTRODUCTION</u> . . . . .	Page 1
<u>SECTION 1</u> Basic Features . . . . .	Page 2
<u>SECTION 2</u> Basic Feature Experiments . . . . .	Page 15
<u>SECTION 3</u> Advanced Features . . . . .	Page 36
<u>SECTION 4</u> Advanced Feature Experiments . . . . .	Page 44
<u>SECTION 5</u> Final Notes . . . . .	Page 74
<u>APPENDIX</u> ASC II Code Table . . . . .	Page 75





# IMPORTANT!

## HOW TO USE THE MMD-2 MANUALS

Your MMD-2 is one of the most sophisticated microcomputer training and development systems available. Using a combination of different manuals aimed at different levels of understanding and different types of readers, we have tried to describe the computer and its operating software (EXEC C) as comprehensively as possible.

The manuals have been designed to support several different routes to understanding the MMD-2 and how to use it. The choice of one of these depends largely on your experience with microcomputer hardware and programming.

1. If you are starting from scratch, the best approach is probably to use the MMD-2 in conjunction with E & L's Technibooks V and VI, and the MMD-2 manual entitled "USING THE MMD-2 WITH THE TECHNIBOOKS". Technibooks V and VI are written around E & L's earlier MMD-1 trainer, but with minor differences will work with the MMD-2. These differences are covered on a chapter-by-chapter/step-by-step basis in "USING THE MMD-2 WITH THE TECHNIBOOKS". You will also need the manual entitled "MMD-2 TUTORIAL USER'S GUIDE", which replaces Chapter 4 of Technibook V.
2. If you've been through the Technibooks before with an MMD-1, you probably should read the chapter comparing the MMD-1 and MMD-2 systems in the manual entitled, "USING THE MMD-2 WITH THE TECHNIBOOKS", and go through the "MMD-2 TUTORIAL USER'S GUIDE".
3. If you're fairly knowledgeable about micros but know nothing about the MMD-1, you should probably just go through the "MMD-2 TUTORIAL USER'S GUIDE".
4. If you're virtually expert with microprocessor systems, you might skip the preliminaries and simply look at the "MMD-2 QUICK REFERENCE MANUAL".
5. If what you want are the technical details, consult the "MMD-2 REFERENCE MANUAL", together with the "MMD-2 EXEC C SOFTWARE LISTING" and the "MMD-2 SCHEMATICS".
6. The following is a list of manuals available for use with the MMD-2:

Using the MMD-2 with the Technibooks	(Part No. 801-0192)
MMD-2 Schematics	(Part No. 801-0205)
MMD-2 Tutorial User's Guide	(Part No. 801-0213)
MMD-2 Reference Manual	(Part No. 801-0228)
MMD-2 Source Listing	(Part No. 801-0229)
MMD-2 Quick Reference Manual	(Part No. 801-0230)





## MMD-2 TUTORIAL USER'S GUIDE

### Introduction

The MMD-2 user's guide is designed to help you become familiar with the organization, function and operation of the MMD-2 computer system. Each major section of the computer will be examined and its function described. Also included are detailed instructions on the operation of the computer as well as short programs which will help familiarize you with the computer.

This user's guide tells you about two things: How to use the MMD-2 for 8080A machine language programming and how to use the various auxiliary functions of the MMD-2. Therefore, in order to understand this user's guide and use it effectively, it is essential that you have some knowledge of machine language (as this guide does not teach this subject). This guide assumes a knowledge equivalent to that in the first three chapters of Technibook V (E & L Instruments Part Number 345-1020).

This guide is organized using the following format. Section 1 explains the basic structure and features of the computer. Section 2 provides you with a series of experiments to help you become familiar with the operation of the computer. Section 3 introduces you to the computer's more advanced feature. Section 4 includes experiments designed to demonstrate these features.

### Objectives

When you have completed this guide, you should be familiar with the following:

1. You will have an understanding of the major sub-sections of the computer how they interrelate to form a complete system.
2. You will be familiar with the general operation of the computer and understand the functions available from the keypad and associated controls.
3. You will know how to interpret the output displays on the computer.
4. You will be able to enter, modify and run machine language programs.
5. You will understand how to interface the MMD-2 computer to an external terminal and tape recorder.
6. You will be able to use the MMD-2 to program and verify EPROMs.
7. You will know how to use the breakpoints and step features to observe program execution.

## SECTION 1

### THE BASIC FEATURES OF THE MMD-2

#### Introduction

The MMD-2 computer is an 8080A-based computer system. It is a complete, functional computer capable of loading, storing, and running machine language programs. These are programs written in machine language, the most elementary language the computer understands. This chapter will introduce you to the MMD-2 computer's functional components, systems and operating procedures. In using this guide, it is assumed you have access to an MMD-2 computer, and that the computer is in operating condition.

#### Objectives

When you have completed this section, you should be able to do the following:

1. List the different DC voltages needed to operate the computer. You will know how much current is available at each voltage for external experimental circuits you can connect to the computer.
2. Understand the capabilities provided by all the functions available on both keypads.
3. Understand the function of the mode selector switch and reset push button.
4. Know how to read the seven-segment displays for address, data register information and special command words.
5. Understand how to interpret the HALT, HOLD, and INTE indicator LEDs.
6. Understand how to read the binary output LEDs for data and address indications.
7. Describe the operation of each of the main sections of the computer, and understand why they are part of the computer and how they interact with the rest of the computer.
8. Know the difference between random access memory (RAM) and erasable programmable read only memory (EPROM).

#### How the MMD-2 Computer is Used

The MMD-2 (Mini-Micro Designer-2) is a complete microcomputer system. It contains keypads for data and control function entry, seven-segment displays for the display of data, and twenty-four individual LEDs (light emitting diodes) arranged

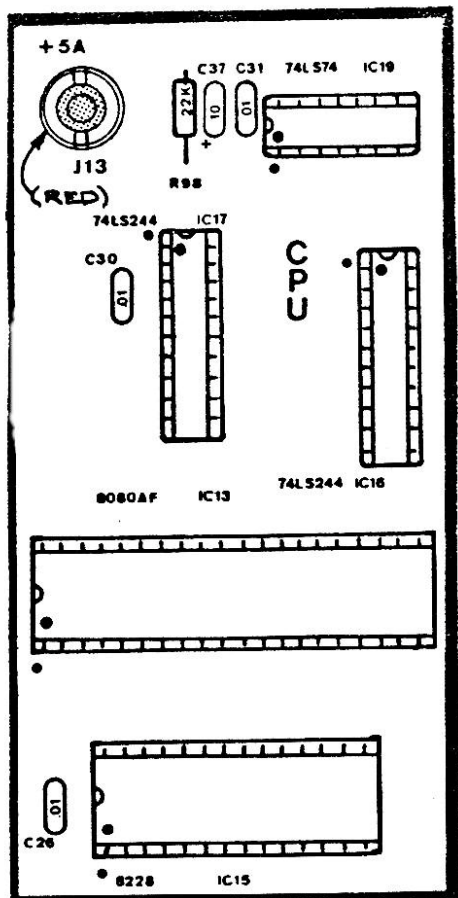
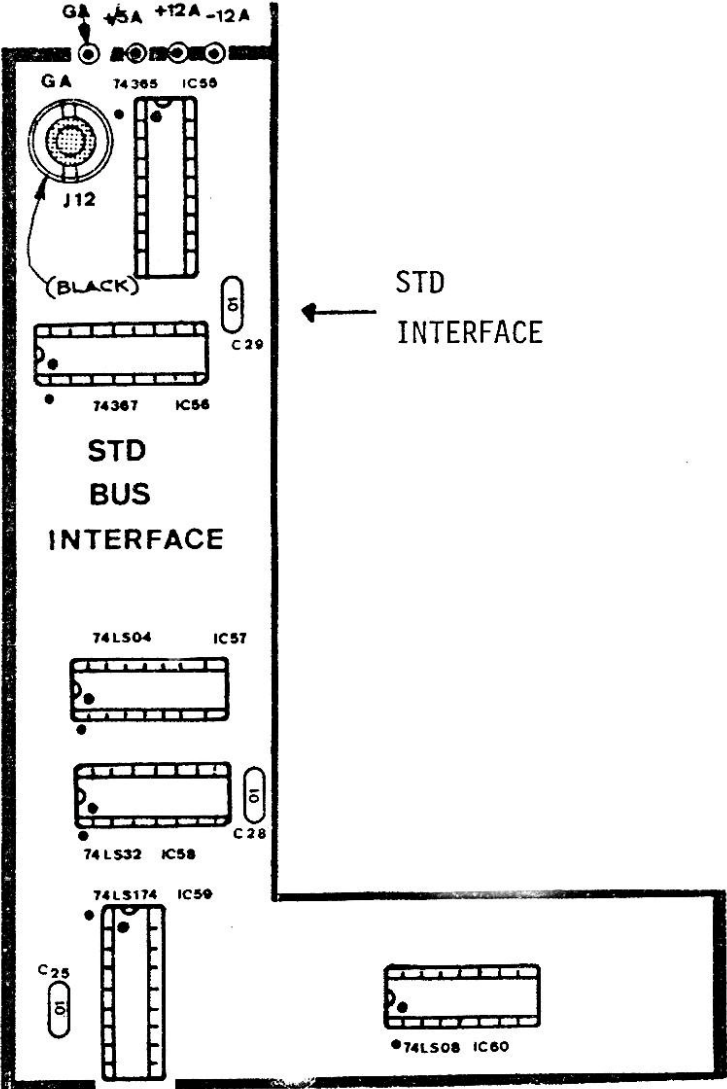


as three parallel output ports (8 LEDs per port) used to display binary data. (An output port is a section of the computer used for sending data to external devices.) The computer has built-in solderless interfacing sockets, which are used to provide convenient access to the signals produced by the MMD-2. These signals are used for interfacing the computer to external circuits, and are discussed in greater detail in E & L's Technibooks Volumes V and VI. Also included are an EPROM programmer, a serial I/O (input/output) port which is used to connect the computer to a teletype or similar terminal device, and a tape recorder interface which is used for saving and loading programs from audio tape cassettes. You will find that it is much simpler to load a program from tape rather than key it in by hand each time you wish to run it. Included as well is an STD interface socket (STD is the designation of a particular bus used in control applications, and is widely used in industry by many manufacturers). The STD bus is used to connect the computer to other external devices. Each of the connections on the STD interface has a specific signal from the computer connected to it which is standardized so that devices using this interface will function with the MMD-2 computer.

The keypads located on the bottom right corner of the computer are used for entering programs and data. Additional uses of the keypads include: examining and changing the computer's memory, executing a stored program, examining and changing the contents of various user storage registers which simulate those inside the microprocessor itself, setting breakpoints which are used in testing the operation of a program, and loading and saving programs on tape. It is not necessary that you understand the operation of the keypads at this time; each of their functions will be explained later.

The RESET button can always be used to return control to you via the keypads. The small eight-element DIP (Dual Inline Package) switch next to the keypad has several uses. Switches one through four, labeled WE0 through WE3, are used to write-protect the RAM. Write-protection means that after a program has been entered into RAM, the computer cannot change (write into) the memory; it can only read it. Switch five, labeled SPARE, has no defined function. Switch six, labeled HEX/OCTAL, is used to select either hexadecimal or octal data entry and display. Switch seven, labeled PUP/RESET, is used to change the function of the RESET button. When switch seven is in the RESET position, the RESET button will perform a system reset. A system RESET allows EXEC C to gain control of the computer, update the displays and wait for input from the keypads. In the PUP (Power UP) position, the EXEC C firmware will perform a complete power-on initialization. The PUP performs all the functions of the RESET and also initializes the USER REGISTERS (explained later) and performs other housekeeping tasks needed to initialize the computer. The PUP is necessary in the event a user program accidentally writes over the system RAM causing EXEC C to malfunction. If this happens, switch seven is placed in the PUP position and the RESET button is pressed. This will perform a complete initialization avoiding the need to turn the computer off, which would destroy a user program in memory. Switch eight, labeled EXEC/USER is used to control the function of output ports 0, 1 and 2. These ports consist of eight LEDs each. This switch controls whether the LEDs are to be used as standard output ports displaying data which has been specifically sent to the output port by a user program, or if they will be used to display the binary information which corresponds to the data on the seven-segment displays which is generated by the system-supplied firmware.

- J9
- J10
- J11
- +5 (A)
- GND(A)
- +12V(A)
- 12V(A)
- 5V
- BA 0
- BA 1
- BA 2
- BA 3
- BA 4
- BA 5
- BA 6
- BA 7
- BA 8
- BA 9
- BA 10
- BA 11
- BA 12
- BA 13
- BA 14
- BA 15
- BD 0
- BD 1
- BD 2
- BD 3
- BD 4
- BD 5
- BD 6
- BD 7
- MEM R
- ACK
- INT
- READY
- INTE
- IN
- OUT
- MEM W
- WAIT
- RESET
- M1
- HALT
- HLDA
- WR
- RD
- IORQ
- MEM RQ
- IOEXP
- MEMEX
- REFRESH
- MCSYNC
- STATUS 0
- BUSAK
- BUS RQ
- NMI RQ
- SYS RESET
- CLOCK
- CNTRL
- PC-0
- PC-1
- RESTART 7
- RESTART 3
- RESTART 2
- RESTART 1
- RESTART 0





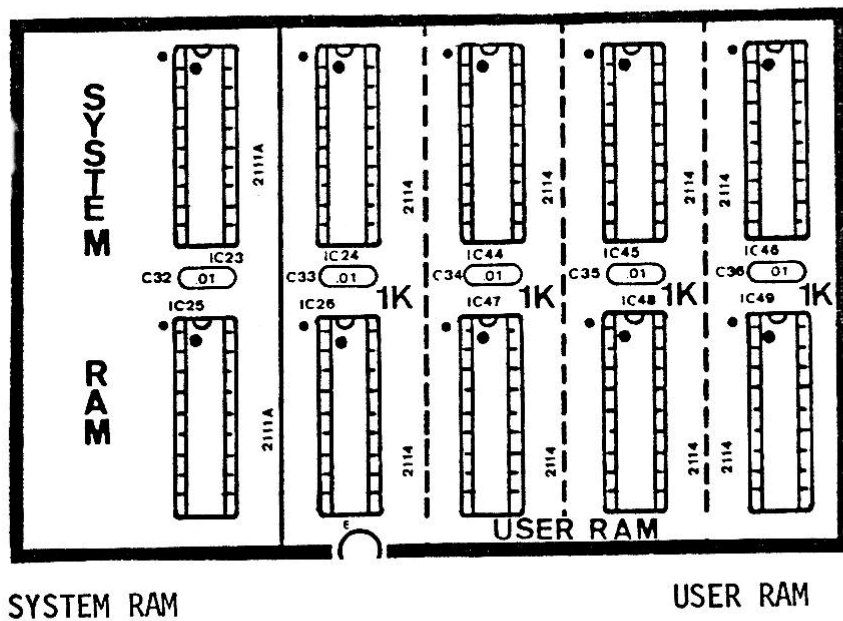
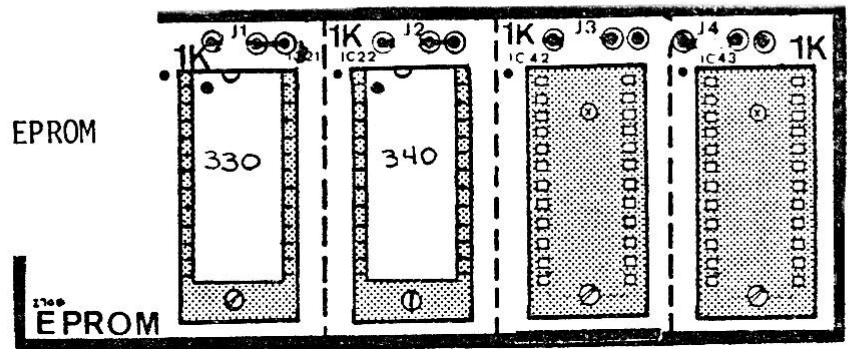
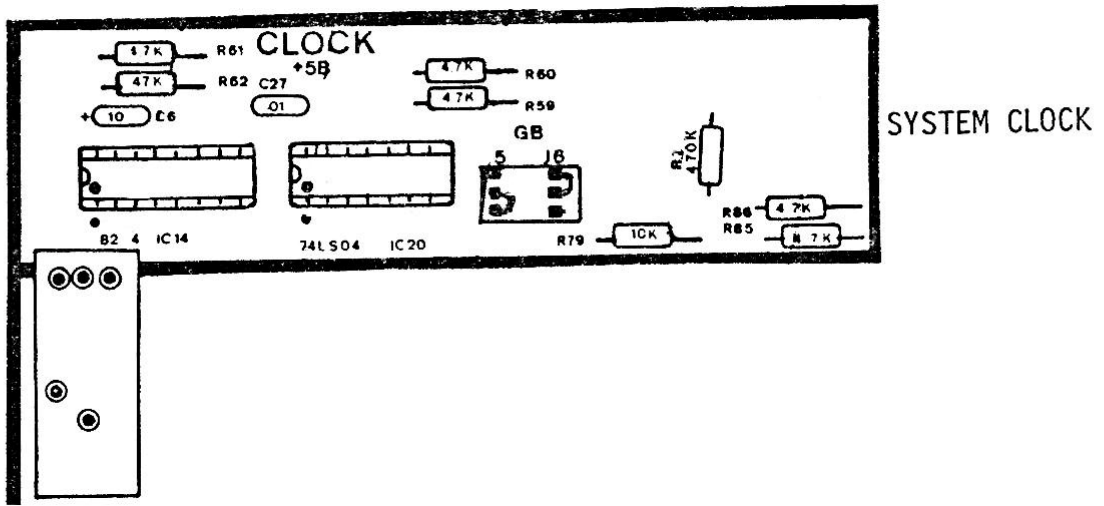
The eight-element DIP switch functions like eight individual SPST switches. They are activated by pressing on the ends of the rockers to switch from one position to the other. For example, to place the HEX/OCTAL switch into the OCTAL position, press down on the right side of the switch.

Three additional LEDs are provided to monitor specific operating conditions of the MMD-2 computer. When the HALT LED is on, it indicates that the computer has executed a HALT (HLT) instruction and that the computer has stopped. The easiest way to return from a halt is to press RESET. You could turn the computer off and then on again, but your program will be lost by this method. HOLD is a condition similar to the HALT condition in which the processor is stopped and will remain inactive while the BUSRQ control line is held low (grounded). Releasing the BUSRQ line will cause the processor to resume normal operations. When the instruction "Enable-Interrupts" (EI) is executed, the ENABLE-INTERRUPTS (INTE) LED will turn on.

These controls and indicators will be discussed in more detail later. On the following pages there are pictures of the MMD-2 computer in which the functional areas are displayed. Each section has a specific purpose. You will gain an understanding of each of the sections and how they function as you progress through this chapter.

#### The Major Computer Blocks Are:

1. POWER SUPPLY. The computer power supply is located inside the chassis. The computer is turned on by the switch on the back left hand side of the chassis. It glows red when turned on. It has been located on the rear of the MMD-2 to prevent accidentally turning the unit off. The power supply converts the 115 or 230 VAC line voltage coming into the computer into the lower DC voltages needed to run the computer. These voltages include +5 VDC at 3 amps of which 500 Ma is available for user experiments. Also supplied are +12 and -12 VDC at 500 Ma of which 250 Ma is available for user experiments. A -5 VDC supply is derived from the -12 VDC supply and is also available for experiments, although its use reduces the 250 Ma available from the -12 VDC supply. There is also a 26 VDC supply for programming EPROMS which is not available for user experiments. All power supply voltages are current limited and short circuit protected. Regulation is within 5% on all supplies, except the 26 VDC supply, which is within 2%.
2. STD INTERFACE. The STD interface is an industry standard interfacing scheme for many different 8-bit computers. This interface standardizes the physical and electrical aspects of modular 8-bit computer plug-in circuit card systems. It provides an orderly interconnection scheme which many manufacturers are currently using. This means that devices manufactured by different companies which employ this STD bus structure can be connected directly to the MMD-2. This feature provides easy expansion of the computer system with additional memory or other special devices used to extend the usefulness of the basic computer.
3. CENTRAL PROCESSING UNIT (CPU). This section is the "heart" of the computer. The 8080A microprocessor and its system controller (8228) act as the central clearinghouse for all information passing through the computer. There are few functions not directly controlled by the CPU.

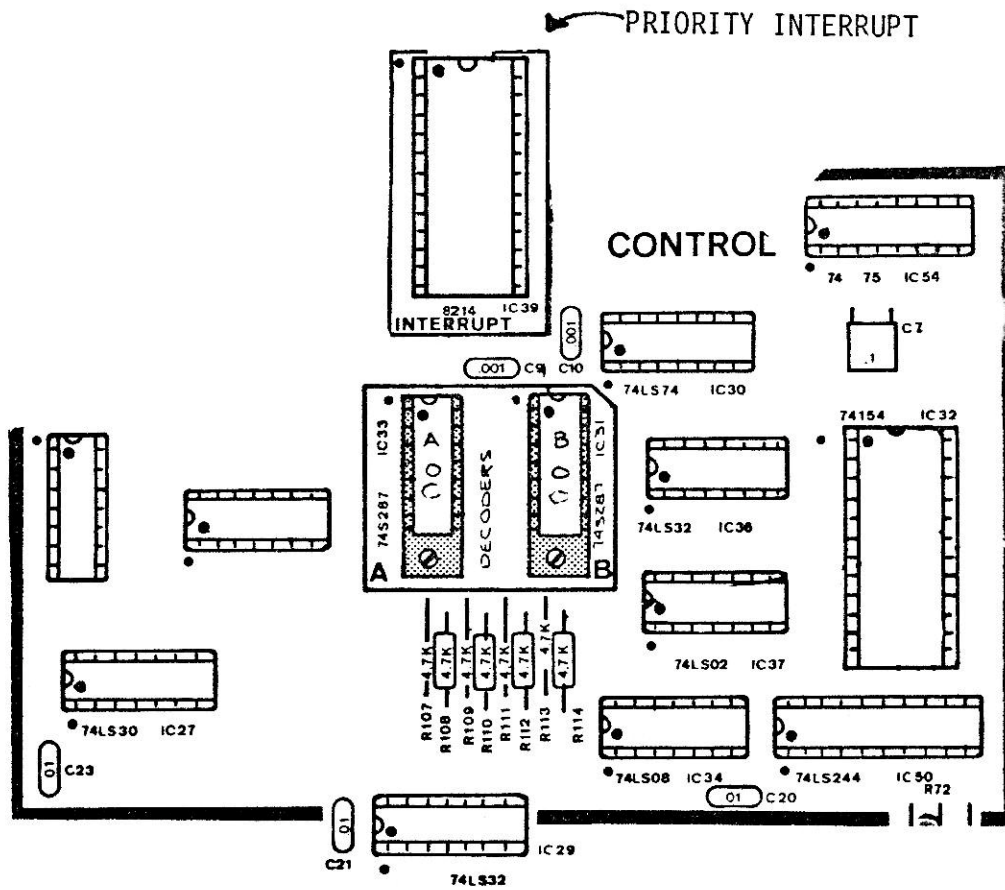
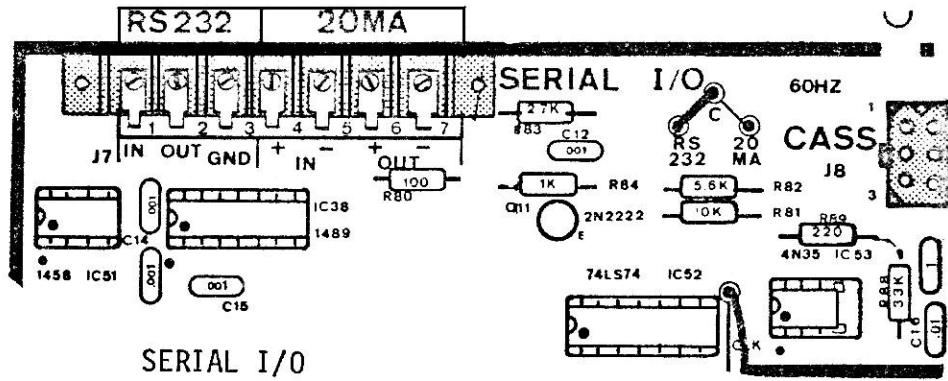




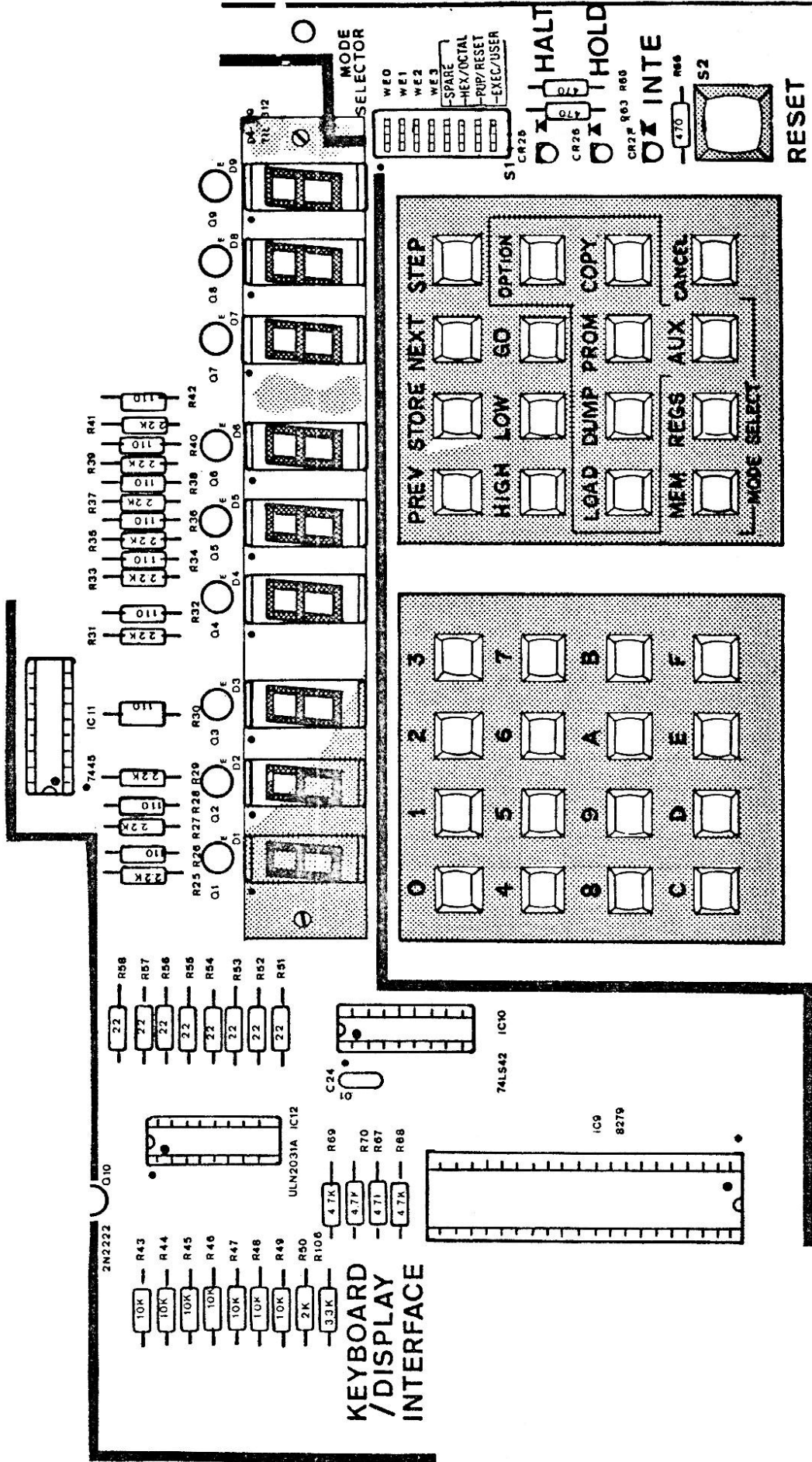
4. CLOCK. The microprocessor's clock produces the timing pulses which keep the entire computer operating in sequence. Clock signals are used to control the operation of the CPU chip, the timing of the serial I/O interface sections and generally to provide standard timing pulses for all sections of the computer to insure that all sections function in synchronization with each other. The quartz crystal in the clock section has a frequency of 6.75 megahertz (Mhz). This is used to control the 8224 clock generator chip. This chip produces the actual clock frequency used by the 8080A microprocessor. The 8224 chip provides two separate clock pulse trains called phase one clock and phase two clock. The crystal (XTAL) frequency is divided by nine to produce an actual clock frequency of 750 kilohertz (Khz). The maximum clock frequency for which the 8080A microprocessor is designed is 2 Mhz which would require an 18 Mhz crystal. The MMD-2 computer is capable of operating at 2 MHz. The slower clock rate was chosen to be compatible with the MMD-2 computer (the original training computer produced by E & L Instruments, Inc.) and E & L's Technibook series of tutorials.
5. EPROM. This is the Erasable Programmable Read Only Memory, which contains the operating system. The operating system is a program which monitors the keypads and the mode selector switches and performs the pre-defined functions assigned to the keypads. The operating system will be discussed later. The program in this EPROM was developed by E & L Instruments, Inc., and is called EXEC C. The remaining one or two sockets in this section are spare, allowing an additional EPROM (or EPROMs) to be installed as needed. The spare EPROM sockets allow you to add your own programs to the system.

Note that an identification tag covers the window area of the EPROM. If you lift the cover paper, you will see the actual integrated circuit chip. The cover is left in place to prevent ultraviolet light from accidentally erasing the program stored inside the EPROM. Both 2708 and 2716 EPROMs can be used in the MMD-2; however, they cannot be inter-mixed. The 2708 EPROM is an eight-bit by 1K device, while the 2716 is an eight-bit by 2K device. This means that twice as much information can be stored in the 2716 EPROM than in the 2708 EPROM. To select the appropriate connections for 2708 or 2716 EPROMs, consult your Reference Manual (E & L Part Number 801-0228).

6. RAM. These Random Access Memory (RAM) devices are the read/write memory devices that will contain the programs you generate while using the MMD-2 computer. RAMs are also known as volatile memory, because when you turn the trainer off, the programs contained in these devices are lost. The programs stored in the EPROM memory are not lost by power failure. The first RAM section is labeled SYSTEM RAM and consists of two 256 by four-bit devices which are connected to produce a "scratch pad" memory of 256 eight-bit words. This area is used by EXEC C and should not be modified by the user. The next section, called USER RAM, consists of two 1K by four-bit RAMs, which provide you with 1024 eight-bit words of storage for your programs. The last three sockets are provided for expansion to 4K (4096 eight-bit words) of user storage. Most MMD-2 units will come from the factory complete with all 4K of RAM in place.



















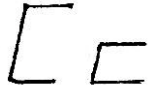














KEYPAD/DISPLAY INTERFACE













7. SERIAL I/O. The MMD-2 computer has two basic methods of communicating with external devices. These methods are called PARALLEL I/O (which will be discussed later) and SERIAL I/O. Many common I/O devices, including TTYs, CRT (cathode ray tube) terminals, tape recorders and printers use serial I/O techniques to interface with the computer. The MMD-2 provides two different types of electrical connections between the computer and peripherals. The first type is called CURRENT LOOP and requires that data signals from the computer be converted to the presence of a 20 Ma current or the absence of that current to represent logic 1 or logic 0 signals. This signal is commonly used to interface with a teletype. The other output signal, RS-232, requires that the digital signals from the computer be converted to a positive voltage level to indicate a logic 1, and to a negative voltage level to indicate a logic 0. The MMD-2 uses +12 and -12 volts for this purpose. RS-232 is commonly used in interfacing to CRT terminals or other devices located within about 10 feet of the computer. CURRENT LOOP can be used to send signals many hundreds of feet. The serial I/O section of the computer also has the interface circuitry to convert digital signals to audio tones, which are used to store data and programs on cassette tape. The audio conversion conforms to the Kansas City Standard, which is one of several common methods used to store data on audio cassette tape.
8. CONTROL. This section contains the circuits for input/output and memory address decoding; that is, these circuits take multiple address and input/output lines and select a specific input port, output port or memory address for reading or writing. The exact method used for address decoding is unique to the MMD-2 and will be discussed in greater detail in the chapter on advanced features.
9. PRIORITY INTERRUPT. This section of the computer contains the 8214 priority interrupt controller chip. This chip and its eight associated inputs are used in processing interrupts. The techniques for interrupt handling will be covered in the advanced features section.
10. KEYPAD AND DISPLAY INTERFACE. This section contains the circuits required to decode the data and function keys of the keypad and the circuits used to encode the data for the seven-segment displays. This section also contains the seven-segment displays. The transistors located above the displays are called the display drivers. These circuits supply the drive current needed by the displays. The displays are called "seven-segment displays" because each display contains seven separate segments as shown in the illustration captioned SEVEN-SEGMENT DISPLAYS (Page 10).

Each segment is independently driven, thus providing the capability to construct the patterns of decimal digits and most alphabetic characters accurately. However, to avoid confusion between certain digits and alphabetic characters, such as B (  ) and 8 (  ), special "codings" are required. Figure 1 shows segment arrangements which are used with the MMD-2. Note that the digits are the same for both octal and hexadecimal. The unique interpretations are required for some alphabetical characters. A complete character set for the seven-segment displays is provided in Appendix IV of the MMD-2 Reference Manual (E & L Part Number 801-0228).



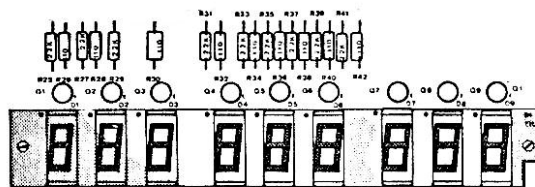
A = 	G = 	M = 	S = 	X = 
B = 	H = 	N = 	T = 	Y = 
C = 	I = 	O = 	U = 	Z = 
D = 	J = 	P = 	V = 	
E = 	K = 	Q = not possible	W = 	
F = 	L = 	R = 		

0 = 	2 = 	4 = 	6 = 	8 = 
1 = 	3 = 	5 = 	7 = 	9 = 

SEVEN-SEGMENT CHARACTER SET

Figure 1.



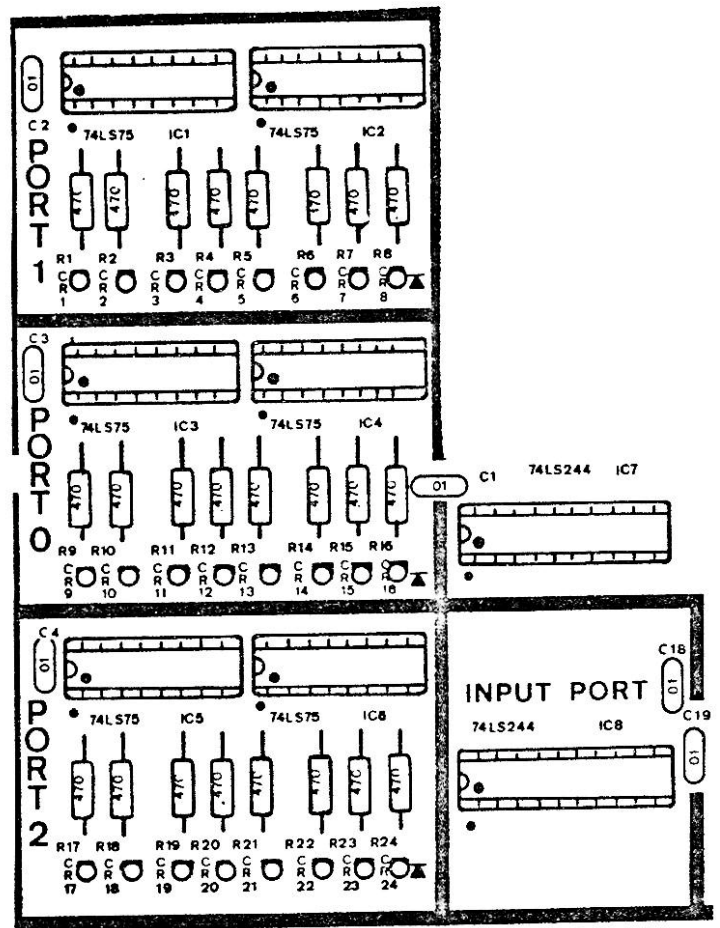
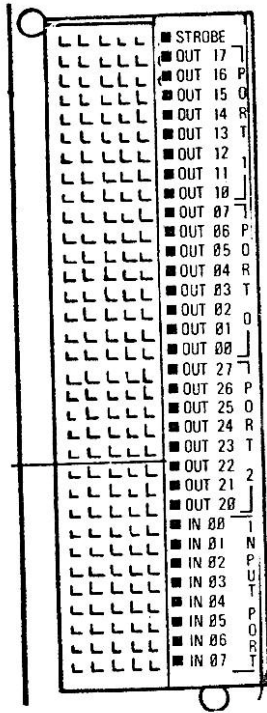
SEVEN-SEGMENT DISPLAYS

11. INPUT PORT. This port is provided to allow the user a convenient method of inputting data from external circuits to the computer. The port can be used as an interface between circuits you construct and the computer.
12. PORT 0, PORT 1, PORT 2. These are the output ports which control the LEDs located in three rows of eight on the lower left-hand side of the computer. The eight connections from each output port are also available on the solderless breadboarding strip on the left side of the computer. These ports are used to send data from the computer to external circuits. When the EXEC/USER switch is in the EXEC position, these LEDs show the binary code for the numbers displayed on the seven-segment displays by the EXEC C firmware. The following is the format for this output:

- \* Port 1 indicates the upper 8 bits of the memory address. It is blank in REGS and AUX mode.
- \* Port 0 indicates the lower 8 bits of the memory address. It is blank in REGS and AUX mode.
- \* Port 2 indicates the 8-bit data word contained in the memory address. It can also indicate the eight-bit data word displayed in REGS and AUX modes.

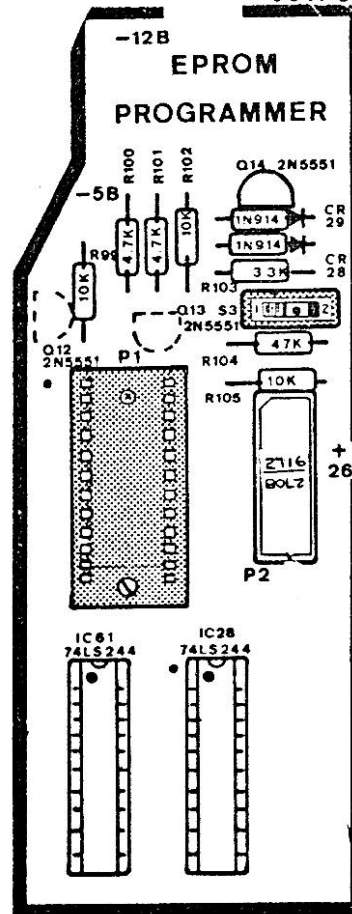
You will notice that the output port LEDs do not change their values when you change the HEX/OCTAL switch. This is because both hexadecimal and octal are different methods of interpreting the same binary information. When the EXEC/USER switch is in the USER position, the LEDs will not change until data is specifically sent to them by your program. When the instructions OUT 0, OUT 1 or OUT 2 are executed, the data from the accumulator in the microprocessor chip will be displayed on the selected output port LEDs and the corresponding data will be available at the electrical connections to the specified port on the solderless plug strip. Note that the least significant bit is on the right and the most significant bit is on the left. When all the LEDs are lit, the code indicated is 377 or FF, depending on which number base you are using, octal or hex respectively.

13. EPROM PROGRAMMER. The EPROM programmer works in conjunction with EXEC C to perform the functions necessary to program and test both 2708 and 2716 EPROMs. The EPROM programmer allows you to: verify that an EPROM is erased, to actually program the EPROM, and to test to make sure that the programming was successful. The EPROM programmer can be used when you have written a program which you want to store on a permanent basis for immediate use. It is much more convenient to store a program, especially a long one, in EPROM rather than on tape. Loading a long program from a tape takes time while accessing it from EPROM requires no loading.
14. FUNCTION KEYPAD. The Function Keypad specifies what functions EXEC C will perform. The following text lists the various functions that are available and explains their uses.



OUTPUT PORT

INPUT PORT



EPROM PROGRAMMER



PREV--This is a decrement address function which selects the memory address one position back from the address being displayed. Each time the key is depressed, the new address and data are displayed. It also selects the previous register in the REGS mode or the previous auxiliary register in the AUX mode.

STORE--Writes (stores) the data entered from the data keys into the RAM memory. It also stores data in the REGS and AUX modes. When STORE is pressed, three dashes are displayed momentarily in the data display to indicate the data was stored properly.

NEXT--This is an increment address function which selects the next sequential memory address to either write or read data. The new address and data are displayed. It also displays the next register in the REGS mode or the next auxiliary register in the AUX mode.

STEP--This mode of operation enables you to see the sequence of instructions as they are executed, and to have a clearer understanding of how your program is functioning.

HIGH--This key is used to enter the high-order 8 bits of an address.

LOW--This key is used to enter the low-order 8 bits of an address.

GO--This key is used to initiate execution of a program. When GO is pressed, execution commences at the address specified.

CANCEL--This key is basically a "clear entry" which returns the displayed data back to the original data if the STORE key has not been depressed.

The next three keys: MEM, REGS, AUX, change the entire behavior of the computer. The keys can be considered mode selectors. Each mode of operation incorporates a different set of functions.

MEM (Memory): This key selects the standard mode of operation in which data can be written into or read from memory at the address specified (displayed). Memory mode is the most common operation mode you will use. This key is also used to return from REGS or AUX mode to MEM mode.

REGS (Register): This key selects Register Mode, in which data will be written into or read from the specified (displayed) register. This mode is used to display or alter the simulated internal registers in the 8080A microprocessor chip.

AUX (Auxiliary): This key selects the auxiliary mode. This mode is used for storing information needed by the advanced features and will be discussed in detail in the section on advanced features.

The following keys are only active when the computer is in the auxiliary mode:

LOAD: This key is used to read program information from a cassette tape and place it in the computer's memory.

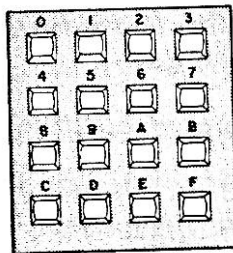
DUMP: The DUMP key is used to store programs on cassette tape for later recall.

PROM: This key is used to program EPROMs inserted in the EPROM programmer socket.

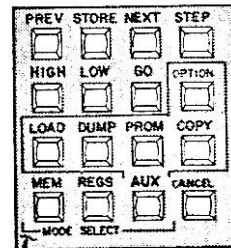
COPY: The COPY key is used to copy a block of memory from one location to another.

OPTION: This key has many functions in connection with the AUX MODE. These functions include setting memory addresses, the length of data to be copied or moved, the type of EPROM to be programmed, the address of the breakpoint, the function of the STEP key, and the BAUD rate of the serial I/O port. These functions will be discussed in detail later.

15. DATA KEYPAD. The data keypad allows you to enter numeric information. When the HEX/OCTAL switch (switch number eight of the mode selector) is in the OCTAL mode, the DATA INPUT keys zero through seven are usable and are interpreted as octal digits. The remaining keys (eight through F) will have no effect. When the HEX/OCTAL switch is in the HEX mode, the DATA INPUT keys zero through F are useable and are interpreted as hexadecimal digits.



DATA INPUT KEYPAD



FUNCTION INPUT KEYPAD

SECTION 2  
BASIC FEATURE EXPERIMENTS

The experiments contained in this section are designed to help you become familiar with the operation of the MMD-2 microcomputer. They will help you understand the operation of the basic keypad functions and demonstrate how to enter and run programs. All you will need to perform these experiments is the MMD-2 microcomputer.

OBJECTIVES

1. To provide you with an understanding of the operation of the MMD-2 computer.
2. To help you become familiar with the functions available in the MEM (Memory) Mode.
3. To demonstrate how to enter, correct and run machine language programs.
4. To gain familiarity with the way the MMD-2 displays data.

SUMMARY OF EXPERIMENTS IN THIS SECTION

1. Demonstration of the initial operating condition of the computer when it is turned on. The function of the HEX/OCTAL switch, EXEC/USER switch and RESET button will be explained. Operation of the output port LEDs and seven-segment displays will also be demonstrated.
2. Operation of the DATA KEYPAD, and MEM KEY will be demonstrated.
3. Operation of the NEXT, PREV and STORE keys will be demonstrated, as well as the function of the MEMORY PROTECT (WE0 - WE4) switches.
4. The functions of the HIGH and LOW keys will be investigated to show how they are used to access different memory locations.
5. A simple program will be entered into the computer's memory. The GO and STEP keys will be used to run the program first at full speed and then to execute it one step at a time.
6. The function of the REG MODE will be demonstrated. You will learn how to view and alter the contents of the 8080A's internal registers while stepping through the program.



## EXEC B EXPERIMENTS

### EXPERIMENT 1

The purpose of this experiment is to help you become familiar with the operation of the following controls on the MMD-2: RESET, HEX/OCTAL and EXEC/USER. You will also become familiar with the seven-segment displays and the output port LEDs.

STEP 1. Set the MODE SELECTOR switches as follows: All switches(1-7) should be set by pressing down on the RIGHT side of the switch, except for switch eight-- you should press on the LEFT side of this switch. This is considered the NORMAL operating position. Switches one through four write enable the RAM memory, switch five is not used, switch six sets the display format to OCTAL, switch seven implements a normal system RESET, and switch 8 copies the seven-segment information in the output port LEDs.

STEP 2. Apply power to the MMD-2. You will note that the power switch (located on the back panel) will glow red when the unit is operating. You should see "EXEC READY" on the seven-segment displays, and the three output ports will also have some of the LEDs lit. The meaning of all these lights will be explained as we go along.

Press the MEM key on the function keypad.

The binary output port LEDs are arranged in the following order

<u>Most Significant Digit</u>		<u>Least Significant Digit</u>	
00	000	000	HI ADDRESS
00	000	000	LO ADDRESS
00	000	000	DATA

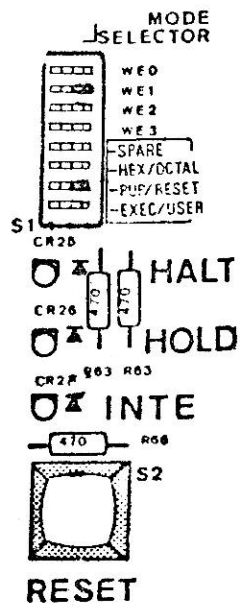
When the MMD-2 is turned on, the system begins to perform a program which is in its EPROMs, and is called EXEC C. This program's initial job includes a great many "housekeeping" functions, which set up the computer and get it ready for your use. One of these functions is setting the starting program address to the specific RAM location 003 000 (octal), 03 00 (hex). This address will be displayed on the seven-segment displays and the output ports.

Question: What does the high address display equal?

Answer: The value is 003 (03 hex).

Question: What does the low address display equal?

Answer: The value is 000 (00 hex).





Question: What does the data display equal?

Answer: The data display can have any value. The address 003 00 (03 00 hex) is one location in the computer's RAM, and because the RAM has not been preprogrammed, it can have any value when the computer is turned on. It will be your job to replace the random numbers in these memory locations with specific numbers which will form your programs.

STEP 3. Change the HEX/OCTAL switch to the HEX position by pressing down on the left side of the switch.

Question: Did you observe any change in the displays?

Answer: No. The displays will change to the new number system after another key is pressed.

STEP 4. Press RESET.

Question: What does the high address display equal?

Question: What does the low address display equal?

Question: What does the data display equal?

Answers: We observed the high address displayed 03., the low address displayed 00. We observed a hex number with two digits in the data display. This hex number will have the same binary value that the previous octal number had before RESET was pressed.

STEP 5. Look at the output port LEDs; some of the LEDs will be lit in Ports 1 and 2.

Question: Mark which LEDs are lit in Port 1: 00 000 000 (LSB).

Question: Mark which LEDs are lit in Port 0: 00 000 000 (LSB).

Question: Mark which LEDs are lit in Port 2: 00 000 000 (LSB).

Question: Do the values indicated in the output port LEDs agree with the hex values indicated by the seven-segment displays? Port 1 is the HI address, Port 0 is the LO address and Port 2 is the DATA address.

Answer: Port 1 is 00 000 011. The number "1" indicates the LED is lit, representing a logic one. The number "0" indicates a logic zero, with the LED off.

Answer: Port 0--00 000 000.

Answer: Port 2--The value displayed here will depend on what value data is stored in location 03 00 H.

Answer: The seven-segment displays only encode the binary data to make it more readable. The values indicated on the output port LEDs and the seven-segment displays should agree.

STEP 6. Return the HEX/OCTAL switch to the OCTAL position and press RESET.

Question: Did the seven-segment displays change?

Answer: Yes, we observed that the display is again indicating octal values.

Question: Did the output port LEDs change?

Answer: No, the binary values are constant; HEX and OCTAL are alternate ways of displaying the same binary numbers.

STEP 7. Change the position of the EXEC/USER switch to USER by pressing down on the right side of the switch. Press RESET.

Question: What change in the output port LEDs did you observe?

Answer: All the LEDs are off. Now the output port LEDs are under user program control and will only display numbers when specifically instructed to do so by a program you write.

STEP 8. Return the EXEC/USER switch to the EXEC position.

## EXPERIMENT 2

This experiment will demonstrate the operation of the DATA KEYPAD and the MEM key.

STEP 1. Set switches to their normal starting positions and turn the MMD-2 computer off. Then wait two seconds; turn the computer on.

STEP 2. Press the MEM key on the FUNCTION INPUT keypad. The MEM key selects the MEMORY mode of operation. This enables you to read/write data into the computer's RAM.

STEP 3. Press the 0 key on the DATA INPUT KEYPAD three (3) times.

Question: Describe what happened to the seven-segment displays and the output port 2's LEDs.

Answer: We observed that each time the zero key was pressed an additional zero appeared in the data display and the corresponding LEDs when off on Port 2.

STEP 4. Press the 3 key, then press the 2 key, then press the 7 key.

Question: What number is displayed in the data display?

QUESTION: What is the binary code displayed in Port 2? 00 000 000.

Answers: We observed the number 327 on the data display, and Port 2 looked like this: 11 010 111. The LEDs indicate the binary code represented by the octal value 327.

You have now selected one byte of data which could be entered into the system's memory.

STEP 5. Change the HEX/OCTAL switch to the HEX position and press RESET. You are now ready to enter code in the HEX mode.

STEP 6. Press the DATA keys D, 7.

Question: What number is displayed in the seven-segment displays, and what is the pattern of the LEDs on Port 2? 00 000 000.

Answer: We observed the values D 7 on the data display, and the LED pattern looked like this: 11 010 111. This is the same pattern displayed when we entered 327 in octal. Remember, whichever method you use to enter data (hex or octal) the result is an identical binary bit pattern for the data.

STEP 7. Return the HEX/OCTAL switch to the OCTAL position.

STEP 8. Press the DATA INPUT keys 4, 1, 3 (in that order).

Question: What is the code displayed on the seven-segment displays, and on output Port 2? 00 000 000.

Question: Why is the number on the seven-segment displays and on Port 2 different than what you entered?

Answers: We observed the number 013 on the seven-segment display, and the following pattern of LEDs were lit on Port 2: 00 001 011.

A complete octal digit requires 3 bits. In an eight-bit computer, the eight bits are broken down into two groups of three bits each, leaving two bits left over to form an octal digit. Using 2 bits, the only octal numbers you can generate are 0, 1, 2 and 3. The numbers between 400 and 777 require 9 bits. Since this is an 8-bit system, the ninth (or most significant bit) is lost. You cannot enter the octal codes for numbers between 400 and 777.

### EXPERIMENT 3

This experiment will help you become familiar with the process of storing data into the system's memory, how to view that memory, and how that memory protect switches function.

STEP 1. Verify that all mode selector switches are in the normal positions and apply power to the system.



STEP 2. Press the MEM key on the FUNCTION INPUT KEYPAD to place the computer into the MEMORY mode.

STEP 3. Enter the digits 257 on the DATA KEYPAD.

STEP 4. Depress the STORE key on the FUNCTION INPUT KEYPAD.

Question: What numbers are displayed on the seven-segment displays?

Answer: We observed the numbers 003 000 257.

Pressing the STORE key actually entered the value 257 (AF hex) into the system's memory at location 003 000. When the STORE key was pressed, the DATA display momentarily flashed three dashes to confirm that the data was properly stored. If the data was not stored (for example, if you attempt to store data where there is no memory) then (NOT.STORED) will be displayed on the seven-segment displays.

STEP 5. Press the NEXT key on the FUNCTION INPUT KEYPAD.

Question: What do the seven-segment displays indicate now?

Answer: We observed 003 001 ????. The value in the data display is any random number, because we haven't stored any number in that location.

STEP 6. Enter the number 074 on the DATA INPUT KEYPAD. Press STORE. This procedure will enter the number 074 into memory location 003 001.

STEP 7. Press NEXT to increment the memory address.

STEP 8. In a similar manner, enter the numbers 323, 002, 303, 001 003, by entering the number and pressing STORE, NEXT. At this point, you have entered a short program into the system's memory.

Question: What final values are displayed on the seven-segment displays?

Answer: We saw 003 007 ???.

STEP 9. In this step we will use the PREV key on the FUNCTION INPUT KEYPAD to review the data stored in memory. Press PREV.

Question: What do the seven-segment displays show?

Answer: We saw the number 003 006 003. This shows the address of the last memory location our program used, 003 006, and the value of the data stored there, 003.

STEP 10. Each time you press the PREV key, you should see the address value decrease by one, and the data byte stored at that location. Press the PREV key six (6) times. At this point, you should be back at location 003 000.

STEP 11. Press PREV one more time.

Question: What number will be displayed in the address section?

Answer: We observed 002 377 ???, which is the next lower address and the data in the system's memory.

STEP 12. Press the NEXT key on the FUNCTION INPUT KEYPAD.

Question: What does the display show now?

Answer: The display indicates the number 003 000 257.

STEP 13. Press the NEXT key several times and verify that each time it is pressed, the address is incremented by one and the data display shows the data stored in that location.

The PREV and NEXT keys provide you with an easy method for looking through a program to check that data was entered properly or to view a section of the program. If you discover an error while looking through your program, enter the correct code and then press STORE. The new data will replace the old data.

STEP 14. Turn the system off for about ten seconds and then turn it back on. Look through your program starting at 003 000.

Question: What values are stored in the locations 003 000 to 003 006?

Question: What happened to your program?

Answers: The values stored where your program was are now random values. Your program was lost when power was removed from the system.

STEP 15. Change the HEX/OCTAL switch to HEX and press RESET.

Question: What does the display show?

Answer: The display shows 03.00. ??. Note: ?? represents the hex code corresponding to the value stored at location 03 00.

STEP 16. Enter the following codes by first entering the numbers on the DATA INPUT KEYPAD and then pressing STORE, NEXT: AF, 3C, D3, 02, C3, 01, 03. You have entered the same program as before using HEX notation instead of OCTAL. Press PREV until the address is 03 00 and verify that each byte in the data display is correct.

STEP 17. Return the HEX/OCTAL switch to the OCTAL position and press RESET.

STEP 18. Using NEXT, look through the program and verify that the octal values are the same as were entered previously.

Question: If you find a difference between these octal values and the ones you originally entered, what could be a cause of the difference.

Answer: Possibly you entered one or more digits incorrectly. It is most important that you use care in entering programs. One small error will prevent the program from functioning properly.



STEP 19. Press RESET.

STEP 20. Press down on the left side of the first write protect switch labeled WE0 to write protect the first 1K of RAM. The purpose of these four switches is to prevent the microprocessor from writing any data into memory. Write protecting prevents the accidental erasure of your program through a programming error. One possible error is your program writing useless data over itself. Write protection is used from time to time to protect a program entered into memory which hasn't been "debugged" (corrected). Debugging is the process of checking and testing a program to verify that it is functioning correctly. Write protect will not protect a program from erasure if the computer is turned off; it only protects from accidental writing of data by the processor.

STEP 21. Using the STORE key, enter 000.

Question: What did you observe in the display?

Answer: We observed (NOT.STORED). When the memory is write protected, it is impossible to store new values. It is possible, however, to restore the value already at a memory location.

Each memory protect switch protects a group 1024 memory locations.

<u>SWITCH</u>	<u>MEMORY PROTECTED (OCTAL)</u>	<u>MEMORY PROTECTED(HEX)</u>
WE0	000 000 - 003 377	00 00 - 03 FF
WE1	004 000 - 007 377	04 00 - 07 FF
WE2	010 000 - 013 377	08 00 - 0B FF
WE3	014 000 - 017 377	0C 00 - 0F FF

The message (NOT.STORED) will also be displayed if you try to store data in non-existent memory or different data in EPROM memory.

STEP 22. Return switch WE0 to its former position.

#### EXPERIMENT 4

The purpose of this lab is to demonstrate the function of the HIGH and LOW keys. The HIGH and LOW keys are another address selection method rather than starting from 003 000 and pressing NEXT or PREV to go to the desired address. If you have to move more than a very few addresses, these methods become tedious at best.

The MMD-2, as well as most other 8-bit computers, is capable of directly addressing 65,536 different memory locations. The range is from 000 000 to 377 377 (00 00 to FF FF HEX). To address this range, the microprocessor needs 16 bits of address data. Usually, this data is in the form of two eight-bit bytes, which are treated as one 16-bit word by the system. The MMD-2 allows us to enter an address from the

keypads using the HIGH and LOW keys. The HIGH key enters the eight most significant bits of the address, and the LOW key enters the eight least significant bits.

STEP 1. Make sure that all the switches on the system are set in the normal positions and apply power to the computer.

STEP 2. Enter the number 002 (02 HEX) from the DATA INPUT KEYPAD.

STEP 3. Press HIGH on the FUNCTION KEYPAD.

Question: What change did you see in the display?

STEP 4. Enter the number 100 (40 HEX) from the DATA INPUT KEYPAD.

STEP 5. Press LOW on the FUNCTION KEYPAD.

Question: What change did you see in the display?

Answers: After pressing HIGH, we saw the HIGH section of the address display change from 003 (03 HEX) to 002 (02 HEX). There was a change in the number in the DATA display, but this is just random data. When we pressed LOW, the low section of the address display changed to 100 (40 Hex). The data display will again display the random number stored at the address 002 100 (02 40 hex).

STEP 6. Using the STORE and NEXT keys, enter the following numbers starting at location 002 100 (02 40 hex): 000 001 002 003 004 005 (01 02 03 04 05 hex) in sequential locations.

STEP 7. Return to your starting address 002 100 (02 40 hex) using the procedure outlined in steps 2 through 5. Look at the output port LEDs. In Port 0 you should see 00 000 010, Port 1 should be 01 000 000 and Port 2 should be 00 000 000. The ports again display the equivalent binary data represented by the seven-segment displays. Use the NEXT key to examine the memory locations where you stored the numbers. While you examine the memory locations, look at the output ports to verify that they are changing and displaying the same numbers as the display.

Question: Were the numbers stored properly?

Answer: We observed the correct numbers. The technique of using the HIGH and LOW keys to specify an address allows the programmer the flexibility to go to any memory location in the computer's memory conveniently.

STEP 8. Press RESET.

Question: What happened to the address display?

Answer: The display returned to 003 000 (03 00 hex).



STEP 9. Change the HEX/OCTAL switch to the HEX position and press RESET.

STEP 10. Set up the number 3F, using the DATA INPUT KEYPAD.

STEP 11. Press HIGH.

STEP 12. Set up the number 00, using the DATA INPUT KEYPAD.

STEP 13. Press Low.

Question: What number is displayed on the address display?

Question: What number is displayed on the data display?

Answers: We observed the numbers 3F 00 on the address display, and an FF on the data display. Look at the output ports. The LEDs on Ports 0 and 1 are the binary values for the address display (00 111 111, 00 000 000). Port 2 has the binary value of the data displayed on it (11 111 111). You will note that all the LEDs are on. This is the normal indication you will see whenever you try to display nonexistent memory. In the MMD-2, only a small portion of the total address space actually has memory devices (RAM or EPROM) installed. The rest of the address space is available for future expansion through the use of the STD BUS and external memory cards.

STEP 14. Return the HEX/OCTAL switch to the OCTAL position.

## EXPERIMENT 5

In this lab you will enter a small program and observe how it operates both while running at full speed and stepping through it one step at a time. The SINGLE STEP KEY is a very useful for debugging your programs. When you run a program at full speed, it is difficult to find a mistake. Using the single step key, you can run the program one instruction at a time and see exactly what is happening.

The STEP key allows a user to execute a program in the MMD-2 which simulates the execution of all 8080 instructions, with the exception of the HALT instruction, which when executed performs a true HALT function. The software single stepper is quite different from the hardware single stepper described in Technibook Vol. V, Chapter 11, Experiment 5. The software single step is more useful debugging software, than the hardware single step, which is commonly used in testing computer hardware.

### PROGRAM

ADDRESS	CODE	LABELS	INSTRUCTION	OPERAND	COMMENTS
003 000	257		XRA	A	;CLEAR THE ACCUMULATOR
003 001	323	LOOP:	OUT		;OUTPUT TO DATA PORT 2
003 002	002		002		
003 003	074		INR	A	;INCREMENT A REGISTER
003 004	303		JMP		;GO BACK AND DO IT AGAIN
003 005	001		LOOP		
003 006	003		--		



This listing is called an ASSEMBLY LISTING, and is the standard format used to describe an assembly language program. The first two columns contain the address of the instruction. The next column is the numeric value of the instruction. NOTE: An instruction can be 1, 2 or 3 bytes in length. The next column is reserved for program labels. This column may be empty. The next column contains any operand or address needed by the instruction. This column may be empty. The last column provides the programmer with a running commentary of what the program is doing and what is the function of each instruction.

### Entering a Program

Up to now, we have been entering data into the system, but we have not actually entered and run a program.

STEP 1. Change the EXEC/USER switch to USER.

STEP 2. Press RESET.

Question: What change has taken place in the output port LEDs?

Answer: All the LEDs have gone off. The ports are now controlled by the user's program.

STEP 3. Store the number 257 (AF hex) into the RAM memory at location 003 000 (03 00 hex).

Question: How do you store the number?

Question: How do you increment to the next memory location?

Answers: To store the number, press STORE. To advance to the next memory location, press NEXT.

STEP 4. Enter the following numbers sequentially into memory starting at location 003 001 (03 01 hex): 323, 002, 074, 303, 001, 003 (D3, 02, 3C, C3, 01, 03 hex).

STEP 5. Verify the program. Using the HIGH and LOW keys, set the computer to the beginning of the program (location 003, 000; 03 00 hex). It is good practice to step through a newly entered program to verify that the data was entered correctly. To verify the program you just entered:

STEP 6. Does the DATA display equal 257 (AF hex)?

STEP 7. Press NEXT.

Question: Does the DATA display equal 323 (D3 hex)?

STEP 8. Press NEXT.

Question: Does the DATA display equal 002 (02 hex)?

STEP 9. Press NEXT.

Question: Does the DATA display equal 074 (3C hex)?

STEP 10. Press NEXT.

Question: Does the DATA display equal 303 (C3 hex)?

STEP 11. Press NEXT.

Question: Does the DATA display equal 001 (01 hex)?

STEP 12. Press NEXT.

Question: Does the DATA display equal 003 (03 hex)?

Answers: If any of the above answers was NO, you have a code entry error. To correct a code entry error, you have several options available to get to the address that contains the error.

These include:

-Depressing the NEXT key to advance to the address of the incorrect data.

-Depressing the PREV key to back up to the address of the incorrect data.

-Selecting the HIGH and LOW address code of the address of the incorrect data.

NOTE: If the address to be selected has the same high-order address (i.e., 003, 03 hex) you only have to select the low order address byte. For example, to change the code at address 003 002 (03 02 hex) to read 001 (01 hex) instead of 002 (02 hex) perform the following two steps:

STEP 13. Enter 002 (02 hex), press LOW.

Question: What do the displays read now?

Answer: 003 002 002 (03 02 02 hex).

STEP 14. Enter 001 (01 hex), press STORE.

You have now replaced the data 002 (02 hex) with the data 001 (01 hex).

Return the data to 002 (02 hex) using the same process.

It should be mentioned that this method for correcting a program works as long as you are only exchanging one byte for another. If you have accidentally left a byte out of your program, the method for correcting it is more complex. To accomplish this, you have to go to the address where the missing byte has to be stored and enter the byte. Next, you have to reenter the remainder of your program, because the remainder of the program now has to be shifted one higher memory location to

make room for the missing byte. The problem becomes even more complex if you have accidentally omitted a program step when you originally coded your program. If that has occurred, then all jump and call instructions in the program will have to be checked and the addresses possibly recalculated. This type of problem is generally minimized through the use of a program called an ASSEMBLER, which automatically makes this type of address change to correct for modification of the program.

STEP 15. You will now execute the program you have entered and verified. This program is a simple routine which counts in binary and displays the binary number on output Port 2.

Question: What is the starting address for this program?

Answer: 003 000 (03 00 hex).

Question: The easiest method to get to this address would be to press the \_\_\_\_\_ button. Two additional methods would be to key \_\_\_\_\_ Low or depress the \_\_\_\_\_ key.

Answer: RESET, 000, PREV (until you arrive at the starting address).

STEP 16. Press RESET GO.

Question: Describe what happened.

Answer: (003.000.GO) was displayed and output Port 2 shows all LEDs glowing. The program is counting in binary and displaying the numbers on Port 2. The values are changing too rapidly for you to see the individual numbers.

Question: How do you terminate the operation of the program?

Answer: Press RESET. Pressing RESET will always terminate the operation of any program and restart EXEC C. Your program will still be stored in the computer and can be executed again by pressing GO.

If you are using the system in the HEX mode of operation, the steps followed to enter and run a program are the same. The only difference is that the HEX/OCTAL switch will be in the HEX position.

To actually see how a program functions, the system has a single step key which gives you the opportunity of running your program one instruction at a time. To experiment with the SINGLE STEP key, we will use the same program we have already entered.

STEP 17. Press RESET. The display should show 003 000 257 (03 00 AF hex), which is the first instruction in the program.

STEP 18. Press STEP.

Question: What is indicated on the displays?

Answer: The displays indicate 003 001 323 (03 01 D3 hex). The first instruction, XRA A, 257 (AF hex) was performed. This instruction is one method of putting the value zero in the



system's accumulator.

STEP 19. Press STEP.

Question: What happened to the display?

Answer: Now the display indicates 003 003 074 (03 03 3C hex). The instruction OUT 2, 323 002 (D3 02 hex) was performed. This instruction took the value in the accumulator and displayed it on output Port 2's LEDs. Since there was a zero in the accumulator, all the LEDs remain off.

STEP 20. Press STEP.

Question: What does the display indicate now?

Answer: The display indicates 003 004 303 (03 04 C3 hex). The instruction INR A, 074 (3C hex) was performed. This instruction adds one to the accumulator.

Question: What value will be in the accumulator?

Answer: 001 (01 hex).

STEP 21. Press STEP.

Question: What does the display indicate?

Answer: We saw the value 003 001 323 (03 01 D3 hex). The system executed the JMP LOOP, 303 001 003 (C3 01 03 hex) instruction. This instruction caused the computer to jump from location 003 004 (03 04 hex) to the location 003 001 (03 01 hex).

STEP 22. Press STEP.

Question: What does the display indicate?

Answer: The display indicates 003 003 074 (03 03 3C hex).

Question: What does Port 2 indicate?

Answer: The least significant bit on Port 2 is lit. Again the value of the accumulator (now 001, 01 hex) has been moved to the output port and displayed.

STEP 23. Continue to press the STEP key. Each time you go completely through the program, the value indicated on output Port 2 will increase by one. This program is an example of an infinite loop, because the program will continue doing the same thing as long as it is allowed to run.

When you are using the STEP key, it is important to make sure that you are executing a valid 8080 instruction. The STEP key will not execute any of the undefined opcodes. These opcodes are: 010, 020, 030, 040, 050, 060, 070, 313, 331, 335, 355, 375 (08, 10, 18, 20, 28, 30, 38, CB, D9, DD, ED, FD hex). If you attempt to execute any of these codes, the seven-segment displays will have the address of the undefined instruction and the data



display will show (ERR.). A second important point is to make sure that you do not execute the second or third byte of a multi-byte instruction or a stored data byte. In this case, the STEP program will execute the data byte as if it were an instruction with unpredictable results.

You will now change the previous program. This program will be longer than the last program and is more complex, consisting of a main routine and a subroutine. The program will operate like the last one; however, we will add a subroutine called DELAY. The purpose of this routine is simply to waste time. Each time the program enters the DELAY routine, the system will waste 10 milliseconds (msec) by executing a series of instructions in a loop. After the subroutine has been executed, the routine will jump back to the main program and continue.

#### MAIN PROGRAM

```

003 000 257      XRA  A      ;CLEAR ACCUMULATOR
003 001 323 LOOP:  OUT          ;OUTPUT TO PORT
003 002 002      002          ;NUMBER 2
003 003 074      INR  A      ;INCREMENT REGISTER A
003 004 315      CALL         ;GO TO SUBROUTINE
003 005 030      DELAY        ;LO SUBROUTINE ADDRESS
003 006 003      --          ;HI SUBROUTINE ADDRESS
003 007 303      JMP          ;JUMP TO LOOP
003 010 001      LOOP        ;LO JUMP ADDRESS
003 011 003      --          ;HI JUMP ADDRESS

```

#### TIME DELAY SUBROUTINE

```

003 030 365 DELAY:  PUSH PSW   ;STORE THE A REGISTER AND FLAGS
                   ;ON THE STACK
003 031 325      PUSH D      ;STORE THE D&E REGISTER PAIR ON
                   ;THE STACK
003 032 021      LXI D      ;ENTER THE FOLLOWING DATA IN THE
                   ;D&E REGISTER PAIR
003 033 046      --          ;LOAD THE E REGISTER WITH 046
003 034 001      --          ;LOAD THE D REGISTER WITH 001
003 035 033 LOOPER: DCX D      ;DECREMENT D&E BY ONE.
003 036 172      MOV A,D     ;MOVE THE CONTENTS OF REGISTER D
                   ;TO THE ACCUMULATOR
003 037 263      ORA E      ;LOGICALLY OR THE CONTENTS OF
                   ;REGISTER E WITH THE ACCUMULATOR
003 040 302      JNZ         ;IF D&E NOT EQUAL TO 0 GO BACK

```

```

003 041 035      LOOPER      ;TO LOW-ORDER ADDRESS
003 042 003      --          ;HIGH-ORDER ADDRESS
003 043 321      POP D        ;RESTORE D&E TO ORIGINAL VALUE
003 044 361      POP PSW     ;RESTORE THE A REGISTER AND
                          ;FLAGS
003 045 311      RET          ;GO BACK TO MAIN PROGRAM

```

Question: What is the starting address of the main program?

Answer: 003 000 (03 00 hex).

Question: What is the starting address of the time delay subroutine?

Answer: 003 030 (03 18 hex).

You will now enter and execute the program. We will delete the specific steps required to enter the program. This will be the way you will enter programs during the rest of the course. If you are not sure now to enter the program, review Experiment 3.

STEP 24. Select the main program starting address 003 000 (03 00 hex).

STEP 25. Enter the following code:

HI	LO	OCTAL	HI	LO	HEX
003	000	257	03	00	AF
	001	323		01	D3
	002	002		02	02
	003	074		03	3C
	004	315		04	CD
	005	030		05	18
	006	003		06	03
	007	303		07	C3
	010	001		08	01
	011	003		09	03

STEP 26. Select the starting address of the subroutine 003 030 (03 18 hex).

STEP 27. Enter the following program:

<u>HI</u>	<u>LO</u>	<u>OCTAL</u>	<u>HI</u>	<u>LO</u>	<u>HEX</u>
003	030	365	03	18	F5
	031	325		19	D5
	032	021		1A	11
	033	046		1B	26
	034	001		1C	01
	035	033		1D	1B
	036	172		1E	7A
	037	263		1F	B3
	040	302		20	C2
	041	035		21	1D
	042	003		22	03
	043	321		23	D1
	044	361		24	F1
	045	311		25	C9

STEP 28. Look through the program and subroutine to verify that they were entered properly.

STEP 29. Select the main program starting address 003 000 (03 00 hex) and execute the program by pressing G0.

Question: What is happening to the LEDs on Port 2?

Answer: We observed the LEDs incrementing in the binary counting pattern. If you didn't observe this result, check your program and subroutine to make sure that it was entered correctly.

STEP 30. To change the delay time of the LEDs, modify the contents of address location 003 034 (03 1C hex). Decreasing this number reduces the delay while increasing this number increases the delay.

Question: If the number at location 003 034 (03 1C hex) were changed from 001 to 010 (01 to 08 hex), will the LEDs count faster or slower?

Answer: The LEDs will count slower because we have increased the time delay between counts.

STEP 31. Press RESET and use the HIGH and LOW keys to go to the beginning of the subroutine, 003 030 (03 18 hex).

STEP 32. Press STEP several times.

Question: Are you still in the subroutine, or did you return to the main program?



Answer: We observed that after pressing the STEP key many times, we were still in the subroutine.

Question: How many times would you have to press STEP to completely execute the subroutine?

Answer: We calculated it would take 1164 presses to completely execute the subroutine! The next experiment will demonstrate a method for avoiding this problem.

At this point, you should have an understanding of how to enter data and programs into the MMD-2. You should know how to verify the data you entered, how to go to different addresses in the computer, how to run a program at full speed, and how to run a program one step at a time.

## EXPERIMENT 6

The purpose of this experiment is to help you become familiar with the REGS FUNCTION. The REGS function, in conjunction with the STEP key, form the basis of a good system for debugging programs. The REGS function allows you to view the values stored in the internal registers of the 8080 microprocessor chip. Registers are used to temporarily store values during the execution of a program. Registers can also be used to control the operation of a program. The values in these registers cannot be viewed after each program step, unless a program is available which will show you the values that would be in the registers. The MMD-2 includes special software called a "debugger", which will allow you to view the processor's internal registers and change the values stored there. This procedure is commonly used in testing a program to verify that it is working properly.

We will use the following short program to demonstrate the operation of the REGS function of the computer:

```
003 000 076      MVI A      ;MOVE THE NEXT BYTE TO THE
                   ;ACCUMULATOR
003 001 010      --
003 002 075 LOOP: DCR A      ;DECREMENT A BY ONE
003 003 303      JMP        ;DO IT AGAIN
003 004 002      LOOP      ;LOW HALF OF ADDRESS
003 005 003      --        ;HIGH HALF OF ADDRESS
```

STEP 1. Enter the following program into memory, starting at address 003 000 (03 00 hex).

<u>HI</u>	<u>LO</u>	<u>OCTAL</u>	<u>HI</u>	<u>LO</u>	<u>HEX</u>
003	000	076	03	00	3E
	001	010		01	08
	002	075		02	3D
	003	303		03	C3
	004	002		04	02
	005	003		05	03



The program will load a number 010 (03 hex) into the A register. Next, the program will decrement the A register by one and jump back to the decrement instruction. The program is again an infinite loop and will continue until YOU stop it. The REGS key will allow you to see the contents of the various registers in the microprocessor. The REG KEY selects the REG MODE.

STEP 2. Press RESET to bring the computer back to the beginning of the program.

STEP 3. Press REGS. The display should now read (REG A. 000). Each time you press the NEXT key, the next successive register and its contents will be displayed. The order of these registers is A, B, C, D, E, H, L, SH, SL, FL (repeat). The SH and SL registers are the high and low bytes of the stack pointer register and FL will display the flag register.

### Flag Byte Assignments

STEP 4. Press NEXT to verify the order of registers to be displayed. When you reach the A register again, stop.

STEP 5. Press STEP. The A register should show 010 (08 hex) as the data because the first step in the program loads that number into the A register.

STEP 6. Press STEP. When you depress STEP, the A register was decremented by one.

Question: What is the contents of the A register now?

Answer: 007 (07 hex). The A register has been decremented by one.

STEP 7. Continue to press the STEP key. On every other press of the key, the A register will be decremented by one. Notice that the A register will change from 000 to 377 (00 to FF hex) and will continue to count down. The program will continue to run in this loop until you stop it, because the system is not testing for any special condition, such as a zero value in the accumulator or a negative number (sign). NOTE: In this case the zero condition would occur when the number in the accumulator changed from 001 to 000 (01 to 00 hex) and the negative number (sign) would occur at the 000 to 377 (00 to FF hex) change. These conditions are called FLAGS. Flags can be tested using different instructions; on the basis of the flag tested, the system can take different courses of action. For more information about FLAGS, consult Technibook Vol. VI. Unit 23.

STEP 8. Press RESET, press REGs. Place the EXEC/USER switch in the EXEC position.

STEP 9. Press STEP until the A register attains the value 001 (01 hex).

STEP 10. Press NEXT until the FL register is being displayed. Look at Port 2. You will see the binary representation of the microprocessor's flags. Look at the seventh LED from the right. This LED indicates the condition of the ZERO FLAG. The LED is off because the value in the accumulator is NOT ZERO.

STEP 11. Press STEP twice.

Question: What happened to the ZERO FLAG LED?

Answer: The LED went on, indicating that the value in the accumulator is now zero.

STEP 12. Press STEP twice.

Question: What changes did you observe in the LEDs in Port 2?

Answer: LED 7 went off, indicating that the accumulator has a non-zero number in it and that the SIGN FLAG LED 8 went on indicating a negative number. (You will also see the AUX CARRY, LED 5, change from ON to OFF.) In computers, negative numbers are indicated by having the most significant bit set to a one. Codes between 200 and 377 (80 and FF hex) therefore are considered to be negative numbers.

Besides being able to view the values in the internal registers of the 8080A microprocessor, you also have the ability to change these registers. This function is useful if you are stepping through a loop which repeats many times and you wish to get to the end of the loop rapidly. For example, consider the following change in the previous program:

```
003 000 076          MVI A      ;MOVE THE NEXT BYTE TO THE
                          ;ACCUMULATOR
003 001 377          --
003 002 075  LOOP:  DCR A      ;DECREMENT A BY ONE
003 003 302          JNZ       ;KEEP DECREMENTING UNTIL A=0
003 004 002          LOOP      ;ADDRESS TO
003 005 003          --       ;JUMP TO
003 006 166          HLT       ;HALT, DUMMY INSTRUCTION IN
                          ;PLACE OF ADDITIONAL PROGRAM
```

You will note that the JMP instruction has been replaced by a JNZ instruction, which will execute while the zero flag is not set (off) indicating that the accumulator has a number in it which is non-zero. When the accumulator's value goes to zero, the program will not jump to LOOP, but will execute the HLT instruction instead. To test this program to verify that it functions correctly, you have two options. First, you could step through the loop until the A register counted down from 377 to 0 (FF to 00 hex), which would take a considerable amount of time. The other approach is to step through the loop once or twice to verify that the loop is functioning, then change the value in the A register to 001 (01 hex) and then step through the program to see if the exit from the loop operates correctly. This method takes less time and provides the same information as the first method.



STEP 13. Change the byte at location 003 003 from 303 to 302. Change the byte at location 003 008 to 166. Change the byte at location 003 001 from 001 to 377. Press RESET, GO.

Question: What happened when you pressed GO?

Answer: We observed that the HALT LED turned on immediately. To actually see what is occurring in this program you have to use the STEP KEY.

STEP 15. Reset the computer and enter the REGS MODE. Press STEP several times.

Question: What did you observe happening to the value of the A register?

Answer: We observed that the values started at 377 (FF hex) and counted down 376 (FE hex), 375 (FD hex), etc.

STEP 16. Enter the value 001 (01 hex) and press STORE.

Question: What change did you see in the display?

Answer: We saw the value in the A register change from 375 to 001 (FD to 01 hex).

STEP 17. Press STEP twice.

Question: What changes occurred?

Answer: We observed the value in the A register change from 001 to 000 (01 to 00 hex).

STEP 18. Press STEP twice.

Question: What change occurred?

Answer: We observed the HALT LED turn on indicating the program did not execute the JNZ instruction since the value in the accumulator is zero. The program skipped that instruction and executed the next instruction, which was the HLT 166 (76 hex) instruction.

You can use the DELAY subroutine for additional practice using STEP and REG modes. You may want to enter different values in the DE register pair and observe their effect on the loop.

At this point you have been introduced to the basic functions of the MMD-2 computer. You now have an understanding of how to enter and test your programs.



## SECTION 3

### THE ADVANCED FEATURES OF THE MMD-2

#### Introduction

In addition to the basic features discussed in the previous section, the MMD-2 microcomputer includes advanced features designed to make the computer more versatile. The CASSETTE INTERFACE permits storage of programs on magnetic tape for later re-use. Program development is facilitated by storing programs on tape from one session to the next, because the time needed to enter the program each time is eliminated. Saving a program on tape also avoids having to re-enter it after a power failure or program "wipe out".

An EPROM PROGRAMMER is available. This device is used to "pop" (store) a program into either 2708 or 2716 EPROMs. This feature is useful when you have a program which is going to be used many times, or if you want a program available whenever the computer is turned on. For instance, EXEC C is stored in EPROM and is used each time the computer is activated.

Another advanced feature is the TELETYPE INTERFACE. The teletype interface is used to connect a teletype (or similar terminal) to the MMD-2. Terminals are necessary if you are using a high-level language, such as BASIC.

An additional feature is the DYNAMIC ADDRESSING SYSTEM, which allows the user to relocate RAM and EPROM memory conveniently. This feature is automatically used during the start-up sequence. It can be used to simplify the development of programs which will run in EPROM. The specific techniques for memory relocation will be discussed in Section 4.

The AUX MODE provides the user with several options to enhance the operation of the computer. Included are: a BREAKPOINT which can be used in debugging programs, software control of the SINGLE STEP function to include a MULTI-STEPPER option, and selection of BAUD RATE for the serial interface.

The final advanced feature is the STD BUS INTERFACE. This interface is used to connect STD BUS accessories to the MMD-2.

#### Objectives

To introduce you to the MMD-2's advanced features and how they function.

To provide a general understanding of how these advanced features will aid in program development.

To understand the memory maps of the computer and how they can be changed by alternate hardware decoding.

To provide a general understanding of the EPROM programmer.

To understand the function of the tape recorder interface for saving and loading programs.

To introduce the serial communications port, including its electrical specification and operating modes.

To introduce the concept of breakpoints and how they are used in software debugging.

To introduce the multi-stepper function and how it is used for viewing program execution.

## General Description of the Advanced Features

### 1. Memory Decoding and Readdressing.

The MMD-2 microcomputer has the ability to dynamically alter the location of RAM and EPROM memory. Three different memory maps are available under software control. Map 1 is automatically selected when the computer is first turned on or when RESET is pressed. In this map, the EPROM memory is mapped into the address space 000 000 (00 00 hex) through 037 377 (1F FF hex) if the computer is using 2716 EPROMs, or from 000 000 (00 00 hex) through 017 377 (0F FF hex) if 2708 EPROMs are used. In Map 1, RAM memory is located from 330 000 (D8 00 hex) to 347 377 (E7 FF hex) in the 2716 version, and from 330 000 (D8 00 hex) to 347 377 (E7 FF hex) in the 2708 version.

When the computer is first turned on or reset, the microprocessor begins executing the instructions in EPROM starting at address 000 000. One of the first operations performed is to enable Map 2. Memory Map 2 swaps the EPROM and RAM memory. This map locates the RAM memory at 000 000 and the EPROM is relocated to high memory (see memory maps for specific addresses). This is the standard map which you will normally use. Map 3 allows you to swap EPROM 3 with the first 1K block of RAM in a 2708 system or with the first 2K of RAM in a 2716 system. This feature permits you to develop a program starting at address 000 000 with the computer in its normal operating mode (Map 2) and then pop the program into EPROM. After your program has been stored in EPROM, you can verify its operation by enabling Map 3. This will swap the RAM memory starting at 000 000 with the EPROM located in socket 3. You will then be able to check out the operation of the program stored in EPROM. This feature will be of particular interest to people developing process control programs which generally reside in EPROM starting at address 000 000. You can enable any of the memory maps by executing the following instructions: IN 5 enables Map 1, IN 6 enables MAP 2 and IN 7 enables Map 3. You also have the ability to deselect any or all of the on-board memory and use external memory devices. The instruction OUT 7 with bit 0 high deselects (turns off) the user RAM and OUT 7 with bit 1 high will deselect the EPROM. The SYSTEM RAM can be deselected by executing an OUT 5 with bit 3 set high. The ability to change the configuration of the memory in the computer significantly increases the flexibility of the system and enhances



MMD-2 MEMORY MAP (2716 Version)  
 (Decoder PROMs Labeled A06-B06)

Octal				Hex
375	-Hole-	-Hole-	-Hole-	Fd
374	Scratchpad (1/4K)	Scratchpad (1/4K)	Scratchpad (1/4K)	Fc
	-Hole-	-Hole-	-Hole-	F8
		PROM 3 (2K)	RAM 1 (1K)	F4
			RAM 0 (1K)	F0
		PROM 2 (2K)	PROM 2 (2K)	E8
350	RAM 3 (1K)	PROM 1 (2K)	PROM 1 (2K)	E0
344	RAM 2 (1K)			
340	RAM 1 (1K)			
334	RAM 0 (1K)	PROM 0 (2K)	PROM 0 (2K)	d8
330	-Hole-			
040	PROM 3 (2K)	-Hole-	-Hole-	
030	PROM 2 (2K)			
020	PROM 1 (2K)	RAM 3 (1K)	RAM 3 (1K)	10
014		RAM 2 (1K)	RAM 2 (1K)	0c
010		RAM 1 (1K)	PROM 3 (2K)	08
004	PROM 0 (2K)	RAM 0 (1K)		
000				00

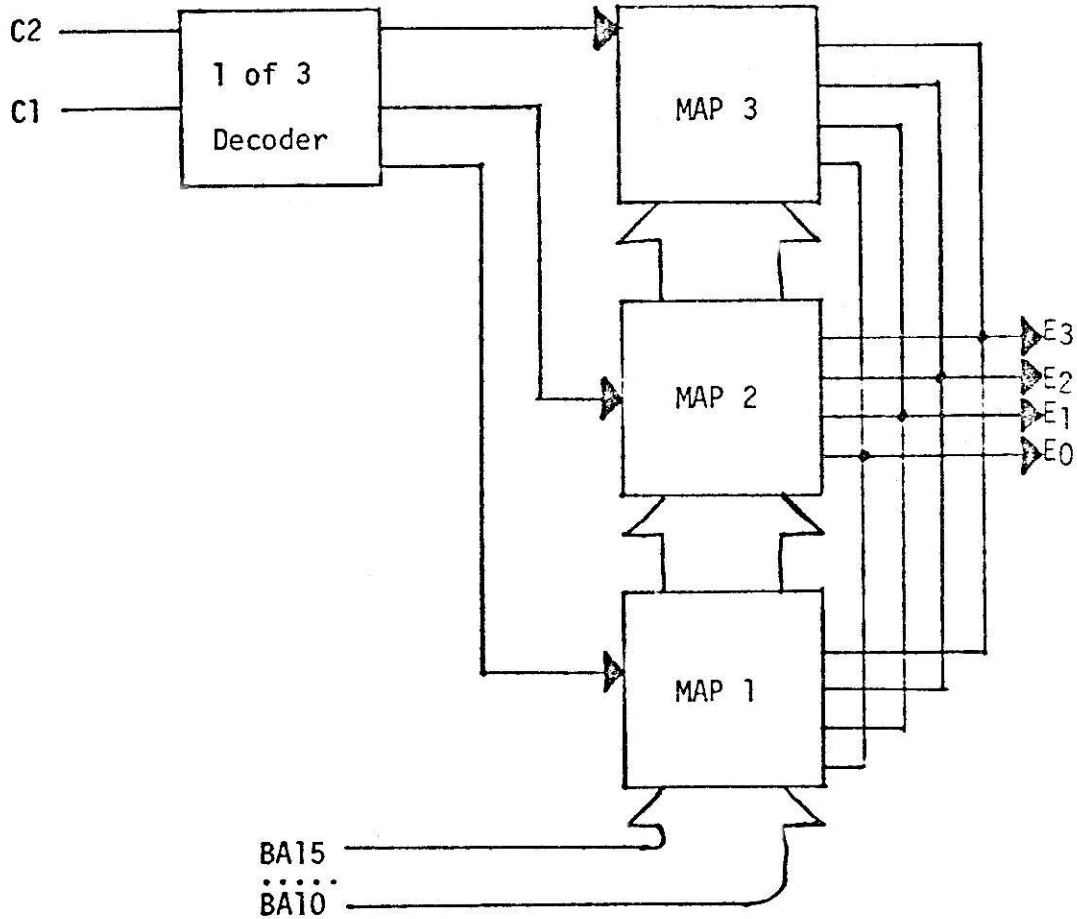


MMD-2 MEMORY MAP (2708 Version)

(Decoder PROMs Labeled A07-B07)

Octal	MAP 1 (IN 005 or RESET)	MAP 2 (IN 006)	MAP 3 (IN 007)	Hex
375	-Hole-	-Hole-	-Hole-	Fd
374	Scratchpad (1/4K)	Scratchpad (1/4K)	Scratchpad (1/4K)	Fc
350	-Hole-	-Hole-	-Hole-	E8
344	RAM 3 (1K)	PROM 3 (1K)	RAM 0 (1K)	E4
340	RAM 2 (1K)	PROM 2 (1K)	PROM 2 (1K)	E0
334	RAM 1 (1K)	PROM 1 (1K)	PROM 1 (1K)	dc
330	RAM 0 (1K)	PROM 0 (1K)	PROM 0 (1K)	d8
020	-Hole-	-Hole-	-Hole-	10
014	PROM 3 (1K)	RAM 3 (1K)	RAM 3 (1K)	0c
010	PROM 2 (1K)	RAM 2 (1K)	RAM 2 (1K)	08
004	PROM 1 (1K)	RAM 1 (1K)	RAM 1 (1K)	04
000	PROM 0 (1K)	RAM 0 (1K)	PROM 3 (1K)	00

EQUIVALENT FUNCTION OF DECODER PROM



its ability to operate with different memory requirements. For example, if your computer is equipped with E & L BASIC, available as an accessory from E & L Instruments, Inc., memory Maps 1 and 2 will be different from those supplied with EXEC C. (See memory maps on preceding pages.)

Address decoding is accomplished using 256x4 fused link type PROMs. Since the memory is made up of 1K blocks, just the HIGH order 6 bits of the address code from the microcomputer are used. These signals are connected to the LOW order address lines of the decoder PROMs. The two most significant address lines are connected to the map control signals C1 and C2. These signals can have the values 01, 10, or 11, depending on which map is enabled. The value 00 is not decoded. The states of C1 and C2 are determined by a three-line to two-line multiplexer circuit, which takes the map control commands IN 5, IN 6 or IN 7, and generates the C1 and C2 signals needed to enable one of the maps. Each map consists of a 65-byte region in each PROM. One PROM is used to select the RAM devices and the other selects the EPROM devices according to the data stored in the MAP PROMs. (See Function of Decoder Diagram on following page.)

The PROMs are factory-programmed to meet the map and location assignments shown in the memory maps. These PROMs cannot be reprogrammed. If your requirements differ from the standard maps, it will be necessary for you to procure additional PROMs and program them to fit your requirements. Specific information about the map PROMs can be found in the Reference Manual, Section 2, Theory of Operation--Hardware, Map Selector Circuitry.

## 2. Software and User EPROMs, 2708 and 2716 Operation.

The basic software supplied with the MMD-2 microcomputer is called EXEC C. This program provides you with an operating system and several subroutines which can be called from your program to perform basic input and output functions. The source for EXEC C is supplied in the Source Listings with some of the user callable subroutines marked. The basic MMD-2 computer is equipped with 2716 EPROMs installed in the first two sockets. The last two sockets are available for your use. NOTE: You cannot mix 2708 and 2716 EPROMs. You must have the same type of EPROM in each socket. If you switch from one variety to the other, you will have to change the jumpers located above the EPROM sockets (refer to the Reference Manual, Appendix III). If you're popping EPROMs, there's another header block which has to be positioned correctly, depending on the type of EPROM being programmed. This header block is located next to the socket in the EPROM PROGRAMMER area.

## 3. EPROM Programmer.

The EPROM programmer is a peripheral device directly accessed by the computer. EXEC C contains routines designed to facilitate the programming and testing functions. An EPROM can be given the CLEAR TEST. This test verifies that the EPROM has been erased properly and all bit positions contain ones. EXEC C has the POPPING sequence. This is basically a copy operation, where the specified block of memory is transferred, one byte at a time, to the EPROM. While the data is available to the EPROM PROGRAMMER, the computer turns on a programming pulse of 26 VDC for a specified length of time, which stores (programs) the byte in the EPROM. The programming sequences are different for the two types of EPROMs. The proper programming sequence is selected by storing



the EPROM type in the PROM auxiliary register. This procedure will be explained fully in the next chapter. Programming takes about three minutes for a 2708 EPROM, and 1.75 minutes for a 2716 EPROM. EXEC C can perform a DUPLICATION TEST to verify that the data has been stored correctly. You can program either type EPROM by placing the header block in the correct position and telling EXEC C which type of EPROM is in the programmer's socket. Specific instructions for programming will be given in the next section.

#### 4. Cassette Interface.

The cassette interface is another peripheral which has been provided to make the computer more useful. This device will write blocks of memory to a cassette tape recorder or read blocks from the recorder and store them in the computer's RAM memory. If you have written a program you wish to save on tape to use at a later time, you would use the DUMP command to store your program. The LOAD command will take your program and reload it back into memory at the same memory locations where it was originally.

#### 5. Teletype Interface.

The MMD-2 contains the necessary hardware and software to interface the computer to a teletype or CRT-type terminal. Both types of terminals communicate with the computer via a serial communications port. This port converts a byte of information from the microprocessor into a string of data bits, including a start bit, the data bits, an optional parity bit and stop bit (or bits). The conversion from a parallel data byte into a serial data string can be done using hardware or software. The MMD-2 uses the latter technique. The conversion and timing are performed by subroutines in EXEC C. These subroutines are available for use in your software. Terminals are designed to accept data at certain speeds called BAUD RATES. EXEC C is designed to provide the BAUD RATES 110, 150, 300, 600 and 1200. These BAUD RATES can be selected using the AUX function BAUD in EXEC C, or directly under your software's control. These techniques will be explained in the next section. After the parallel to serial conversion has taken place, another conversion of the signal has to occur before the data can be transmitted to the terminal. The TTL logic levels must be converted to either a 20 Ma current signal or an RS-232 voltage signal. The MMD-2 has both available at the terminal block located along the top of the computer. The choice of which interface to use is determined by the terminal to be connected to the computer. Teletypes are generally current loop devices, while CRT terminals can be either current loop or RS-232. The one disadvantage of RS-232 is that the terminal cannot be located more than a few feet from the computer, while current loop allows a much greater separation.

Data can be transmitted in either direction using this interface, but only in one direction at a time. In other words, you cannot be typing on the terminal and have the computer transmitting other information to the terminal at the same time.

## 6. Breakpoint.

The MMD-2 computer provides you with one breakpoint, which is used in software debugging. When a breakpoint is used, the programmer inserts it in the program at a location where he wants the program execution to be interrupted and return control to EXEC C. When the program reaches the breakpoint, the display will show (BREAK PT.1), indicating that EXEC C is again in control. You can then look at the stored registers, view the program, reset the breakpoint to a different location, or single step through the selected portion of the program. Using the breakpoint allows you to control the execution of a program for the purpose of testing selected portions of the program conveniently. The breakpoint cannot be used with programs stored in EPROM memory.

## 7. Step.

The single step key has a multi-step function accessed through the AUX MODE register STEP, which will control the rate the single step key will execute instructions when it is held down. Speeds from about one instruction every three seconds to several hundred instructions per second are available. Information on how to use the multi-step function is in Section 4.

## 8. STD Bus.

The STD BUS interface provided on the side of the MMD-2 allows the user to interface a wide variety of devices with the computer. The STD bus is provided on the MMD-2 computer to give you the capability for developing and testing STD circuit cards. The STD bus is commonly used in process control computers. The MMD-2 computer provides the hardware and software to conveniently test and develop these cards. The MMD-2 can also be used to verify the operation of the STD-based cards by plugging them into the computer and running diagnostic programs to check their operation. The STD signals are also available on the solderless plug strips on the MMD-2 for easy monitoring.

This combination of advanced features, in addition to the basic operating features of the MMD-2, provide you with a versatile but easy-to-use computer system for educational and developmental work.



## SECTION 4

### ADVANCED FEATURE EXPERIMENTS

#### Introduction

The advanced features allow you a great deal of flexibility in using the MMD-2 computer. These features in EXEC C are accessed using the AUX key. The five keys: COPY, LOAD, DUMP, PROM and OPTION become operational when you press AUX. In addition, the PREV, STORE, NEXT and CANCEL keys function in the same manner as in the memory or register modes. Depressing the AUX key places the MMD-2 in the auxiliary mode. By depressing the NEXT key, the user can walk through the parameter registers needed by the auxiliary functions in the same manner you walk through the 8080's internal registers in the REG mode. The AUX registers, however, are not hardware registers in the 8080 microprocessor, but storage locations used to save the parameters you enter. The following functions can be performed after you have entered the required data into the AUX FUNCTION registers. These functions will be described completely later on in the experiments.

LOAD. This function loads programs from cassette tape to memory. There are no variables to set when this function is used. The program is transferred from the cassette recorder and placed in its original memory locations. The LOAD command transfers in increments of one block (256 bytes) starting on page boundaries.

DUMP. The DUMP command permits the saving of programs on tape. You have to provide the high address where you wish to begin saving the memory and the length in blocks. This information is stored in the appropriate AUX REGISTERS.

PROM. This key is used to program EPROMs. It is capable of performing any or all of three different functions selected by you and stored in the AUX registers. The functions are: CLEAR TEST (used to test for a blank EPROM), POP PROM (used to program the EPROM), and DUPLICATION TEST (to test that the EPROM was successfully programmed). These functions are selected in the AUX mode, along with the high address byte where the source code starts and the type of EPROM to program or test.

COPY. The COPY function allows the duplication of data from one location in memory to a different location. To use this function, you must provide the SOURCE address, DESTINATION address and the number of BLOCKS to copy.

OPTION. This key displays the options associates with each AUX register. Each time the OPTION key is pressed, the next option for that particular register is displayed.

In addition to these functions, parameters for the following operations are stored in the auxiliary registers: BREAKPOINT ADDRESS, MULTI-STEP RATE, BAUD RATE and MEMORY MAP SELECTION. These functions will be demonstrated in the experiments later on in this text.



An additional advanced feature included in the MMD-2 computer is the on-board real time clock and priority interrupt controller circuitry. This device will accept interrupts from the real time clock, the keypad logic, serial input or from external interrupts.

### Objectives

To help you understand how the auxiliary features are used.

To provide you with experiments which demonstrate the operation of these features.

Demonstrate how these features may facilitate your use of the computer.

### Summary of the Experiments

Experiment 1 is designed to introduce you to the AUXILIARY REGISTERS and OPTION KEY, and how they are used to store data in the auxiliary registers.

Experiment 2 introduces you to the CASSETTE TAPE INTERFACE and how it is used to load and dump information to a tape.

Experiment 3 demonstrates the operation of the EPROM PROGRAMMER for programming 2708 EPROMs.

Experiment 4 demonstrates the operation of the EPROM PROGRAMMER for programming 2716 EPROMs.

Experiment 5 introduces you to the MEMORY MAP and how it can be altered.

Experiment 6 demonstrates how the SERIAL INTERFACE is interfaced to a terminal and how to change the BAUD RATE of the interface.

Experiment 7 demonstrates how to use the SERIAL PORT driver software for transmitting information to and from a terminal connected to the MMD-2.

Experiment 8 introduces you to the COPY function.

Experiment 9 provides information on changing EPROMs and ADDRESS DECODERS.

Experiment 10 demonstrates the operation of the MULTI-STEP option for controlling program execution.

Experiment 11 demonstrates the operation of the on-board priority interrupt controller. Program segments are included to demonstrate the keyboard interrupt signal and the on-board real time clock interrupt.

Experiment 12 demonstrates the use of the BREAKPOINT for debugging software and controlling program execution.

## EXPERIMENT 1

The purpose of this experiment is to help you become familiar with the Auxiliary Mode Registers. These registers store data needed by the AUX functions. In this experiment you will walk through the different registers and store variables in each register.

All auxiliary registers will be used in this experiment.

STEP 1. Turn on your MMD-2 microcomputer and press AUX. At this point, you are viewing the first AUX register. The display should read: (BR1 HI. OFF). This is the breakpoint high address register. It is used by the breakpoint routine. The high half of the breakpoint address is stored here.

STEP 2. Press OPTION.

Question: what does the display indicate?

Answer: We saw the OFF change to 000 (00 hex). At this point, you can enter and store the high memory address needed by the breakpoint function.

STEP 3. Enter 100 (40 hex) and press STORE. The value 100 (40 hex) has been stored in the BR1 HI register. NOTE: In order to actually save any data in an AUX register, you MUST press STORE.

STEP 4. Press OPTION.

Question: What does the display indicate?

Answer: We observed the 100 (40 hex) was replaced by OFF, the other option for this register. Pressing STORE will eliminate the 100 and store OFF in the register. Do not press STORE.

STEP 5. Press NEXT. The display should read (BR1 LO. 000).

STEP 6. This register is used to store the low half of the breakpoint address. NOTE: if the BR1 HI option is OFF, the BR1 LO option will also be off. You cannot store only one-half of an address; data must be stored in both registers to activate the breakpoint.

STEP 7. Press STORE. You have now stored 000 as the low half of the breakpoint address. Pressing OPTION would change the 000 to OFF, which could be restored to deactivate the breakpoint. NOTE: By storing the option OFF in either BR1 registers, OFF is stored in both registers.

STEP 8. Press NEXT. The display will indicate (STEP. OFF). This register stores a value which controls how fast the single stepper will execute instructions if the STEP key is held down.

- STEP 9. Press OPTION. The OFF will change to 000, the fastest speed. Enter the value 100 (40 hex). Press STORE.
- STEP 10. Press NEXT. The display should read (SRC HI. OFF). This is the source register which is used with the LOAD, PROM and COPY functions. The high half of the data source address is stored here. (There is no low half; it is assumed to be 000.)
- STEP 11. Press OPTION.

Question: What does the display indicate?

Answer: We saw the OFF change to 000 (00 hex). At this point, you can enter and store the source memory address needed by the COPY, PROM and DUMP functions.

- STEP 12. Enter 100 (40 hex) and press STORE. The value 100 (40 hex) has been stored in the SRC HI register.
- STEP 13. Press NEXT. The display should read (DES HI. OFF). This is the destination register. It is used only by the COPY function to store the high address of the data's destination.
- STEP 14. Press OPTION. You will see the OFF change to 000 (00 hex). You can now enter a destination address and store it.
- STEP 15. Enter and store the value 300 (C0 hex) using the STORE key. You have now saved the destination address for the copy function. Pressing OPTION will again restore OFF to the display. The value 300 (C0 hex), however, will remain in the register unless STORE is pressed.
- STEP 16. Press NEXT. The display should read (LEN HI. OFF). This register stores the length of the data in blocks. It is used with the COPY and DUMP functions.
- STEP 17. Enter a length of 010 (08 hex) in the same manner as Steps 4 through 7.
- STEP 18. Press NEXT. The display should read (CLR TST. OFF.). This is the first register associated with the EPROM function. This test checks the EPROM for all ones (blank).
- STEP 19. Press OPTION. The OFF should change to ON indicating the test has been selected. Pressing OPTION will toggle between OFF and ON.
- STEP 20. Select the test by pressing STORE while ON is displayed.



- STEP 21. Press NEXT. The display should read (POP PRM. OFF). Pressing OPTION will toggle between OFF and ON. This function will program the EPROM if it is ON.
- STEP 22. Select ON by pressing STORE.
- STEP 23. Press NEXT. The display should read (DUP TST. OFF). This test is performed after an EPROM has been programmed to verify that it was programmed accurately. Pressing OPTION will cause the display to toggle from OFF to ON.
- STEP 24. Store the ON condition as you did in previous steps.
- STEP 25. Press NEXT. The display should read (PROM. 2708). Pressing the OPTION key will toggle the display between 2708 and 2716. This register selects which variety of EPROM you are going to program.
- STEP 26. Select 2716 by pressing STORE as before.
- STEP 27. Press NEXT. The display should indicate (MEM MAP. RAM). This option allows you to select either Memory Map 2 or Memory Map 3. These maps swap the first 2K of RAM memory with the last 2716 EPROM socket, or the first 1K of RAM with the last 2708 EPROM socket, depending on which type of EPROM you are using. This feature is used when you have to run a program stored in EPROM which has a starting address of 000 000 (00 00 hex). Pressing the OPTION key will change the RAM to ROM indicating Map 3 will be enabled when STORE is pressed.
- STEP 28. Store ROM as before.
- STEP 29. Press NEXT. The display should indicate (BAUD. 300). This is the default baud rate for the serial output port.
- STEP 30. Press OPTION five (5) times.
- Question: What did you observe?
- Answer: We observed the value change from 300 to 600, 1200, 110, 150 and back to 300 BAUD. Each number represents a speed which data will be transmitted over the serial port.
- STEP 31. Press OPTION.
- STEP 32. Press STORE. You now have stored the BAUD RATE 600.
- STEP 33. Press NEXT. The display should return to the original register (BR1 HI.) and the value displayed should be 100 (40 hex).
- STEP 34. Press NEXT to step through the registers. As you step through the registers, you should see the values you stored using the OPTION and STORE keys: BR1 HI 100 (40 hex), BR1 LO. 000, STEP. 100 (40 hex), SRC HI. 100 (40 hex), DES HI. 300 (C0 hex), LEN HI. 010 (08 hex), CLR TST. ON, POP PRM. ON, DUP TST. ON, PROM. 2716, MEM MAP. ROM, BAUD. 600.

STEP 35. Press PREV. You will note that you now are examining the registers in the reverse order. This function operates in the same manner as PREV did in the REG mode.

At this point, you should have an understanding of how to change and store information in the auxiliary registers, and how the OPTION and STORE keys function.

## EXPERIMENT 2

The purpose of this experiment is to help you become familiar with the CASSETTE interface for LOADING and DUMPING information to and from a tape recorder. You will use the AUX function to store the necessary parameters for the DUMP command. You will also learn how to reload your program from tape.

The DUMP command uses the SOURCE HI and LENGTH auxiliary registers. The LOAD command uses no auxiliary registers.

STEP 1. In this step, you will enter the program which blinks the LEDs on Port 2. (See Experiment 5, Section 2). First, power down the MMD-2 to reset all the AUX mode registers to the OFF state, then power back on. Enter the following code, starting at address 003 000 (03 00 H):

<u>HI ADDRESS</u>	<u>LO ADDRESS</u>	<u>INSTRUCTION BYTE</u>	<u>HI ADDRESS</u>	<u>LO ADDRESS</u>	<u>INSTRUCTION BYTE</u>
003	000	257	03	00	AF
	001	323		01	D3
	002	002		02	02
	003	074		03	3C
	004	315		04	CD
	005	030		05	18
	006	003		06	03
	007	303		07	C3
	010	001		08	01
	011	003		09	03

Enter the following code at 003 030 (03 18 hex):

HI ADDRESS	LO ADDRESS	INSTRUCTION BYTE	HI ADDRESS	LO ADDRESS	INSTRUCTION BYTE
003	030	365	03	18	F5
	031	325		19	D5
	032	021		1A	11
	033	046		1B	26
	034	001		1C	01
	035	033		1D	1B
	036	172		1E	7A
	037	263		1F	B3
	040	302		20	C2
	041	035		21	1D
	042	003		22	03
	043	321		23	D1
	044	361		24	F1
	045	311		25	C9

STEP 2. Press RESET, Go. The LEDs in output Port 2 should be counting. If not, check your program for errors and correct them.

STEP 3. Press RESET, AUX, DUMP.

Question: What happened?

Answer: We observed the display change to (STATE. OFF). This message will be displayed if either the SOURCE, LENGTH or both of these auxiliary registers have been left in the OFF state. Before you can DUMP to tape, both the SOURCE and LENGTH registers must have data stored in them. If you get the display STATE. OFF, press CANCEL to display the previously displayed data. At this point, enter the missing data in the appropriate register.

STEP 4. You are now ready to store the necessary values in the AUXILIARY registers needed to DUMP your program to cassette tape.

STEP 5. Press NEXT three (3) times. The display should read SRC HI OFF.

STEP 6. Press OPTION, the OFF will change to 000.



- STEP 7. Enter the high half of the program's starting address. Programs are dumped starting on page boundaries; i.e., in this example, the memory starting from 003 000 (03 00 hex) will be recorded on tape. The value you should enter is 003. Remember to press STORE after entering the number, or it will not be retained in the register.
- STEP 8. Press NEXT two (2) times to advance to the LENGTH register. Press OPTION to change the OFF to 000. At this point, you will enter the number of 256-byte blocks to be DUMPED to the tape. Since the program you entered is less than one block in length, you should enter and store 001 to save one block of memory on the cassette recorder.
- STEP 9. Make sure that the cassette recorder has been properly connected to the computer with the RED jack connected to the MIC (input) and the BLACK jack to the EAR (output) on the recorder. The other end of the cable is connected to the six-pin connector on the upper right hand corner of the MMD-2. Start the recorder in the RECORD mode and wait a few seconds to go past the leader.
- STEP 10. Press DUMP. The display should display (DUMPING..). The MMD-2 is now recording the block of memory on the cassette tape. When the display reads (DUMP DONE), stop the recorder. Your program has been stored on tape. Generally, it is a good practice to dump your program more than once. This procedure is called backing up and is done to provide you with additional copies of your program. Sometimes, tape can be damaged and a program lost. Having backup copies will help prevent the loss of your programs. To record another copy of your program, it is necessary only to leave the recorder running and press DUMP a second time.
- S 11. Turn the MMD-2 off for five (5) seconds.

Question: What has happened to your program in the computer?

Answer: Turning off the computer destroys any information stored in RAM memory.

- STEP 12. Press AUX, then LOAD keys. Displays should read (LOADING...). At this point, rewind the tape and press the PLAY key on the recorder to start the tape. When the display reads (003 TO 004) stop the recorder and rewind the tape. During the time the program is loading, you will hear some sound coming from the recorder. The recording technique uses audio tones to store data on the tape. If the sound stops and the display doesn't indicate 003 TO 004, stop the recorder, press RESET, press AUX, rewind the tape, and try again. This happens when the data is not read correctly from the tape. The numbers which appear after a successful load indicate the beginning and end HIGH addresses of the recorded data. In this case, the display indicates we loaded one block, from 003 HIGH to 004 HIGH.
- STEP 13. Press the CANCEL key and then the MEM key. You have now exited the AUXILIARY mode and are in the MEMORY mode.

STEP 14. Press GO.

Question: What did you observe?

Answer: We observed the LEDs on Port 2 counting as before. We have successfully stored and reloaded a program using cassette tape.

The tape interface can be used to save any amount of memory by just providing the high half of the start address and the number of blocks to be saved on tape.

### EXPERIMENT 3

The purpose of this experiment is to help you become familiar with the EPROM programmer. To perform this experiment, you will need a 2708 EPROM (not supplied). You may also need facilities to erase the EPROM. There are several commercial ultra-violet EPROM erasers available.

WARNING: DO NOT LOOK AT THE ULTRA-VIOLET LIGHT OR ATTEMPT TO BYPASS THE SAFETY INTERLOCKS ON THE EPROM ERASER! TO DO SO CAN RESULT IN SERIOUS EYE INJURIES.

The PROM PROGRAMMER function uses the following AUXILIARY REGISTERS: SRC HI., CLR TST., POP PRM., DUP TST., PROM.

STEP 1. Open the EPROM PROGRAMMER SOCKET by turning the bottom screw 1/4 turn to the left. With the MMD-2 off, insert a 2708 EPROM in the EPROM PROGRAMMER socket. MAKE SURE THAT IT IS INSERTED CORRECTLY WITH PIN ONE IN THE UPPER LEFT HAND CORNER. If you insert the EPROM incorrectly, it will be destroyed. Make sure that the header block is inserted in the proper direction with 2708 on top. After you have inserted the EPROM, be sure to turn the small screw on the bottom of the socket 1/4 turn to the right.

STEP 2. Turn on the MMD-2 and press AUX. NOTE: When the computer is turned on, the PROM register will contain 2708.

STEP 3. Press NEXT six (6) times to get to the CLEAR TEST register. Press OPTION and STORE to turn on the CLEAR TEST. The display should read: (CLR TST. ON).

STEP 4. Press the PROM key. The display should read (ONES GOOD). If the display reads (ONES BAD), the EPROM in the socket is not blank. If this happens, you will have to erase the EPROM or use one which is blank.

STEP 5. Press the CANCEL key. Press OPTION and STORE to turn off the CLEAR TEST.

STEP 6. Press NEXT. The display should read (POP PRM. OFF). Press OPTION and STORE to turn on the EPROM POP function. The display should change from POP PRM. OFF to POP PRM. ON.

STEP 7. Press the PREV four (4) times to go to the SOURCE REGISTER. Press OPTION and enter the high address of the starting location of the data to be popped into the EPROM. Enter the value 330 (D8 hex) and press STORE. We will use the starting address of the first EXEC C EPROM for our data.



STEP 8. Turn on the programming voltage switch in the EPROM PROGRAMMER area by sliding it to the right.

STEP 9. Press the PROM key.

Question: What do you observe on the display and the LED output ports?

Answer: We observed (POPPING...) in the display and activity in the output port LEDs. It will take approximately 3.5 minutes to program the EPROM.

STEP 10. When the display reads (POP DONE), turn the programming voltage switch off. At this point, you have programmed the EPROM. The final part of the procedure is to test the EPROM to verify that it was programmed correctly. NOTE: Turn POP PRM register off first!

STEP 11. Press the CANCEL key. Press NEXT four times. This will bring you back to the POP PROM register. Press OPTION, STORE to turn this register off.

STEP 12. Press the CANCEL key. Press NEXT. This will bring you to the DUP TST register. Press OPTION, STORE to turn on the duplication test. The display should read (DUP TST. ON).

STEP 13. Press the PROM key. The display should read (DATA GOOD). This indicates that the EPROM was programmed successfully. If the display indicated (DATA BAD) it means that the EPROM has not been correctly programmed. You will have to erase the EPROM and try the experiment again.

It is possible to include all three programming options in one operation by turning all three options ON, storing the HIGH memory address of the data in the SRC HI register and then pressing PROM. The computer will perform each command in order. Testing the EPROM, programming and verifying it. If the EPROM fails the ONES TEST, the sequence will stop without performing the other functions.

As with the DUMP function, if the SOURCE register is in the OFF state, neither the POP PRM. function or the DUP TST function will operate. The display will show STATE OFF, and the function will be aborted.

The PROM programmer makes no test for the proper EPROM in the programming socket during any programming function. It is up to you to verify that the proper EPROM is in the socket, that the header is inserted properly and that the PROM auxiliary register has the proper EPROM type stored.

#### EXPERIMENT 4--Programming 2716 EPROMs

The purpose of this experiment is to help you become familiar with the technique for programming 2716 EPROMs. To perform this experiment, you will need a 2716 EPROM (not supplied) and an EPROM eraser.

The AUXILIARY REGISTERS needed for this experiment are: SRC HI., CLR TST., POP PRM., DUP TST., PROM.



- STEP 1. With the MMD-2 turned off, insert the EPROM in the socket in the programming area and position the header block with 2716 on top to configure the programmer for the 2716 EPROM. Make sure that the EPROM is inserted properly with pin one in the upper left hand corner. Failure to insert the EPROM properly will result in its destruction.
- STEP 2. Turn the MMD-2 on and press AUX. In this experiment, we will set all of the EPROM parameters at once and allow the entire programming sequence to progress automatically.
- STEP 3. Enter the high byte of the starting location in the SRC HI. register. We will again use 330 (D8 hex) as our source.
- STEP 4. Press NEXT until the CLEAR TEST register is displayed and store "ON" in the register.
- STEP 5. Press NEXT and store "ON" in the POP PROM register.
- STEP 6. In the same manner, store "ON" in the DUP TEST register.
- STEP 7. Press NEXT and store 2716 in the PROM register.
- STEP 8. Make sure that the PROGRAMMING VOLTAGE is on by sliding it to the right. If you forget to turn this switch on, the EPROM will not be programmed.
- STEP 9. Press PROM. At this time, the MMD-2 will perform the three functions you selected. The sequence is: VERIFY ONES to make sure the EPROM is erased, POP PROM by programming the data into the EPROM (it will take approximately 1.75 minutes for programming), and DUPLICATE TEST to verify that the programming sequence is terminated.
- STEP 10. After the EPROM has been programmed, turn off the PROGRAMMING VOLTAGE TO PREVENT ACCIDENTAL PROGRAMMING.

As in the previous experiment, EXEC C makes no test for the proper EPROM in the programming socket, or if the header has been inserted correctly, or if the proper EPROM type has been stored in the PROM register. EXEC C does check that the SRC HI register has an address stored in it. If not, the display will show STATE OFF, and the PROM programming sequence will be aborted after the ONES TEST.

At this point, you should understand how to program both 2716 and 2708 EPROMs. It is important to understand that the MMD-2 is designed to program the entire EPROM. You cannot program only a section of an EPROM.

### EXPERIMENT 5--Memory Map

The purpose of this experiment is to demonstrate how to swap the first 2K of RAM memory with the last 2K of EPROM memory (if 2716 EPROMs are being used), or the first 1K of RAM with the last 1K of EPROM in systems using 2708 EPROMs. This feature is useful when you have a program located in EPROM which is designed to execute in the first 1 or 2K of memory.

This experiment uses the MEM MAP. register.

STEP 1. Store several bytes of zeros, starting at location 000 000. You will use this as a reference to verify that the memory has been swapped with the EPROM.

STEP 2. Press AUX. Press NEXT until you are at the register MEM MAP. RAM.

STEP 3. Press OPTION, STORE. The display should now indicate (MEM MAP. ROM). You have now swapped the first 1 or 2K of RAM and the last EPROM socket by enabling Map 3.

STEP 4. Press MEM to exit from the AUX mode.

STEP 5. Examine the memory starting at location 000 000.

Question: What values did you see stored at the locations you had previously stored zeros?

Answer: We observed the value 377 (FF hex) in these locations. In our MMD-2, the last EPROM socket is empty and nonexistent memory is displayed as 377 (FF hex). If you have an EPROM in the last socket, the code you will see at location 000 000 will be the code stored in that EPROM.

STEP 6. Examine the memory at location 360 000 (F0 00 hex) if you are using 2716 EPROMs. If you are using 2708 EPROMs, examine 344 000 (E4 00 hex).

Question: What values did you observe in the first few bytes of memory?

Answer: We observed the zeros we had previously stored at location 000 000.

The memory map will stay in the last map you put it in, even if you press RESET. You can only change the memory map by changing the MEM MAP register (or turning the MMD-2 off and then on; or doing a PUP RESET which is described later).

## EXPERIMENT 6--Setting Teletype Parameters

The purpose of this experiment is to help you become familiar with the serial interface. You will learn how to select the proper baud rate for communication with your terminal. Baud rate refers to the speed with which characters are transmitted to and received from an external terminal connected to the MMD-2. The higher the baud rate, the more rapidly characters will be transmitted. EXEC C can transmit data at standard rates including: 110, 150, 300, 600 and 1200 BAUD. To determine the proper baud rate for your terminal, you will have to consult the owner's manual. Most CRT terminals have selectable baud rates, while TTY terminals are fixed at 110 baud. The default rate for the MMD-2 is 300 baud. The other rates are available by changing the value of the BAUD auxiliary register.

This program will be used to demonstrate the effects of BAUD RATE changes on terminal operation.



```

003 000 041 020 003  START:   LXI H,MSG      ;POINTER TO MESSAGE
003 003 176           LOOP:   MOV A,M        ;GET CHARACTER
003 004 376 000           CPI 0          ;END OF MESSAGE
003 006 312 000 003       JZ  START      ;DO IT AGAIN
003 011 315 141 343       CALL TTYOUT    ;OUTPUT ROUTINE
003 014 043           INX H          ;BUMP POINTER
003 015 303 003 003       JMP  LOOP      ;GO GET NEXT CHARACTER
003 020 115 115 104  MSG:   DB 'MMD-2', 012Q,015Q,0 ;MESSAGE 0 = OCTAL
003 023 055 062 012
003 026 015 000
                                END

```

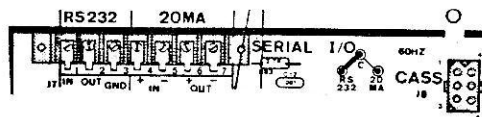
The program will continually print the message 'MMD-2' on the terminal connected to the serial port.

To perform this experiment, you will need a serial terminal with selectable baud rate capability. You will also need a cable which will connect your terminal to the MMD-2 computer (not supplied). The cable will need a connector on one end to match the connector on the terminal. This connector can be wired for either RS-232 or CURRENT LOOP operation. Your choice of interface will be determined by the terminal you are going to connect to the MMD-2. The MMD-2 can interface using either CURRENT LOOP or RS-232 signals. The cable should be connected to the MMD-2 terminal strip along the upper right hand side of the unit. If your terminal is using the CURRENT LOOP interface, you will have to connect four wires to the computer. The IN wires will come from the keyboard of your terminal, and the OUT wires will go to the terminal's printer connections. Proper polarity should be observed for correct operation. If you are using the RS-232 interface, you will only have to connect three wires: GROUND, IN, OUT. The IN wire comes from your terminal's keyboard and the OUT wire goes to the printer connection. You will also have to install the proper jumper connection located to the right of the terminal strip. Connect a short piece of solid #22 gauge wire between the pin marked "C" and either RS-232 or CURRENT LOOP, depending on which interface you have selected.

When these connections have been properly made, the computer will be ready to communicate with your terminal.

This experiment uses the BAUD register.

STEP 1. Enter the following program in memory, starting at location 003 000. (Program starts on following page.)



MMD-2 SERIAL COMMUNICATIONS TERMINAL STRIP AND JUMPER AREA



HI ADDRESS	LO ADDRESS	INSTRUCTION BYTE	HI ADDRESS	LO ADDRESS	INSTRUCTION BYTE
003	000	041	03	00	21
	001	020		01	10
	002	003		02	03
	003	176		03	7E
	004	376		04	FE
	005	000		05	00
	006	312		06	CA
	007	000		07	00
	010	003		08	03
	011	315		09	CD
	012	141		0A	61
	013	343		0B	E3
	014	043		0C	23
	015	303		0D	C3
	016	003		0E	03
	017	003		0F	03
	020	115		10	4D
	021	115		11	4D
	022	104		12	44
	023	055		13	2D
	024	062		14	32
	025	012		15	0A
	026	015		16	0D
	027	000		17	00

STEP 2. If your terminal has selectable BAUD rates, follow the instructions and set it to 300 BAUD. If this BAUD RATE is not available, set it to a BAUD RATE which is available on the MMD-2.

STEP 3. Press AUX. Press NEXT to advance to the BAUD register.

STEP 4. Press OPTION until the BAUD rate you have selected for the terminal is displayed.

STEP 5. Press STORE to save the new rate.

STEP 6. Connect your terminal to the MMD-2. Press RESET, GO.

Question: What did you observe?

Answer: We observed the terminal printing 'MMD-2' over and over.

STEP 7. Press RESET to stop the computer. The program is another example of an infinite loop.

You may wish to experiment with this program by changing the message printed or by adding a delay loop to slow down the printing speed. You may also wish to only print the message once or print it a specific number of times.

STEP 8. At this point, select a different BAUD RATE with which to output data to the terminal. Make sure that you choose a rate which your terminal can accept. For our terminal, an ADM 3A, we selected 1200 BAUD. Set your terminal for the baud rate you selected.

STEP 9. Press AUX. You should be at the BAUD register. Using the OPTION key, select the BAUD rate the terminal has been set to and store the new value.

STEP 10. Press RESET, GO.

Question: What did you observe on the terminal?

Answer: We observed the same message being printed at a higher speed. The speed you observe will be determined by the baud rate you selected. The higher the baud rate, the faster the characters will be displayed.

STEP 11. Press RESET.

STEP 12. Change the BAUD rate to a different value by pressing AUX, then OPTION, STORE.

STEP 13. Run the same program. This time, the baud rate of the terminal does not match the baud rate stored in the MMD-2.

Question: What did you observe on the terminal?

Answer: We observed the terminal was printing random characters, "garbage". For proper operation, the baud rate of the computer MUST match that of the terminal.

At this point, you should understand the techniques for changing the baud rate of the MMD-2 computer. You should also be familiar with the methods for connecting a terminal to the computer.

#### EXPERIMENT 7--Performing TTY I/O

The purpose of this experiment is to help you understand the techniques for performing console input/output using a terminal and the TTY I/O routines in EXEC C. You will probably use these routines if you write programs requiring data entry from the terminal's keyboard, or if you have to print results on the terminal's display.

You will now enter a program which will accept an input from the keyboard, then print the character on the terminal. If a RETURN is pressed, the program will halt. Generally, most input routines will define a character which is used to terminate the input loop. The return character is commonly used; however, any character not needed for data entry can be used.

```

003 000 315 042 343      START:   CALL TTY IN      ;GET A CHARACTER
003 003 376 015          CPI/015      ;TEST FOR CARRIAGE RETURN
003 005 312 016 003      JZ DONE       ;EXIT LOOP
003 010 315 141 343      CALL TTYOUT     ;PRINT CHARACTER
003 013 303 000 003      JMP START     ;DO IT AGAIN
003 016 166              DONE:   HLT        ;END OF PROGRAM

```

END

This experiment uses the BAUD register only, if the terminal you connect to the MMD-2 uses a BAUD rate different from the default value of 300 BAUD. If you have to change the BAUD rate, follow the steps outlined in Experiment 6.

STEP 1. Enter the following program, starting at location 003 000.

HI ADDRESS	LO ADDRESS	INSTRUCTION BYTE	HI ADDRESS	LO ADDRESS	INSTRUCTION BYTE
003	000	315	03	00	CD
	001	042		01	22
	002	343		02	E3
	003	376		03	FE
	004	015		04	0D
	005	312		05	CA
	006	016		07	0E
	007	003		08	03
	010	315		09	CD
	011	141		0A	61
	012	343		0B	E3
	013	303		0C	C3
	014	000		0D	00
	015	003		0E	03
	016	166		0F	76



STEP 2. Make sure that the terminal has been properly connected to the MMD-2, and that the BAUD rate of the computer matches the terminal's BAUD rate.

STEP 3. Run the program. Type some characters on the keyboard.

Question: What happened?

Answer: We observed the characters being printed on the terminal screen. As long as the program is running, you will be able to type and see the characters on the screen. This is another example of a program that loops.

STEP 4. Press the RETURN key.

Question: What did you observe on the MMD-2?

Answer: We observed the HALT LED lit on the computer. The HALT instruction generally will not be used in a program; however, we are using it to simulate the next section of the program which might, for instance, process the data returned to the computer from the terminal.

At this point, you should know how to get data from an external terminal and how to print data on the terminal. Remember, the code for the character to be printed must be a valid ASCII code (see Appendix for ASCII table), and it must be in the A register when TTYOUT is called. The character typed on the keyboard will be an ASCII character and will be in the A register when the program returns from the TTYIN routine.

### EXPERIMENT 8--Setting Copy Parameters

The purpose of this experiment is to help you become familiar with the technique of setting memory copy parameters. This technique is used when you are going to copy a block of memory from one location to another location. For example, if you want to make a patch (change) in the code stored in an EPROM, you can copy the contents of the EPROM down to location 000 000, make the patch and then use the PROM routines to pop a new EPROM using the copied code.

This experiment uses the SRC HI., DES HI., LEN HI. registers.

STEP 1. Press AUX to place the MMD-2 in the AUXILIARY MODE. Press NEXT to advance to the SRC HI register. Store the value 330 (D8 hex) by pressing OPTION, the value, STORE.

STEP 2. Press NEXT. You should see (DES HI. OFF). Store the value 000 in this register using the same technique.

STEP 3. Press NEXT. You should see (LEN HI. OFF). Store the value 020 (10 hex) in this register.

STEP 4. Press COPY.

Question: What did you observe?

Answer: We observed the display change to (BLOCKS 020) (BLOCKS 10)hex. This means that 20 (octal) 256-byte blocks were copied beginning with location 330 000 (D8 00 hex), to location 000 000.

STEP 5. Press RESET and then display memory location 000 000. Copy the values of the first five bytes, starting with location 000 000.

\_\_\_\_\_.  
\_\_\_\_\_.

STEP 6. Display the location 330 000 (D8 00 hex). Copy the first five bytes starting at that location.

\_\_\_\_\_.  
\_\_\_\_\_.

Question: Are the values in agreement?

Answer: We observed the same values in both locations indicating that the memory was copied.

STEP 7. Press AUX. Enter the following values in the appropriate registers: SRC HI. 330 (D8 hex), DES HI. 000 (00 hex), LEN HI. 050 (28 hex).

STEP 8. Press COPY.

Question: What happened?

Answer: We observed the display indicated BLOCKS with the number counting. After about one second, the display changed to (NOT STORED), indicating that the copy operation was unsuccessful. We were attempting to copy a larger block of memory than there was space to receive, and EXEC C will not allow that. The message NOT STORED indicates that an attempt to store data into nonexistent memory or EPROM was attempted. If any or all of the registers used by the COPY function are in the OFF state, when COPY is pressed the display will change to STATE OFF. At this point, you will have to press CANCEL to restore the display and enter the proper information in the register which is off before you can use the COPY function.

At this point, you should know how to copy blocks of memory from one location to another.

#### EXPERIMENT 9--Mounting New Decoder and Program PROMs

The purpose of this experiment is to make you aware of the procedure for changing the DECODER PROMs and PROGRAM PROMs on the MMD-2 computer.

From time to time, you may have to change the on-board EPROMs to run a different

program, for example, BASIC (available as an accessory from E & L Instruments, Inc.). To do this, you have to remove the EPROMs as well as the DECODER PROMs. When removing these devices, it is important to have the MMD-2 turned off. These devices are mounted in zero insertion force sockets. To remove these devices, turn the screw located on the socket to release the chip. At this point, the chip can be lifted out easily. When inserting the new chips, make sure that they are installed in the correct socket and that all pins are inserted properly. After the new chips have been installed, turn on the MMD-2 and verify the operation of the new program. Remember, you must change both the EPROMs and the DECODER PROMs.

## EXPERIMENT 10

The purpose of this experiment is to demonstrate the multi-step option. This feature permits continuous execution of instructions as long as the STEP key is pressed. The speed with which instructions are executed is controlled by the value stored in the STEP auxiliary register. The slowest speed executes one instruction every two seconds and the highest speed executes several hundred instructions per second. Both single step and multi-step execute instructions identically. The multi-step provides you with a convenient method for stepping through a long subroutine or loop without having to press the STEP key many times.

We will use the time delay subroutine to demonstrate the operation of the multi-step function.

HI ADDRESS	LO ADDRESS	INSTRUCTION BYTE		MNEMONIC	DESCRIPTION
003	000	315		CALL	;CALL TIME DELAY
	001	277		TIMOUT	
	002	000			
	003	166		HLT	;STOP PROGRAM
000	277	365	TIMOUT	PUSH PSW	;SAVE REGISTERS
	300	325		PUSH D	
	301	021		LXI D	;LOAD WITH VALUE TO DECREMENT
	302	046		046	
	303	001		001	
	304	033	MORE	DCX	;JUMP IN THIS LOOP UNTIL D&E ARE ;ZERO
	305	172		MOV A,D	
	306	263		ORA E	
	307	302		JNZ	
	310	304		MORE	
	311	000		0	
	312	321		POP D	;RESTORE REGISTERS
	313	361		POP PSW	
	314	311		RET	



This experiment uses the STEP register.

STEP 1. Enter the following code:

HI ADDRESS	LO ADDRESS	INSTRUCTION BYTE	HI ADDRESS	LO ADDRESS	INSTRUCTION BYTE
003	000	315	03	00	CD
	001	277		01	bf
	002	000		02	00
	003	166		03	76
000	277	365	00	BF	F5
	300	325		C0	D5
	301	021		C1	11
	302	046		C2	26
	303	001		C3	01
	304	033		C4	1B
	305	172		C5	7A
	306	263		C6	B3
	307	302		C7	C2
	310	304		C8	C4
	311	000		C9	00
	312	321		CA	D1
	313	361		CB	F1
	314	311		CC	C9

STEP 2. Press AUX and NEXT to advance to the (STEP. OFF) register.

STEP 3. Enter 001 and store in the usual manner

STEP 4. Press MEM. At this point, you are ready to execute the program. The program will call the time delay subroutine and then loop in the routine. When the program returns from the subroutine, the computer will halt. Enter 003 High, 000 Low.

STEP 5. Press STEP and hold the key down.

Question: What happened?

Answer: We observed that the display changed once every two seconds, indicating the execution of a program step. The STEP function will continue to execute instructions at this rate as long as the key remains pressed, or until the STEP function attempts to execute an undefined op code, which will result in the display showing the address of the erroneous instruction

and (ERR), or it will stop single stepping if it single steps a HLT instruction (166).

STEP 6. Press AUX. You should be at the STEP register.

STEP 7. Enter the value 300 (C0 hex) and store it.

STEP 8. Press RESET to return to the program.

STEP 9. Press and hold the STEP key.

Question: What does the display indicate?

Answer: We observed that the multi-stepper is executing instructions at a rate of about 2 instructions per second.

STEP 10. Press AUX, enter the value 350 (EB hex) in the STEP register.

STEP 11. Return to your program and press and hold the STEP key to execute several instructions.

Question: What change did you observe in the rate the instructions were executed?

Answer: We observed the instructions were being executed at an approximate rate of 4 per second.

STEP 12. Using the same technique, enter the value 365 (F5 hex) into the STEP register.

STEP 13. Press RESET and hold the STEP key to execute the program again.

Question: What did you observe in the display?

Answer: We observed the display executing instructions at a rate of about 15 per second. As the number stored in the step register increased in value, the speed of the multi-stepper increases. The speed increase is not linear, but follows a log function where the greatest change in rate occurs at the high end of the range. Also, the speed range starts at 001 and continues to 377 and then 000, which is the highest speed.

STEP 14. At location 000 277 (000 BF hex) enter the value 331 (D9 hex). This is an undefined instruction code.

Press RESET to return to your program.

STEP 15. Press and hold the STEP key.

Question: What did the display change to?

Answer: We observed the display execute the first instruction (CALL TIMEOUT), then the display indicated 000 277 ERR (00 BF ERR hex). The STEP function will not execute an undefined op code.

At this point, you should be able to use both the single step and the multi-step functions of the MMD-2 computer to slowly run a program so you can see what is happening. The ability to execute a program one instruction at a time is a very valuable tool for testing program operation.

## EXPERIMENT 11

The purpose of this experiment is to introduce you to the operation of the on-board priority interrupt controller. The system has on-board interrupts for the real time clock, keypad input and serial data input. This experiment will demonstrate the proper code to initialize the interrupt controller and how to use the real time clock and keypad interrupts. The serial data input interrupt circuit will generate an interrupt at the start of a received character. This interrupt is only reliable at 110 BAUD. Higher BAUD rates cause erratic decoding of the input character due to timing problems. For additional information in interrupts, consult TECHNIBOOK Volume VI, Chapter 23.

The first program segment uses the on-board real time clock to generate an interrupt at the line frequency, which is 60 times per second (60 Hz). Each time the interrupting signal occurs, the computer branches to the interrupt handler routine which, in this case, increments a counter and displays the result. The instructions, MVI A,010, OUT 006, are used to initialize the priority interrupt controller. For additional information about this device, consult the Reference Manual (Appendix I, 8214).

```

003 000 227          SUB A      ;CLEAR ACCUMULATOR
003 001 107          MOV B,A    ;SAVE FOR INTERRUPT ROUTINE
003 002 076 010     LOOP:      MVI A,010 ;SET UP CONTROLLER
003 004 323 006     OUT 006    ;INTERRUPT CONTROLLER PORT
003 006 373          EI        ;ENABLE INTERRUPTS
003 007 303 002 003 JMP LOOP  ;WAIT FOR AN INTERRUPT

```

### Interrupt Service Routine

```

000 060 170          MOV A,B    ;GET COUNTER VALUE
000 061 323 002     OUT 002    ;DISPLAY VALUE
000 063 074          INR A      ;BUMP COUNTER
000 064 107          MOV B,A    ;SAVE FOR NEXT INTERRUPT
000 065 311          RET        ;BACK TO MAIN ROUTINE

```

END



The second routine is used to demonstrate how an interrupt can be used to capture data entered from the keypad. This routine also has the code needed to initialize the interrupt controller. Every time a key is pressed, an interrupt is generated and the keypad service routine is used to input the key code and display the value on Port 2.

```

003 000 076 010      LOOP:    MVI A,010      ;SET UP INTERRUPT
                                           ;CONTROLLER
003 002 323 006                OUT 006      ;CONTROLLER STATUS PORT
003 004 373                EI          ;ENABLE INTERRUPTS
003 005 303 000 003        JMP LOOP   ;WAIT FOR INTERRUPT

```

Interrupt Service Routine

```

000 050 076 100                MVI A,100Q   ;SET UP KEYPAD HANDLER
                                           ;CHIP
000 052 323 003                OUT 003      ;CONTROL PORT FOR 8279
                                           ;CHIP
000 054 333 004                IN 004       ;GET KEY PRESSED CODE
000 056 323 002                OUT 002      ;DISPLAY CODE ON PORT 2
000 060 311                  RET          ;BACK TO MAIN PROGRAM

```

END

This experiment uses no auxiliary registers.

STEP 1. Enter the following code for program one, the real time clock interrupt:

HI ADDRESS	LO ADDRESS	INSTRUCTION BYTE	HI ADDRESS	LO ADDRESS	INSTRUCTION BYTE
003	000	227	03	00	97
	001	107		01	47
	002	076		02	3E
	003	010		03	08
	004	323		04	D3
	005	006		05	06
	006	373		06	FB
	007	303		07	C3
	010	002		08	02
	011	003		09	03

Then enter the following code for program for two, the interrupt service routine:

HI ADDRESS	LO ADDRESS	INSTRUCTION BYTE	HI ADDRESS	LO ADDRESS	INSTRUCTION BYTE
000	060	170	00	30	78
	061	323		31	D3
	062	002		32	02
	063	074		33	3C
	064	107		34	47
	065	311		35	C9

STEP 2. On the upper right hand corner of the system, locate the two pins connected with the line labeled CLK. Make sure that a wire is connected between these pins to connect the real time clock and the interrupt controller.

STEP 3. EXECUTE the program starting at location 003 000 (03 00 hex).

Question: What did you observe?

Answer: First, we observed the green LED labeled INTE turn on. This indicates that the 8080A has executed the EI instruction. Each time a pulse from the real time clock was detected by the interrupt controller, the program branched to the service routine, the value in the B register was moved to the accumulator, displayed on Port 2, incremented and restored in the B register for the next interrupt. In actual practice, the real time interrupt could be used to time external events, maintain a time of day clock with appropriate software, and other time-sensitive applications.

STEP 4. Enter the following code for the keyboard interrupt program:

HI ADDRESS	LO ADDRESS	INSTRUCTION BYTE	HI ADDRESS	LO ADDRESS	INSTRUCTION BYTE
003	000	076	03	00	3E
	001	010		01	08
	002	323		02	D3
	003	006		03	06
	004	373		04	FB
	005	303		05	C3
	006	000		06	00
	007	003		07	03

Then enter the following code:

HI ADDRESS	LO ADDRESS	INSTRUCTION BYTE	HI ADDRESS	LO ADDRESS	INSTRUCTION BYTE
000	050	076	00	28	3E
	051	100		29	40
	052	323		2A	D3
	053	003		2B	03
	054	333		2C	DB
	055	004		2D	04
	056	323		2E	D3
	057	002		2F	02
	060	311		30	C9

STEP 5. Make sure the two pins with the line marked KBRD are connected with a short piece of wire.

STEP 6. Execute the program starting at location 003 000 (03 00 hex).(Again, the the green LED labeled INTE should turn on.)

STEP 7. Press any keys on either keypad.

Question: What did you observe on Port 2?

Answer: We observed the display changed with each key pressed. When a key was pressed, the 8279 chip generated an interrupt signal which caused the interrupt controller chip to transfer program control from the main routine starting at location 003 000 to the interrupt service routine, starting at location 000 050 (00 28 hex). This routine sent a command to the 8279 chip to read the keypad and then input the key code from the chip. That code was then sent to Port 2 for the display.

At this point, you should have an understanding of the interrupt controller and how the on-board interrupts can be used. The interrupts for the keypads, real time clock and serial input are defined on the MMD-2 while the other five are available for user-defined interrupt signals from external devices.

## EXPERIMENT 12

The purpose of this experiment is to introduce the concept of breakpoints, and how they can be used in software debugging. Also, this experiment will provide you with specific instructions for using the breakpoint provided in the system's software. The specific restrictions associated with the use of this function will be discussed and demonstrated.

### Function of Breakpoints

Breakpoints are used by programmers to control a program being tested. A breakpoint is established at an address in the program where the programmer wants to have the debugging program, or monitor (in this case, EXEC C), regain control. When the program reaches the breakpoint address, a jump back to EXEC C is performed, the internal registers of the 8080A are saved and the programmer then has the ability to view the registers, alter the registers' contents, single step the



program, modify the program, or set the breakpoint to a new address. The breakpoint can also provide a convenient method for executing a loop in a program without having to single step through it. For example, if the breakpoint is placed in a loop, each time GO is pressed, the program will begin execution at the instruction after the breakpoint. Each time the breakpoint is encountered, program execution will halt and the programmer will have the option of viewing, modifying or rerunning the program. The breakpoint is very useful for testing new software, as it allows running the program in segments to more easily verify its operation.

### Limitations of Breakpoint Usage

The operation of the MMD-2 breakpoint has several limitations, which have to be understood for its proper use.

The breakpoint operates by inserting a three-byte JMP instruction into the program at the address selected for the breakpoint. When the breakpoint is encountered, the program will execute a JMP to EXEC C.

Since this method of implementing the breakpoint is used, the breakpoint cannot be directly used to debug EPROM programs or programs where the memory is write-protected. This problem is encountered if you are going to run the program using the GO key. However, if you use the multi-step option the breakpoint can also be implemented in write-protected memory or EPROM. The program can be run using the multi-step function and it will respond to the breakpoint. An additional problem is encountered if the breakpoint is to be inserted in the last memory location before nonexistent memory, EPROM, or write-protected memory.

Since the breakpoint requires three bytes, the last two will not be stored, causing the breakpoint to malfunction. This problem is again avoided if the multi-step function is used in conjunction with the breakpoint, because the breakpoint jump is not actually stored in the program. The single stepper compares the breakpoint address against the address of the instruction currently being executed. If the addresses match, EXEC C stops single stepping.

Another problem occurs if the breakpoint is inserted just before a data area which will be used by the program. In this case, the breakpoint insertion will change what is in the data area; or if the data area will be written into by the program, the breakpoint jump instruction will be altered. The first situation will cause a program error, because the expected data has been changed. The latter will cause the breakpoint to jump to an undefined memory location. Another consideration is that a breakpoint cannot be inserted into the second or third byte of an instruction. To do so will alter the program under test and cause unpredictable results. The breakpoint is inoperative in the single step mode. If the multi-step option is used, the breakpoint will function normally.

When RESET is pressed, the inserted breakpoint jump is removed and the original code is restored. In normal operation, you should never see the jump instruction EXEC C inserts to implement the breakpoint.

The program used to demonstrate the breakpoint will clear the accumulator, call the standard delay routine, increment the accumulator, display the accumulator

value on Port 2, and jump back to the delay routine.

```

003 000 227          SUB A          ;CLEAR ACCUMULATOR
003 001 315 277 000  LOOP:        CALL DELAY  ;TIME OUT ROUTINE
003 004 074          INR A          ;BUMP COUNTER
003 005 323 002     OUT 002        ;DISPLAY COUNT
003 007 303 001 003     JMP LOOP    ;DO IT AGAIN

```

### Delay Subroutine

```

000 277 365          TIMEOUT:     PUSH PSW    ;SAVE REGISTERS
000 300 325          PUSH D
000 301 021 046 001  LXI D,046 001 ;LOAD WITH VALUE TO
                                ;DECREMENT
000 304 033          MORE:        DCX D      ;JUMP IN THIS LOOP
                                ;UNTIL D&E ARE ZERO
000 305 172          MOV A,D      ;TEST FOR END OF LOOP
000 306 263          ORA E
000 307 302 304 000  JNZ MORE
000 312 321          POP D        ;RESTORE REGISTERS
000 313 361          POP PSW
000 314 311          RET

```

END

This experiment uses the BR1 HI., BR1 LO., and STEP registers.

STEP 1. Enter the following code starting at location 003 000 (03 00 hex):

HI BYTE	LO BYTE	INSTRUCTION BYTE	HI BYTE	LO BYTE	INSTRUCTION BYTE
003	000	227	03	00	97
	001	315		01	CD
	002	277		02	BF
	003	000		03	00
	004	074		04	3C
	005	323		05	D3
	006	002		06	02
	007	303		07	C3
	010	001		08	01
	011	003		09	03

Enter the following code at location 000 277 (00 BF hex):

<u>HI</u> <u>BYTE</u>	<u>LO</u> <u>BYTE</u>	<u>INSTRUCTION</u> <u>BYTE</u>	<u>HI</u> <u>BYTE</u>	<u>LO</u> <u>BYTE</u>	<u>INSTRUCTION</u> <u>BYTE</u>
000	277	365	00	BF	F5
	300	325		C0	D5
	301	021		C1	11
	302	046		C2	26
	303	001		C3	01
	304	033		C4	1B
	305	172		C5	7A
	306	263		C6	B3
	307	302		C7	C2
	310	304		C8	C4
	311	000		C9	00
	312	321		CA	D1
	313	361		CB	F1
	314	311		CC	C9

STEP 2. Press RESET, press GO.

Question: What did you observe on Port 2?

Answer: We observed the LEDs counting in the binary pattern.

STEP 3. Press RESET, AUX.

STEP 4. We will now insert a breakpoint into the program. In the BRL HI..register, store the value 003 (03 hex). Remember to press STORE.

STEP 5. Press NEXT. You will see the value 000 in this register. Enter the value 004 and store it. This is the address, 003 004 (03 04 hex) of the INR instruction in the program. We will use the BREAKPOINT feature to execute the program and stop at that instruction so we can view the value in the accumulator.

STEP 6. Press RESET. Press GO.

Question: What did you observe?

Answer: We observed (BREAK PT.1) on the display.

STEP 7. Press MEM.

Question: What did you observe?



Answer: We observed the address of the breakpoint (003 004) on the display and the instruction 074 (3C hex).

STEP 8. Press REGS.

Question: What value is in the A register?

Answer: We observed the value 000 in the A register because the instruction INR A has not yet been executed.

STEP 9. Press GO.

STEP 10. Press REGS.

Question: What is the value in the A register?

Answer: We observed the value 001. The program has been executed one complete time and the value has been incremented.

Question: What value is displayed on Port 2?

Answer: The value 001 is being displayed.

STEP 11. Press GO several times.

Question: What do you observe on Port 2?

Answer: Each time we pressed GO, we observed the binary value increase by one indicating that the entire program has been executed one time.

STEP 12. Press AUX. Press NEXT to advance to the STEP register.

STEP 13. Enter the value 377 (FF hex) and store it.

STEP 14. Press RESET, press and hold down the STEP key. The multi-step function will execute the program one step at a time as long as the key is pressed. You will note that it will take some time to step through the time delay before actually encountering the breakpoint.

The use of the breakpoint in conjunction with the multi-step mode is useful when you are debugging EPROM or write-protected memory. The program will function in the normal manner when GO is pressed and any breakpoint cannot be written into protected or EPROM memory. The use of the multi-step function in conjunction with the breakpoint allows the breakpointing of protected or EPROM memory.

If you do decide to press GO with a breakpoint set in EPROM, you will need to do one Power UP (PUP) RESET (via the PUP/RESET switch) in order to clear the breakpoint. (After you do the Power UP RESET, return the PUP RESET switch to the RESET position.)

STEP 15. Using the switch labeled WE $\emptyset$ , write-protect the first 1K of memory. This memory contains the program and time delay subroutine. Press REGS. Press NEXT to advance to the SH register. Enter the value 010 and STORE it.

STEP 16. Press RESET, then GO.

Question: What did you observe on the display and on Port 2?

Answer: We observed that the display had the address 003 000 and GO and Port 2 was counting in the binary pattern. The breakpoint is inoperative.

STEP 17. Press RESET. Press and hold the STEP key.

Question: What did you observe?

Answer: We observed the display flickering for about 10 seconds and then BREAK PT 1 was displayed. EXEC C will test for the breakpoint address if the multi-step function is used to run the program. While this feature won't allow the running of the program at full speed, it does allow the setting of breakpoints in either ROM, RAM or protected memory. In most instances, you will use the BREAKPOINT for debugging RAM memory where you will be able to run the program at full speed.

The breakpoint is very useful for testing conditional jumps or calls. You can set the breakpoint to the address of the jump or call to be tested and run the program. When you encounter the breakpoint, viewing the FLAG register will show whether the condition being tested is true or false. This information will let you determine whether the jump or call will be executed. Using this technique for running through a program, you will be able to follow the operation of the program by setting selected breakpoints and viewing data at the breakpoint.

## SECTION 5

### FINAL NOTES

REMEMBER...

YOU CAN'T DAMAGE A COMPUTER THROUGH PROGRAMMING!

Do not be afraid to try any type of program using the MMD-2 computer. Entering and running any program can have no effect on the computer itself. If you enter a program incorrectly, or if the program you enter isn't constructed properly, the only thing that can happen is that the program won't work. Nothing will happen to the computer hardware.

However, you can damage the system through misuse or physical abuse. The following precautions should be observed while using your computer:

- \* Make sure that it is connected to the proper power source. Connection to a DC power source or the improper line voltage will cause severe damage to the unit.
- \* Protect the components on the printed circuit board from coming in contact with metallic objects, stray pieces of wire, coins, tools, etc. These items can cause short circuits which can damage the computer.
- \* Double check all connections to the STD interface and the solderless breadboard connections. Improper connections to these areas of the computer can cause electrical malfunction or damage to the computer.
- \* Do not drop the computer.
- \* Do not store or operate the computer in areas where the temperature is excessively high or the atmosphere has corrosive elements in it.
- \* Protect the computer from "dirty tricks". Since the entire computer is exposed to allow you to see the actual computer, it is vulnerable to people changing the chips on the computer or plugging them in backwards or other types of vandalism. We feel that the advantage of having the computer exposed so you can see all the sections and components which make up the computer is a valuable asset. Hiding everything in a box makes the computer more mysterious and can make it somewhat intimidating to the beginner.

Again, remember that if you are doing only programming, you cannot possibly hurt the computer. If you are using the computer to interface an external circuit, the possibility for electrical damage does exist and we ask you to be very careful in making your connections to the computer.



APPENDIX

ASC II CODE TABLE (With Parity Bits = 0)

DIGITS		
	OCTAL	HEX
0	060	30
1	061	31
2	062	32
3	063	33
4	064	34
5	065	35
6	066	36
7	067	37
8	070	38
9	071	39

LETTERS					
UPPER CASE	OCTAL	HEX	HEX	OCTAL	LOWER CASE
A	101	41	61	141	a
B	102	42	62	142	b
C	103	43	63	143	c
D	104	44	64	144	d
E	105	45	65	145	e
F	106	46	66	146	f
G	107	47	67	147	g
H	110	48	68	150	h
I	111	49	69	151	i
J	112	4A	6A	152	j
K	113	4B	6B	153	k
L	114	4C	6C	154	l
M	115	4D	6D	155	m
N	116	4E	6E	156	n
O	117	4F	6F	157	o
P	120	50	70	160	p
Q	121	51	71	161	q
R	122	52	72	162	r
S	123	53	73	163	s
T	124	54	74	164	t
U	125	55	75	165	u
V	126	56	76	166	v
W	127	57	77	167	w
X	130	58	78	170	x
Y	131	59	79	171	y
Z	132	5A	7A	172	z

SYMBOLS		
	OCTAL	HEX
!	041	21
"	042	22
#	043	23
\$	044	24
%	045	25
&	046	26
'	047	27
(	050	28
)	051	29
*	052	2A
+	053	2B
,	054	2C
-	055	2D
.	056	2E
/	057	2F
:	072	3A
;	073	3B
<	074	3C
=	075	3D
>	076	3E
?	077	3F
@	100	40
[	133	5B
\	134	5C
]	135	5D
^	136	5E
←	137	5F
✓	140	60
{	173	7B
⋮	174	7C
}	175	7D
~	176	7E