A MULTIPURPOSE INTELLIGENT EPROM PROGRAMMER

by

ESREF DURMUS

B.S. in E.E., Bogazici University, 1981

Submitted to the Institute for Graduate Studies in Science and Engineering in partial fulfilment of the requirements for the degree of

Master of Science

in

Electrical Engineering

Bogazici University

1984

A MULTIPURPOSE INTELLIGENT EPROM PROGRAMMER

APPROVED BY

Y.Doc.Dr. Omer Cerid.....

Doc.Dr. Okyay Kaynak.....

Doc.Dr. Yusuf Tan.....

DATE OF APPROVAL : June 28, 1984

ACKNOWLEDGEMENTS

I consider it a prevailege to acknowledge those people who help and encourage me during my education and specially my thesis. First, I would like to thank Dr. Omer Cerid , my thesis advisor, for his kind help and understanding throughout my undergraduate and graduate study as well as my thesis. I would like to thank Doc. Dr. Yorgo Istefanopulos , my under graduate advisor. I also would like to thank to Mr. Tanju Argun and Mr. Semih Pekol who let me use the facilities of the research and development laboratories of NETAS.

ABSTRACT

This master thesis describes the design and implementation of a multipurpose intelligent EPROM programmer named as MPP.

Today; microprocessors are used in many kind of applications. In microprocessor based design; program editing, execution, debugging and the programming the final software to EPROMs are basic steps. In the market; there are many kind of eproms used as nonvolatile program storage in microprocessor based designs.

MPP system is designed as a microprocessor_based system. It utilizes the program, verify and duplicate most of the EPROMs in the market. It also supports intelligent programming and gang programming facilities.

MPP system is an multipurpose system. It features a data memory which allows data manipulation, program execution and debugging. It also allows communication utulity with external resources such as computers and development systems for data communication.

MPP system is designed as a compact system which provides a keyboard/display unit as a man_machine interface. Beside that an interactive CRT terminal interface is provided.

OZETCE

Bu master tezi MPP olarak isimlendirilen, genel amacli akilli EPROM programlayicisinin tasarimini ve uygulanmasini tanimlar.

Gunumuzde mikroislemciler cesitli uygulamalarda kullanilmaktadir. Mikroislemci tabanli tasarimda programmin yazimi, icra edilmesi, test edilmesi ve sonuc yazilimin EPROMlara yuklenmesi temel islevlerdir.

Mikroislemci tabanli tasarimlarda program depolamek icin imalatci firmalar cesitli ozellikte EPROMlar uretirler.

MPP sistemi mikroislemci tabanli sistem olarak tasarlanmistir. Bu sistem farkli ozellikteki EPROMlari programlamak, programlari dogrulamak ve cogullamak ozelligine sahiptir. Bu sistem ayni zamanda akilli programlama ozelligine de sahiptir

MPP sistemi genel amacli bir sistemdir. Bu sistemin ozellikleri arasinda bellegindeki verileri degistirebilme, programlari icra edebilme ve hata bulabilme vardir. Ayni zamanda Komputer veya mikroislemci gelistirme araci ile veri alisverisinede elverislidir.

MPP sistemi insan_makina iletisimini tus takimi/gosterge birimi veya CRT terminal ile gerceklestirir.

TABLE OF CONTENTS

I. THE MPP SYSTEM AND THE NEED FOR THE MULTIPURPOSE INTELLIGENT EPROM PROGRAMMER

The MPP system is useful in several aspects of microcomputer system development. It may be used to create a programmed EPROM from a source data file stored in a computer or microprocessor development system. The MPP system also may be used to debug and test a program which is loaded in his RAM buffer.

Today; In the market there are many kind of EPROMs supplied by the manufacturers. The programming and verification of different type of EPROMs rises unexpected problems in the development stage of microprocessor_based systems. The development of software of microprocessor based system is realized in development systems. Once the software is finalized an EPROM should be created. As described above; In development stage, two problems rise up. The first one is programming different kind of EPROMS and the second one is data transfer between EPROM programmer and development system.

In the mass production of microprocessor_based products, large amount of EPROMs may be programmed from a master EPROM or from a RAM buffer. If EPROMs are programmed one by one. A large amount of time is wasted. So that gang programming facility is required.

The introduction of higher EPROM densities up to 512Kbit, large programming time is required for programming, up to 28

minute. An intelligent programming algoritm for programming EPROMs save time.

The functions required from an EPROM programmer is summarized below.

i) Programming different kind of EPROMs

ii) Computer/Development system interface

iii) Intelligent programming

iv) Gang programming

v) Data manipulations in buffer memory

vi) Easy usage

An unintelligent EPROM programmer cannot realize all the features given above. So that a microprocessor-based multipurpose intelligent EPROM programmer is designed called as MPP. MPP system has stored program logic. The functions of this system is controlled by the operational system S/W. The selection of software control logic provides the flexibility and upgrating the MPP functions without any modifications on the system hardware.

II. INTRODUCTION

The MPP system is an microprocessor-based intelligent sytem. It provides the following functions.

i) Programming different type of EPROMs

ii) Programming multiple number of EPROMs from buffer

iii) Programming multiple number of EPROMs from a master EPROM

iv) Verify EPROMs with buffer and master EPROM.

v) Transfer EPROM contents to buffer

vi) Check erasure of EPROMs

vii) Provides data communication with development systems.

viii) Provides a keyboard/display unit for man_machine interface

iv) Provides Interactive CRT terminal interface as man-machine interface

v) Provides a monitor S/W for data manipulation, program execution and debugging.

MPP system is a stored program logic. This system is realized in two stage; System software and system hardware. System block diagram is given in figure 1.

MPP hardware mainly consists five circuit pack to achieve the required functions. These are;

i) Processor CP.

ii) Memory CP.

iii) Programmer CP.

iv) Keyboard/Display CP.

v) Programmable power supply.

Processor circuit pack is the main control element in the system. It contains a single chip 8085 microprocessor to execute all logical functions required, and I/O devices.

It contains four I/O unit. Two of them is serial and the others parallel I/Os. One of serial I/O device is used for communication with an interactive CRT terminal and the other one is used for communication with a microcomputer development system. Parallel I/Os are used for communication with keyboard display interface and programmer card.

Memory CP. provides data and program storage area for the system. It consists of 16Kbyte RAM buffer for data loading. and 4 Kybte program storage for system software.

Programmer CP. provides the necessary interface for programming EPROMs under processor card and system S/W control.

Keyboard/Display CP. is one of the man_machine interface of the system. The other one is CRT terminal. These interfaces are strap selectable. Keyboard/Display CP contains a display panel, key pad and necessary driving circuitry. This CP. is controlled and refreshed by the processor.

Programmable power supply provides the necessary voltages required by the EPROM programming under processor control.

These circuit packs are explained in detail in the later chapters.

MPP software mainly consists of three modules. These are ;

i) Monitor module

ii) Programmer module

iii) Foreground tasks

Monitor S/W provides the commands for data manipulation, program execution and degugging. It also contains the computer communication and CRT display communication routines.

Programmer S/W provides commands for processing different type and different number of EPROMs plugged in the programmer card.

Foreground S/W provides the control and refreshing of the display/keybord card. The detailed discussion of the system S/W is given in later sections.

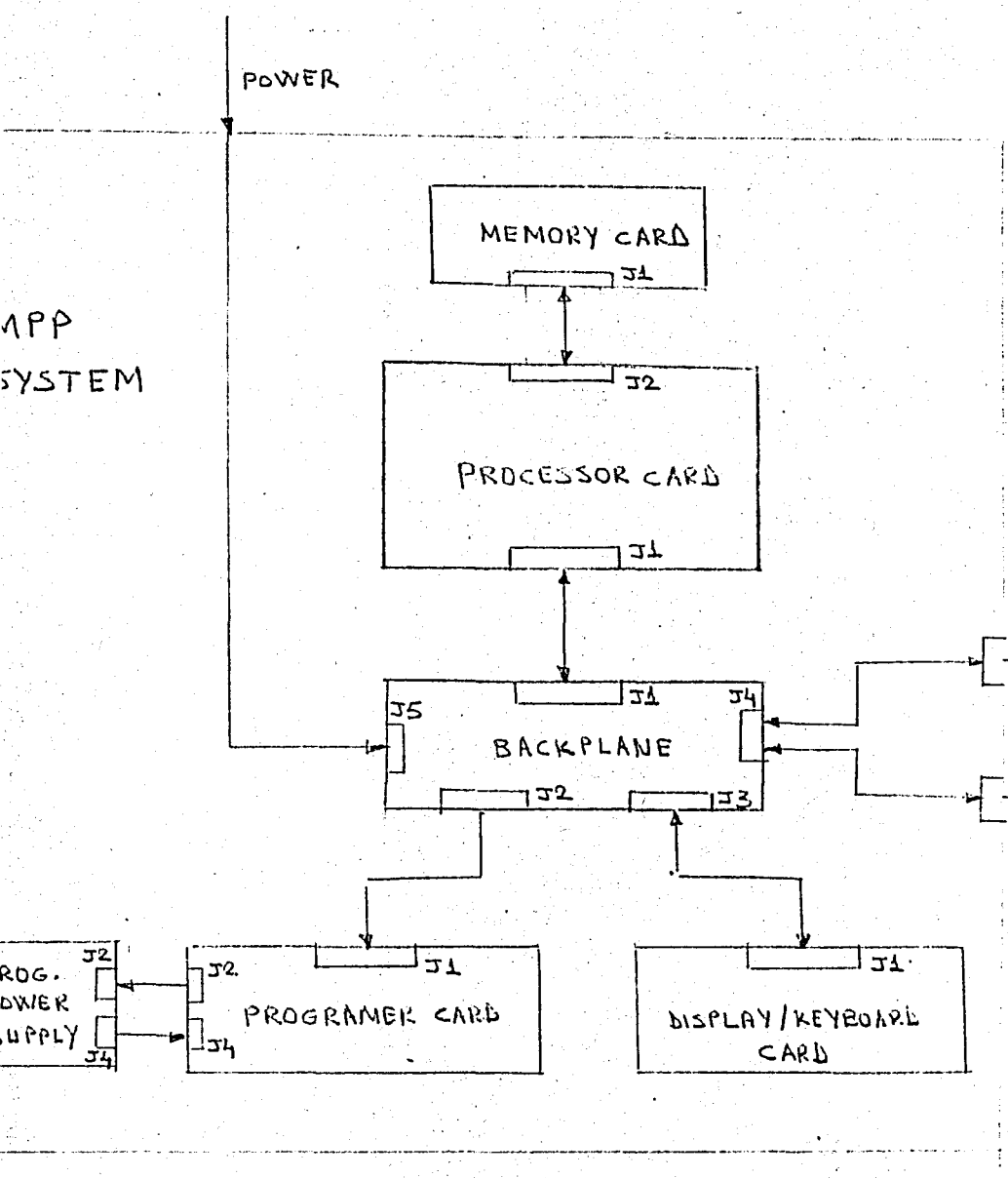


Figure 1. System block diagram

III. PROCESSOR CARD

In MPP system, all system control resides in two circuit packs, namely a PROCESSOR card and a MEMORY card. Memory card provides program and data storage for the system. Processor card contains microprocessor and Input/Output units to communicate with the peripheral cards.

Processor card mainly consists of seven blocks. These are given below. The block diagram of the processor card is given in figure 3.

i) Microprocessor and support circuit

ii) Memory card interface

iii) Programmer card interface

iv) Keyboard/Display card interface

v) CRT terminal intarface

vi) Computer interface

vii) Interrupt

viii) Decoding

3.1 MICROPROCESSOR AND SUPPORT CIRCUIT

Processor card has been designed using the INTEL 8085 microprocessor to execute all the logical functions reguired. This processor is an 8 bit microprocessor, It has 64Kbyte addressing capability and operates at 3.072 MHz clock rate. It

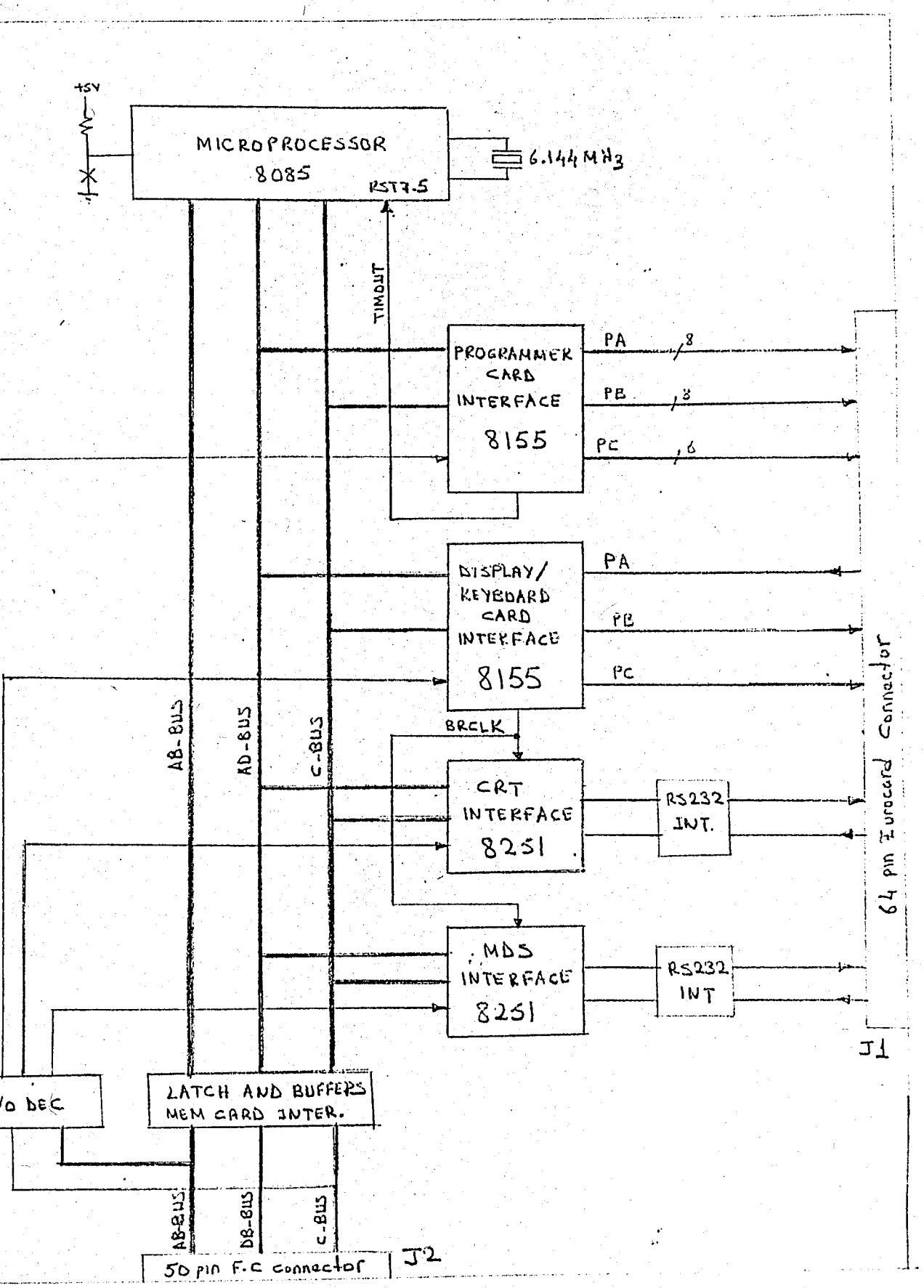


Figure 3. Processor card block diagram.

provides five interrupts inputs. The RST7.5 one is used to interrupt the processor at regular intervals for keyboard/display scanning.

Clock generation for the microprocessor is achived by an 6.144 MHz crystal oscillator and bypass capacitors. Microprocessor divides this clock frequency by two and generates 3.072 MHz system clock.

Reset circuit is used to generate reset signal for the microprocessor and peripheral ICs upon power on and reset switch is pressed.

3.2 MEMORY CARD INTERFACE

Memory card buffering is provided to ensure correct processor memory communication.

Address bus of the processor is buffered by 74LS373 latch buffers. Low part of the address bus is multilex address/data bus so that the latch/buffer on this bus is used also to latch the low part of the address bus. Data bus is buffered by a bidirectional buffer 74LS245. The direction of this buffer is controlled by RD signal. and it is only enabled when memory access is encountered to prevent bus contention.

3.3 PROGRAMMER CARD INTERFACE

Programmer card interface is achieved by an programmable parallel port(8155). This IC provides two 8 bit(PortA,PortB) and one 6 bit(PortC) ports. The portA is used for data bus for latches on the programmer card. The PortC is used for latch control and PortB is used for control signals required for Vpp control etc.

3.4 KEYBOARD CARD INTERFACE

This interface is also achieved by an 8155 IC. The PortA of this IC is used for key detection PortB is used for row driving and PortC is used for column scanning.

3.5 CRT TERMINAL INTERFACE

This interface provides the data transmission path between processor and CRT terminal. This interface is according to RS232 specifications. Asynchronous serial transmission method is used for communication with the CRT A 8251 USART IC is used for this purpose. Baud rate generation of data transfer rate is achieved by the timer section of 8155 IC and software. This baud rate is set up to 2400 baud.

3.6 COMPUTER INTERFACE

This interface provides the data transmission between MPP system and a computer via RS232 bus. This interface is same as CRT interface.

3.6 INTERRUPT LOGIC

A timer interrupt is generated by the timer section of an 8155. The processor is interrupted at regular intervals for keyboard/Display scanning. This interval is set to 1msec by software and causes the processor to enter the foreground mode.

3.8 DECODING

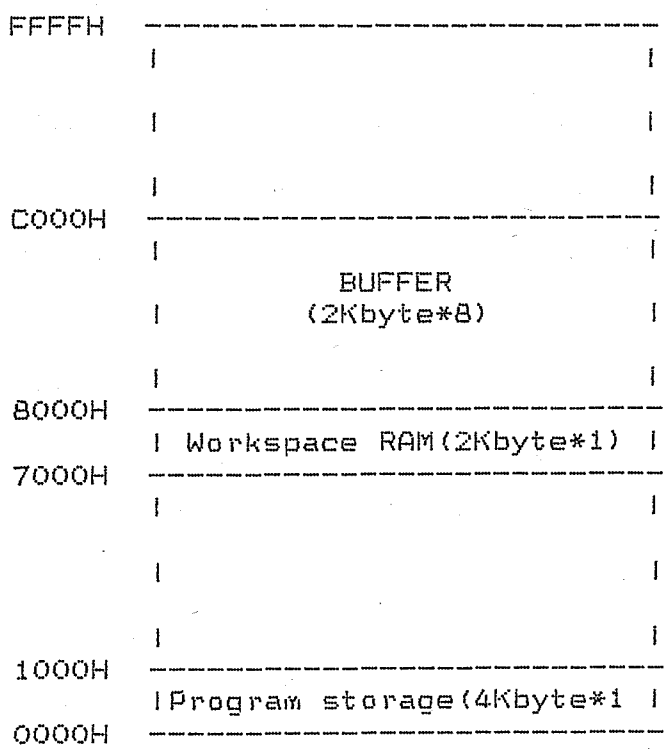
This card has four I/O units. These are two PPIs and two USARTs The address decoding for this units is achieved by using a 3 to 8 decoder. The address of these units are.

I/O Address(Hex)	Device	Function
00	PPI1	Command/Status reg.
01	PPI1	PortA
02	PPI1	PortB
03	PPI1	PortC
04	PPI1	Timer low byte
05	PPI2	Timer high byte
08	PPI2	Command/Status
09	PPI2	PortA
0A	PPI2	PortB
0B	PPI2	PortC
0C	PPI2	Timer low byte
0D	PPI2	Timer high byte
10	USART1	Data register
11	USART1	Command/status
18	USART2	Data register
19	USART2	Command/status

IV. MEMORY CARD

Memory card provides data and program storage area for the system. Present system requires 4Kbyte program storage. A 2732 EPROM is used for this purpose. It is possible to expand this area up to 28Kbyte. 2Kbyte system data storage area and 16Kbyte Buffer area is provided. Data storage is made up of 6116 2Kbyte RAM ICs. Decoding is achieved by 74LS138 decoders.

Memory map is given in figure below.



V. PROGRAMMER CARD

Programmer card is one of the peripheral cards in the system. The function of this card is to provide the interface to program eproms under processor card and software control.

This card is such that it is possible to connect it to any computers which supports 3 8 bit parallel port. In MPP sytem PortA, PortB and PortC of an 8155 on the processor card is used for this purpose.

This card consists of five latch/buffers(74LS373) and one bidirectional buffer. Latch/buffers are used to expand the I/O lines of the processor and to increase the driving capacity. PortA is used for data bus for latch/buffers. PortC is used for latch/buffer control and PortB is used for Vpp, Vcc, PGM and OE control.

The latch process is executed as follows;

1. Disable all latches and buffers
2. Send data to portA
3. Enable reletad latch
4. Disable all latches.

The address information for eproms are generated from latches U3 and U4. The chip select information for eproms are generated from latch U2. and LED information is generated from U1 latch. Data information of EPROMs are bidirectional so that a bidirectional buffer is used It is direction and three state

control is also control from PC. The other control signals such as Vpp control is directly obtained from PB via U6 buffer.
This card provides five sockets for eproms called as master, slave 1, slave 2, slave 3 and slave 4. It is possible to process different kind of eproms by changing the personality module on programmer card and selecting the proper software switches as described in the man-machine interface.

The connection of programmer card other than processor card is programmable voltage supply card which provides the programming voltage Vpp and supply voltage EVcc for eproms. The control inputs for this card is taken from processor card and buffered on programmer card.

VI. DISPLAY/KEYBOARD CARD

Display/keyboard card is one of the peripheral card in the system The function of this card is to provide the man-machine interface of the system. This card is such that it is possible to connect it to any computer which supports 22 I/O lines. In MPP system, This card is connected the PA, PB and PC of an 8155 located on processor card.

The function of this card is to display messages which is sent from the processor card and to constitute a path through the pressed key for recognation of it by the processor. This CP. mainly consists of two circuit. These are; Display circuit and Keyboard circuit.

6.1 DISPLAY CIRCUIT

This circuit consists of a display panel and necessary driving and matrix addressing circuitry. Display panel is made up of 16 seven segment digits and 16 rectangular LED. The following functions are provided by this circuit.

i) Driving sufficient current to light Digits and LEDs

ii) Constituting a path to light a certain Digits or LEDs according to corresponding matrix address supplied by the processor.

iii) Providing a display panel which consists of 7-

segment digits and LEDs.

The elements of the display panel are organized as an LED matrix. Each segment of digits are considered as a single LED in the matrix configuration. This matrix is constituted 16 columns by 8 rows.

Display process is time multiplexed and information is periodically refreshed by the processor. This information is internally represented as the status of the digits and LEDs stored in the processor memory. This table is updated by the processor according to the information change. The processor activates each column of the LED matrix for 1msec at every 16msec intervals. When a column is selected , the rows are made high or low according to the contents of the corresponding state table entry. Thus any segment of digits or LEDs on the current column is turned on/off whether the corresponding row is high or low. Since each column is activated once every 16msec the duty cycle of a single LED is 6.3 % which dictates the use of high efficiency digits and LEDs. The refresh rate is 63Hz which provides a stable flicker_ free display.

In Display/Keyboard card, the portC is used for column information. This information is sent from the processor as coded so that two 74LS156 3 to 8 open collector decoders are used to decode this information to 16 column data because ane column is selected at atime and driving the column drivers. Column drivers are used to drive sufficient current to colums of the LED matrix. Column drivers are realized by BD370 PNP transistors 2K2 pull-up resistors ans 390ohm base current limiting resistors. The portB

is used for row information to the LED matrix and this signals are also buffered by ULN2003 driver and a transistor.

In this card; For seven segment digits MAN72A are used. These are high efficiency light output displays and for LEDs high efficiency rectangular LEDs COW10 are used. MAN72A digits are common anot digits. In the LED matrix the anots of the digits and LEDs are connected to the column drivers. Segments and cathod of LEDs are connected to the rows of the matrix so that columns source current and rows sink the current. Constituting a path to light a certain LED according to corresponding matrix address supplied by the processor is as follows.

According to column information one output of decoders selected. Because of open collector drivers. The selected output goes into low level. so that the emitter_base function of transistor is forward biased and it goes into saturation. so that selected column is high level and source current. In order to light a segment of a digit or LED on the selected column The bit in the row information correspond to this segment must be high because row buffer ULN2003 is inverted buffer. when it is input is one it pulls the outputs to low level so that the corresponding segment is connected to ground by a 24ohm current limiting resistor. According to the value of this resistor and emitter-collector saturation voltage drops of the 2003 and BD370 approximately 100mA current flows through the segments since each column is activated ance every 16msec then 6.3mA average current flows through the LEDs. In the other side if all rows are active then 100mA*8=800mA current is flows through BD370 transistor. This transistor is 1A current capacity thus it meets this

requirement. Since one row is active at a time so that only 100mA current flows through 2003. It has 500mA current capacity and meets the requirement.

6.2 KEYBOARD CIRCUIT

Keyboard circuit includes keys. The function of this circuit is to constitute a path through the pressed key so that processor can locate it in the keyboard matrix.

Keys on the keyboard panel are wired as a matrix confiquration. This matrix is, in fact, the virtuel extension of the display matrix. 5 columns by 8 rows. The colums of the key matrix is driven from the decoder which drives column drivers on the display circuit. The rows of keyboard matrix is connected to the portA of the 8155 IC on the processor card.

The key scanning algorithm is as follws; As described above processor activates each column. Keyboard software checks that this column is also a key column or not if it is also a key column then it reads PortA to check that any key closure. If any key closure is detected location of the pressed key in the key matrix is determined by the processor. Thereafter the processor checks the pressed key periodically for contact debouncing and whether it is an valid key closure. If it is ascertained, the processor process this key.

Sequences in a typical key closure action is given below Keys are scanned at 1msec intervals and if any key closure is detected this closure is scanned at 16msec intervals. Any key closure

action is rejected by the processor unless the following requirements are satisfied.

i) A pressed key should stay in pressed position for at least 16msec.

ii) A pressed key should stay in released position for at least 16msec.

The PortA inputs are pull up to 5V level by 2K2 resistors. As described above decoder which drive colums are active low so that key clousure is detected as low level in portA. If any key is not pressed then portA inputs high level in all bits.

VII. PROGRAMMABLE POWER SUPPLY

The function of this card is to provide necessary voltages for the programmer card. The control inputs of this card is taken from the processor card through the programmer card. This card consists of two circuit. The first one generates the programming(Vpp) voltages for the eproms and the second one generates supply voltages(EVcc) for the eproms.

Eproms are required different voltage levels on Vpp pins for programming. for example 2716 requires 0V-25V, 27128: 0V-21V,27256:0V-12.5V,68764:0V-5V-25V MPP system is capable of the programming most of the eproms in the market so that a software controlled supply is designed This supply is a linear regulator it is output voltage is controlled by changing the feedback resistor. This is provided by inserting or removing resistors in parallel to fixed feedback resistor. The 0V output is taken by grounding the shutdown pin of the regulator. The LM305 regulator is used for this purpose. but it is output current is not enough to supply 250mA current. Eproms are required 50mA current for Vpp inputs and It will be possible to program 5 eproms in parallel. So that 2N5415 and BD137 transistors are used. The reason for the use of LM305 is that it is possible to pull down it is output to 0.7V by grounding shutdown pin and a series diod is replaced to the output to drop this voltages. so that output goes into 0V. The state of the control inputs for selecting different voltages are given below.

Vp4	Vp3	Vp2	Vp1	Vpp
1	0	0	0	0V
0	0	0	0	5V
0	0	0	1	12.5V
0	0	1	0	21V
0	1	0	0	25V

Eproms require 5V on supply pins in normal programming mode. But some eproms provide intelligent programming capability and requires 6V supply voltage Beside that MPP system supports xxxx socket property so that three voltage level is required. 0V, 5V, 6V.

This voltages are generated from a linear regulator called as LM317 by changing its fedback resistance under software control. The control signals of this circuit is also taken from processor card through programmer card.

The state of the control inputs to select different voltages are given below.

Vcc2	Vcc1	EVcc
1	0	0V
0	1	5V
0	0	6V

VIII. SYSTEM SOFTWARE ARCHITECTURE

8.1 INTRODUCTION

MPP system is an microprocessor based system. It has stored_program logic. The functions of this system is provided and controlled by the operational MPP S/W. The selection of S/W controlled logic provides flexibility and simplicity of the MPP H/W. The MPP S/W is structured in a modular fashion so that each system task is responsible for only one of the MPP operation. This feature provides higher reliability and better understanding of the MPP H/W.

MPP Software consist of mainly three modules. These are; monitor S/W, Programmer software and foreground tasks.

Monitor S/W is a supervisor program designed for program evaluation, debugging and testing on MPP processor card using the 8085 microprocessor.

Programmer S/W is a program design for programming different types and different numbers of eproms located on the programmer card.

Foreground S/W is designed to provide the interface between user and the system by the way of keyboard/display CP. beside that a remote terminal(CRT,Development system) interface is

provided in the background S/W.

8.2 S/W DESIGN HIGHLIGHTS

i) Programming language is ASM85

ii) Software design is structured and modular

iii) Organized as two levels, background and foreground

iv) Requires only one interrupt input which is used to activate the foreground

8.3 ARCHITECTURE

The MPP system provides the interface to the user from a CRT terminal or from a keyboard/display path. CRT terminal interface does not require any refreshing processes. It sends to an ASCII character to MPP when a key is pressed or MPP sends ASCII characters to CRT terminal when it wants to display messages. Beside that keyboard/display interface requires refreshing process for pressed key detection and to display messages. So that MPP software is designed in two levels. These are; foreground level which provides Display/Keyboard scanning and background level for non-time critical processes such as monitor and programming software.

The MPP software is designed such as all system process are executed either in foreground or in background levels. Foreground is entered at 1 msec intervals as a result of hardware interrupt. Background is entered whenever the foreground executive releases control. A typical system cycle is shown in Fig 8.1

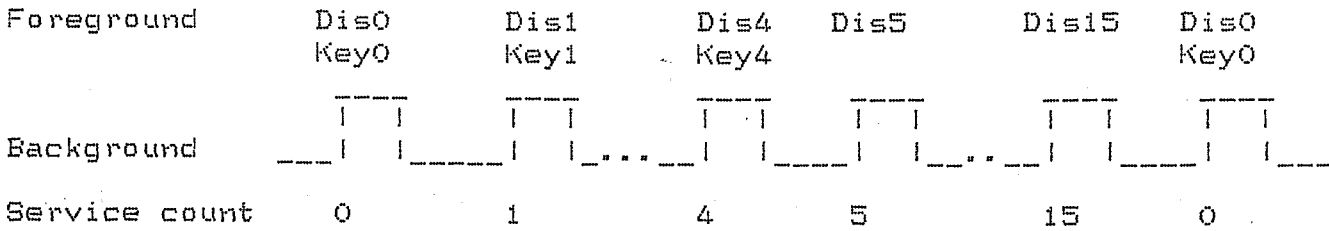


Figure 8.1

MPP S/W hieearchy diagram is given in Fig 8.2 and S/W organization diagram is given in Figure 8.3.

8.4 FOREGROUND

Foreground contains the basic, time critical system tasks; Display refresh and Keyboard scan. All process are sheduled by a shedular, DCLCNT which is controlled by foreground executive. The foreground is entered at every 1msec. Upon the FORGND is activated, current status of CPU registers are saved before the foreground processes are activated. Then processes are activated according to the current value of the DCLCNT. DCLCNT is incremented by one for next execution. The old status of the CPU registers is reassigned. Then control is passed to background level. Flowchart of the foreground level is given in Fig 8.4

Functions of the foreground system tasks are summarized below.

i) Display refresh: To refresh one column of the LED matrix at one time(at 16msec intervals)

ii) Keyboard Scan: To scan the keyboard matrix for any key closure.

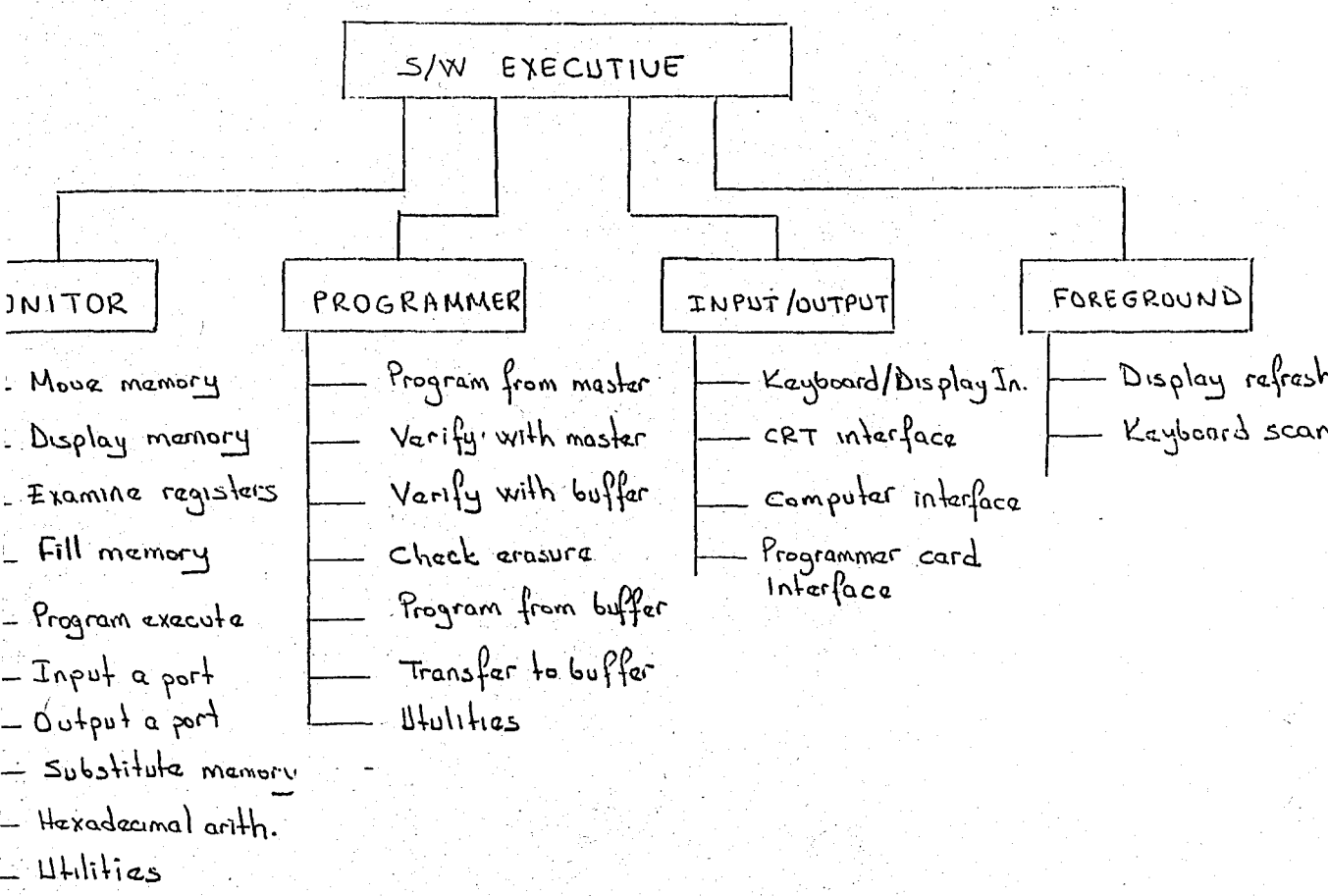


Figure 8.2 S/W hierarchy diagram

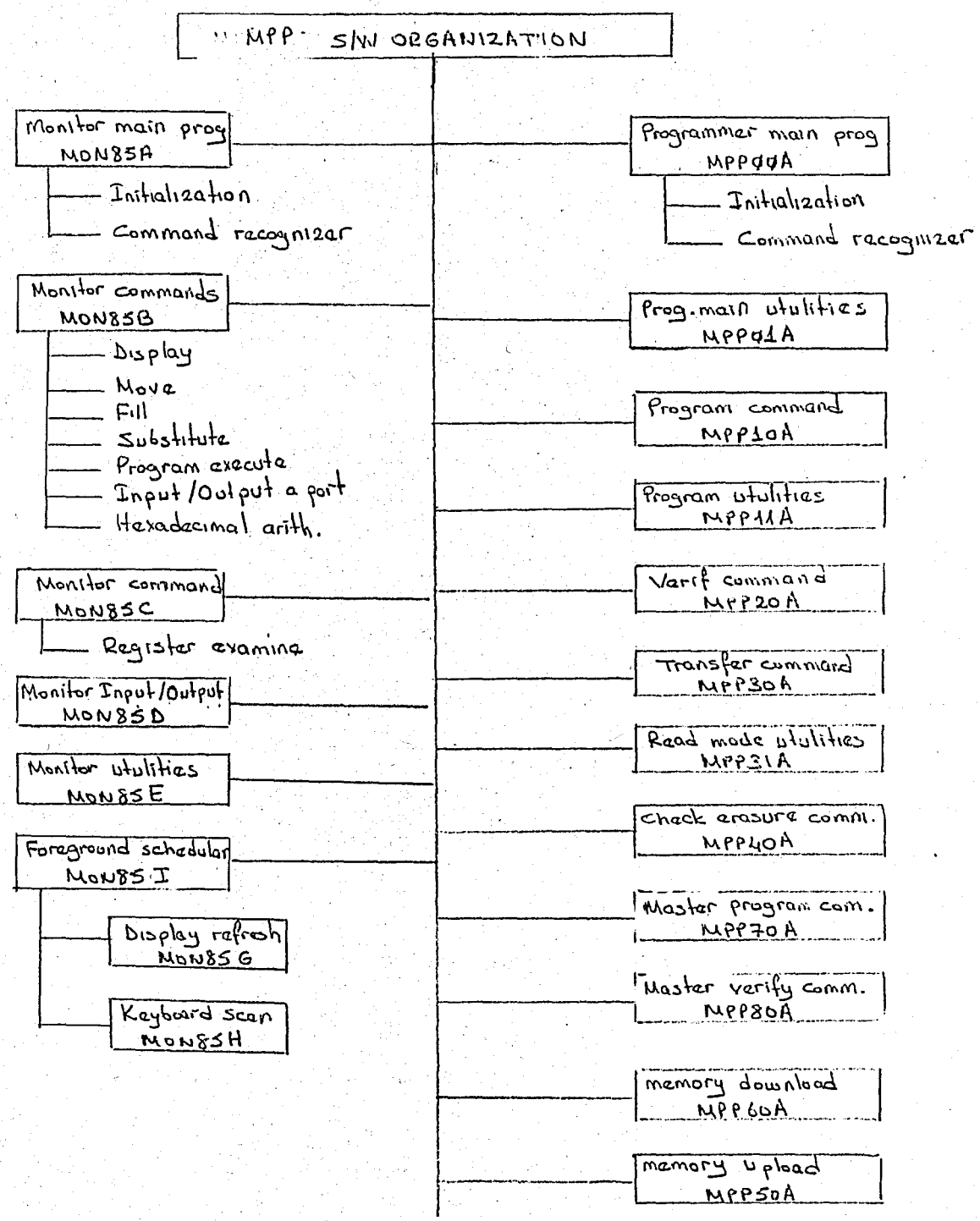


Figure 8.3 S/W Organization Diagram

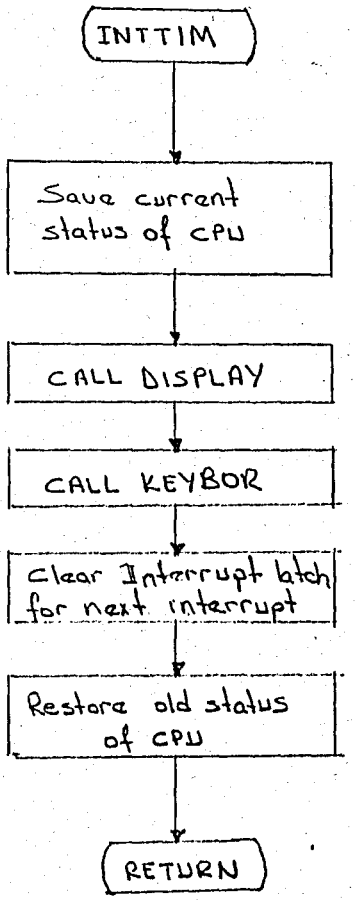


Figure 8.4 Flowchart of foreground executive

8.5 BACKGROUND

Background contains non-time critical system tasks such as monitor S/W and programmer S/W. Their functions are summarized below.

Monitor S/W :To provide commands for program evalutain, debugging and testing. It allows the user to enter, check out and execute his programs. It contains facilities for memory display and modifications, CPU register display and modifications, program initiation, break point insertion and detection. It also provide input/output routines to provide the interface between the user and system.

MPP monitor provides the communication of the MPP sytem with remote computers

Programmer S/W : To provide commands for processing different kind of eproms plugged in programmer card such as program from buffer and master, verify with buffer and master , transfer, compare.

IX. FOREGROUND TASKS

9.1 DISPLAY REFRESH

The function of Display refresh is to refresh one column of the display matrix at each execution

The data bases required by this task is given below.

DISBUF: 16 bytes long. DISBUF is a buffer which is used to keep the state of each LED and segments of digirs in the display panel.

DRWPTR: 2 bytes long. This is a pointer to DISBUF for PortB

DCLCNT: 1 byte long. This an offset to DIGCOL table to obtained pysical column adress from logical column address

DIGCOL: 16 bytes long. This a conversion table which provides the conversion from logical column adr to pysical column adr.

This procedure is executed at every 1 msec. At every execution of this procedure, one column of the display matrix whose column number equals to the current DCLCNT value is refreshed as illustrated in figure 1. As seen in the figure, Foregroud executive shedulere cycles at every 16 msec. Thus each column of the display matrix is refreshed at the same rate which correspond to 63Hz refreshing frequecy.

Detailed flow-chart of the Display refresh procedure is given in figure 9.1. Operation of the process is controlled by a counter DCLCNT. each entry on the current column is turned on/off

lization ; DCLCNT ← 15

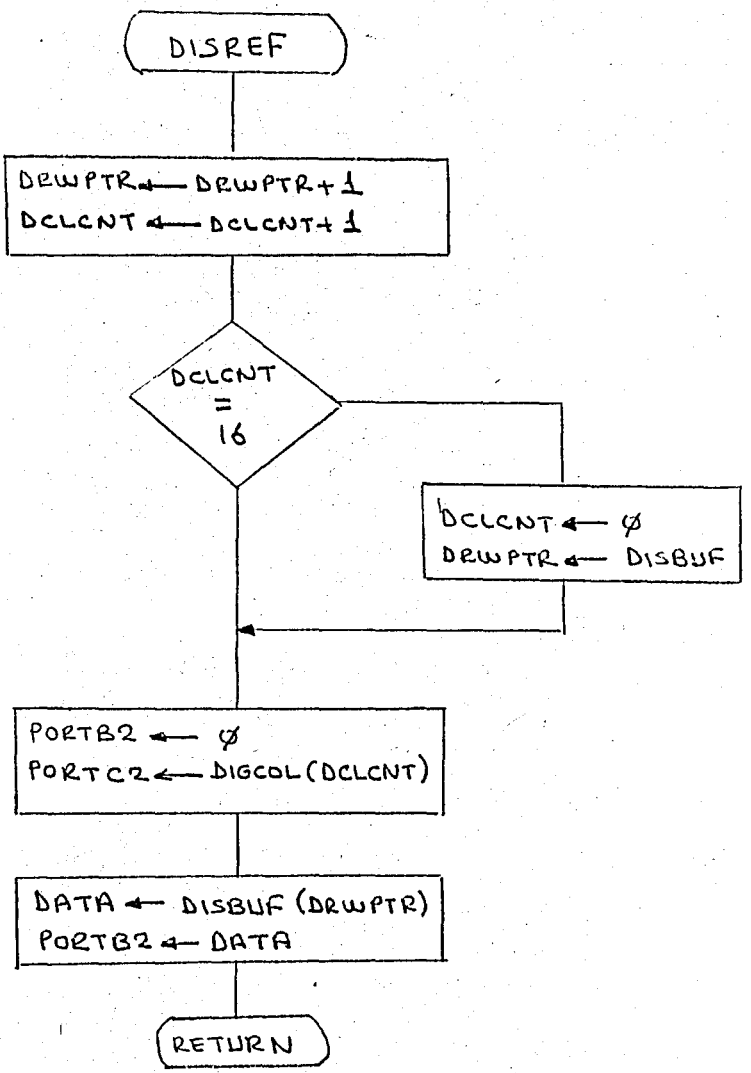


Figure 9.1 Detailed flowchart of DISREF

according to its state buffer, DISBUF. each byte of this buffer represents the information related to column number. Each bit of a byte corresponds to a particu segment or LED on the related column. The pointer, DRWPTR is used to access to the related byte of the DISBUF for the current column. The refreshing process is executed as follws.

i) The real column adr. is taken from DIGCOL table according to the current value of DCLCNT.

ii) Turn off data(00) is outputted from portB(rows of the matrix) to to turn off previous data.

iii) Column is selected according to column adr.

iii) The row data related to DCLCNT is outputted from portB.

At the end of this process the current column is refreshed.

A typical Digit/LED turn off progress would progress as follows.

i) Monitor S/W sends data to output procedure named as CO as ASCII form.

ii) Output routines check that CRT or keyboard/display module is in progress.

iii) If CRT is in progress then it sends this ASCII data to terminal.

iv) If Keybord/Display is in progress then it converts this ASCII data to seven segment form and replace this data according DISBUF entry.

v) Display refresh refreshes the DISBUF entry with this new value.

9.2 KEYBOARD SCAN

The Keyboard scan provides the following functions

i) Scanning the columns of the key matrix to detect any key closure

ii) To check the pressed key for debouncing

iii) Sampling the pressed key to be able to sense when the pressed key is released.

iv) To check the released key for debouncing.

v) To convert the key code to ASCII form

The data bases required by this task is given below.

KEYFLG: 1 byte ;Pressed key status

0 = No pressed key, search key matrix

1 = Check debouncing for pressed key

2 = Pressed key is still in pressed position

3 =Pressed key is still released and check for debouncing

KEYCOL: 1 byte ; Keeps pressed key column number

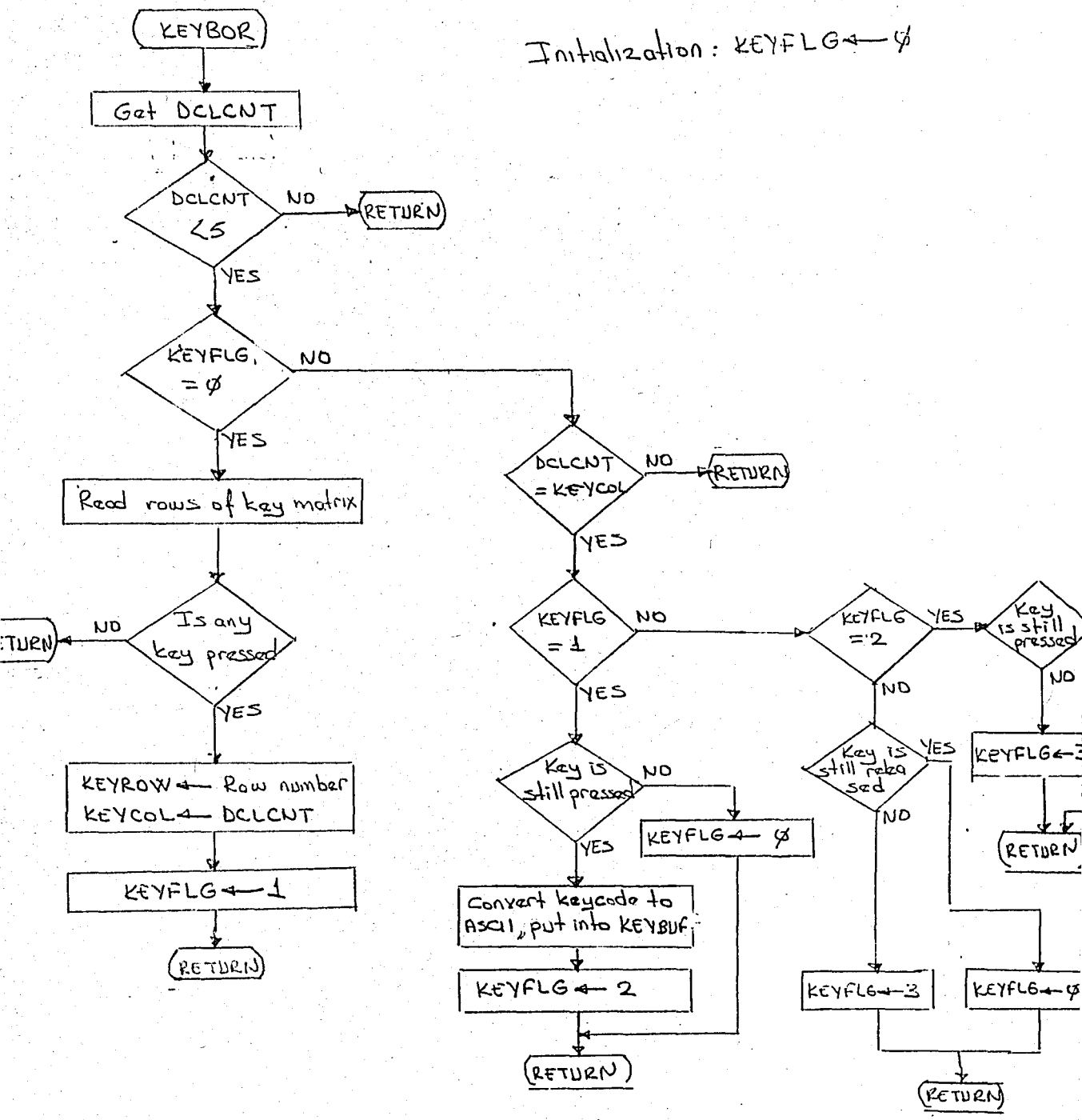
KEYROW: 1 byte ; Keeps pressed key row information

KBNBUF: 1 byte ; Keeps information on pressed key and offse to the CONTBL for ASCII conversion

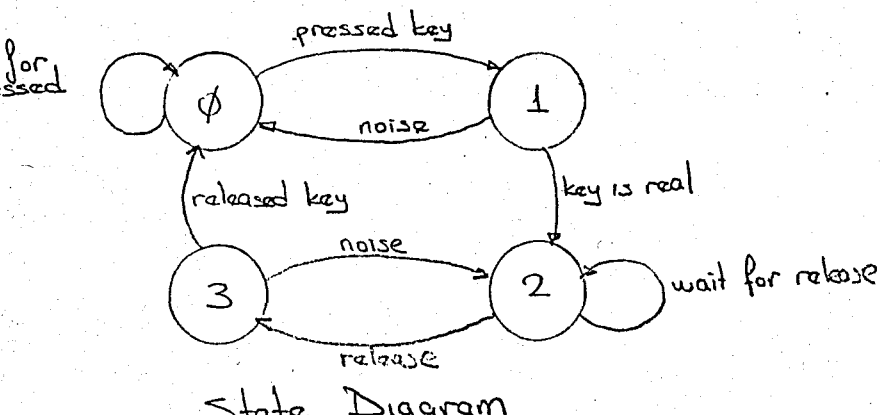
CONTBL: 54 byte ; ASCII coversion table for the pressed key

Keyboard scan is executed according to the value of DCLCNT. Key matrix is replaced in the first five column of sixteen row of the display matrix. So that it is executed in the first five steps of the foreground executive.

A detailed flow-chart and state table of the KEYBOR is given in figure 9.2.



Detailed flow chart of KEYBOR



State Diagram

Figure 9.2

Operation of the procedure is basically controlled by a flag named as KEYFLG. The KEYFLG serves as a switch for four function of the process mentioned above KEYFLG = 0,1,2,3 corresponds to functions i,ii,iii,iv respectively.

As described in the Display scan, Display columns are scanning according to the value of DCLCNT. Keyboard scan is also executed from 0 to 4 of the DCLCNT. When a column is activated related to key matrix keyboard S/W reads portA to check any key pressed or not. If any key pressed it saves the current DCLCNT value in KEYCOL as column number, the value read from PortA as row information and updates KEYFLG to 1. Afterwards key procedure is not executed until DCLCNT is equal to the pressed key column number(KEYCOL) It means that 16 msec time elapse. When DCLCNT is equal KEYCOL it reads portA to check pressed key is still pressed or not. If it is pressed it converts the pressed key code to ASCII form and put into KEYBUF for Bakground input routine(CI). and set KEYFLG to 2. else it is considered as a detection failure.

The same process is applied for release of the key as seen from the state table.

X. MONITOR SOFTWARE

MPP monitor is a supervisor program in the MPP system. It allows the user to enter, checkout and execute his programs and contains facilities for memory and CPU register modifications. This monitor communicates with the user an interactive CRT terminal or a Keyboard/Display pack.

The monitor begins the dialogue by typing the sign on message: SYS READY. Commands are in the form of single letters and are:

A :Move Memory

B :Write a record to external computer.

D :Display memory

E :Examine and modify CPU registers

F :Fill memory with data

G :Progarm execute and break point insertion

I :Input a port

L :Read a record fro an external computer

O :Output a data to a port

P :Enter programmer commands

S :Substitute memory

T :Hexedecimal arithmetic

Monitor S/W has a modular, top down structure, distrubuted in five source files. These are main program, command implementing routines, I/O utulities and miscellenous utulities.

Main program initializes system software and system hardware,display prompt messages, wait and receives an input character from the user and attempts to locate this character in its command table(CTAB). If succesful. The routine corresponding to this character is selected from a table of command routine address. and control is passed to this routine. If the character does not match any entries control is passed to the error handler.

Monitor provides a number of small service routines which may be prove to be useful for the user. The names and discussions of this subroutines and programm listings are given in the listings.

10.1 MONITOR COMMANDS

Move memory(A):

_A <LOW ADR OF MEMORY> <HIGH ADR OF MEMORY> <LOW ADR OF DEST>

This routine expects three hexadecimal parameters from the console. The first and second parameters are the bounds of the memory area to be moved. The third parameter is the destination area.

Write hexedecimal record(B):

_B <LOW ADR OF MEMORY> <HIGH ADR OF MEMORY>

This routine expects two hexedecimal parameters which are interpreted as the bounds of a memory area to be encoded into hexedecimal format and punched on the assigned punch device.

Display memory(D):

_D <LOW ADR OF MEMORY> <HIGH ADR OF MEMORY>

This routine expects two hexadecimal parameters specifiying the bounds of a memory area to be displayed on the list device. The memory area is displayed 16 bytes per line, with the memory address of the first byte printed for reference. All lines are blocked into integral multiples of 16 for clarity, so the first and last lines may be less then 16 bytes in order to syncronize the display.

Examine register(E)

_E<REGISTER NAME>

This routine allows the user to exemine and/or modify the contents of the user program's registers. The register values were stored as a result of a previous breakpoint and will be restored to the user program during a subsequent 'G' command,

Fill memory(F)

_F <LOW ADR OF MEMORY> <HIGH ADR OF MEMORY> <DATA>

This routine expects three hexedecimal parameters, The first and second(16 bits) are interpreted as the bounds of a memory area to be initialized to a constant value, The third parameter (8 bit) is that value.

Program execute(G)

_G <ADDRESS>

The G command is used for transferring control from the monitor

to a user program.

Input port(I)

_I(PORT ADR)

This routines read a port specified by the hexedecimal parameter.

Read a hexedecimal record(L)

_L (ADDRES)

This routine reads a hexadecimal file from the assigned reader device and loads it into memory. One hexedecimal parameter is expected. This parameter is a base address to be added to the memory address of each data byte encountered. In this way, hexedecimal files may be loaded into memory in areas other than that for which they were assembled or complied. All records read are checksummed and compared against the checksum in the record. If a checksum error occurs, The routine takes an error exit.

Output data to a port(O)

_O(ADDRESS) XX-YY

This routine expects 8 bit address input(ADDRESS). When space is pressed It displays the data read from this port(XX). The substituted data(YY) outputted to the port assigned by address.

Enter programmer(P)

_P

This routine enters to programmer routines as soon as P is pressed.

Substitute memory(S)

_S (MEMORY ADDRESS)

This routine expects one parameter from the local console, followed by a space. The parameter is interpreted as a memory location and the routine display the contents of that location, followed by a dash (-). To modify memory, type in the new data followed by a space or a carriage return. If no modification of the location is required , type only a space or a carriage return. If a space was last typed. The next memory location is displayed and modification of it is allowed. If a carriage return was entered, The command is terminated.

Hexadecimal arithmetic(T)

_T(ADDRESS 1) (ADDRESS 2)

This routine expects two hexadecimal parameters. It computes the sum and difference of the two values and displays them on the local console device as follows.

(A1 + A2) (A1 - A2)

10.2 INPUT/OUTPUT ROUTINES

MPP sytem provides two type of I/O device interface for man_machine interface

These are interactive CRT terminal interface and Keyboard/display interface. An interactive CRT terminal is an

intelligent terminal and does not require any refreshing operation. It communicates an ASCII format with devices connected to it. In other words; it sends ASCII format of this key when a key is pressed on it and characters which is displayed on it should be ASCII formatted. Beside that Keyboard/Display circuit is an slave device and requires refreshing. Upper level programs does not know which interface is active . These programs are communicate with the Input/output routines when they are required Input a character from an input device or output a character to an output device. The data Input/output communication between Upper level programs and Input/output routines are in ASCII format.

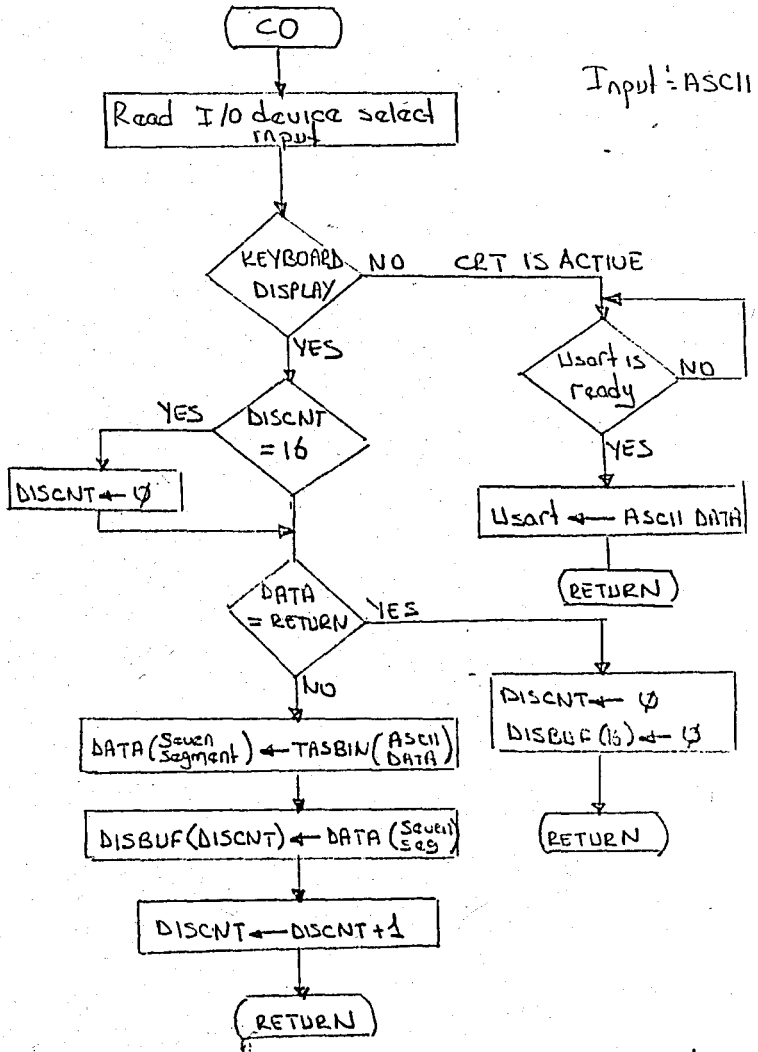
Basic Output Routine(CO):

This routine is called when a character is outputted to a list device. At the begining this routine checks that list device is CRT or display unit. If CRT terminal then it sends ASCII information to CRT terminal sending from upper programs. Otherwise it converts this ASCII data to seven_segment form and put into DISBUF location pointed by DISCNT offset counter. Dislay refresh procedure displays DISBUF with this new value. DISCNT points the first empty location in DISBUF. If the data is return then DISBUF is cleared and DISCNT is set to 0. The detailed flowchart is given IN Figure 10.1.

Basic Input routine(CI):

Input routine is same as output routine. It checks that which punch device is active.If CRT is connected data comes from CRT as ASCII formatted and this data is bypassed to upper programs. If

Keyboard is connected then data is written from KEYBUF and it is also ASCII formatted because keyboard scan convert pressed key code into ASCII format and put into KEYBUF. The detailed flowchart is given in Figure 10.2



Input = ASCII DATA

Fig 10.1. Detailed flowchar of Character output routine

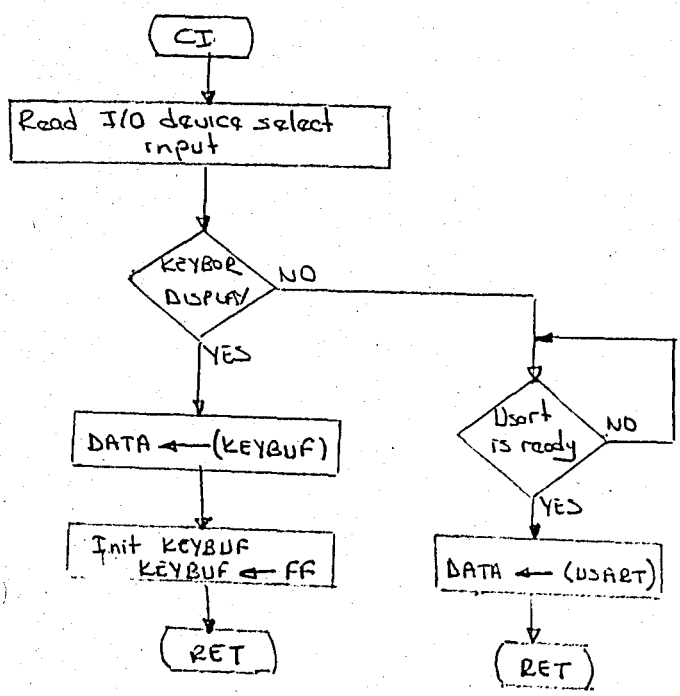


Fig 10.2 Detailed flowchart of Character input rout

10.3 COMPUTER INTERFACE

MPP system provides the communication facility with an external device such as a remote computer or a development system.

MPP system provides two type of standart format for data communication. These are INTEL HEX FORMAT and MOTOROLA S1 FORMAT.

10.3.1 INTEL HEX FORMAT

The INTEL 'HEX' format is used for program downloading/uploading from/to an external device. In this format, absolute object code is transmitted as variable lenght records made up of ASCII characters. MPP system accepts two record types; Data record and End of file record, and ignores the rest.
The format of the data record is shown below.

:AABBBBCCDDDD.....DDDDEE(CR)(LF)

Record Mark Field :

The ASCII code a colon (:) is used to signal the start of a record.

Record Lenght Field(AA):

Two hex digits indicating the number of data bytes in the record.

Starting Load Address Field(BBBB):

Four hex digits indicating the starting address of the block of data to be transferred.

Record Type Field(CC):

Two hex digits. Data records are signified by record type 00. End of file records are signified by record type 01.

Data Field:

2 to 64 hex digits (32 bytes) coded as ASCII 0-9,A-F.

Checksum(EE):

A two digit hex number representing the two's complement of the 8 bit sum (modulo 256) of the 8- bit bytes that result from the record lenght field to tha last data byte, inclusive.

End Of File Record Format:

Record mark = same as data record.

Record lenght = Always 00

Starting load address = Four hex digits (Zeros)

Record type = Always 01, identifies the end-of-file record.

Checksum = calculated the same way as the data record

10.3.2 MOTOROLA S FORMAT

The MOTOROLA format is a method of encoding data inASCII form. There are two type of record; Data record(S1) and End of file record.

The format of the Data Record is given below.

S1AABBBBCCCC.....CCCCDD(CR)(LF)

Header(S1):

An S1

Byte Count(AA):

Two hex digits indicating the number of bytes in the record from

the byte count to the last data byte.

Address(BBBB):

Four Hex digits indicating the start address of the block of data to be transferred.

Data:

2 to 64 Hex digits(32 data bytes) coded as ASCII 0-9 of A-F.

Checksum(DD):

A two digit Hex number representing the one's complement of the 8-bit sum(modula 256) of the 8-bit bytes that result from the byte count to the last data byte inclusive.

End Of File Record:

Header = S9

Byte count = Always 03

Address = Always 0000

Checksum = same as Data record

XI. PROGRAMMER SOFTWARE

MPP system provides two mode of operation. The first one is monitor mode and the other one is programmer mode. Monitor mode provides the modifications on memory and registers, communication with computers and program execution.

Programmer software is a program which allows the user to program, verify, compare, transfer eproms which plugged in the programmer card. Programmer mode is entered via P command provided by the monitor mode of operation. Programmer software is a table driven software. It uses same software to processes different type and different number of eproms.

The programmer mode begins with the dialogue by typing the UPP READY message. Then it displays the type message(TYPE=) to obtain the kind of eprom which will be processed. After getting type message it updates table pointers according to EPROM type. and display function message(FUNC=) to obtain which command is entered.

The commands provided by the programmer is given below. X is a variable and inform programmer which socket or sockets are processed. There are five sockets on the programmer card. These are called as master socket(M), slave1(S1),slave2(S2),slave3(S3) and slave4(S4).

AX : Program from master

BX : Verify with master

CX : Verify with buffer

EX : Check erasure

PX : Program from buffer

TX : Transfer to buffer

R : Return to monitor

11.1 PROGRAMMER COMMANDS

As explained above programmer ask to user the type of EPROM which will be operated by typing TYPE= message. The kind of EPROMS which programmer provides the operation are given below. X is a character entered by the user.

TYPE = X	X	Selected EPROM
	0	27256 Intelligent
	1	27128 Normal
	2	27128 Intelligent
	3	2764 Normal
	4	2764 Intelligent
	5	2732A Normal
	6	2732 Normal
	7	2716 Normal
	8	M2716 Normal
	9	M2532 Normal
	10	68764 Normal

After getting type command processor ask the user which function command is take place by typing function message(FUNC=). Programmer commands are in the form of single letters. Beside that all commands except Return to monitor(R) command requires a second character which follows the command to learn which sockets

are processed. This variable character is called as X.

Program from master(AX):

FUNC = AX <CR>

X	Function
0	M --> S1
1	M --> S1 & S2
2	M --> S1 & S2 & S3
3	M --> S1 & S2 & S3 & S4

This routine expects one character named as X. According to the value of X It copies the content of master eprom to the slaves. At end of copy, It verifies the content of master with slaves. If an unmatch is found it displays the socket number, location and data values relatad to unmatch.

Verify with master(BX)

FUNC = BX <CR>

X	Function
0	M <--> S1
1	M <--> S1 & S2
2	M <--> S1 & S2 & S3
3	M <--> S1 & S2 & S3 & S4

This routine verify the content of master EPROM with slaves. If an unmatch is found then It informs the user by giving the location and data values.

Verify with Buffer(CX)

FUNC = CX (CR)

) (Buffer start adr) (Buffer end adr) (Eprom start adr)

X	Function
0	B (--) M
1	B (--) S1
2	B (--) S2
3	B (--) S3
4	B (--) S4

This routine expects a character for socket information and expects three hexadecimal parameters for address information. The first and second paramaters of address information are the bounds of buffer to be verified with the eprom which start address is given as the third parameter. This routine displays the location and data if any verification error is found.

Check erasure(EX)

FUNC = EX (CR)

X	Function
0	M
1	S1
2	S2
3	S3
4	S4

This routine expects only socket information. The function of this routine is to check the eprom erasure. If EPROM is erase, It

display ERASED messages otherwise It displays NOT ERASED message. It compare EPROM content with FF.

Program from buffer(PX)

FUNC = PX (CR)

) (Buffer start adr) (Buffer end adr) (EPROM start adr)

X	Function
0	B --) M
1	B --) M & S1
2	B --) M & S1 & S2
3	B --) M & S1 & S2 & S3
4	B --) M & S1 & S2 & S3 & S4

This routine expects three hexadecimal parameter together with socket information. The first and second parameters are interpreted as the bounds of buffer to be programmed to the EPROM which it is start address is give by the third parameter. At the end of programming, EPROMS whichis programmed are verified with the buffer. If an error is found. It is displayed.

Transfer to buffer

FUNC = TX (CR)

) (EPROM start adr) (EPROM end adr) (buffer start adr)

X	Function
0	B (-- M
1	B (-- S1
2	B (-- S2

3 B <-- S3

4 B <-- S4

This routine expects three hexadecimal parameteres togather with socket information. The first and second parameters are interpreted as the bounds of EPROM which is transfered to the buffer. Buffer start address is given in the third parameter.

11.2 DATA STRUCTURES.

One of the major steps in S/W design procedure is to specify the format and content of each data base required by the S/W. Thus, the data bases required by the programmer S/W is discussed in this section.

Data bases of the programmer S/W classified in two groups;

i) Device addresses , sytem constants and tables

ii) System variables

Device adresses are given in section 7 (software/hardware interface)

System constants:

Programmer S/W requires system constants for Input/Output routines. These are latch control on the programmer card. Given below.

```
DISALL EQU 08H ; Disables all latches and buffers
ENAL   EQU 0AH ; Enable latch which provides Address low  byte
ENAH   EQU 0AH ; Enable latch which provides Address high byte
DDRIN  EQU 08H ; Puts Data buffer in input mode
DDROUT EQU 0CH ; Puts data buffer in output mode
ENDATI EQU 00H ; Enables data buffer in input mode
```

```
ENDATO EQU 04H ; Enables data buffer in output mode
ENCS   EQU 18H ; Enable latch which provides chip select
ENLED  EQU 28H ; Enable latch which provides LED information
```

System Tables:

As described in the previous sections. Programmer software is a table driven software. There are two tables. These are named as PDRTBL and PCSTBL. These tables provides the programming/reading information for different type of EPROMs.

PDRTBL support the Vpp control(different type of eproms require different Vpp signals), EVcc control, PGM and OE control.

The bit definitions of each byte is as follows

PGM	OE	VcC2	VcC1	VpC4	VpC3	VpC2	VpC1

EVcc control — Vpp control

This table has 6 entries per prom type. These are;

1. READ STANDBY
2. READ
3. PROGRAM STANDBY
4. PROGRAM PREPARATION
5. PROGRAM
6. PROGRAM VERIFY

These signals are outputted from the portB to programmer card.

PCSTBL contains the chip select information according to the different type of EPROMS. The bit definition of each byte is as follows.

-	-	-	S4CS	S3CS	S2CS	S1CS	MCS

This table has also 6 entries per prom type . These are same as PDRTBL.

Variables:

BUFBEG: 2 bytes long , Keeps buffer begining address entered by by the user

BUFEND: 2 bytes long , Keeps buffer end address entered by the user

ROMBEG: 2 bytes long, Keeps EPROM beging address entered by the user

ROMEND: 2 bytes long, Keeps EPROM end address entered by the user

PDRBUF: 2 bytes long, Keeps the base pointer which points to the related PDRTBL entry according to EPROM type .

PCSBUF: 2 bytes long, Keeps the base pointer which points to the related PCSTBL entry according to EPROM type.

PCMBUF: 6 bytes long. It keeps the information on chip select signals related to EPROM type and socket information.

PCMBUF (-- PCSTBL(PCSBUF) & CSMODE

TYPE: 1 bytes long. It contains the type of EPROM.

PRTYPE: 1 bytes long. A flag, which keeps information on the programming mode; intelligent or normal.

PRTYPE = FF Intelligent
PRTYPE = 00 NORMAL

PCTYPE: 1 bytes long; It keeps the maximum number of 1 msec pulses for intelligent programming.

PCTYPE = 15 for 2764 & 27128
PCTYPE = 25 for 27256

PCOUNT: 1 bytes long , programming pulse(1 msec) counter for intelligent programming.

EPEND: 1 bytes long, It keeps the selected EPROM high byte+1

CSMODE: 1 bytes long, It keeps the information on selected sockets

Programmer main program:

Monitor command recognizer transfers the control of the system to programmer main program when P command is entered. Programmer main program displays the programmer beginning message and ask the user the type of EPROM which is programmed. According the type information, related pointers(PDRBUF,PCSBUF), counters(PCTYPE), flags(PRTYPE) and buffers(EPEND)are initialized. At this stage There is no difference for the programmer between different type of EPROMs. Then asked that which function is processed. According to the input, command recognizer of programmer transfers the control related command routine.

11.3 DATA PROGRAM ROUTINES

Programmer provides two type of programming technique. The fisrt one is conventional 50 msec algorithm and the other one is intelligent algorithm.

In conventional algorithm; programming is a simple matter of applying a nominal 50 msec pulse per EPROM byte, resulting in programming time of approximately 1.75 msec for the entire 16K chip. This algorithm was easy to implement and became the

industry standard technique for programming. It has persisted through the introduction of higher EPROM densities up to the 128 KBIT 27128. This algorithm requires that programming time increase linearly with memory size. If the same algorithm were to continue to be used through the 256K bit density, the result would be an unmanageable 28 minute programming time. In stead Intelligent algorithm accelerates programming by up to a factor of six compared to conventional 50 msec algorithm. The key ingredient of the intelligent programming algorithm is its closed_loop margin checking techique. By programming in short, one msec pulses until an adequate margin is achieved, the algorithm assures reliability while rapidly programming the cell. Only those cells requiring longer programming times receive additional pulses. This unique approach to programming id the crux of the intelligent programming algorithm's ability to save time and cut costs.

The algorithm begins by setting Vcc to 6.0 V to obtain the correct level of margin testing. Next, Vpp is set to the correct voltage. Then , an iterative sequence of one msec pulsese, each followed by a check of the EPROM's output begins. Once the byte has been verified, a final overprogramming pulse is applied. The overprogramming pulse is equal to a multiple of the number of one msec pulses which that byte had received. This additional pulse helps insure that the cell has received additional programming margin for reliable operation. The detailed flowchart is given in figure 11.3

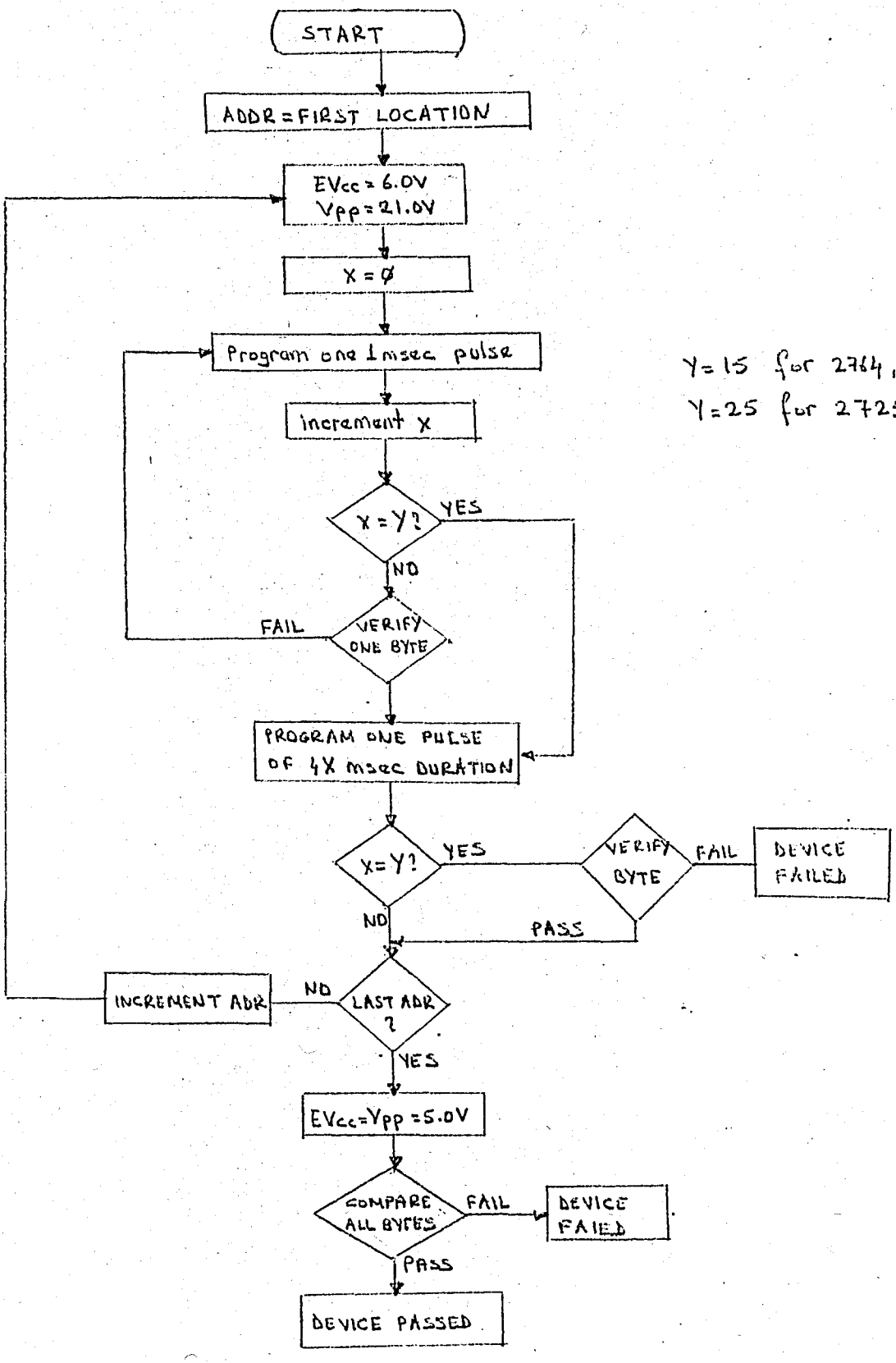


Fig. 11.3 Intelligant programming algorithm

XII. CONCLUSION

With the design of the MPP system Different type and different number of EPROMs can be easily programmad in a short time in respect to conventional programmers. MPP system also offers the possibility of executing and debugging programs before programming to EPROMs. So that debugging time is saved.

MPP system has been designed as a modular system. These modules are realized as different cards and connections between them has been provided by a backplane. So that it is possible to use each card of the sytem in different type of applications such as processor/memory and keyboard card can be used as intelligent motor controller or role driver.

APPENDIX A

COST ANALYSIS

1.	COMPONENTS	100,000.-
2.	FILM AND PRINTED CIRCUIT COSTS	100,000.-
3.	ENGINEERING COST	400,000.-
5.	MISCELLANEOUS	20,000.-
	TOTAL	620,000.- TL

APPENDIX B

PIN ASSIGNMENTS OF THE CARDS

PROCESSOR CARD

PIN NO	SIGNAL NAME	PIN NO	SIGNAL NAME
1A	+12V	1C	+12V
2A	1PA0	2C	1PA1
3A	1PA2	3C	1PA3
4A	1PA4	4C	1PA5
5A	1PA6	5C	1PA7
6A	1PC5	6C	1PC4
7A	1PC0	7C	1PC1
8A	1PC2	8C	1PC3
9A	1PB7	9C	1PB6
10A	1PB5	10C	1PB4
11A	1PB3	11C	1PB2
12A	1PB1	12C	1PB0
13A	1PB5	13C	2PB4
14A	2PB7	14C	2PB6
15A	-12V	15C	-12V
16A	12GRD	16C	12GRD
17A	2PB3	17C	2PB2
18A	2PB1	18C	2PB0
19A	2PA5	19C	2PA4
20A	2PA3	20C	2PA2
21A	COMRXD	21C	2PA1
22A	COMTXD	22C	2PA0
23A	2PC0	23C	2PA6
24A	2PC2	24C	2PA7
25A	DTR	25C	2PC5
26A	CRTRXD	26C	2PC4
27A	CRTTXD	27C	2PC1
28A	DSR	28C	2PC3
29A	+5V	29C	+5V
30A	+5V	30C	+5V
31A	5GRD	31C	5GRD
32A	5GRD	32C	5GRD

J1 connector.

PIN NO	SIGNAL NAME	PIN NO	SIGNAL NAME
1	+5V	50	+5V
2	+5V	49	+5V
3	DB7	48	DB6
4	DB5	47	DB4
5	DB3	46	DB2
6	DB1	45	DB0
7	-	44	-
8	-	43	-
9	-	42	-
10	-	41	-
11	AB12	40	AB11
12	AB13	39	AB10
13	AB14	38	AB9
14	AB15	37	AB8
15	AB3	36	AB4
16	AB5	35	AB2
17	AB6	34	AB1
18	AB7	33	AB0
19	-	32	-
20	-	31	-
21	PCLK	30	-
22	WR	29	RST
23	IO/M	28	RD
24	5GRD	27	5GRD
25	5GRD	26	5GRD

J2 connector

MEMORY CARD

There is an 50 pin flat-cable connector on the memory card for connection with processor card. The pin assignment of this connector is same as the memory card connector(J2) on the processor card.

PROGRAMMER CARD

PIN NO	SIGNAL NAME	PIN NO	SIGNAL NAME
1	5GRD	2	1PB7
3	"	4	1PB6
5	"	6	1PB5
7	"	8	1PB4
9	"	10	1PB3
11	"	12	1PB2
13	"	14	1PB1
15	"	16	1PB0
17	"	18	1PC3
19	"	20	1PC2
21	"	22	1PC1
23	"	24	1PC0
25	"	26	1PC4
27	"	28	1PC5
29	"	30	1PC6
31	"	32	1PC7
33	"	34	1PA7
35	"	36	1PA6
37	"	38	1PA5
39	"	40	1PA4
41	"	42	1PA3
43	"	44	1PA2
45	"	46	1PA1
47	"	48	1PA0
49	5GRD	50	-

J1 connector.

PIN NO	SIGNAL NAME
1	+5V
2	+5V
3	5GRD
4	5GRD
5	VpC1
6	VpC2
7	VcC2
8	VpC3
9	VcC1
10	VpC4

J2 connector

PIN NO	SIGNAL NAME
1	Vpp
2	EVcc
3	EA
4	VDD
6	Prog

J4 connector

PIN NO	SIGNAL NAME
1	+5V
2	+5V
3	5GRD
4	5GRD

J3 connector

DISPLAY/KEYBOARD CARD

PIN NO	SIGNAL NAME	PIN NO	SIGNAL NAME
1	5GRD	2	2PB7
3	"	4	2PB6
5	"	6	2PB5
7	"	8	2PB4
9	"	10	2PB3
11	"	12	2PB2
13	"	14	2PB1
15	"	16	2PB0
17	"	18	2PC3
19	"	20	2PC2
21	"	22	2PC1
23	"	24	2PC0
25	"	26	2PC4
27	"	28	2PC5
29	"	30	2PC6
31	"	32	2PC7
33	"	34	2PA7
35	"	36	2PA6
37	"	38	2PA5
39	"	40	2PA4
41	"	42	2PA3
43	"	44	2PA2
45	"	46	2PA1
47	"	48	2PA0
49	5GRD	50	-

J1 connector.

1	+5V
2	+5V
3	5GRD
4	5GRD

J2 connector.

PROGRAMMABLE POWER SUPPLY

J2 and J4 connector pin assignments are same as programmer card J2 and J4 connector pin assignments.

1	+12V
2	12GRD
3	+30V
4	30GRD

J3 connector.

BACKPLANE

There are five connectors on the backplane. One of them is an 64 pin euro-card connector(J1) for processor connection. Two ones are 50 pin flatcable connectors for programmer and keyboard/display connection named as J2 and J3. The other ones are used for power, CRT and MDS connections.

J1 = Processor card connection

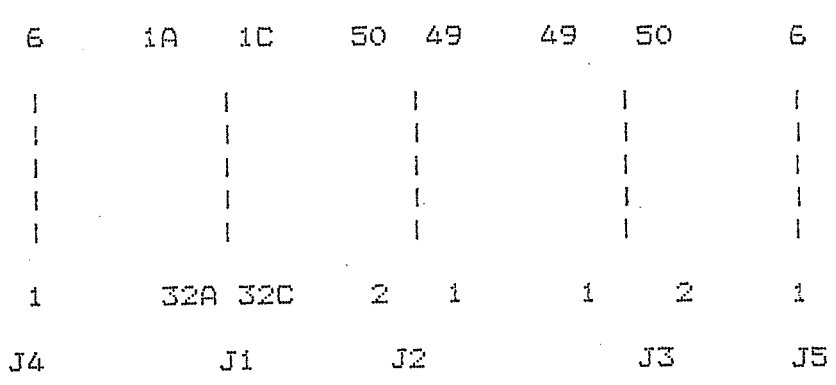
J2 = Programmer card connection

J3 = Display/Keyboard card connection

J4 = CRT & MDS connection

J5 = Power connection.

The front view of the backplane is given below.



J1 connector pin assignment is same as processor card J1 connector.

J2 connector pin assignment is same as Programmer card J1 connector.

J3 connector pin assignment is same as Display/Keyboard card J1 connector.

1	CRTRXD
2	CRTTXD
3	COMRXD
4	COMTXD
5	5GRD
6	-

J4 connector

1	+5V
2	5GRD
3	+12V
4	-12V
5	12GRD
6	-

J5 connector

APPENDIX C

PERSONALITY MODULE CONNECTIONS

The pin assignment is given below.

PIN NO	SIGNAL NAME	PIN NO	SIGNAL NAME
1	-	14	-
2	OE	13	FS1
3	A13	12	FS4
4	A14	11	FS2
5	A11	10	FS3
6	A12	9	EVcc
7	PGM	8	Vpp

The internal connections of the personality module according to EPROMs type is given below.

27256 : 2-13, 5-12, 3-11, 4-10

27128 : 2-13, 5-12, 3-11, 7-10

2764 : 2-13, 5-12, 7-10

2732 & 2732A : 5-12, 9-11, 8-13

2716 & MCM2716 : 2-13, 9-11 , 8-12

MCM2532 : 2-13, 8-12

MCM68764 : 6-12, 8-13

APPENDIX D

MONITOR/PROGRAMMER COMMAND SUMMARY

MONITOR COMMANDS

COMMAND	PARAMETERES	FUNCTION
A	(LOW ADR OF MEM.) (HIGH ADR OF MEM) (LOW ADR OF DEST)	MOVE
B	(LOW ADR OF MEM) (HIGH ADR OF MEM)	WRITE
D	(LOW ADR OF MEM) (HIGH ADR OF MEM)	DISPLAY
E	(REGISTER NAME)	REG. EXAM
F	(LOW ADR OF MEM) (HIGH ADR OF MEM) (DATA)	FILL
G	(PROGRAM STARTING ADR) (PROGRAM END ADR)	PROG EXEC.
I	(PORT ADR)	INPUT PORT
O	(PORT ADR) (DATA)	OUTPUT PORT
P	----	ENTER PROG.
S	(MEMORY ADR) (DATA)	SUBSTITUTE
T	(ADDRESS 1) (ADDRESS 2)	HEX ADD.

PROGRAMMER COMMANDS

AX	-----	PRO. FRO. MAS.
BX	-----	VER. WIT. MAS.
CX	(BUF STR ADR) (BUF END ADR) (EP STR ADR)	VER.WIT.BUF.
EX	-----	CHK ERASURE

PX	(BUF STR ADR)	(BUF END ADR)	(EP STR ADR)	PRO.FRO.BUF
TX	(EP STR ADR)	(EP END ADR)	(BUF STR ADR)	TRA FRO EP
R	-----			RET TO MON

APPENDIX E

8085 INSTRUCTION SET

The following is a summary of the instruction set:

8080/85 CPU INSTRUCTIONS IN OPERATION CODE SEQUENCE

OP CODE	MNEMONIC	OP CODE	MNEMONIC	OP CODE	MNEMONIC
00	NOP	2B	DCX H	56	MOV D,M
01	LXI B,D16	2C	INR L	57	MOV D,A
02	STAX B	2D	DCR L	58	MOV E,B
03	INX B	2E	MVI L,D8	59	MOV E,C
04	INR B	2F	CMA	5A	MOV E,D
05	DCR B	30	SIM	5B	MOV E,E
06	MVI B,D8	31	LXI SPD16	5C	MOV E,H
07	RLC	32	STA Adr	5D	MOV E,L
08	—	33	INX SP	5E	MOV E,M
09	DAD B	34	INR M	5F	MOV E,A
0A	LDAX B	35	DCR M	60	MOV H,B
0B	DCX B	36	MVI M,D8	61	MOV H,C
0C	INR C	37	STC	62	MOV H,D
0D	DCR C	38	—	63	MOV H,E
0E	MVI C,D8	39	DAD SP	64	MOV H,H
0F	RRC	3A	LDA Adr	65	MOV H,L
10	—	3B	DCX SP	66	MOV H,M
11	LXI D,D16	3C	INR A	67	MOV H,A
12	STAX D	3D	DCR A	68	MOV L,B
13	INX D	3E	MVI A,D8	69	MOV L,C
14	INR D	3F	CMC	6A	MOV L,D
15	DCR D	40	MOV B,B	6B	MOV L,E
16	MVI D,D8	41	MOV B,C	6C	MOV L,H
17	RAL	42	MOV B,D	6D	MOV L,L
18	—	43	MOV B,E	6E	MOV L,M
19	DAD D	44	MOV B,H	6F	MOV L,A
1A	LDAX D	45	MOV B,L	70	MOV M,B
1B	DCX D	46	MOV B,M	71	MOV M,C
1C	INR E	47	MOV B,A	72	MOV M,D
1D	DRC E	48	MOV C,B	73	MOV M,E
1E	MVI E,D8	49	MOV C,C	74	MOV M,H
1F	RAR	4A	MOV C,D	75	MOV M,L
20	RIM	4B	MOV C,E	76	HLT
21	LXI H,D16	4C	MOV C,H	77	MOV M,A
22	SHLD Adr	4D	MOV C,L	78	MOV A,B
23	INX H	4E	MOV C,M	79	MOV A,C
24	INR H	4F	MOV C,A	7A	MOV A,D
25	DCR H	50	MOV D,B	7B	MOV A,E
26	MVI H,D8	51	MOV D,C	7C	MOV A,H
27	DAA	52	MOV D,D	7D	MOV A,L
28	—	53	MOV D,E	7E	MOV A,M
29	DAD H	54	MOV D,H	7F	MOV A,A
2A	LHLD Adr	55	MOV D,L	80	ADD B

OP CODE	MNEMONIC	OP CODE	MNEMONIC	OP CODE	MNEMONIC
81	ADD C	AC	XRA H	D7	RST 2
82	ADD D	AD	XRA L	D8	RC
83	ADD E	AE	XRA M	D9	—
84	ADD H	AF	XRA A	DA	JC Adr
85	ADD L	B0	ORA B	DB	IN D8
86	ADD M	B1	ORA C	DC	CC Adr
87	ADD A	B2	ORA D	DD	—
88	ADC B	B3	ORA E	DE	SBI D8
89	ADC C	B4	ORA H	DF	RST 3
8A	ADC D	B5	ORA L	E0	RPO
8B	ADC E	B6	ORA M	E1	POP H
8C	ADC H	B7	ORA A	E2	JPO Adr
8D	ADC L	B8	CMP B	E3	XTHL
8E	ADC M	B9	CMP C	E4	CPO Adr
8F	ADC A	BA	CMP D	E5	PUSH H
8G	SUB B	BB	CMP E	E6	ANI D8
91	SUB C	BC	CMP H	E7	RST 4
92	SUB D	BD	CMP L	E8	RPE
93	SUB E	BE	CMP M	E9	PCHL
94	SUB H	BF	CMP A	EA	JPE Adr
95	SUB L	C0	RNZ	EB	XCHG
96	SUB M	C1	POP B	EC	CPE Adr
97	SUB A	C2	JNZ Adr	ED	—
98	SBB B	C3	JMP Adr	EE	XRI D8
99	SBB C	C4	CNZ Adr	EF	RST 5
9A	SBB D	C5	PUSH B	F0	RP
9B	SBB E	C6	ADI D8	F1	POP PSW
9C	SBB H	C7	RST 0	F2	JP Adr
9D	SBB L	C8	RZ	F3	DI
9E	SBB M	C9	RET Adr	F4	CP Adr
9F	SBB A	CA	JZ	F5	PUSH PSW
A0	ANA B	CB	—	F6	ORI D8
A1	ANA C	CC	CZ Adr	F7	RST 6
A2	ANA D	CD	CALL Adr	F8	RM
A3	ANA E	CE	ACI D8	F9	SPHL
A4	ANA H	CF	RST 1	FA	JM Adr
A5	ANA L	D0	RNC	FB	EI
A6	ANA M	D1	POP D	FC	CM Adr
A7	ANA A	D2	JNC Adr	FD	—
A8	XRA B	D3	OUT D8	FE	CPI D8
A9	XRA C	D4	CNC Adr	FF	RST 7
AA	XRA D	D5	PUSH D		
AB	XRA E	D6	SUI D8		

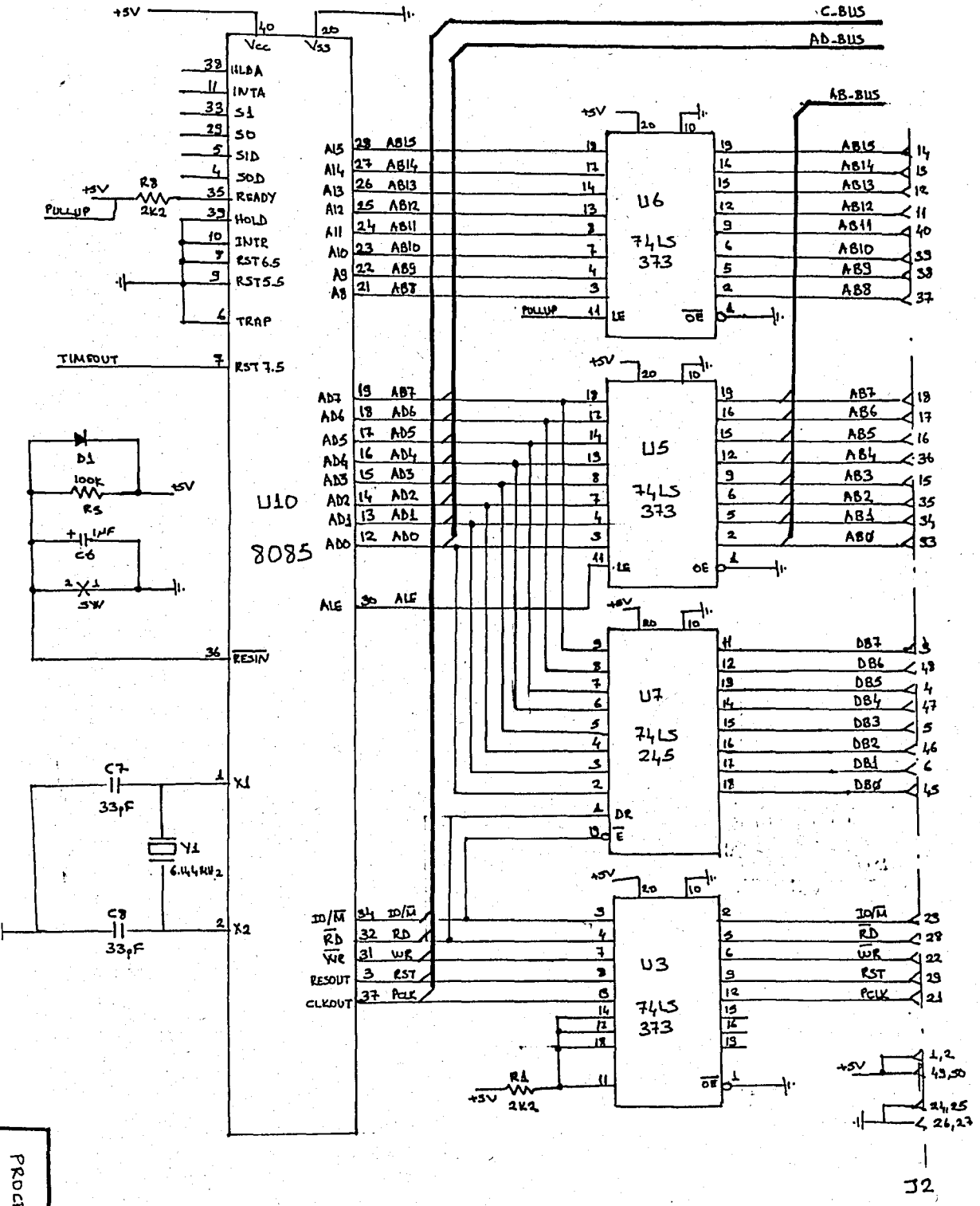
D8 = constant, or logical/arithmetic expression that evaluates to an 8 bit data quantity.

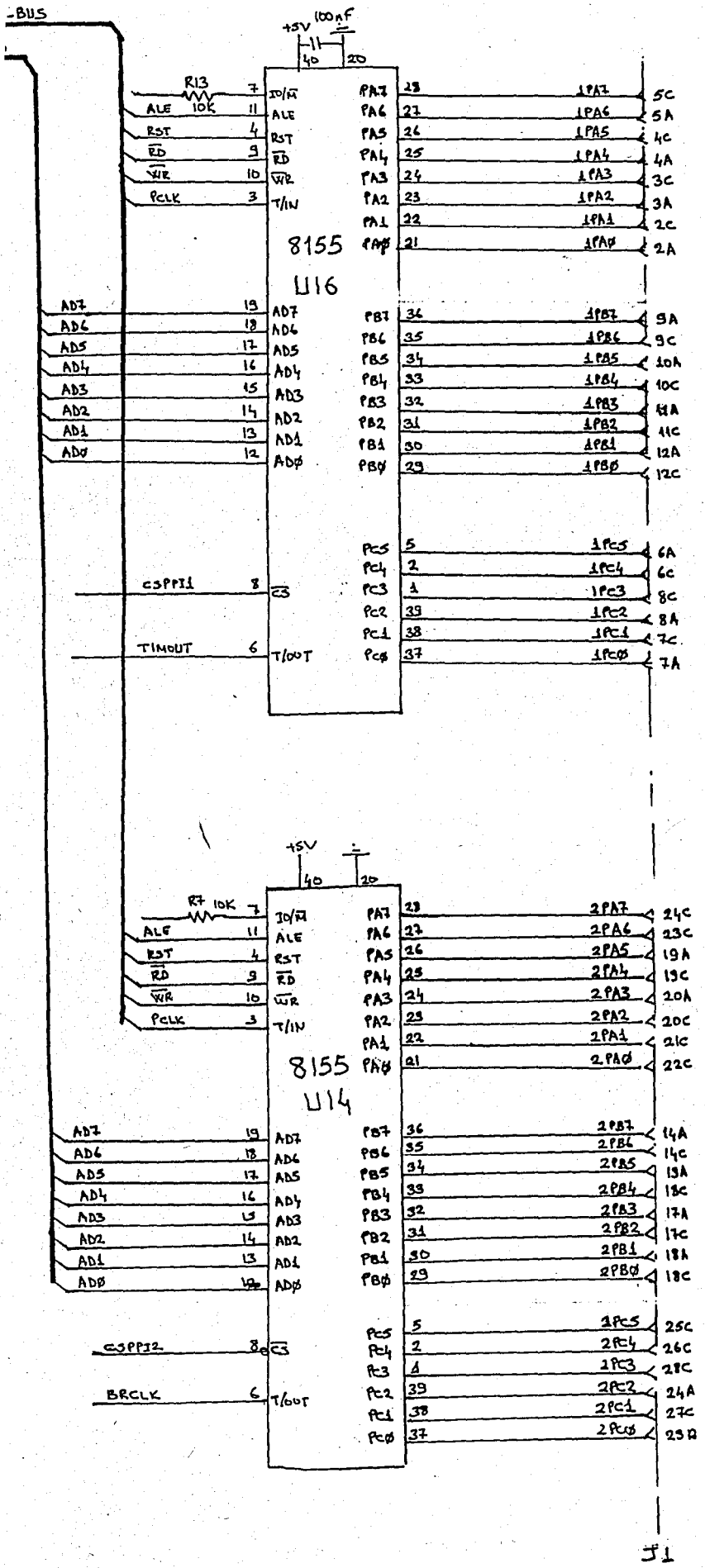
D16 = constant, or logical/arithmetic expression that evaluat to a 16 bit data quantity

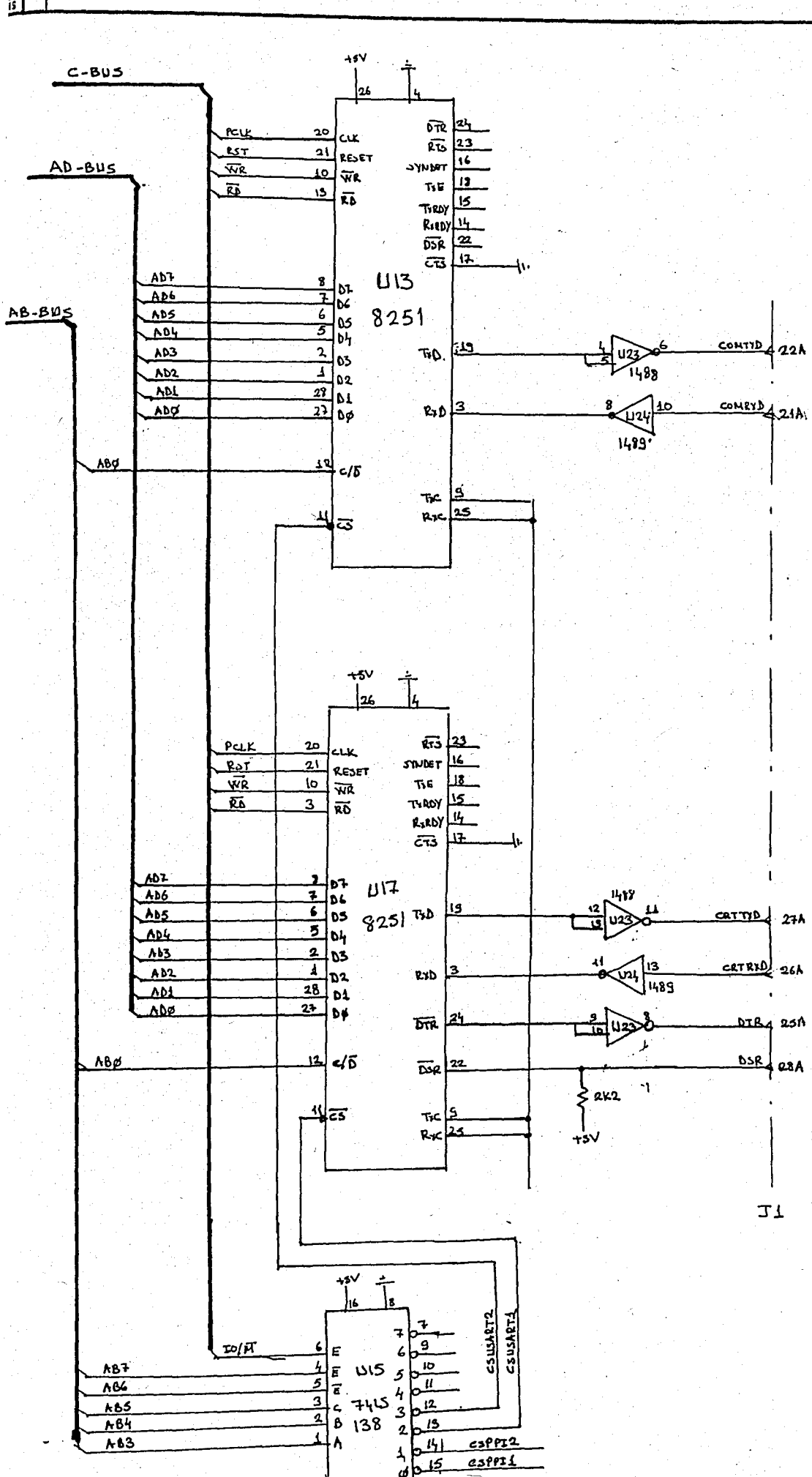
Adr = 16-bit address

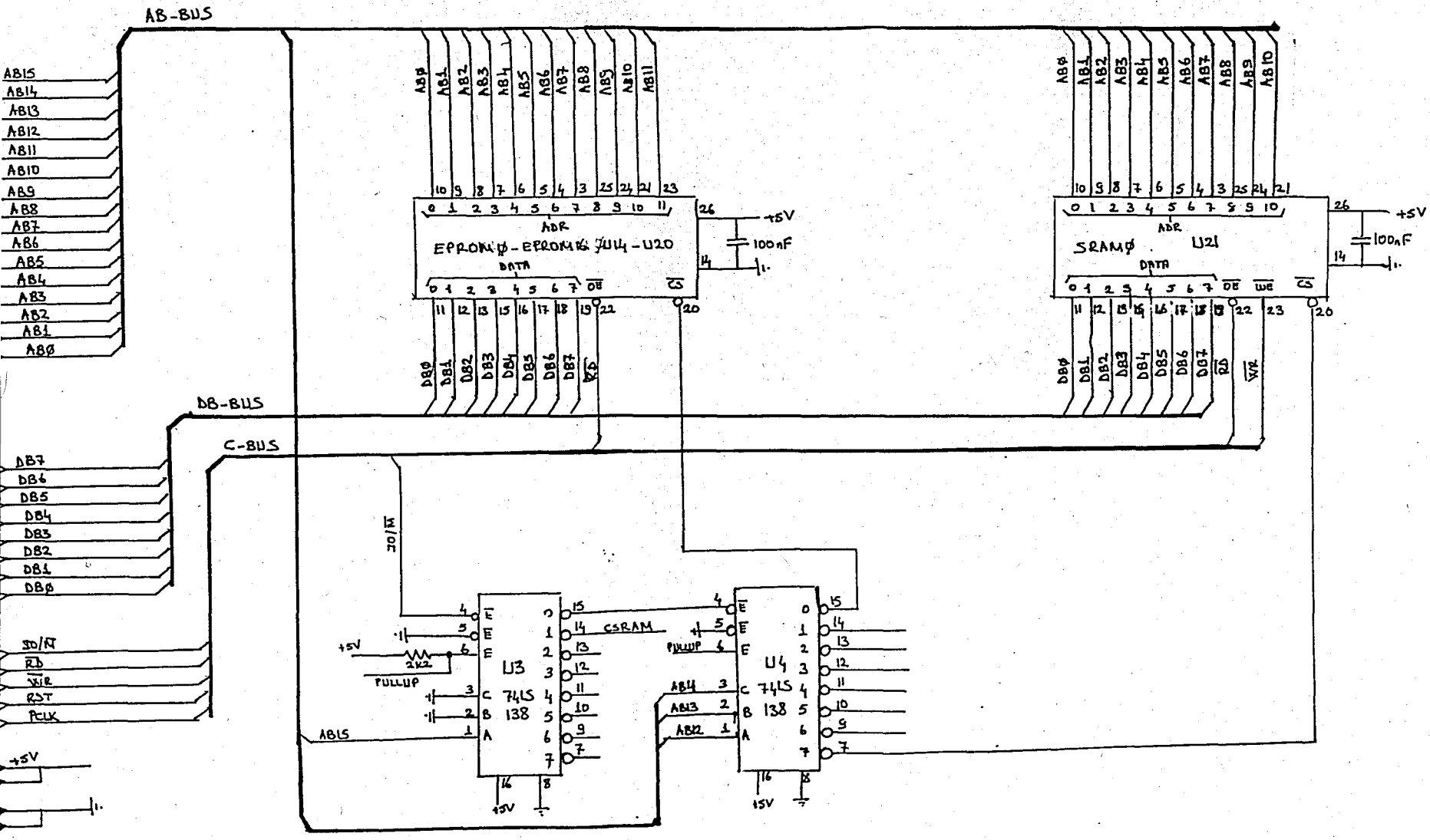
APPENDIX F

CIRCUIT SHEMATICS

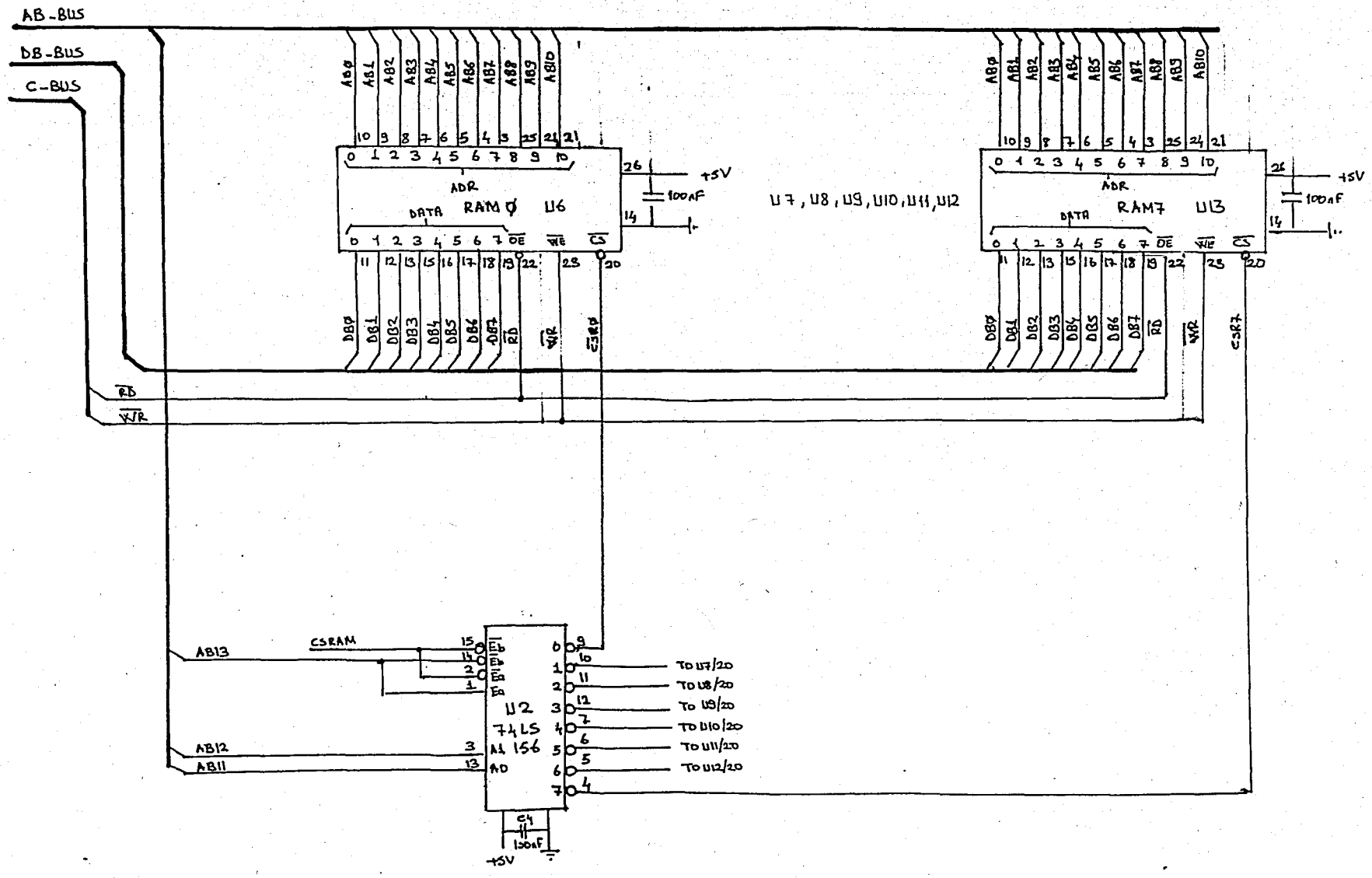

C-BUS
AD-BUS
AB-BUS
U10
8085
U6
74LS
373
U5
74LS
373
U7
74LS
245
U3
74LS
373
+5V
PULLUP
R3
2K2
R4
2K2
R5
100K
D1
C6
1MF
SW
C7
33pF
C8
33pF
Y1
6.144MHz
TIMEDOUT
RESIN
RESOUT
CLKOUT
ALE
HLDA
INTA
S1
S0
SID
SOD
READY
HOLD
INTR
RST6.5
RST5.5
TRAP
RST7.5
X1
X2
Vcc
Vss
J2
PROCESSOR CARD


-BUS
+5V
100nF
40
20
R13
10K
IO/M
ALE
RST
RD
WR
T/IN
PA7
PA6
PA5
PA4
PA3
PA2
PA1
PA0
8155
U16
1PA7
1PA6
1PA5
1PA4
1PA3
1PA2
1PA1
1PA0
5C
5A
4C
4A
3C
3A
2C
2A
AD7
AD6
AD5
AD4
AD3
AD2
AD1
AD0
PB7
PB6
PB5
PB4
PB3
PB2
PB1
PB0
1PB7
1PB6
1PB5
1PB4
1PB3
1PB2
1PB1
1PB0
9A
9C
10A
10C
11A
11C
12A
12C
CSPPI1
TIMOUT
CS
T/OUT
PC5
PC4
PC3
PC2
PC1
PC0
1PC5
1PC4
1PC3
1PC2
1PC1
1PC0
6A
6C
8C
8A
7C
7A
+5V
R7 10K
2PA7
2PA6
2PA5
2PA4
2PA3
2PA2
2PA1
2PA0
24C
23C
19A
19C
20A
20C
21C
22C
U14
2PB7
2PB6
2PB5
2PB4
2PB3
2PB2
2PB1
2PB0
14A
14C
15A
15C
17A
17C
18A
18C
CSPPI2
BRCLK
2PC5
2PC4
2PC3
2PC2
2PC1
2PC0
25C
26C
28C
24A
27C
23A
J1

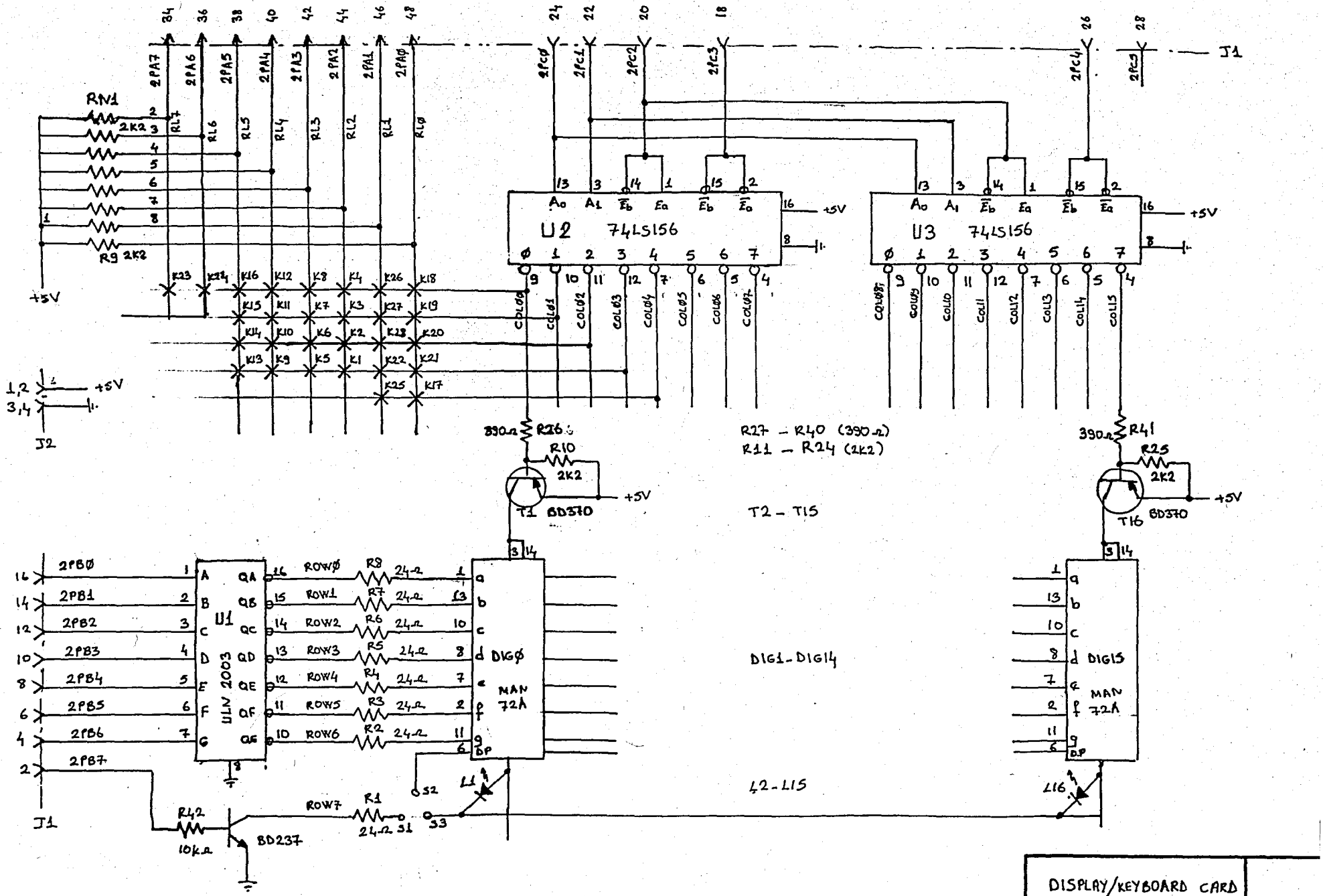

C-BUS
AD-BUS
AB-BUS
PROCESSOR CARD
Netas
SHEET
SHEETS 3
25
+5V
U13
8251
U17
8251
U15
74LS 138
PCLK
RST
WR
RD
CLK
RESET
DTR
RTS
SYNDET
TxE
TxRDY
RxRDY
DSR
CTS
AD7
AD6
AD5
AD4
AD3
AD2
AD1
AD0
D7
D6
D5
D4
D3
D2
D1
D0
TxD
RxD
C/D
CS
TxC
RxC
U23
U24
1488
1489
CONTYD
CONRYD
CRTTYD
CRTRYD
DTR
DSR
22A
21A
27A
26A
25A
23A
2K2
J1
IO/M
E
E
C
B
A
CSUSART2
CSUSART1
CSPPI2
CSPPI1

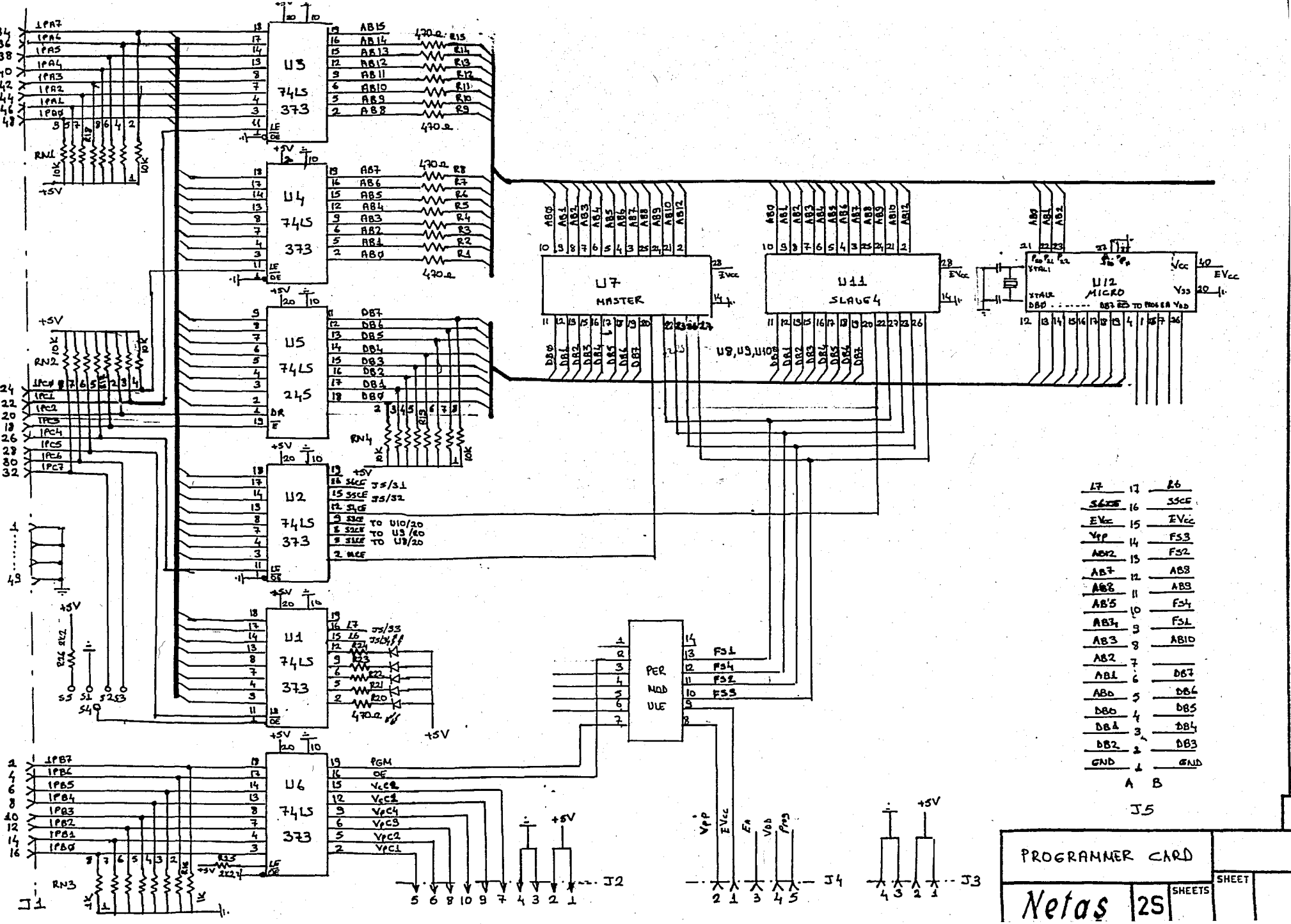


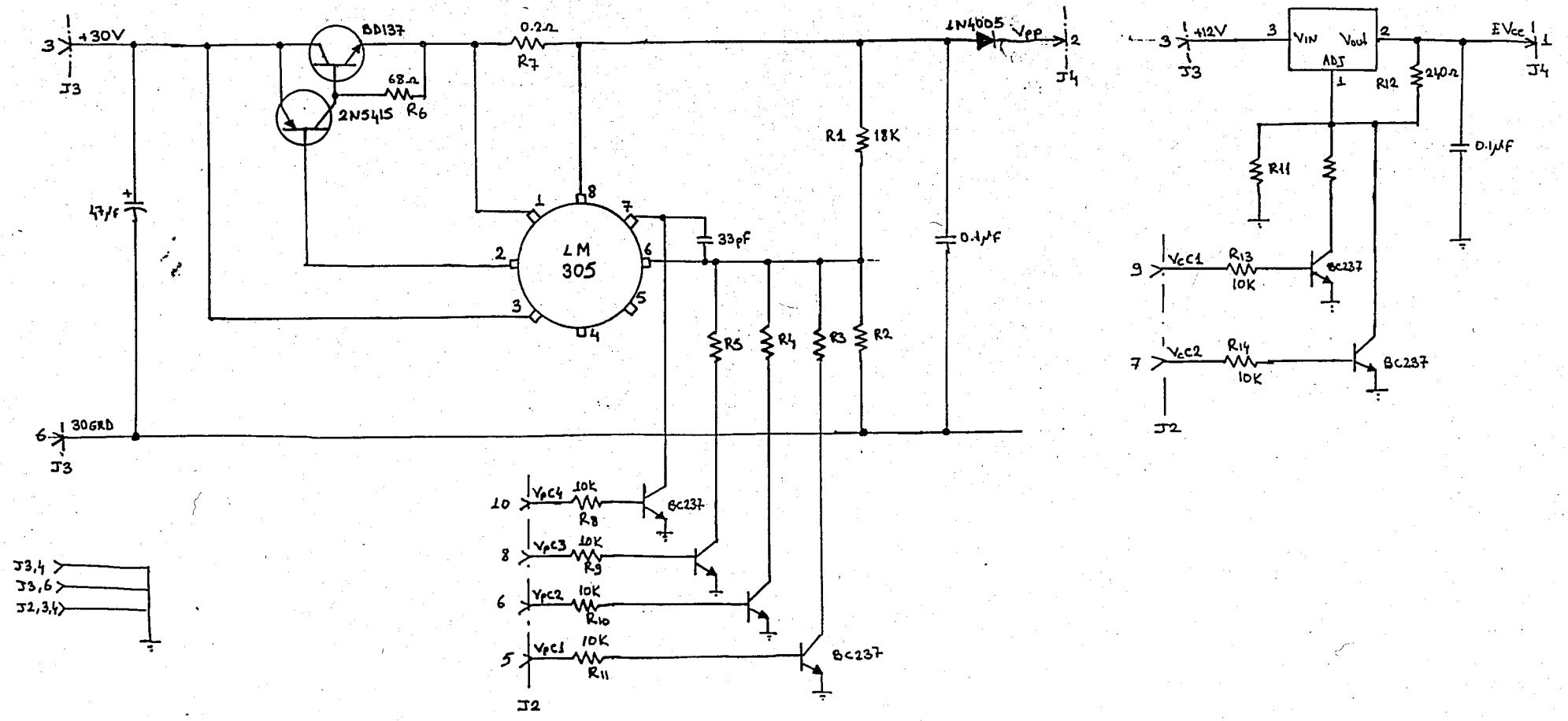
MEMORY CARD			
Netas	2S	SHEETS 1	SHEET



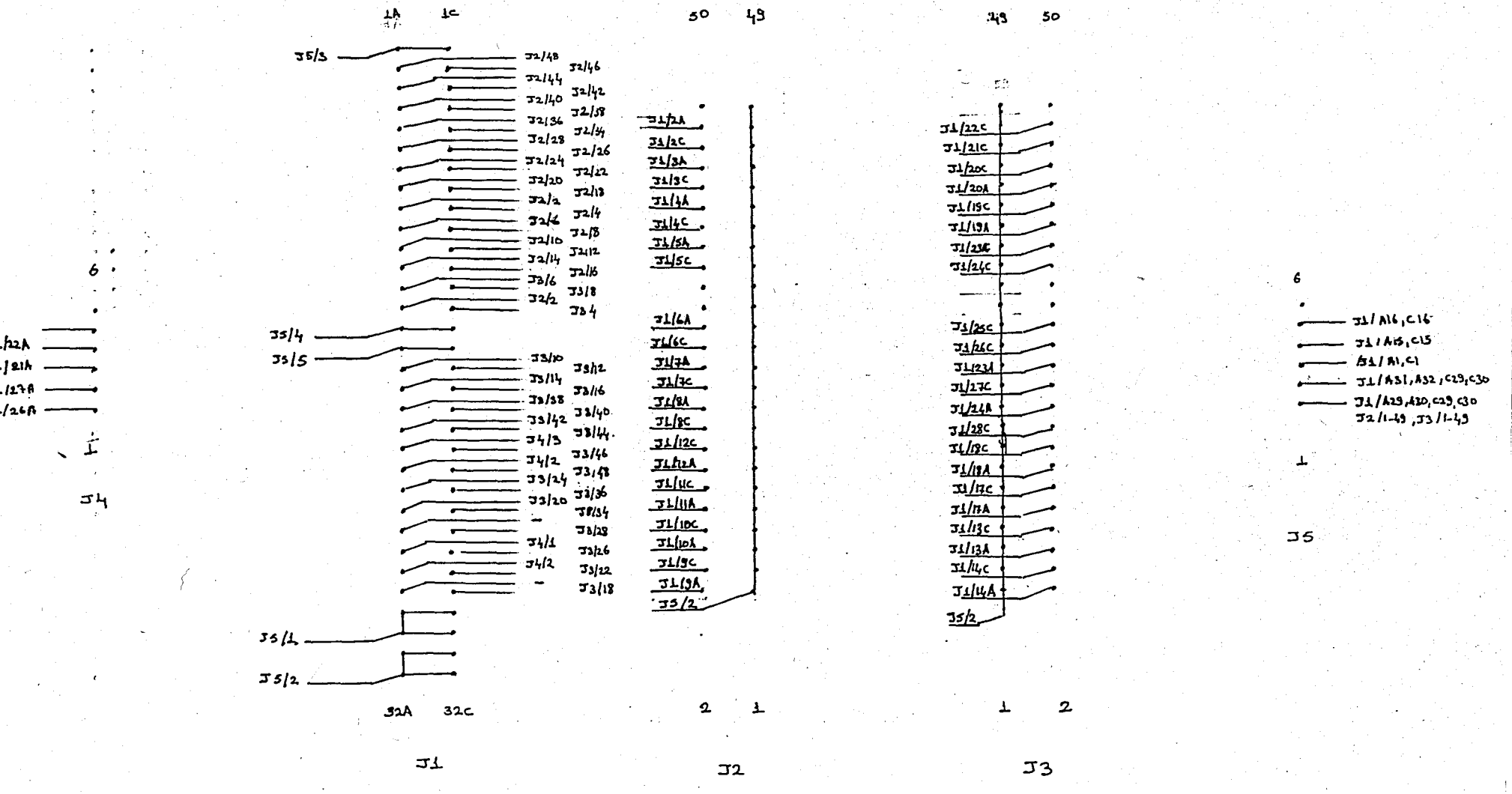
MEMORY CARD			SHEET
Netas	2S	SHEETS 2	


RN1
2K2
R9 2K2
+5V
J2
U2 74LS156
U3 74LS156
R26 390Ω
R41 390Ω
R10 2K2
R25 2K2
T1 BD370
T16 BD370
R27 – R40 (390Ω)
R11 – R24 (2K2)
T2 – T15
U1 ULN 2003
R8 24Ω
R7 24Ω
R6 24Ω
R5 24Ω
R4 24Ω
R3 24Ω
R2 24Ω
R1 24Ω
R42 10kΩ
BD237
DIG0
DIG15
MAN 72A
DIG1-DIG14
L2-L15
L1
L16
J1
DISPLAY/KEYBOARD CARD
SHEET
SHEETS


U3 74LS 373
U4 74LS 373
U5 74LS 245
U2 74LS 373
U1 74LS 373
U6 74LS 373
U7 MASTER
U11 SLAVE 4
U12 MICRO
PER MOD ULE
RN1
RN2
RN3
RN4
470Ω
10K
2K
J1
J2
J3
J4
J5
PROGRAMMER CARD
Netas
2S
SHEETS
SHEET



PROGRAMMABLE POWER SUPPLY CARD		
Notes	2S SHEETS	SHEET



BACKPLANE			
Notes	25	SHEETS	SHEET

APPENDIX G

SOURCE LISTINGS

```
ISM80 MON85A.001 NOOBJECT PRINT(:LP:)

SIS-II 8080/8085 MACRO ASSEMBLER. V2.0          MPP85A   PAGE    1

 LOC  OBJ          SEQ          SOURCE STATEMENT

                     1 $PAGEWIDTH(80)
                     2 :*****************************************************
                     3 :**
                     4 :**                  MPP    MONITOR  V2
                     5 :**
                     6 :** PROG.NO:200-09-01                DATE  : 26.6.1984
                     7 :** FILE   :MON85A.001               MODULE: MPP85A
                     8 :** AUTHOR :E.DURMUS
                     9 :**
                    10 :*****************************************************
                    11 :
                    12          NAME    MPP85A
                    13 :
                    14          PUBLIC  MONINI,GETCM,TEMP,TEMP1
                    15          PUBLIC  PSAVE,SSAVE,LSAVE,BRKFL
                    16          PUBLIC  MSTAK,RTAB,SCRTH,ERROR,EXIT
                    17          PUBLIC  USART1,PORTA2,PORTB2,PORTC2
                    18          PUBLIC  PORTA1,PORTB1,PORTC1,PPI1
                    19 :
                    20          EXTRN   ECHO,ADRD,CROUT,CO,GETCH
                    21          EXTRN   TBLSER,MESOUT,INTTIM,CRWAIT
                    22          EXTRN   ACMD,BCMD,DCMD,MPPMON
                    23          EXTRN   ECMD,FCMD,GCMD,ICMD,LCMD
                    24          EXTRN   OCMD,SCMD,DCLCNT,DISCNT
                    25          EXTRN   TCMD,KEYBUF,DISCLR,KEYFLG
                    26          EXTRN   STNDBY,SNDLED
                    27 :
                    28 :********   SYSTEM DEPENDENT DEFINITIONS   *********
                    29 :
0010                30 USART1   EQU     10H
0018                31 USART2   EQU     18H
0000                32 PPI1     EQU     00H     :MPP
0008                33 PPI2     EQU     08H     :KEY/DIS
                    34 :
0004                35 TIML1    EQU     PPI1+4
0005                36 TIMH1    EQU     PPI1+5
000C                37 TIML2    EQU     PPI2+4
000D                38 TIMH2    EQU     PPI2+5
                    39 :
0001                40 PORTA1   EQU     PPI1+1
0002                41 PORTB1   EQU     PPI1+2
0003                42 PORTC1   EQU     PPI1+3
0009                43 PORTA2   EQU     PPI2+1
000A                44 PORTB2   EQU     PPI2+2
000B                45 PORTC2   EQU     PPI2+3
                    46 :
                    47 :************    RAM AREA DEFINITIONS  **************
                    48 :
                    49
                    50          DSEG
                    51 :
0030                52 STKBOT:  DS      48      :STACK AREA
```

```
LOC  OBJ          SEQ          SOURCE STATEMENT

0030          D    53 MSTAK   EQU     $        :TOP OF MONITOR STACK
0030          D    54 SCRTH   EQU     $        :RAM PAGE ADR.
0001               55 ESAVE:  DS      1        :E REGISTER SAVE LOCATION
0001               56 DSAVE:  DS      1        :D REGISTER SAVE LOCATION
0001               57 CSAVE:  DS      1        :C REGISTER SAVE LOCATION
0001               58 BSAVE:  DS      1        :B REGISTER SAVE LOCATION
0001               59 FSAVE:  DS      1        :FLAGS SAVE LOCATION
0001               60 ASAVE:  DS      1        :A REGISTER SAVE LOCATION
0001               61 LSAVE:  DS      1        :L REGISTER SAVE LOCATION
0001               62 HSAVE:  DS      1        :H REGISTER SAVE LOCATION
0002               63 PSAVE:  DS      2        :PGM COUNTER SAVE LOCATION
003A 0200          64 SSAVE:  DW      2        :USER SP SAVE LOCATION
0001               65 TEMP:   DS      1        :TEMPORARY MONITOR CELL
0001               66 TEMP1:  DS      1
0001               67 BRKFL:  DS      1
0003               68 USERBR: DS      3
                   69 :
                   70 :*********  BEGINNING OF MONITOR  ********************
                   71 :
                   72         CSEG
                   73 :
                   74 : MONITOR COLD START
                   75 :
0000 3E8F          76 MONINI: MVI     A.08FH
0002 D311          77         OUT     LOW USART1+1
0004 C33F00    C   78         JMP     INUST
0007 00            79         NOP
                   80 :
                   81 : BREAKPOINT RETURN SERVICING
                   82 :
0008 223600    D   83 GO:     SHLD    LSAVE
000B E1            84         POP     H
000C F5            85         PUSH    PSW
000D 2B            86         DCX     H
000E 223800    D   87         SHLD    PSAVE
0011 3A3E00    D   88         LDA     BRKFL
0014 B7            89         ORA     A
0015 CA1C00    C   90         JZ      NOBRK
0018 3A3D00    D   91         LDA     TEMP1
001B 77            92         MOV     M.A
001C 210200        93 NOBRK:  LXI     H.2
001F 39            94         DAD     SP
0020 223A00    D   95         SHLD    SSAVE
0023 F1            96         POP     PSW
0024 313600    D   97         LXI     SP.ASAVE+1
                   98 :*
                   99 :*  ADROUT SAVES THE USER REGISTERS AND
                  100 :*  OUTPUTS TO THE CONSOLE THE USER P
                  101 :*  COUNTER AFTER A RST 4 INSTRUCTION
                  102 :*
0027 F5           103 ADROUT: PUSH   PSW      :SAVE A AND FLAGS
0028 C5           104         PUSH   B
0029 D5           105         PUSH   D
002A 0E23         106         MVI    C.'#'
002C CD0000    E  107         CALL   ECHO
```

```
 LOC  OBJ          SEQ          SOURCE STATEMENT

 002F 2A3800   D    108          LHLD    PSAVE
 0032 CD0000   E    109          CALL    ADRD
 0035 C3D700   C    110          JMP     EXIT
                    111 ;
                    112 ;*****************************************************
                    113 ;
 0038 C33F00   D    114          JMP     USERBR  ;RST 7
 003B 00            115          NOP
                    116 ;
 003C C30000   E    117          JMP     INTTIM  ;RST 7.5
                    118 ;
                    119 ;*****************************************************
                    120 ;*
                    121 ;* INUST OUTPUTS TO THE USARTS THE COMMAND WORD
                    122 ;*
 003F 3E8F          123 INUST:   MVI     A,08FH
 0041 D319          124          OUT     LOW USART2+1
 0043 3ECF          125          MVI     A,0CFH
 0045 D311          126          OUT     LOW USART1+1
 0047 D311          127          OUT     LOW USART1+1
 0049 D319          128          OUT     LOW USART2+1
 004B D319          129          OUT     LOW USART2+1
 004D 3E25          130          MVI     A,025H
 004F D311          131          OUT     LOW USART1+1
 0051 D319          132          OUT     LOW USART2+1
                    133 ;
                    134 ;INITIALIZE PROGRAMMER I/O(8155) AND INT. TIMER
                    135 ;
 0053 3E00          136          MVI     A,00
 0055 D304          137          OUT     LOW TIML1
 0057 3E4C          138          MVI     A,4CH
 0059 D305          139          OUT     LOW TIMH1
 005B 3E1B          140          MVI     A,1BH
) 005D 30           141          SIM
(   0)
 005E 3ECF          142          MVI     A,0CFH
 0060 D300          143          OUT     LOW PPI1
                    144 ;
                    145 ;INITIALIZE KEY/DIS I/O(8155) AND BAUD RATE TIMER
                    146 ;
 0062 3E14          147          MVI     A,14H
 0064 D30C          148          OUT     LOW TIML2
 0066 3E40          149          MVI     A,40H
 0068 D30D          150          OUT     LOW TIMH2
 006A 3E1B          151          MVI     A,1BH   ;UNMASK RST 7.5
) 006C 30           152          SIM
( 141)
 006D 3ECE          153          MVI     A,0CEH
 006F D308          154          OUT     LOW PPI2
                    155 ;
 0071 AF            156 WRMSTR:  XRA     A
 0072 323E00   D    157          STA     BRKFL
 0075 211C00   D    158          LXI     H,MSTAK-20
 0078 223A00   D    159          SHLD    SSAVE
 007B 3EC7          160          MVI     A,0C7H
```

```
LOC  OBJ          SEQ          SOURCE STATEMENT

007D 323F00   D    161          STA     USERBR
0080 313000   D    162          LXI     SP.MSTAK
                   163 SOMSG:
0083 FB            164          EI
0084 CD0000   E    165          CALL    DISCLR
0087 3E0F          166          MVI     A.15
0089 320000   E    167          STA     DCLCNT
008C 3EFF          168          MVI     A.0FFH
008E 320000   E    169          STA     KEYBUF
0091 AF            170          XRA     A
0092 320000   E    171          STA     KEYFLG
0095 320000   E    172          STA     DISCNT
0098 CD0000   E    173          CALL    STNDBY
009B 21DD00   C    174          LXI     H.SGNON
009E CD0000   E    175          CALL    MESOUT
00A1 CD0000   E    176          CALL    CRWAIT
                   177 :
                   178 :*****************************************************
                   179 :*
                   180 :*
                   181 :*           COMMAND RECOGNIZING ROUTINE
                   182 :*
                   183 :*
                   184 :*****************************************************
                   185 :*
                   186 :* GETCM RECEIVES AN INPUT CHARACTER FROM THE
                   187 :* USER AND ATTEMPTS TO LOCATE THIS CHARACTER
                   188 :* IN ITS COMMAND CHARACTER TABLE.
                   189 :* IF SUCCESSFUL.THE ROUTINE CORRESPONDING TO
                   190 :* THIS CHARACTER IS SELECTED FROM A TABLE OF
                   191 :* COMMAND ROUTINE ADDRESSES. AND CONTROL IS
                   192 :* TRANSFERRED TO THIS ROUTINE. IF THE
                   193 :* CHARACTER DOES NOT MATCH ANY ENTRIES.
                   194 :* CONTROL IS PASSED TO THE ERROR HANDLER.
                   195 :*
                   196 GETCM:
00A4 313000   D    197          LXI     SP.MSTAK        :ALWAYS WANT TO RESE
                   198                                  :/STACK PTR TO MONITOR STAR
                   199                                  :/VALUE SO ROUTINES NEEDN'T
      66 0C        200                                  :/CLEAN UP
00A7 CD0000   E    201          CALL    STNDBY
00AA CD0000   E    202          CALL    DISCLR
00AD 3E35          203          MVI     A.35H
00AF CD0000   E    204          CALL    SNDLED
00B2 CD0000   E    205          CALL    CROUT
00B5 0E2E          206          MVI     C.'.'   :PROMPT CHARACTER TO C
00B7 CD0000   E    207          CALL    ECHO    :SEND PROMPT CHARACTER TO U
                   208                          :/TERMINAL
00BA CD0000   E    209 GTC03:   CALL    GETCH
00BD CD0000   E    210          CALL    ECHO
00C0 21ED00   C    211          LXI     H.CTAB
00C3 CD0000   E    212          CALL    TBLSER
00C6 DAD200   C    213          JC      ERROR
00C9 7E            214          MOV     A.M
00CA 23            215          INX     H
```

```
LOC  OBJ          SEQ          SOURCE STATEMENT

00CB 66             216          MOV     H,M
00CC 6F             217          MOV     L,A
00CD 01D700    C    218          LXI     B,EXIT
00D0 C5             219          PUSH    B
00D1 E9             220          PCHL
                    221 ;*****************************************************
                    222 ;*
                    223 ;* ERROR PRINTS THE ERROR CHARACTER(CURRENTLY
                    224 ;* AN CROSS-HATCH) ON THE CONSELE,FOLLOWED BY A
                    225 ;* CARRIAGE RETURN-LINE FEED, AND THEN RETURNS
                    226 ;* CONTROL TO THE COMMAND RECOGNIZER.
                    227 ;*
00D2 0E23           228 ERROR:  MVI     C,'#'
00D4 CD0000    E    229         CALL    ECHO
00D7 CD0000    E    230 EXIT:   CALL    CROUT
00DA C3A400    C    231         JMP     GETCM
                    232 ;
                    233 ;*****************************************************
                    234 ;********** MONITOR TABLES ***********************
                    235 ;*****************************************************
                    236 ;
00DD 0D             237 SGNON:  DB      0DH,0AH,'SYS IS READY ',0
00DE 0A
00DF 53595320
00E3 49532052
00E7 45414459
00EB 20
00EC 00
                    238 ;-------------------
                    239 CTAB:                           ;TABLE OF COMMAND CHARACTERS
00ED 0C             240         DB      12              ;NUMBER OF VALID COMMANDS
00EE 0300           241         DW      3               ;NUMBER OF BYTES PER ENTRY
00F0 41             242         DB      'A'             ;MOVE
00F1 0000      E    243         DW      ACMD
00F3 42             244         DB      'B'             ;WRITE TO MDS
00F4 0000      E    245         DW      BCMD
00F6 44             246 LD:     DB      'D'             ;DISPLAY
00F7 0000      E    247         DW      DCMD
00F9 45             248         DB      'E'             ;REG EXAM
00FA 0000      E    249         DW      ECMD
00FC 46             250         DB      'F'             ;FILL
00FD 0000      E    251         DW      FCMD
00FF 47             252         DB      'G'             ;GO
0100 0000      E    253         DW      GCMD
0102 49             254         DB      'I'             ;INPUT PORT
0103 0000      E    255         DW      ICMD
0105 4C             256         DB      'L'             ;LOAD FROM MDS
0106 0000      E    257         DW      LCMD
0108 4F             258         DB      'O'             ;OUTPUT PORT
0109 0000      E    259         DW      OCMD
010B 53             260         DB      'S'             ;SUBSTITUTE
010C 0000      E    261         DW      SCMD
010E 54             262         DB      'T'             ;HEX ADD/SUBT
010F 0000      E    263         DW      TCMD
0111 50             264         DB      'P'
```

```
 OC  OBJ           SEQ          SOURCE STATEMENT

)112 0000      E   265          DW      MPPMON
                   266 ;-------------------
                   267 ;
                   268 RTAB:                        ;TABLE OF REGISTER INFORMATIO
)114 41            269          DB      'A'. LOW ASAVE
)115 35        D
)116 42            270          DB      'B'. LOW BSAVE
)117 33        D
)118 43            271          DB      'C'. LOW CSAVE
)119 32        D
)11A 44            272          DB      'D'. LOW DSAVE
011B 31        D
011C 45            273          DB      'E'. LOW ESAVE
011D 30        D
011E 46            274          DB      'F'. LOW FSAVE
011F 34        D
0120 48            275          DB      'H'. LOW HSAVE
0121 37        D
0122 4C            276          DB      'L'. LOW LSAVE
0123 36        D
0124 4D            277          DB      'M'. LOW HSAVE
0125 37        D
0126 50            278          DB      'P'. LOW PSAVE+1
0127 39        D
0128 53            279          DB      'S'. LOW SSAVE+1
0129 3B        D
012A 00            280          DB      00.00
012B 00
                   281 ;
                   282 ;*****************************************************
                   283 ;
                   284          END

UBLIC SYMBOLS
RKFL   D 003E    ERROR  C 00D2    EXIT   C 00D7    GETCM  C 00A4
AVE    D 0036    MONINI C 0000    MSTAK  D 0030    PORTA1 A 0001
ORTA2  A 0009    PORTB1 A 0002    PORTB2 A 000A    PORTC1 A 0003
ORTC2  A 000B    PPI1   A 0000    PSAVE  D 0038    RTAB   C 0114
CRTH   D 0030    SSAVE  D 003A    TEMP   D 003C    TEMP1  D 003D
ART1   A 0010

ITERNAL SYMBOLS
CMD    E 0000    ADRD   E 0000    BCMD   E 0000    CO     E 0000
ROUT   E 0000    CRWAIT E 0000    DCLCNT E 0000    DCMD   E 0000
ISCLR  E 0000    DISCNT E 0000    ECHO   E 0000    ECMD   E 0000
CMD    E 0000    GCMD   E 0000    GETCH  E 0000    ICMD   E 0000
NTTIM  E 0000    KEYBUF E 0000    KEYFLG E 0000    LCMD   E 0000
ISOUT  E 0000    MPPMON E 0000    OCMD   E 0000    SCMD   E 0000
NDLED  E 0000    STNDBY E 0000    TBLSER E 0000    TCMD   E 0000

SER SYMBOLS
CMD    E 0000    ADRD   E 0000    ADROUT C 0027    ASAVE  D 0035
CMD    E 0000    BRKFL  D 003E    BSAVE  D 0033    CO     E 0000
ROUT   E 0000    CRWAIT E 0000    CSAVE  D 0032    CTAB   C 00ED
CLCNT  E 0000    DCMD   E 0000    DISCLR E 0000    DISCNT E 0000
```

```
BAVE   D 0031    ECHO   E 0000    ECMD   E 0000    ERROR  C 00D2
SAVE   D 0030    EXIT   C 00D7    FCMD   E 0000    FSAVE  D 0034
CMD    E 0000    GETCH  E 0000    GETCM  C 00A4    GO     C 0008
TCO3   C 00BA    HSAVE  D 0037    ICMD   E 0000    INTTIM E 0000
NUST   C 003F    KEYBUF E 0000    KEYFLG E 0000    LCMD   E 0000
D      C 00F6    LSAVE  D 0036    MESOUT E 0000    MONINI C 0000
PPMON  E 0000    MSTAK  D 0030    NOBRK  C 001C    OCMD   E 0000
ORTA1  A 0001    PORTA2 A 0009    PORTB1 A 0002    PORTB2 A 000A
ORTC1  A 0003    PORTC2 A 000B    PPI1   A 0000    PPI2   A 0008
SAVE   D 0038    RTAB   C 0114    SCMD   E 0000    SCRTH  D 0030
GNON   C 00DD    SNDLED E 0000    SOMSG  C 0083    SSAVE  D 003A
TKBOT  D 0000    STNDBY E 0000    TBLSER E 0000    TCMD   E 0000
EMP    D 003C    TEMP1  D 003D    TIMH1  A 0005    TIMH2  A 000D
IML1   A 0004    TIML2  A 000C    USART1 A 0010    USART2 A 0018
SERBR  D 003F    WRMSTR C 0071
```

SSEMBLY COMPLETE. 2 ERRORS (152)

```
M80 MON85B.001 NOOBJECT PRINT(:LP:)

IS-II 8080/8085 MACRO ASSEMBLER, V2.0          MPP85B   PAGE    1

LOC  OBJ          SEQ          SOURCE STATEMENT

                    1 $PAGEWIDTH(80)  MOD85
                    2 ;*****************************************************
                    3 ;**
                    4 ;**                  MPP   MONITOR  V1
                    5 ;**
                    6 ;** PROG.NO:200-09-02                  DATE:   25.06.198
                    7 ;** FILE   :MON85B.001                 MODULE: MPP85B
                    8 ;** AUTHOR :E.DURMUS
                    9 ;**
                   10 ;*****************************************************
                   11 ;
                   12          NAME    MPP85B
                   13 ;
                   14          PUBLIC  DCMD,GCMD,ICMD,FCMD
                   15          PUBLIC  TCMD,SCMD,ACMD,DSUB
                   16          PUBLIC  DCM05,OCMD
                   17 ;
                   18          EXTRN   GETCM,ADRD,NMOUT
                   19          EXTRN   HILO,GETHX,ERROR,PSAVE
                   20          EXTRN   TEMP1,BRKFL,RSTTF,GETCH
                   21          EXTRN   VALDL,VALDG,CNVBN,STHLF
                   22          EXTRN   TEMP,STHFO,RTAB,SCRTH
                   23          EXTRN   USART1,KEYBUF,CRWAIT
                   24          EXTRN   CI,CO,SPCOUT,ECHO,GETNM,CROUT
                   25 ;
                   26          DSEG
                   27 ;
0003               28 OUTBUF:  DS      3
                   29 ;
                   30
                   31          CSEG
                   32 ;*****************************************************
                   33 ;
                   34 ;
                   35 ;           COMMAND IMPLEMENTING ROUTINES
                   36 ;
                   37 ;
                   38 ;*****************************************************
                   39 ;*
                   40 ;*PROCEDURE:DCMD
                   41 ;*FUNCTION :DCMD,IMPLEMENTS THE DISPLAY MEMORY
                   42 ;*            (D) COMMAND
                   43 ;*INPUTS    :NONE
                   44 ;*OUTPUTS   :NONE
                   45 ;*CALLS     :ECHO,NMOUT,HILO,GETCM,CROUT,GETNM
                   46 ;*DESTROYS  :ALL
                   47 ;*
                   48 DCMD:
0000 0E02          49 DSUB:    MVI     C,2         ;GET 2 NO FROM INPUT STREAM
0002 CD0000    E   50          CALL    GETNM
0005 D1            51          POP     D           ;ENDING ADDRESS TO DE
0006 E1            52          POP     H           ;STARTING ADDRESS TO HL
```

```
LOC  OBJ          SEQ          SOURCE STATEMENT

0007 CD0000    E     53 DCM05:  CALL     CROUT
000A CD0000    E     54         CALL     ADRD
000D CD0000    E     55 DCM10:  CALL     SPCOUT
0010 7E              56         MOV      A,M
0011 CD0000    E     57         CALL     NMOUT
0014 C34501    C     58         JMP      CIBRK
0017 E67F            59 DCM15:  ANI      07FH
0019 FE1B            60         CPI      01BH
001B C8              61         RZ
001C CD0000    E     62 BRK:    CALL     HILO
001F D8              63         RC
0020 23              64         INX      H
0021 7D              65         MOV      A,L
0022 E60F            66         ANI      00FH
0024 C20D00    C     67         JNZ      DCM10
0027 C30700    C     68         JMP      DCM05
                     69 ;*****************************************************
                     70 ;*
                     71 ;*PROCEDURE:GCMD
                     72 ;*FUNCTION :GCMD IMPLEMENTS THE BEGIN EXECUTION
                     73 ;*          (G) COMMAND
                     74 ;*INPUTS    :NONE
                     75 ;*OUTPUTS   :NONE
                     76 ;*CALLS     :ERROR,GETHX,RSTTF
                     77 ;*DESTROYS  :ALL
                     78 ;*
                     79 GCMD:
002A CD0000    E     80         CALL     GETHX    ;GET ADDR.FROM INPUT STREAM
002D D23600    C     81         JNC      GCM05
0030 210000    E     82         LXI      H,PSAVE
0033 71              83         MOV      M,C
0034 23              84         INX      H
0035 70              85         MOV      M,B
0036 7A              86 GCM05:  MOV      A,D
0037 FE0D            87         CPI      00DH
0039 3E00            88         MVI      A,0
003B CA4B00    C     89         JZ       GCM10
003E CD0000    E     90 GCM15:  CALL     GETHX
0041 D20000    E     91         JNC      ERROR
0044 0A              92         LDAX     B
0045 320000    E     93         STA      TEMP1
0048 3ECF            94         MVI      A,0CFH
004A 02              95         STAX     B
004B 320000    E     96 GCM10:  STA      BRKFL
004E C30000    E     97         JMP      RSTTF
                     98 ;*****************************************************
                     99 ;*
                    100 ;*PROCEDURE:ICMD
                    101 ;*FUNCTION :ICMD IMPLEMENTS THE INPUT COMMMAND
                    102 ;*          MEMORY (I) COMMAND
                    103 ;*INPUTS    :NONE
                    104 ;*OUTPUTS   :NONE
                    105 ;*CALLS     :,
                    106 ;*
                    107 ;*DESTROYS  :ALL
```

```
LOC  OBJ          SEQ          SOURCE STATEMENT

                   108 ;*
                   109 ICMD:
0051 CD0000    E   110          CALL    GETHX
0054 C5            111 ICMO5:   PUSH    B
0055 7A            112          MOV     A,D
0056 FE20          113          CPI     ' '
0058 CA6000    C   114          JZ      ICM10
005B FE2C          115          CPI     ','
005D C20000    E   116          JNZ     GETCM
0060 CDA100    C   117 ICM10:   CALL    INPORT
0063 CD0000    E   118          CALL    NMOUT
0066 0E2D          119          MVI     C,'-'
0068 CD0000    E   120          CALL    ECHO
006B CD0000    E   121          CALL    GETHX
006E D27200    C   122          JNC     ICM15
0071 C9            123          RET
0072 C1            124 ICM15:   POP     B
0073 0C            125          INR     C
0074 C35400    C   126          JMP     ICMO5
                   127 ;
                   128 ;*****************************************************
                   129 ;*PROCEDURE:OCMND
                   130 ;*FUNCYION :OCMND IMPLEMENTS THE OUTPUT COMMAND
                   131 ;
0077 CD0000    E   132 OCMD:    CALL    GETHX
007A C5            133 OCMO5:   PUSH    B
007B 7A            134          MOV     A,D
007C FE20          135          CPI     ' '
007E CA8600    C   136          JZ      OCM10
0081 FE2C          137          CPI     ','
0083 C20000    E   138          JNZ     GETCM
0086 CDA100    C   139 OCM10:   CALL    INPORT
0089 CD0000    E   140          CALL    NMOUT
008C 0E2D          141          MVI     C,'-'
008E CD0000    E   142          CALL    ECHO
0091 CD0000    E   143          CALL    GETHX
0094 D29C00    C   144          JNC     OCM15
0097 D1            145          POP     D
0098 D5            146          PUSH    D
0099 CDAF00    C   147          CALL    OUPORT
009C C1            148 OCM15:   POP     B
009D 0C            149          INR     C
009E C37A00    C   150          JMP     OCMO5
                   151 ;
                   152 ;*************************************************
                   153 ;*PROCEDURE:INPORT
                   154 ;*FUNCTION :PROCESS INPUT PORT
                   155 ;
00A1 210000    D   156 INPORT:  LXI     H,OUTBUF
00A4 36DB          157          MVI     M,0DBH
00A6 23            158          INX     H
00A7 71            159          MOV     M,C
00A8 23            160          INX     H
00A9 36C9          161          MVI     M,0C9H
00AB CD0000    D   162          CALL    OUTBUF
```

```
LOC  OBJ          SEQ          SOURCE STATEMENT

00AE C9             163          RET
                    164 ;
                    165 ;*****************************************************
                    166 ;*PROCEDURE:OUPORT
                    167 ;*FUNCTION :OUTPUT PORT
                    168 ;
00AF 210000    D    169 OUPORT: LXI     H,OUTBUF
00B2 36D3           170         MVI     M,0D3H
00B4 23             171         INX     H
00B5 73             172         MOV     M,E
00B6 23             173         INX     H
00B7 36C9           174         MVI     M,0C9H
00B9 79             175         MOV     A,C
00BA CD0000    D    176         CALL    OUTBUF
00BD C9             177         RET
                    178 ;
                    179 ;*****************************************************
                    180 ;*
                    181 ;*PROCEDURE:FCMD
                    182 ;*FUNCTION :FILLS A MEMORY BLOCK BLOCK WITH DATA
                    183 ;*INPUTS    :NONE
                    184 ;*OUTPUTS   :NONE
                    185 ;*CALLS     :GETNM,HILO
                    186 ;*DESTROYS  :ALL
                    187 ;*
00BE 0E03           188 FCMD:   MVI     C,3
00C0 CD0000    E    189         CALL    GETNM
00C3 C1             190         POP     B
00C4 D1             191         POP     D
00C5 E1             192         POP     H
00C6 71             193         MOV     M,C
00C7 E5             194         PUSH    H
00C8 1B             195         DCX     D
00C9 D5             196         PUSH    D
00CA 23             197         INX     H
00CB E5             198         PUSH    H
00CC C3FC00    C    199         JMP     MCMD7
                    200 ;*****************************************************
                    201 ;*
                    202 ;*PROCEDURE:TCMD
                    203 ;*FUNCTION :ADDS/SUBRACTS TWO 16 BIT HEX NO
                    204 ;*INPUTS    :NONE
                    205 ;*OUTPUTS   :NONE
                    206 ;*CALLS     :GETHX,ADRD,SPCOUT
                    207 ;*DESTROYS  :ALL
                    208 ;*
00CF CD0000    E    209 TCMD:   CALL    GETHX
00D2 D20000    E    210         JNC     ERROR
00D5 C5             211         PUSH    B
00D6 D1             212         POP     D
00D7 C5             213         PUSH    B
00D8 E1             214         POP     H
00D9 CD0000    E    215         CALL    GETHX
00DC D20000    E    216         JNC     ERROR
00DF C5             217         PUSH    B
```

```
.OC  OBJ          SEQ          SOURCE STATEMENT

)0E0 E5            218          PUSH    H
)0E1 09            219          DAD     B
)0E2 CD0000    E   220          CALL    ADRD
)0E5 CD0000    E   221          CALL    SPCOUT
)0E8 D1            222          POP     D
)0E9 C1            223          POP     B
)0EA 7B            224          MOV     A,E
00EB 91            225          SUB     C
00EC 6F            226          MOV     L,A
00ED 7A            227          MOV     A,D
00EE 98            228          SBB     B
00EF 67            229          MOV     H,A
00F0 CD0000    E   230          CALL    ADRD
00F3 CD0000    E   231          CALL    CRWAIT
00F6 C9            232          RET
                   233 ;*****************************************************
                   234 ;*
                   235 ;*PROCEDURE:ACMD
                   236 ;*FUNCTION :ACMD IMPLEMENTS THE MOVE DATA IN MEMORY
                   237 ;*           (A) COMMAND.
                   238 ;*INPUTS    :NONE
                   239 ;*OUTPUTS   :NONE
                   240 ;*CALLS     :GETCM,HILO,GETNM
                   241 ;*DESTROYS :ALL
                   242 ;*
                   243 ACMD:
00F7 0E03          244          MVI     C,3
00F9 CD0000    E   245          CALL    GETNM    ;GET 3 NO FROM INPUT STREAM
00FC C1            246 MCMD7:   POP     B
00FD E1            247          POP     H
00FE D1            248          POP     D
00FF 1A            249 MCMD5:   LDAX    D
0100 02            250          STAX    B
0101 03            251          INX     B
0102 78            252          MOV     A,B
0103 B1            253          ORA     C
0104 C8            254          RZ
0105 13            255          INX     D
0106 CD0000    E   256          CALL    HILO
0109 D0            257          RNC
010A C3FF00    C   258          JMP     MCMD5
                   259 ;*****************************************************
                   260 ;*
                   261 ;*PROCEDURE:SCMD
                   262 ;*FUNCTION :SCMD, IMPLEMENTS THE SUBSTITUTE INTO
                   263 ;*           MEMORY (S) COMMAND.
                   264 ;*INPUTS    :NONE
                   265 ;*OUTPUTS   :NONE
                   266 ;*CALLS     :GETHX,GETCM,NMOUT,ECHO
                   267 ;*DESTROYS :ALL
                   268 ;*
                   269 SCMD:
010D CD0000    E   270          CALL    GETHX    ;GET A NO, IF PRESENT, FROM IN
0110 C5            271          PUSH    B
0111 E1            272          POP     H
```

```
LOC  OBJ           SEQ          SOURCE STATEMENT

0112 7A            273          MOV     A,D
0113 FE20          274          CPI     ' '
0115 CA1D01    C   275          JZ      SCM10
0118 FE2C          276          CPI     ','
011A C20000    E   277          JNZ     GETCM
011D 7E            278 SCM10:   MOV     A,M
011E CD0000    E   279          CALL    NMOUT
0121 0E2D          280          MVI     C,'-'
0123 CD0000    E   281          CALL    ECHO
0126 CD0000    E   282          CALL    GETHX
0129 D22D01    C   283          JNC     SCM15
012C 71            284          MOV     M,C
012D 7A            285 SCM15:   MOV     A,D
012E FE20          286          CPI     ' '
0130 CA3801    C   287          JZ      SCM20
0133 FE2C          288          CPI     ','
0135 C20000    E   289          JNZ     GETCM
0138 CD0000    E   290 SCM20:   CALL    CROUT
013B 23            291          INX     H
013C CD0000    E   292          CALL    ADRD
013F CD0000    E   293          CALL    SPCOUT
0142 C31D01    C   294          JMP     SCM10
                   295 ;
                   296 ;*****************************************************
                   297 ;
0145 DB01      E   298 CIBRK:   IN      LOW USART1+1
0147 17            299          RAL
0148 DA5701    C   300          JC      CIBKEY
014B DB01      E   301          IN      LOW USART1+1
014D E602          302          ANI     002H
014F CA1C00    C   303          JZ      BRK
0152 DB00      E   304          IN      LOW USART1
0154 C31700    C   305          JMP     DCM15
0157 3A0000    E   306 CIBKEY:  LDA     KEYBUF
015A FEFF          307          CPI     0FFH
015C CA1C00    C   308          JZ      BRK
015F C31700    C   309          JMP     DCM15
                   310 ;*********************************************
                   311          END

UBLIC SYMBOLS
CMD    C 00F7    DCM05  C 0007    DCMD   C 0000    DSUB   C 0000
CMD    C 00BE    GCMD   C 002A    ICMD   C 0051    OCMD   C 0077
CMD    C 010D    TCMD   C 00CF

XTERNAL SYMBOLS
DRD    E 0000    BRKFL  E 0000    CI     E 0000    CNVBN  E 0000
D      E 0000    CROUT  E 0000    CRWAIT E 0000    ECHO   E 0000
RROR   E 0000    GETCH  E 0000    GETCM  E 0000    GETHX  E 0000
ETNM   E 0000    HILO   E 0000    KEYBUF E 0000    NMOUT  E 0000
SAVE   E 0000    RSTTF  E 0000    RTAB   E 0000    SCRTH  E 0000
PCOUT  E 0000    STHF0  E 0000    STHLF  E 0000    TEMP   E 0000
EMP1   E 0000    USART1 E 0000    VALDG  E 0000    VALDL  E 0000

SER SYMBOLS
```

MD	C 00F7	ADRD	E 0000	BRK	C 001C	BRKFL	E 0000
	E 0000	CIBKEY	C 0157	CIBRK	C 0145	CNVBN	E 0000
	E 0000	CROUT	E 0000	CRWAIT	E 0000	DCM05	C 0007
M10	C 000D	DCM15	C 0017	DCMD	C 0000	DSUB	C 0000
HO	E 0000	ERROR	E 0000	FCMD	C 00BE	GCM05	C 0036
M10	C 004B	GCM15	C 003E	GCMD	C 002A	GETCH	E 0000
TCM	E 0000	GETHX	E 0000	GETNM	E 0000	HILO	E 0000
M05	C 0054	ICM10	C 0060	ICM15	C 0072	ICMD	C 0051
PORT	C 00A1	KEYBUF	E 0000	MCM05	C 00FF	MCMD7	C 00FC
OUT	E 0000	OCM05	C 007A	OCM10	C 0086	OCM15	C 009C
MD	C 0077	OUPORT	C 00AF	OUTBUF	D 0000	PSAVE	E 0000
TTF	E 0000	RTAB	E 0000	SCM10	C 011D	SCM15	C 012D
M20	C 0138	SCMD	C 010D	SCRTH	E 0000	SPCOUT	E 0000
HF0	E 0000	STHLF	E 0000	TCMD	C 00CF	TEMP	E 0000
MP1	E 0000	USART1	E 0000	VALDG	E 0000	VALDL	E 0000

SEMBLY COMPLETE, NO ERRORS

```
$M80 MON85C.001 NOOBJECT PRINT(:LP:)

ISIS-II 8080/8085 MACRO ASSEMBLER, V2.0          MPP85C    PAGE     1


 LOC  OBJ           SEQ          SOURCE STATEMENT

                      1 $PAGEWIDTH(80)
                      2 ;*****************************************************
                      3 ;**
                      4 ;**                NET85   MONITOR  V1
                      5 ;**
                      6 ;** PROG.NO:200-09-030                      1982.03.05
                      7 ;** FILE    :MON85C.001                     MODULE:MPP85C
                      8 ;** AUTHOR  :E.DURMUS
                      9 ;**
                     10 ;*****************************************************
                     11 ;
                     12           NAME     MPP85C
                     13 ;
                     14           PUBLIC   ECMD
                     15 ;
                     16           EXTRN    GETCM, NMOUT, EXIT, ERROR
                     17           EXTRN    TEMP, ECHO, RTAB, SCRTH, GETHX
                     18           EXTRN    GETCH, CROUT, SPCOUT
                     19 ;
                     20           CSEG
                     21 ;*****************************************************
                     22 ;*
                     23 ;*PROCEDURE:ECMD
                     24 ;*FUNCTION :ECMD IMPLEMENTS THE REGISTER EXAMINE AND
                     25 ;*            CHANGE (E) COMMAND.
                     26 ;*INPUTS    :NONE
                     27 ;*OUTPUTS   :NONE
                     28 ;*CALLS     :GETCH, ECHO, REGDS, GETCM, ERROR, RGADR, NMOUT
                     29 ;*            CROUT, GETHX
                     30 ;*DESTROYS  :ALL
                     31 ;*
                     32 ECMD:
0000 CD0000      E   33           CALL     GETCH    ;GET REGISTER IDENTIFIER
0003 CD0000      E   34           CALL     ECHO     ;ECHO IT
0006 FE0A            35           CPI      00AH
0008 C23D00      C   36           JNZ      XCMOS
                     37 ;*****************************************************
                     38 ;*
                     39 ;*PROCEDURE:REGDS
                     40 ;*FUNCTION :REGDS DISPLAYS THE CONTENTS OF THE
                     41 ;*            REGISTER SAVE LOCATIONS, IN FORMATTED FRO
                     42 ;*            ON THE CONSOLE.
                     43 ;*            SAVE LOCATIONS, IN FORMATTED FROM, ON THE
                     44 ;*            THE DISPLAY IS DRIVEN FROM A TABLE, RTAB,
                     45 ;*            WHICH CONTAINS THE REGISTER'S PRINT
                     46 ;*            SYMBOL, SAVE LOCATION ADDRESS, AND LENGTH
                     47 ;*            (8 OR 16 BITS).
                     48 ;*INPUTS    :NONE
                     49 ;*OUTPUTS   :NONE
                     50 ;*CALLS     :ECHO, NMOUT, ERROR, CROUT
                     51 ;*DESTROYS  :ALL
                     52 ;*
```

```
LOC  OBJ          SEQ          SOURCE STATEMENT

000B 210000   E    53 REGDS:  LXI     H,RTAB
000E 4E            54 REG05:  MOV     C,M
000F 79            55         MOV     A,C
0010 B7            56         ORA     A
0011 C21A00   C    57         JNZ     REG10
0014 CD0000   E    58         CALL    CROUT
0017 C30000   E    59         JMP     GETCM
001A CD0000   E    60 REG10:  CALL    ECHO
001D F5            61         PUSH    PSW
001E 0E3D          62         MVI     C,'='
0020 CD0000   E    63         CALL    ECHO
0023 23            64         INX     H
0024 5E            65         MOV     E,M
0025 1600     E    66         MVI     D,HIGH SCRTH
0027 1A            67         LDAX    D
0028 CD0000   E    68         CALL    NMOUT
002B F1            69         POP     PSW
002C FE4D          70         CPI     'M'
002E FA3600   C    71         JM      REG15
0031 1B            72         DCX     D
0032 1A            73         LDAX    D
0033 CD0000   E    74         CALL    NMOUT
0036 CD0000   E    75 REG15:  CALL    SPCOUT
0039 23            76         INX     H
003A C30E00   C    77         JMP     REG05
                   78 XCM05:
                   79 ;*****************************************************
                   80 ;*
                   81 ;*PROCEDURE:RGADR
                   82 ;*FUNCTION :RGADR TAKES A SINGLE CHARACTER AS INPUT.
                   83 ;*          THIS CHARACTER DENOTES A REGISTER. RGADR
                   84 ;*          SEARCHES THE TABLE RTAB FOR A MATCH ON
                   85 ;*          THE INPUT ARGUMENT. IF ONE OCCURS, RGADR
                   86 ;*          RETURNS THE ADDR. OF THE ADDR. OF THE
                   87 ;*          SAVE LOCATION CORRESPONDING TO THE REG.
                   88 ;*          THIS ADDR. POINTS INTO RTAB. IF NO MATCH
                   89 ;*          OCCURS, THEN THE REGISTER IDENTIFIER IS
                   90 ;*          ILLEGAL AND CONTROL IS PASSED TO THE
                   91 ;*          ERROR ROUTINE.
                   92 ;*INPUTS   :(C)=CHARACTER DENOTING REGISTER
                   93 ;*OUTPUTS  :(BC)=ADDRESS OF ENTRY IN RTAB CORRESPOND
                   94 ;*          TO REGISTER
                   95 ;*CALLS    :ERROR
                   96 ;*DESTROYS :ALL
                   97 ;*
                   98 RGADR:
003D 210000   E    99         LXI     H,RTAB   :HL GETS ADDR. OF TABLE STAR
0040 110200       100         LXI     D,2
0043 7E           101 RGA05:  MOV     A,M
0044 B7           102         ORA     A
0045 CA0000   E   103         JZ      ERROR
0048 B9           104         CMP     C
0049 CA5000   C   105         JZ      RGA10
004C 19           106         DAD     D
004D C34300   C   107         JMP     RGA05
```

```
LOC  OBJ          SEQ          SOURCE STATEMENT

0050 23            108 RGA10:  INX     H
0051 CD0000    E   109         CALL    SPCOUT
0054 320000    E   110         STA     TEMP
0057 3A0000    E   111 XCM10:  LDA     TEMP
005A FE20          112         CPI     ' '
005C CA6400    C   113         JZ      XCM15
005F FE2C          114         CPI     ','
0061 C20000    E   115         JNZ     GETCM
0064 7E            116 XCM15:  MOV     A,M
0065 B7            117         ORA     A
0066 CA0000    E   118         JZ      EXIT
0069 5E            119         MOV     E,M
006A 1600      E   120         MVI     D,HIGH SCRTH
006C EB            121         XCHG
006D 7E            122         MOV     A,M
006E CD0000    E   123         CALL    NMOUT
0071 EB            124         XCHG
0072 E5            125         PUSH    H
0073 2B            126         DCX     H
0074 7E            127         MOV     A,M
0075 D5            128         PUSH    D
0076 EB            129         XCHG
0077 FE4D          130         CPI     'M'
0079 FA8100    C   131         JM      XCM20
007C 2B            132         DCX     H
007D 7E            133         MOV     A,M
007E CD0000    E   134         CALL    NMOUT
0081 0E2D          135 XCM20:  MVI     C,'-'
0083 CD0000    E   136         CALL    ECHO
0086 CD0000    E   137         CALL    GETHX
0089 D2A600    C   138         JNC     XCM30
008C 7A            139         MOV     A,D
008D 320000    E   140         STA     TEMP
0090 D1            141         POP     D
0091 E1            142         POP     H
0092 E5            143         PUSH    H
0093 2B            144         DCX     H
0094 7E            145         MOV     A,M
0095 EB            146         XCHG
0096 FE4D          147         CPI     'M'
0098 FA9D00    C   148         JM      XCM25
009B 70            149         MOV     M,B
009C 2B            150         DCX     H
009D 71            151 XCM25:  MOV     M,C
009E 110200        152 XCM27:  LXI     D,2
00A1 E1            153         POP     H
00A2 19            154         DAD     D
00A3 C35700    C   155         JMP     XCM10
00A6 7A            156 XCM30:  MOV     A,D
00A7 320000    E   157         STA     TEMP
00AA D1            158         POP     D
00AB C39E00    C   159         JMP     XCM27
                   160 ;
                   161 ;*******************************************************
                   162 ;
```

```
 LOC  OBJ          SEQ          SOURCE STATEMENT

                    163          END

PUBLIC SYMBOLS
ECMD    C 0000

EXTERNAL SYMBOLS
CROUT   E 0000     ECHO    E 0000     ERROR   E 0000     EXIT    E 0000
GETCH   E 0000     GETCM   E 0000     GETHX   E 0000     NMOUT   E 0000
RTAB    E 0000     SCRTH   E 0000     SPCOUT  E 0000     TEMP    E 0000

USER SYMBOLS
CROUT   E 0000     ECHO    E 0000     ECMD    C 0000     ERROR   E 0000
EXIT    E 0000     GETCH   E 0000     GETCM   E 0000     GETHX   E 0000
NMOUT   E 0000     REG05   C 000E     REG10   C 001A     REG15   C 0036
REGDS   C 000B     RGA05   C 0043     RGA10   C 0050     RGADR   C 003D
RTAB    E 0000     SCRTH   E 0000     SPCOUT  E 0000     TEMP    E 0000
XCM05   C 003D     XCM10   C 0057     XCM15   C 0064     XCM20   C 0081
XCM25   C 009D     XCM27   C 009E     XCM30   C 00A6

ASSEMBLY COMPLETE,  NO ERRORS
```

```
:SM80 MON85D.001 NOOBJECT PRINT(:LP:)

ISIS-II 8080/8085 MACRO ASSEMBLER, V2.0          MPP85D   PAGE    1

 LOC  OBJ          SEQ          SOURCE STATEMENT

                      1 $PAGEWIDTH(80)
                      2 ;*******************************************************
                      3 ;**
                      4 ;**                  MPP   MONITOR  V1
                      5 ;**
                      6 ;** PROG.NO:200-09-04              DATE:   26.06.1984
                      7 ;** FILE   :MON85D.001             MODULE: MPP85D
                      8 ;** AUTHOR :E.DURMUS
                      9 ;**
                     10 ;*******************************************************
                     11 ;
                     12          NAME    MPP85D
                     13 ;
                     14          PUBLIC  CI, CO, SPCOUT, CROUT, ADRD, NMOUT
                     15          PUBLIC  ECHO, GETCH, GETNM, PTSER
                     16          PUBLIC  TBLSER, MESOUT
                     17 ;
                     18          EXTRN   USART1, GETCM, MSTAK
                     19          EXTRN   SSAVE, PSAVE, LSAVE, TEMP
                     20          EXTRN   ERROR, HILO, GETHX
                     21          EXTRN   KEYBUF, DISCNT, DISBUF
                     22          EXTRN   DISCLR
                     23 ;
                     24          CSEG
                     25 ;*******************************************************
                     26 ;*
                     27 ;*PROCEDURE:CI
                     28 ;*FUNCTION :WAIT AND RECEIVE A CHARACTER FROM CONSOL
                     29 ;*INPUTS   :NONE
                     30 ;*OUTPUTS  :(A)=CHARACTER FROM CONSOLE
                     31 ;*CALLS    :NONE
                     32 ;*DESTROYS :A
                     33 ;*
0000 DB01      E     34 CI:      IN      LOW USART1+1
0002 17              35          RAL
0003 DA1000    C     36          JC      CIKEY
0006 DB01      E     37 CICRT:   IN      LOW USART1+1
0008 E602            38          ANI     002H
000A CA0600    C     39          JZ      CICRT
000D DB00      E     40          IN      LOW USART1
000F C9              41          RET
0010 3A0000    E     42 CIKEY:   LDA     KEYBUF
0013 FEFF            43          CPI     OFFH
0015 CA1000    C     44          JZ      CIKEY
0018 4F              45          MOV     C,A
0019 3EFF            46          MVI     A,OFFH
001B 320000    E     47          STA     KEYBUF
001E 79              48          MOV     A,C
001F C9              49          RET
                     50 ;
                     51 ;*******************************************************
                     52 ;*
```

```
LOC  OBJ            SEQ         SOURCE STATEMENT

                     53 ;*PROCEDURE:SPCOUT
                     54 ;*FUNCTION :WRITE A SPACE CHARACTER ON CONSOLE
                     55 ;*INPUTS   :NONE
                     56 ;*OUTPUTS  :NONE
                     57 ;*CALLS    :CO
                     58 ;*DESTROYS :A,C
                     59 ;*
0020 0E20            60 SPCOUT: MVI     C,' '
                     61 ;
                     62 ;*************************************************
                     63 ;*
                     64 ;*PROCEDURE:CO
                     65 ;*FUNCTION :WAIT AND SEND A CHARACTER TO CONSOLE
                     66 ;*INPUTS   :(C)=CHARACTER TO OUTPUT TO CONSOLE
                     67 ;*OUTPUTS  :NONE
                     68 ;*CALLS    :NONE
                     69 ;*DESTROYS :A,F
                     70 ;*
0022 DB01      E     71 CO:     IN      LOW USART1+1
0024 17              72         RAL
0025 DA3300    C     73         JC      CODIS
0028 DB01      E     74 COCRT:  IN      LOW USART1+1
002A E601            75         ANI     001H
002C CA2800    C     76         JZ      COCRT
002F 79              77         MOV     A,C
0030 D300      E     78         OUT     LOW USART1
0032 C9              79         RET
0033 E5              80 CODIS:  PUSH    H
0034 D5              81         PUSH    D
0035 C5              82         PUSH    B
0036 3A0000    E     83         LDA     DISCNT
0039 FE10            84         CPI     16
003B C24200    C     85         JNZ     CO01
003E AF              86         XRA     A
003F 320000    E     87         STA     DISCNT
0042 79              88 CO01:   MOV     A,C
0043 FE0D            89         CPI     0DH
0045 CA4D00    C     90         JZ      CO03
0048 FE0A            91         CPI     0AH
004A C25800    C     92         JNZ     CO02
004D CD0000    E     93 CO03:   CALL    DISCLR
0050 AF              94         XRA     A
0051 320000    E     95         STA     DISCNT
0054 C1              96         POP     B
0055 D1              97         POP     D
0056 E1              98         POP     H
0057 C9              99         RET
0058 79             100 CO02:   MOV     A,C
0059 E63F           101         ANI     3FH
005B 4F             102         MOV     C,A
005C 0600           103         MVI     B,0
005E 217A00    C    104         LXI     H,TASBIN
0061 09             105         DAD     B
0062 4E             106         MOV     C,M
0063 3A0000    E    107         LDA     DISCNT
```

```
LOC  OBJ          SEQ          SOURCE STATEMENT

0066 5F           108          MOV     E,A
0067 1600         109          MVI     D,0
0069 210000   E   110          LXI     H,DISBUF
006C 19           111          DAD     D
006D 71           112          MOV     M,C
006E 3A0000   E   113          LDA     DISCNT
0071 3C           114          INR     A
0072 320000   E   115          STA     DISCNT
0075 C1           116          POP     B
0076 D1           117          POP     D
0077 E1           118          POP     H
0078 79           119          MOV     A,C
0079 C9           120          RET
                  121 ;
007A 00           122 TASBIN:  DB      00H,77H,7CH,39H,5EH,79H,71H,3DH
007B 77
007C 7C
007D 39
007E 5E
007F 79
0080 71
0081 3D
0082 76           123          DB      76H,04H,0EH,00H,38H,00H,54H,5CH
0083 04
0084 0E
0085 00
0086 38
0087 00
0088 54
0089 5C
008A 73           124          DB      73H,00H,50H,6DH,78H,3EH,00H,00H
008B 00
008C 50
008D 6D
008E 78
008F 3E
0090 00
0091 00
0092 00           125          DB      00H,6EH,00H,00H,00H,00H,00H,08H
0093 6E
0094 00
0095 00
0096 00
0097 00
0098 00
0099 08
009A 00           126          DB      00H,00H,00H,49H,49H,00H,00H,00H
009B 00
009C 00
009D 49
009E 49
009F 00
00A0 00
00A1 00
00A2 00           127          DB      00H,00H,00H,00H,00H,40H,08H,00H
```

```
LOC  OBJ           SEQ          SOURCE STATEMENT

00A3 00
00A4 00
00A5 00
00A6 00
00A7 40
00A8 08
00A9 00
00AA 3F             128          DB      3FH,06H,5BH,4FH,66H,6DH,7DH,07H
00AB 06
00AC 5B
00AD 4F
00AE 66
00AF 6D
00B0 7D
00B1 07
00B2 7F             129          DB      7FH,6FH,48H,00H,00H,48H,4CH,53H
00B3 6F
00B4 48
00B5 00
00B6 00
00B7 48
00B8 4C
00B9 53
                    130 ;
                    131 ;*******************************************************
                    132 ;*
                    133 ;*PROCEDURE:CROUT
                    134 ;*FUNCTION :SEND A CR-LF TO CONSOLE
                    135 ;*INPUTS   :NONE
                    136 ;*OUTPUTS  :NONE
                    137 ;*CALLS    :ECHO
                    138 ;*DESTROYS :A,B,C,F
                    139 ;*
00BA 0E0D           140 CROUT:  MVI     C,00DH
                    141 ;
                    142 ;*******************************************************
                    143 ;*
                    144 ;*PROCEDURE:ECHO
                    145 ;*FUNCTION :SENDS A CHARACTER TO CONSOLE.A CARRIAGE
                    146 ;*           RETURN IS ECHOED AS A CARRIAGE RETURN
                    147 ;*           LINE FEED,AND AN ESCAPE CHARACTER IS
                    148 ;*           ECHOED AS $.
                    149 ;*INPUTS   :(C)=CHARACTER TO ECHO TO TERMINAL
                    150 ;*OUTPUTS  :(C)=CHARACTER ECHOED TO TERMINAL
                    151 ;*CALLS    :CO
                    152 ;*DESTROYS :A,B,F
                    153 ;*
00BC 41             154 ECHO:   MOV     B,C
00BD 3E1B           155         MVI     A,01BH
00BF B8             156         CMP     B
00C0 C2C500    C    157         JNZ     ECHO5
00C3 0E24           158         MVI     C,024H
00C5 CD2200    C    159 ECHO5:  CALL    CO
00C8 78             160         MOV     A,B
00C9 FE0D           161         CPI     0DH
```

```
LOC  OBJ          SEQ          SOURCE STATEMENT

00CB C2D300    C   162          JNZ     ECH10
00CE 0E0A          163          MVI     C,00AH
00D0 CD2200    C   164          CALL    CO
00D3 48            165 ECH10:   MOV     C,B
00D4 C9            166          RET
                   167 ;
                   168 ;*****************************************************
                   169 ;*
                   170 ;*PROCEDURE:GETCH
                   171 ;*FUNCTION :RECEIVE A CHARACTER FROM CONSOLE
                   172 ;*           WITHOUT PARITY
                   173 ;*INPUTS    :NONE
                   174 ;*OUTPUTS   :(C)=CHARACTER FROM CONSOLE
                   175 ;*CALLS     :CI
                   176 ;*DESTROYS :A,C,F
                   177 ;*
                   178 GETCH:
00D5 CD0000    C   179          CALL    CI          ;GET CHARACTER FROM TERMINAL
00D8 E67F          180          ANI     7FH         ;TURN OFF PARITY BIT IN CASE
                   181                              ;/SET BY CONSOLE
00DA 4F            182          MOV     C,A         ;PUT VALUE IN C REGISTER FOR
                   183                              ;/RETURN
00DB C9            184          RET
                   185 ;
                   186 ;*****************************************************
                   187 ;*
                   188 ;*PROCEDURE:GETNM
                   189 ;*FUNCTION :FINDS A SPECIFIED COUNT OF NUMBERS,
                   190 ;*           BETWEEN 1 AND 3, INCLUSIVE, IN THE
                   191 ;*           INPUT STREAM AND RETURNS THEIR ON THE
                   192 ;*           STACK. IF 2 OR MORE NUMBERS ARE REQUESTED
                   193 ;*           THEN THE FIRST MUST BE LEES THAN OR EQUAL
                   194 ;*           TO THE SECOND, OR THE FIRST AND SECOND
                   195 ;*           NUMBERS WILL BE SET EQUAL. THE LAST
                   196 ;*           NUMBER REQUESTED MUST BE TERMINATED BY
                   197 ;*           A CARRIAGE RETURN OR AN ERROR INDICATION
                   198 ;*           WILL RESULT.
                   199 ;*INPUTS   :(C)=COUNT OF NUMBERS TO FIND IN INPUT STRE
                   200 ;*OUTPUTS  :(STACK)=NUMBERS FOUND IN REVERSE ORDER(LAS
                   201 ;*           ON TOP OF STACK)
                   202 ;*CALLS    :GETHX,HILO
                   203 ;*DESTROYS:ALL
                   204 ;*
                   205 GETNM:
00DC 2E03          206          MVI     L,003H
00DE 79            207          MOV     A,C
00DF E603          208          ANI     003H
00E1 C8            209          RZ
00E2 67            210          MOV     H,A
00E3 CD0000    E   211 GNM05:   CALL    GETHX
00E6 D20000    E   212          JNC     ERROR
00E9 C5            213          PUSH    B
00EA 2D            214          DCR     L
00EB 25            215          DCR     H
00EC 7A            216          MOV     A,D
```

```
LOC  OBJ          SEQ          SOURCE STATEMENT

00ED CAF800    C   217          JZ      GNM10
00F0 FE0D          218          CPI     0DH
00F2 CA0000    E   219          JZ      ERROR
00F5 C3E300    C   220          JMP     GNM05
00F8 FE0D          221 GNM10:   CPI     0DH
00FA C20000    E   222          JNZ     ERROR
00FD 01FFFF        223          LXI     B,0FFFFH
0100 7D            224          MOV     A,L
0101 B7            225          ORA     A
0102 CA0A01    C   226          JZ      GNM20
0105 C5            227 GNM15:   PUSH    B
0106 2D            228          DCR     L
0107 C20501    C   229          JNZ     GNM15
010A C1            230 GNM20:   POP     B
010B D1            231          POP     D
010C E1            232          POP     H
010D CD0000    E   233          CALL    HILO
0110 D21501    C   234          JNC     GNM25
0113 54            235          MOV     D,H
0114 5D            236          MOV     E,L
0115 E3            237 GNM25:   XTHL
0116 D5            238          PUSH    D
0117 C5            239          PUSH    B
0118 E5            240          PUSH    H
0119 3D            241 GNM30:   DCR     A
011A F8            242          RM
011B E1            243          POP     H
011C E3            244          XTHL
011D C31901    C   245          JMP     GNM30
                   246 ;
                   247 ;*****************************************************
                   248 ;*
                   249 ;*PROCEDURE:ADRD
                   250 ;*FUNCTION :OUTPUTS TO THE CONSOLE THE ADDR.
                   251 ;*           CONTAINED IN THE H,L REGISTER.
                   252 ;*INPUTS    :(HL)=ADDRESS TO BE DISPLAYED
                   253 ;*OUTPUTS   :NONE
                   254 ;*CALLS     :NMOUT
                   255 ;*DESTROYS :NONE
                   256 ;*
0120 7C            257 ADRD:    MOV     A,H
0121 CD2501    C   258          CALL    NMOUT
0124 7D            259          MOV     A,L
                   260 ;
                   261 ;*****************************************************
                   262 ;*
                   263 ;*PROCEDURE:NMOUT
                   264 ;*FUNCTION :CONVERT BINARY TO ASCII-HEX AND
                   265 ;*           SEND TO CONSOLE
                   266 ;*INPUTS    :(A)=8 BIT UNSIGNED INTEGER
                   267 ;*OUTPUTS   :NONE
                   268 ;*CALLS     :ECHO,PRVAL
                   269 ;*DESTROYS :A,B,C,F
                   270 ;*
0125 F5            271 NMOUT:   PUSH    PSW
```

```
LOC  OBJ          SEQ          SOURCE STATEMENT

0126 0F            272          RRC
0127 0F            273          RRC
0128 0F            274          RRC
0129 0F            275          RRC
012A CD2E01    C   276          CALL    PRVAL
012D F1            277          POP     PSW
                   278 ;
                   279 ;*****************************************************
                   280 ;*
                   281 ;*PROCEDURE:PRVAL
                   282 ;*FUNCTION :CONVERT A NIBBLE TO ASCII-HEX
                   283 ;*INPUTS    :(A)=INTEGER, RANGE 0 TO F
                   284 ;*OUTPUTS   :(A)=ASCII CHARACTER
                   285 ;*CALLS     :NONE
                   286 ;*DESTROYS :NONE
                   287 ;*
012E E60F          288 PRVAL:  ANI     00FH
0130 C690          289         ADI     090H
0132 27            290         DAA
0133 CE40          291         ACI     040H
0135 27            292         DAA
0136 4F            293         MOV     C,A
0137 C3BC00    C   294         JMP     ECHO
                   295 ;
                   296 ;*****************************************************
                   297 ;*
                   298 ;*PROCEDURE:TBLSER
                   299 ;*FUNCTION :SEE PTSER EXCEPT NO OF ENTRIES IN (TABLE)
                   300 ;*           NO OF BYTES PER ENTRY IN (TABLE+1,+2)
                   301 ;*INPUTS    :(A)=ARGUMENT BYTE
                   302 ;*           (HL)=ADDRESS OF TABLE
                   303 ;*OUTPUTS   :(HL)=ADDRESS OF FIRST VALUE BYTE
                   304 ;*           (CY_FLAG)=RESET IF MATCH FOUND
                   305 ;*CALLS     :PTSER
                   306 ;*DESTROYS :ALL EXCEPT A
                   307 ;*
013A 46            308 TBLSER: MOV     B,M
013B 23            309         INX     H
013C 5E            310         MOV     E,M
013D 23            311         INX     H
013E 56            312         MOV     D,M
013F 23            313         INX     H
                   314 ;
                   315 ;*****************************************************
                   316 ;*
                   317 ;*PROCEDURE:PTSER
                   318 ;*FUNCTION :MATCHES AN ARGUMENT WITH VALUES IN A
                   319 ;*           TABLE WITH SEQUENTIAL SEARCH, AND DELIVERS
                   320 ;*           THE POINTER TO THE FIRST VALUE BYTE
                   321 ;*INPUTS    :(A)=ARGUMENT BYTE
                   322 ;*           (B)=NO OF ENTRIES IN TABLE
                   323 ;*           (DE)=NO OF BYTES PER ENTRY
                   324 ;*           (HL)=ADDRESS OF TABLE
                   325 ;*OUTPUTS   :(HL)=ADDR OF FIRST VALUE BYTE
                   326 ;*           (CY_FLAG)=RESET IF MATCH FOUND
```

```
LOC  OBJ          SEQ        SOURCE STATEMENT

                   327 ;*CALLS     :NONE
                   328 ;*DESTROYS  :B, D, E, F
                   329 ;*
0140 BE            330 PTSER:  CMP     M
0141 CA4A01    C   331         JZ      PTFIN
0144 19            332         DAD     D
0145 05            333         DCR     B
0146 C24001    C   334         JNZ     PTSER
0149 37            335         STC
014A 23            336 PTFIN:  INX     H
014B C9            337         RET
                   338 ;
                   339 ;*******************************************************
                   340 ;*
                   341 ;*PROCEDURE:MESOUT
                   342 ;*FUNCTION  :WRITE MESSAGE ON CONSOLE
                   343 ;*           MESSAGE ANDS WITH 0
                   344 ;*INPUTS    :(HL)=POINTER TO START OF MESSAGE
                   345 ;*OUTPUTS   :NONE
                   346 ;*CALLS     :CO
                   347 ;*DESTROYS  :HL, A, C
                   348 ;*
014C 7E            349 MESOUT: MOV     A, M
014D B7            350         ORA     A
014E C8            351         RZ
014F 4F            352         MOV     C, A
0150 CD2200    C   353         CALL    CO
0153 23            354         INX     H
0154 C34C01    C   355         JMP     MESOUT
                   356 ;
                   357 ;*******************************************************
                   358 ;
                   359         END

UBLIC SYMBOLS
ORD    C 0120    CI      C 0000    CO      C 0022    CROUT   C 00BA
CHO    C 00BC    GETCH   C 00D5    GETNM   C 00DC    MESOUT  C 014C
MOUT   C 0125    PTSER   C 0140    SPCOUT  C 0020    TBLSER  C 013A

XTERNAL SYMBOLS
ISBUF  E 0000    DISCLR  E 0000    DISCNT  E 0000    ERROR   E 0000
ETCM   E 0000    GETHX   E 0000    HILO    E 0000    KEYBUF  E 0000
SAVE   E 0000    MSTAK   E 0000    PSAVE   E 0000    SSAVE   E 0000
EMP    E 0000    USART1  E 0000

SER SYMBOLS
ORD    C 0120    CI      C 0000    CICRT   C 0006    CIKEY   C 0010
O      C 0022    COO1    C 0042    COO2    C 0058    COO3    C 004D
OCRT   C 0028    CODIS   C 0033    CROUT   C 00BA    DISBUF  E 0000
ISCLR  E 0000    DISCNT  E 0000    ECHO5   C 00C5    ECH10   C 00D3
CHO    C 00BC    ERROR   E 0000    GETCH   C 00D5    GETCM   E 0000
ETHX   E 0000    GETNM   C 00DC    GNMO5   C 00E3    GNM10   C 00F8
NM15   C 0105    GNM20   C 010A    GNM25   C 0115    GNM30   C 0119
ILO    E 0000    KEYBUF  E 0000    LSAVE   E 0000    MESOUT  C 014C
STAK   E 0000    NMOUT   C 0125    PRVAL   C 012E    PSAVE   E 0000
```

TFIN	C 014A	PTSER	C 0140	SPCOUT	C 0020	SSAVE	E 0000
ASBIN	C 007A	TBLSER	C 013A	TEMP	E 0000	USART1	E 0000

ISSEMBLY COMPLETE, NO ERRORS

```
$M80 MON85E.001 NOOBJECT PRINT(:LP:)

$IS-II 8080/8085 MACRO ASSEMBLER, V2.0          MPP85E    PAGE     1

 LOC  OBJ          SEQ          SOURCE STATEMENT

                     1 $PAGEWIDTH(80)
                     2 ;*************************************************
                     3 ;**
                     4 ;**                  MPP   MONITOR  V1
                     5 ;**
                     6 ;** PROG.NO:200-09-04                DATE:   07.04.1984
                     7 ;** FILE   :MON85E.001               MODULE: MPP85E
                     8 ;** AUTHOR :E.DURMUS
                     9 ;**
                    10 ;*************************************************
                    11 ;
                    12           NAME    MPP85E
                    13 ;
                    14           PUBLIC  GETHX,CNVBN,HILO,RSTTF
                    15           PUBLIC  STHFO,STHLF,VALDG
                    16           PUBLIC  VALDL,FRET,SRET,BREAK
                    17 ;
                    18           EXTRN   USART1,GETCM,MSTAK
                    19           EXTRN   SSAVE,PSAVE,LSAVE,TEMP
                    20           EXTRN   ERROR,GETCH,ECHO
                    21 ;
                    22           CSEG
                    23 ;*************************************************
                    24 ;*
                    25 ;*PROCEDURE:CNVBN
                    26 ;*FUNCTION :CNVBN CONVERTS THE ASCII REPRESENTATION
                    27 ;*           OF A HEX CHARACTER INTO ITS CORRESPONDING
                    28 ;*           BINARY VALUE. CNVBN DOES NOT CHECK THE
                    29 ;*           VALIDITIY OF ITS INPUT.
                    30 ;*INPUTS   :(C)=ASCII CHARACTER '0'-'9' OR 'A'-'F'
                    31 ;*OUTPUTS  :(A)=0 TO F HEX
                    32 ;*CALLS    :NONE
                    33 ;*DESTROYS :A,F
                    34 ;*
0000 79             35 CNVBN:   MOV     A,C
0001 D630           36          SUI     030H
0003 FE0A           37          CPI     00AH
0005 F8             38          RM
0006 D607           39          SUI     007H
0008 C9             40          RET
                    41 ;
                    42 ;*************************************************
                    43 ;*
                    44 ;*PROCEDURE:FRET
                    45 ;*FUNCTION :FRET IS JUMPED TO BY ANY ROUTINE THAT
                    46 ;*           WISHES TO INDICATE FAILURE ON RETURN.
                    47 ;*           FRET SETS THE CARRY FALSE,DENOTING
                    48 ;*           FAILURE,AND THEN RETURNS TO THE CALLER
                    49 ;*           OF THE ROUTINE INVOKING FRET.
                    50 ;*INPUTS   :NONE
                    51 ;*OUTPUTS  :(CY-FLAG)=ALWAYS 0
                    52 ;*CALLS    :NONE
```

```
LOC  OBJ          SEQ          SOURCE STATEMENT

                    53 ;*DESTROYS :F
                    54 ;*
0009 A7             55 FRET:    ANA     A
000A C9             56          RET
                    57 ;
                    58 ;*****************************************************
                    59 ;*
                    60 ;*PROCEDURE:GETHX
                    61 ;*FUNCTION :GETHX ACCEPTS A STRING OF HEX DIGITS FROM
                    62 ;*           THE INPUT STREAM AND RETURNS THEIR VALUE
                    63 ;*           AS A 16 BIT BINARY INTEGER. IF MORE THAN
                    64 ;*           4 HEX DIGITS ARE ENTERED, ONLY THE LAST
                    65 ;*           4 ARE USED. THE NUMBER TERMINATES WHEN A
                    66 ;*           VALID DELIMITER IS ENCOUNTERED. THE
                    67 ;*           DELIMITER IS ALSO RETURNED AS AN OUTPUT
                    68 ;*           OF THE FUNCTION. ILLEGAL CHARACTERS
                    69 ;*           (NOT HEX DIGITS OR DELIMITERS) CAUSE AN
                    70 ;*           ERROR INDICATION. IF THE FIRST (VALID)
                    71 ;*           CHARACTER ENCOUNTERED IN THE INPUT
                    72 ;*           STREAM IS NOT A DELIMITER, GETHX WILL
                    73 ;*           RETURN WITH THE CARRY BIT SET TO 1;
                    74 ;*           OTHERWISE, THE CARRY BIT IS SET TO 0 AND
                    75 ;*           THE CONTENTS OF BC ARE UNDEFINED.
                    76 ;*INPUTS    :NONE
                    77 ;*OUTPUTS   :(BC)=16 BIT INTEGER
                    78 ;*           (D)=CHARACTER WHICH TERMINATED THE INTEGE
                    79 ;*           (CY-FLAG)=1 IF FIRST CHARACTER NOT DELIMI
                    80 ;*                    =0 IF FIRST CHARACTER IS  DELIMI
                    81 ;*CALLS     :GETCH, ECHO, VALDL, VALDG, CNVBN, ERROR
                    82 ;*DESTROYS :ALL
                    83 ;*
000B E5             84 GETHX:   PUSH    H
000C 210000         85          LXI     H,0
000F 5C             86          MOV     E,H
0010 CD0000    E    87 GHX05:   CALL    GETCH
0013 CD0000    E    88          CALL    ECHO
0016 CD9600    C    89          CALL    VALDL
0019 C22500    C    90          JNZ     GHX10
001C 51             91          MOV     D,C
001D E5             92          PUSH    H
001E C1             93          POP     B
001F E1             94          POP     H
0020 7B             95          MOV     A,E
0021 B7             96          ORA     A
0022 C8             97          RZ
                    98 ;*****************************************************
                    99 ;*
                   100 ;*PROCEDURE:SRET
                   101 ;*FUNCTION :SRET IS JUMPED TO BY ROUTINES WISHING TO
                   102 ;*           RETURN SUCCESS. SRET SETS THE CARRY TRUE
                   103 ;*           AND RETURNS TO THE CALLER OF THE ROUTINE
                   104 ;*           CALLER OF THE ROUTINE INVOKING SRET.
                   105 ;*INPUTS    :NONE
                   106 ;*OUTPUTS   :(CY-FLAG)=1
                   107 ;*CALLS     :NONE
```

```
LOC  OBJ          SEQ         SOURCE STATEMENT

                  108 ;*DESTROYS :F
                  109 ;*
                  110 SRET:
0023 37           111         STC             ;SET CARRY TRUE
0024 C9           112         RET             ;RETURN APPROPRIATELY
                  113 ;
                  114 ;*******************************************************
                  115 ;
0025 CD8300    C  116 GHX10:  CALL    VALDG
0028 D20000    E  117         JNC     ERROR
002B CD0000    C  118         CALL    CNVBN
002E 1EFF         119         MVI     E,0FFH
0030 29           120         DAD     H
0031 29           121         DAD     H
0032 29           122         DAD     H
0033 29           123         DAD     H
0034 0600         124         MVI     B,000H
0036 4F           125         MOV     C,A
0037 09           126         DAD     B
0038 C31000    C  127         JMP     GHX05
                  128 ;
                  129 ;*******************************************************
                  130 ;*
                  131 ;*PROCEDURE:HILO
                  132 ;*FUNCTION :HILO COMPARES THE 2 16 BIT INTEGERS IN
                  133 ;*          HL AND DE. THE INTEGERS ARE TREATED AS
                  134 ;*          UNSIGNED NUMBERS. THE CARRY BIT IS SET
                  135 ;*          ACCORDING TO THE RESULT OF THE
                  136 ;*          COMPARISION
                  137 ;*INPUTS   :(DE)=16 BIT INTEGER
                  138 ;*          (HL)=16 BIT INTEGER
                  139 ;*OUTPUTS  :(CY-FLAG)=0 IF HL<DE
                  140 ;*                    =1 IF HL>=DE
                  141 ;*CALLS    :NONE
                  142 ;*DESTROYS :A,F
                  143 ;*
003B C5           144 HILO:   PUSH    B
003C 47           145         MOV     B,A
003D 23           146         INX     H
003E 7C           147         MOV     A,H
003F B5           148         ORA     L
0040 2B           149         DCX     H
0041 37           150         STC
0042 CA4A00    C  151         JZ      HIL05
0045 7D           152         MOV     A,L
0046 93           153         SUB     E
0047 7C           154         MOV     A,H
0048 9A           155         SBB     D
0049 3F           156         CMC
004A 78           157 HIL05:  MOV     A,B
004B C1           158         POP     B
004C C9           159         RET
                  160 ;
                  161 ;*******************************************************
                  162 ;*
```

```
LOC  OBJ          SEQ        SOURCE STATEMENT

                   163 ;*PROCEDURE:RSTTF
                   164 ;*FUNCTION :RSTTF RESTORES ALL CPU REGISTER,FLIP/FLOP,
                   165 ;*          STACK POINTER AND PROGRAM COUNTER FROM
                   166 ;*          THEIR RESPECTIVE SAVE LOCATIONS IN MEMORY.
                   167 ;*          THE ROUTINE THEN TRANSFERS CONTROL TO THE
                   168 ;*          LOCATION SPECIFIED BY THE PROGRAM COUNTER
                   169 ;*          (I.E. THE RESTORED VALUE). THE ROUTINE
                   170 ;*          EXITS WITH THE INTERRUPTS ENABLED.
                   171 ;*INPUTS   :NONE
                   172 ;*OUTPUTS  :NONE
                   173 ;*CALLS    :NONE
                   174 ;*DESTROYS :ALL
                   175 ;*
                   176 RSTTF:
004D F3            177         DI              ;DISABLE INTERRUPTS WHILE
                   178                         ;/RESTORING THINGS
004E 310000    E   179         LXI     SP,MSTAK
0051 D1            180         POP     D
0052 C1            181         POP     B
0053 F1            182         POP     PSW
0054 2A0000    E   183         LHLD    SSAVE
0057 F9            184         SPHL
0058 2A0000    E   185         LHLD    PSAVE
005B E5            186         PUSH    H
005C 2A0000    E   187         LHLD    LSAVE
005F FB            188         EI
0060 C9            189         RET
                   190 ;
                   191 ;*****************************************************
                   192 ;*
                   193 ;*PROCEDURE:STHFO
                   194 ;*FUNCTION :STHFO CHECKS THE HALF BYTE FLAG IN TEMP T
                   195 ;*          SEE IF IT IS SET TO LOWER.
                   196 ;*          IF SO STHFO STORES A 0 TO PAD OUT THE
                   197 ;*          LOWER HALF OF THE ADDRESSED BYTE;
                   198 ;*          OTHERWISE,THE ROUTINE TAKES NO ACTION.
                   199 ;*INPUTS   :(DE)=16 BIT ADDRESS OF BYTE TO BE
                   200 ;*                  STORED INTO
                   201 ;*OUTPUTS  :NONE
                   202 ;*CALLS    :NONE
                   203 ;*DESTROYS :ALL
                   204 ;*
0061 3A0000    E   205 STHFO:  LDA     TEMP
0064 B7            206         ORA     A
0065 C0            207         RNZ
0066 4F            208         MOV     C,A
                   209 ;*****************************************************
                   210 ;*
                   211 ;*PROCEDURE:STHLF
                   212 ;*FUNCTION :STHLF TAKES THE 4 BIT VALUE IN C AND STO
                   213 ;*          IT IN HALF OF THE BYTE ADDRESSED BY REG.
                   214 ;*          DE.THE HALF BYTE USED(EITHER UPPER OR LO
                   215 ;*          IS DENOTED BY THE VALUE OF THE FLAG IN T
                   216 ;*          STHLF ASSUMES THAT THIS FLAG HAS BEEN
                   217 ;*          PREVIOUSLY SET (NOMINALLY BY ICMD).
```

```
LOC  OBJ          SEQ        SOURCE STATEMENT

                   218 :*INPUTS   :(C)=4 BIT VALUE TO BE STORED IN HALF BYTE
                   219 :*          (DE)=16 BIT ADR.OF BYTE TO BE STORED INTO
                   220 :*OUTPUTS  :NONE
                   221 :*CALLS    :NONE
                   222 :*DESTROYS :ALL
                   223 :*
0067 79            224 STHLF:  MOV     A,C
0068 E60F          225         ANI     00FH
006A 4F            226         MOV     C,A
006B 3A0000    E   227         LDA     TEMP
006E B7            228         ORA     A
006F 1A            229         LDAX    D
0070 C27800    C   230         JNZ     STH05
0073 E6F0          231         ANI     0F0H
0075 B1            232         ORA     C
0076 12            233         STAX    D
0077 C9            234         RET
0078 E60F          235 STH05:  ANI     00FH
007A 47            236         MOV     B,A
007B 79            237         MOV     A,C
007C 0F            238         RRC
007D 0F            239         RRC
007E 0F            240         RRC
007F 0F            241         RRC
0080 B0            242         ORA     B
0081 12            243         STAX    D
0082 C9            244         RET
                   245 :
                   246 :*******************************************************
                   247 :*
                   248 :*PROCEDURE:VALDG
                   249 :*FUNCTION :VALDG RETURNS SUCCESS IF ITS INPUT ARGUME
                   250 :*          IS AN ASCII CHARACTER REPRESENTING A VALI
                   251 :*          HEX DIGIT (0-9,A,F), AND FAILURE OTHERWISE
                   252 :*INPUTS   :(C)=ASCII CHARACTER
                   253 :*OUTPUTS  :(CY-FLAG)=1  IF CHARACTER REPRESENTES VALI
                   254 :*                       HEX DIGIT
                   255 :*                   =0 OTHERWISE
                   256 :*CALLS    :NONE
                   257 :*DESTROYS :A,F
                   258 :*
0083 79            259 VALDG:  MOV     A,C
0084 FE30          260         CPI     '0'
0086 FA0900    C   261         JM      FRET
0089 FE3A          262         CPI     '9'+1
008B FA2300    C   263         JM      SRET
008E FE41          264         CPI     'A'
0090 FA0900    C   265         JM      FRET
0093 FE47          266         CPI     'G'
0095 C9            267         RET
                   268 :
                   269 :*******************************************************
                   270 :*
                   271 :*PROCEDURE:VALDL
                   272 :*FUNCTION :VALDL RETURNS SUCCESS IF ITS INPUT ARGUM
```

```
 LOC  OBJ          SEQ          SOURCE STATEMENT

                    273 ;*          IS A VALID DELIMITER CHARACTER (SPACE,
                    274 ;*          COMMA,CARRIAGE RETURN) AND FAILURE
                    275 ;*          OTHERWISE.
                    276 ;*INPUTS    :(C)=CHARACTER
                    277 ;*OUTPUTS   :(CY-FLAG)=1 IF INPUT ARGUM.VALID DELIMITE
                    278 ;*                      =0 OTHERWISE
                    279 ;*CALLS     :NONE
                    280 ;*DESTROYS :A,F
                    281 ;*
 0096 79            282 VALDL:  MOV     A,C
 0097 FE2C          283         CPI     ','
 0099 C8            284         RZ
 009A FE0D          285         CPI     0ODH
 009C C8            286         RZ
 009D FE20          287         CPI     ' '
 009F C9            288         RET
                    289 ;
                    290 ;*******************************************************
                    291 ;
 00A0 C9            292 BREAK:  RET
                    293 ;
                    294         END

PUBLIC SYMBOLS
BREAK   C 00A0    CNVBN   C 0000    FRET    C 0009    GETHX   C 000B
HILO    C 003B    RSTTF   C 004D    SRET    C 0023    STHFO   C 0061
STHLF   C 0067    VALDG   C 0083    VALDL   C 0096

EXTERNAL SYMBOLS
ECHO    E 0000    ERROR   E 0000    GETCH   E 0000    GETCM   E 0000
LSAVE   E 0000    MSTAK   E 0000    PSAVE   E 0000    SSAVE   E 0000
TEMP    E 0000    USART1  E 0000

USER SYMBOLS
BREAK   C 00A0    CNVBN   C 0000    ECHO    E 0000    ERROR   E 0000
FRET    C 0009    GETCH   E 0000    GETCM   E 0000    GETHX   C 000B
GHX05   C 0010    GHX10   C 0025    HILO5   C 004A    HILO    C 003B
LSAVE   E 0000    MSTAK   E 0000    PSAVE   E 0000    RSTTF   C 004D
SRET    C 0023    SSAVE   E 0000    STH05   C 0078    STHFO   C 0061
STHLF   C 0067    TEMP    E 0000    USART1  E 0000    VALDG   C 0083
VALDL   C 0096

ASSEMBLY COMPLETE.  NO ERRORS
```

```
3M80 MON85G.001 NOOBJECT PRINT(:LP:)

3IS-II 8080/8085 MACRO ASSEMBLER. V2.0          MPP85G    PAGE     1

LOC  OBJ           SEQ          SOURCE STATEMENT

                     1 $PAGEWIDTH(80)
                     2 :*****************************************************
                     3 :*                                                   *
                     4 :*              MPP MONITOR V1                       *
                     5 :*                                                   *
                     6 :*            DISPLAY REFRESH                        *
                     7 :*                                                   *
                     8 :* PROG NO:                    DATE:06.5.1984       *
                     9 :* FILE    :MON85G.001         MODULE:MPP85G        *
                    10 :* AUTHOR :E.D                                       *
                    11 :*****************************************************
                    12 :
                    13          NAME    MPP85G
                    14 :
                    15          PUBLIC  DISREF,DISCLR,DISBUF,DISCNT,DCLCNT
                    16 :
                    17          EXTRN   PORTB2,PORTC2
                    18 :
                    19          DSEG
                    20 :
0001                21 DCLCNT:  DS      1
0002                22 DRWPTR:  DS      2
0010                23 DISBUF:  DS      16
0001                24 DISCNT:  DS      1
                    25 :
                    26          CSEG
                    27 :*****************************************************
                    28 DISREF:
0000 2A0100     D   29          LHLD    DRWPTR
0003 23             30          INX     H
0004 220100     D   31          SHLD    DRWPTR
0007 3A0000     D   32          LDA     DCLCNT
000A 3C             33          INR     A
000B 320000     D   34          STA     DCLCNT
000E FE10           35          CPI     16
0010 C21D00     C   36          JNZ     DIS01
0013 AF             37          XRA     A
0014 320000     D   38          STA     DCLCNT
0017 210300     D   39          LXI     H,DISBUF
001A 220100     D   40          SHLD    DRWPTR
                    41 DIS01:
001D AF             42          XRA     A
001E D300       E   43          OUT     LOW PORTB2
0020 57             44          MOV     D,A
0021 3A0000     D   45          LDA     DCLCNT
0024 5F             46          MOV     E,A
0025 213300     C   47          LXI     H,DIGCOL
0028 19             48          DAD     D
0029 7E             49          MOV     A,M
002A D300       E   50          OUT     LOW PORTC2
002C 2A0100     D   51          LHLD    DRWPTR
002F 7E             52          MOV     A,M
```

```
LOC  OBJ           SEQ          SOURCE STATEMENT

0030 D300      E    53           OUT     LOW PORTB2
0032 C9             54           RET
                    55 ;*************************************************
                    56 ;
0033 10             57 DIGCOL: DB       10H, 11H, 12H, 13H, 14H, 15H, 16H, 17H
0034 11
0035 12
0036 13
0037 14
0038 15
0039 16
003A 17
003B 08             58           DB     08H, 09H, 0AH, 0BH, 0CH, 0DH, 0EH, 0FH
003C 09
003D 0A
003E 0B
003F 0C
0040 0D
0041 0E
0042 0F
                    59 ;*************************************************
                    60 ;
                    61 DISCLR:
0043 0E10           62           MVI     C, 16
0045 AF             63           XRA     A
0046 210300    D    64           LXI     H, DISBUF
0049 77             65 DISC01: MOV      M, A
004A 23             66           INX     H
004B 0D             67           DCR     C
004C C24900    C    68           JNZ     DISC01
004F C9             69           RET
                    70 ;*************************************************
                    71 ;
                    72           END

UBLIC SYMBOLS
CLCNT D 0000    DISBUF D 0003    DISCLR C 0043    DISCNT D 0013
ISREF C 0000

XTERNAL SYMBOLS
ORTB2 E 0000    PORTC2 E 0000

SER SYMBOLS
CLCNT D 0000    DIGCOL C 0033    DIS01  C 001D    DISBUF D 0003
ISC01 C 0049    DISCLR C 0043    DISCNT D 0013    DISREF C 0000
RWPTR D 0001    PORTB2 E 0000    PORTC2 E 0000

SSEMBLY COMPLETE.  NO ERRORS
```

```
ЗМ80 MON85H.001 NOOBJECT PRINT(:LP:)

ЗIS-II 8080/8085 MACRO ASSEMBLER, V2.0          MPP85H    PAGE     1

LOC  OBJ           SEQ          SOURCE STATEMENT

                     1 $PAGEWIDTH(80)
                     2 :*********************************************
                     3 :*                                           *
                     4 :*              MPP MONITOR                  *
                     5 :*                                           *
                     6 :*            KEYBOARD  SCAN                 *
                     7 :*                                           *
                     8 :* PROG. NO:              DATE    :05.6.1984  *
                     9 :* FILE     :MON85H.001   MODULE :MPP85H      *
                    10 :* AUTHOR   :E.D                             *
                    11 :*********************************************
                    12 :
                    13          NAME    MPP85H
                    14 :
                    15          PUBLIC  KEYBOR,KEYBUF,KEYFLG
                    16 :
                    17          EXTRN   PORTA2,DCLCNT
                    18 :
                    19          DSEG
                    20 :
0001                21 KEYFLG:  DS      1
0001                22 KEYROW:  DS      1
0001                23 KEYCOL:  DS      1
0001                24 KBNBUF:  DS      1
0001                25 KEYBUF:  DS      1
                    26 :
                    27          CSEG
                    28 :*********************************************
                    29 KEYBOR:
0000 3A0000    E    30          LDA     DCLCNT
0003 FE05           31          CPI     5
0005 F0             32          RP
                    33 :
0006 3A0000    D    34          LDA     KEYFLG
0009 FE00           35          CPI     0
000B C22800    C    36          JNZ     KEY30
                    37 :
                    38 :.......KEYFLG = 0.........................
                    39 :
000E DB00      E    40          IN      LOW PORTA2
0010 5F             41          MOV     E,A
0011 E63F           42          ANI     3FH
0013 FE3F           43          CPI     3FH
0015 C8             44          RZ
0016 7B             45          MOV     A,E
0017 320100    D    46          STA     KEYROW
001A 3A0000    E    47          LDA     DCLCNT
001D E607           48          ANI     07H
001F 320200    D    49          STA     KEYCOL
0022 210000    D    50          LXI     H,KEYFLG
0025 3601           51          MVI     M,1
0027 C9             52          RET
```

```
LOC  OBJ            SEQ          SOURCE STATEMENT

                     53 ;
                     54 KEY30:
0028 3A0000    E     55           LDA     DCLCNT
002B 210200    D     56           LXI     H,KEYCOL
002E BE              57           CMP     M
002F C0              58           RNZ
0030 3A0000    D     59           LDA     KEYFLG
0033 FE01            60           CPI     1
0035 C24D00    C     61           JNZ     KEY40
                     62 ;
                     63 ;......KEYFLG = 1 ..............................
                     64 ;
0038 CD6E00    C     65           CALL    KEYCHK
003B CA4400    C     66           JZ      KEY35
003E 210000    D     67           LXI     H,KEYFLG
0041 3600            68           MVI     M,0
0043 C9              69           RET
                     70 KEY35:
0044 CD7500    C     71           CALL    KCONV
0047 210000    D     72           LXI     H,KEYFLG
004A 3602            73           MVI     M,2
004C C9              74           RET
                     75 KEY40:
004D FE02            76           CPI     2
004F C25C00    C     77           JNZ     KEY45
                     78 ;
                     79 ;.......KEYFLG = 2.................................
                     80 ;
0052 CD6E00    C     81           CALL    KEYCHK
0055 C8              82           RZ
0056 210000    D     83           LXI     H,KEYFLG
0059 3603            84           MVI     M,3
005B C9              85           RET
                     86 KEY45:
005C CD6E00    C     87           CALL    KEYCHK
005F C26800    C     88           JNZ     KEY50
0062 210000    D     89           LXI     H,KEYFLG
0065 3602            90           MVI     M,2
0067 C9              91           RET
                     92 KEY50:
0068 210000    D     93           LXI     H,KEYFLG
006B 3600            94           MVI     M,0
006D C9              95           RET
                     96 ;*************************************************
                     97 KEYCHK:
006E DB00      E     98           IN      LOW PORTA2
0070 210100    D     99           LXI     H,KEYROW
0073 BE             100           CMP     M
0074 C9             101           RET
                    102 ;************************************************
                    103 KCONV:
0075 3A0200    D    104           LDA     KEYCOL
0078 07             105           RLC
0079 07             106           RLC
007A 07             107           RLC
```

```
LOC  OBJ          SEQ          SOURCE STATEMENT

007B 210300   D    108          LXI     H.KBNBUF
007E 77           109          MOV     M.A
007F 0EFF         110          MVI     C.0FFH
0081 3A0100   D    111          LDA     KEYROW
0084 0C           112 KCONO1:  INR     C
0085 0F           113          RRC
0086 DA8400   C    114          JC      KCONO1
0089 79           115          MOV     A.C
008A B6           116          ORA     M
008B 77           117          MOV     M.A
008C 3A0100   D    118          LDA     KEYROW
008F 07           119          RLC
0090 DA9900   C    120          JC      KCONO2
                   121 :....KEY MESS IS READY FOR ASCII CONVERSION.....
0093 21CC00   C    122          LXI     H.CONTBL+24H
0096 C39C00   C    123          JMP     KCONO3
0099 21A800   C    124 KCONO2:  LXI     H.CONTBL
009C 3A0300   D    125 KCONO3:  LDA     KBNBUF
009F 5F            126          MOV     E.A
00A0 1600          127          MVI     D.0
00A2 19            128          DAD     D
00A3 7E            129          MOV     A.M
00A4 320400   D    130          STA     KEYBUF
00A7 C9            131          RET
                   132 :***********************************************
00A8 49            133 CONTBL:  DB      49H.54H.41H.42H.43H.46H
00A9 54
00AA 41
00AB 42
00AC 43
00AD 46
00AE FF            134          DB      0FFH.0FFH.4CH.20H.33H.36H
00AF FF
00B0 4C
00B1 20
00B2 33
00B3 36
00B4 39            135          DB      39H.45H.0FFH.0FFH.4FH.0DH
00B5 45
00B6 FF
00B7 FF
00B8 4F
00B9 0D
00BA 32            136          DB      32H.35H.38H.30H.0FFH.0FFH
00BB 35
00BC 38
00BD 30
00BE FF
00BF FF
00C0 50            137          DB      50H.52H.31H.34H.37H.44H
00C1 52
00C2 31
00C3 34
00C4 37
00C5 44
```

```
LOC  OBJ          SEQ          SOURCE STATEMENT

00C6 FF            138          DB      0FFH,0FFH,47H,53H,0FFH,0FFH
00C7 FF
00C8 47
00C9 53
00CA FF
00CB FF
                   139 ;            SHIFT PAGE
00CC 48            140          DB      48H,55H,4AH,0FFH,0FFH,0FFH
00CD 55
00CE 4A
00CF FF
00D0 FF
00D1 FF
00D2 FF            141          DB      0FFH,0FFH,4AH,1BH,0FFH,0FFH
00D3 FF
00D4 4A
00D5 1B
00D6 FF
00D7 FF
00D8 FF            142          DB      0FFH,0FFH,0FFH,0FFH,4EH
00D9 FF
00DA FF
00DB FF
00DC 4E
                   143 ;
                   144 ;***************************************************
                   145          END

UBLIC SYMBOLS
EYBOR C 0000    KEYBUF D 0004    KEYFLG D 0000

XTERNAL SYMBOLS
CLCNT E 0000    PORTA2 E 0000

SER SYMBOLS
ONTBL C 00A8    DCLCNT E 0000    KBNBUF D 0003    KCONO1 C 0084
CONO2 C 0099    KCONO3 C 009C    KCONV  C 0075    KEY30  C 0028
EY35  C 0044    KEY40  C 004D    KEY45  C 005C    KEY50  C 0068
EYBOR C 0000    KEYBUF D 0004    KEYCHK C 006E    KEYCOL D 0002
EYFLG D 0000    KEYROW D 0001    PORTA2 E 0000

SSEMBLY COMPLETE.  NO ERRORS
```

```
480 MON85I.001 NOOBJECT PRINT(:LP:)

IS-II 8080/8085 MACRO ASSEMBLER. V2.0          MPP85I    PAGE     1

LOC  OBJ           SEQ          SOURCE STATEMENT

                     1 $PAGEWIDTH(80)
                     2 :*************************************************
                     3 :*                                               *
                     4 :*          MPP MONITOR                          *
                     5 :*                                               *
                     6 :*        FOREGROUND SHEDULAR                    *
                     7 :*                                               *
                     8 :* PROG NO:                      DATE:28.2.1983  *
                     9 :* FILE   :MON85I.001           MODULE:MPP85I    *
                    10 :* AUTHOR :E.D                  DEPT   :870      *
                    11 :*************************************************
                    12 :
                    13          NAME    MPP85I
                    14 :
                    15          PUBLIC  INTTIM
                    16 :
                    17          EXTRN   DISREF.KEYBOR
                    18 :
                    19          CSEG
                    20 :*************************************************
                    21 INTTIM:
0000 F5             22          PUSH    PSW
0001 C5             23          PUSH    B
0002 D5             24          PUSH    D
0003 E5             25          PUSH    H
0004 CD0000    E    26          CALL    DISREF
0007 CD0000    E    27          CALL    KEYBOR
000A 3E10           28          MVI     A.10H
000C 30             29          SIM
(   0)
000D E1             30          POP     H
000E D1             31          POP     D
000F C1             32          POP     B
0010 F1             33          POP     PSW
0011 FB             34          EI
0012 C9             35          RET
                    36 :*************************************
                    37          END

UBLIC SYMBOLS
NTTIM C 0000

XTERNAL SYMBOLS
ISREF E 0000    KEYBOR E 0000

SER SYMBOLS
ISREF E 0000    INTTIM C 0000    KEYBOR E 0000

SSEMBLY COMPLETE.    1 ERROR (  29)
```

```
;M80 MPP00A.001 NOOBJECT PRINT(:LP:)

;IS-II 8080/8085 MACRO ASSEMBLER. V2.0          PPMAIN    PAGE     1

 LOC  OBJ          SEQ          SOURCE STATEMENT

                     1 $PAGEWIDTH(80)
                     2 :***********************************************
                     3 :*                                             *
                     4 :*       MULTIPURPOSE PROGRAMMER               *
                     5 :*           MAIN PROGRAM                      *
                     6 :*                                             *
                     7 :* PROG.NO:200-MM-00       DATE  :26.06.1984 *
                     8 :* FILE    :MPP00A.001     MODULE:PPMAIN      *
                     9 :* AUTHOR :E.DURMUS                           *
                    10 :*                                             *
                    11 :***********************************************
                    12 :
                    13           NAME    PPMAIN
                    14 :
                    15           PUBLIC  MPPMON.MSGFIN.ERRMSG.CR.LF
                    16           PUBLIC  PCMBUF.PDRBUF.PRTYPE.ROMBEG
                    17           PUBLIC  ROMEND.BUFBEG.BUFEND.PCOUNT
                    18           PUBLIC  PCTYPE.TYPE.EPEND.DISALL.ENAL
0C66                19           PUBLIC  ENAH.DDRIN.DDROUT.ENDATI.ENDATO
                    20           PUBLIC  ENCS.ENLED.CSMODE.PCSBUF
                    21 :
                    22           EXTRN   ECHO.MESOUT.GETTYP.GETCH.TBLSER
                    23           EXTRN   ACMND.BCMND.CCMND.ECMND.PCMND
                    24           EXTRN   SCMND.TCMND.WAIT
                    25 :
                    26 :***********************************************
                    27 :*
000D                28 CR        EQU     0DH
000A                29 LF        EQU     0AH
0008                30 DISALL    EQU     08H
0009                31 ENAL      EQU     09H
000A                32 ENAH      EQU     0AH
0008                33 DDRIN     EQU     08H
000C                34 DDROUT    EQU     0CH
0000                35 ENDATI    EQU     00H
0004                36 ENDATO    EQU     04H
0018                37 ENCS      EQU     18H
0028                38 ENLED     EQU     28H
                    39 :
                    40 :***********************************************
                    41 :
                    42 :         RAM DEFINITIONS
                    43 :
                    44           DSEG
                    45 :
0002                46 BUFBEG:   DS      2
0002                47 BUFEND:   DS      2
0002                48 ROMBEG:   DS      2
0002                49 ROMEND:   DS      2
0002                50 PDRBUF:   DS      2
0001                51 TYPE:     DS      1
0002                52 PCSBUF:   DS      2
```

```
LOC  OBJ            SEQ         SOURCE STATEMENT

0006                 53 PCMBUF: DS      6
0001                 54 PRTYPE: DS      1
0001                 55 PCTYPE: DS      1
0001                 56 PCOUNT: DS      1
0001                 57 EPEND:  DS      1
0001                 58 CSMODE: DS      1
                     59 ;
                     60         CSEG
                     61 ;*********************************************
                     62 ;*
                     63 ;* PROCEDURE:MPPMON
                     64 ;* FUNCTHON :MPP MAIN PROGRAN
                     65 ;* INPUTS   :NONE
                     66 ;* OUTPUTS  :NONE
                     67 ;* CALLS    :ECHO.MESOUT.GETTYP.GETCH.TBLSER
                     68 ;* DESTROYS :ALL
                     69 ;
0000 216100    C     70 MPPMON: LXI     H.UPPMSG :DISPLAY UPP
0003 CD0000    E     71         CALL    MESOUT
0006 CD0000    E     72         CALL    WAIT     :WAIT UNTIL CR IS PRESSED
0009 0E3E            73         MVI     C.'>'    :DISPLAY PROMPT MSG.
000B CD0000    E     74         CALL    ECHO
000E 214100    C     75         LXI     H.TYPMSG :DISPLAY TYPE MSG
0011 CD0000    E     76         CALL    MESOUT
0014 CD0000    E     77         CALL    GETTYP   :GET EPROM TYPE AND SET FLAGS
0017 214700    C     78 LOOP1:  LXI     H.MODMSG :DISPLAY MODE MSG
001A CD0000    E     79         CALL    MESOUT
001D CD0000    E     80         CALL    GETCH    :GET OPERATION MODE
0020 CD0000    E     81         CALL    ECHO
0023 217100    C     82         LXI     H.PCTBL :RESARCH INPUT MODE IS VALID
0026 CD0000    E     83         CALL    TBLSER
0029 DA3800    C     84         JC      ERROR    :IF NOT FOUND DISPLAY ERROR M
002C 7E              85         MOV     A.M
002D 23              86         INX     H        :ELSE JUMP TO OP MODE
002E 66              87         MOV     H.M
002F 6F              88         MOV     L.A
0030 013500    C     89         LXI     B.RETADR
0033 C5              90         PUSH    B
0034 E9              91         PCHL
0035 D21700    C     92 RETADR: JNC     LOOP1
0038 214E00    C     93 ERROR:  LXI     H.ERRMSG
003B CD0000    E     94         CALL    MESOUT
003E C31700    C     95         JMP     LOOP1
                     96 ;
0041 54595045        97 TYPMSG: DB      'TYPE='.0
0045 3D
0046 00
0047 2046554E        98 MODMSG: DB      ' FUNC='.0
004B 433D
004D 00
004E 0D              99 ERRMSG: DB      CR.LF.'ERROR !'.0
004F 0A
0050 4552524F
0054 522021
0057 00
```

```
LOC  OBJ          SEQ         SOURCE STATEMENT

0058 0D            100 MSGFIN: DB      CR,LF,'FINISH',0
0059 0A
005A 46494E49
005E 5348
0060 00
0061 0D            101 UPPMSG: DB      CR,LF,' UPP IS READY',0
0062 0A
0063 20555050
0067 20495320
006B 52454144
006F 59
0070 00
                   102 ;
                   103 ;*****************************************************
                       **********
                   104 ;*
                   105 ;*TABLE    :PCTBL
                   106 ;*FUNCTION:PROGRAMMER COMMAND TABLE
                   107 ;*
0071 08            108 PCTBL:  DB      8           :NUMBER OF VALID COMMANDS
0072 0300          109         DW      3           :NUMBER OF BYTES PER ENTRY
0074 52            110         DB      'R'         :EXIT TO MONITOR
0075 8C00      C   111         DW      RCMND
0077 50            112         DB      'P'         :PROGRAM BUFFER TO EPROMS
0078 0000      E   113         DW      PCMND
007A 41            114         DB      'A'         :PROGRAM MASTER TO SLAVES
007B 0000      E   115         DW      ACMND
007D 43            116         DB      'C'         :VERIFY EPROMS WIA BUFFER
007E 0000      E   117         DW      CCMND
0080 54            118         DB      'T'         :TRANSFER EPROMS TO BUFFER
0081 0000      E   119         DW      TCMND
0083 45            120         DB      'E'         :CHECK EPROM ERASED OR NOT
0084 0000      E   121         DW      ECMND
0086 53            122         DB      'S'         :CALCULATE CHECKSUM OF EPROMS
0087 0000      E   123         DW      SCMND
0089 42            124         DB      'B'         :VERIFY MASTER WITH SLAVES
008A 0000      E   125         DW      BCMND
                   126 ;
                   127 ;*****************************************************
                   128 ;*
                   129 ;*PROCEDURE:RCMND
                   130 ;*FUNCTION :EXITS MPP AND RETURN TO MONITOR
                   131 ;*
008C C1            132 RCMND:  POP     B           :PULL RETURN ADR OF PPMAIN
008D C9            133         RET                 :RETURN TO MONITOR
                   134 ;
                   135 ;*****************************************************
                   136 ;
                   137         END

PUBLIC SYMBOLS
BUFBEG D 0000    BUFEND D 0002    CR     A 000D    CSMODE D 0017
DDRIN  A 0008    DDROUT A 000C    DISALL A 0008    ENAH   A 000A
ENAL   A 0009    ENCS   A 0018    ENDATI A 0000    ENDATO A 0004
ENLED  A 0028    EPEND  D 0016    ERRMSG C 004E    LF     A 000A
```

```
PPMON  C 0000    MSGFIN C 0058    PCMBUF D 000D    PCOUNT D 0015
PCSBUF D 000B    PCTYPE D 0014    PDRBUF D 0008    PRTYPE D 0013
ROMBEG D 0004    ROMEND D 0006    TYPE   D 000A

EXTERNAL SYMBOLS
ACMND  E 0000    BCMND  E 0000    CCMND  E 0000    ECHO   E 0000
ECMND  E 0000    GETCH  E 0000    GETTYP E 0000    MESOUT E 0000
PCMND  E 0000    SCMND  E 0000    TBLSER E 0000    TCMND  E 0000
WAIT   E 0000

USER SYMBOLS
ACMND  E 0000    BCMND  E 0000    BUFBEG D 0000    BUFEND D 0002
CCMND  E 0000    CR     A 000D    CSMODE D 0017    DDRIN  A 0008
DDROUT A 000C    DISALL A 0008    ECHO   E 0000    ECMND  E 0000
ENAH   A 000A    ENAL   A 0009    ENCS   A 0018    ENDATI A 0000
ENDATO A 0004    ENLED  A 0028    EPEND  D 0016    ERRMSG C 004E
ERROR  C 0038    GETCH  E 0000    GETTYP E 0000    LF     A 000A
LOOP1  C 0017    MESOUT E 0000    MODMSG C 0047    MPPMON C 0000
MSGFIN C 0058    PCMBUF D 000D    PCMND  E 0000    PCOUNT D 0015
PCSBUF D 000B    PCTBL  C 0071    PCTYPE D 0014    PDRBUF D 0008
PRTYPE D 0013    RCMND  C 008C    RETADR C 0035    ROMBEG D 0004
ROMEND D 0006    SCMND  E 0000    TBLSER E 0000    TCMND  E 0000
TYPE   D 000A    TYPMSG C 0041    UPPMSG C 0061    WAIT   E 0000

ASSEMBLY COMPLETE.  NO ERRORS
```

```
M80 MPP01A.001 NOOBJECT PRINT(:LP:)

IS-II 8080/8085 MACRO ASSEMBLER. V2.0          UTMAIN    PAGE     1

LOC  OBJ          SEQ          SOURCE STATEMENT

                     1 $PAGEWIDTH(80)
                     2 :***********************************************
                     3 :*                                             *
                     4 :*        MULTIPURPOSE PROGRAMMER              *
                     5 :*          MAIN PROG. UTULITIES               *
                     6 :*                                             *
                     7 :* PROG.NO:200-MM-01       DATE  :26.06.1984 *
                     8 :* FILE    :MPP01A.001     MODULE:UTMAIN      *
                     9 :* AUTHOR  :E.DURMUS                          *
                    10 :*                                             *
                    11 :***********************************************
                    12 :
                    13          NAME    UTMAIN
                    14 :
                    15          PUBLIC  GETTYP,PGETSC,MODYCS,DEL1M
                    16          PUBLIC  PDRTBL,PCSTBL,MDYSOC
                    17 :
                    18          EXTRN   GETCH,ECHO,TYPE,MESOUT,EPEND
                    19          EXTRN   PDRBUF,PCSBUF,CSMODE,PCMBUF
                    20          EXTRN   PRTYPE,PCTYPE
                    21 :
                    22          CSEG
                    23 :***********************************************
                    24 :*
                    25 :*PROCEDURE:GETTYP
                    26 :*FUNCTION :ASK FOR THE EPROM TYPE AND SET FLAGS
                    27 :*INPUTS   :NONE
                    28 :*OUTPUTS  :NONE
                    29 :*CALLS    :GETCH,ECHO,MESOUT
                    30 :*DESTROYS :ALL
                    31 :*
0000 CD0000    E    32 GETTYP: CALL    GETCH
0003 CD0000    E    33         CALL    ECHO
0006 79             34         MOV     A,C
0007 320000    E    35         STA     TYPE
000A FE30           36         CPI     '0'
000C CA4A00    C    37         JZ      I27256  :THEN 27256 INTELLIGENT
000F FE31           38         CPI     '1'
0011 CA6600    C    39         JZ      N27128  :THEN 27128 NORMAL
0014 FE32           40         CPI     '2'
0016 CA7D00    C    41         JZ      I27128  :THEN 27128 INTELLIGENT
0019 FE33           42         CPI     '3'
001B CA9900    C    43         JZ      N2764   :THEN 2764 NORMAL
001E FE34           44         CPI     '4'
0020 CAB000    C    45         JZ      I2764   :THEN 2764 INTELLIGENT
0023 FE35           46         CPI     '5'
0025 CACC00    C    47         JZ      T2732A  :THEN 2732A
0028 FE36           48         CPI     '6'
002A CADE00    C    49         JZ      T2732   :THEN 2732
002D FE37           50         CPI     '7'
002F CAF000    C    51         JZ      T2716   :THEN 2716
0032 FE38           52         CPI     '8'
```

```
LOC  OBJ          SEQ          SOURCE STATEMENT

0034 CA0201   C    53           JZ      TM2716  :THEN MCM2716
0037 FE39          54           CPI     '9'
0039 CA1401   C    55           JZ      TM2532  :THEN MCM2532
003C FE41          56           CPI     'A'
003E CA2001   C    57           JZ      T68764  :THEN MCM68764
0041 21C301   C    58           LXI     H.TERMSG :IF NON THEN WRITE TYPE ERRO
0044 CD0000   E    59           CALL    MESOUT  :MESSAGE
0047 C30000   C    60           JMP     GETTYP  :TRY AGAIN
004A 3E80          61 I27256:   MVI     A,80H
004C 320000   E    62           STA     EPEND
004F 3EFF          63           MVI     A,0FFH
0051 320000   E    64           STA     PRTYPE
0054 3E19          65           MVI     A,25
0056 320000   E    66           STA     PCTYPE
0059 216901   C    67           LXI     H,PDRTBL
005C 220000   E    68           SHLD    PDRBUF
005F 219F01   C    69           LXI     H,PCSTBL
0062 220000   E    70           SHLD    PCSBUF
0065 C9            71           RET
                   72 ;
0066 3E40          73 N27128:   MVI     A,40H   :TYPE IS 40H FOR 27128
0068 320000   E    74           STA     EPEND   :SAVE TYPE
006B 3E00          75           MVI     A,0
006D 320000   E    76           STA     PRTYPE
0070 216F01   C    77           LXI     H,PDRTBL+6 :GET RELATED PROM DRIVER T
                        E
0073 220000   E    78           SHLD    PDRBUF  :ADR. STORE IT ON PDRBUF
0076 21A501   C    79           LXI     H,PCSTBL+6 :(DE)=CORRES. CS TABLE POI
                        R
0079 220000   E    80           SHLD    PCSBUF
007C C9            81           RET
                   82 ;
007D 3E40          83 I27128:   MVI     A,40H
007F 320000   E    84           STA     EPEND
0082 3EFF          85           MVI     A,0FFH
0084 320000   E    86           STA     PRTYPE
0087 3E0F          87           MVI     A,15
0089 320000   E    88           STA     PCTYPE
008C 217501   C    89           LXI     H,PDRTBL+12
008F 220000   E    90           SHLD    PDRBUF
0092 21AB01   C    91           LXI     H,PCSTBL+12
0095 220000   E    92           SHLD    PCSBUF
0098 C9            93           RET
                   94 ;
0099 3E20          95 N2764:    MVI     A,20H
009B 320000   E    96           STA     EPEND
009E 3E00          97           MVI     A,0
00A0 320000   E    98           STA     PRTYPE
00A3 216F01   C    99           LXI     H,PDRTBL+6
00A6 220000   E   100           SHLD    PDRBUF
00A9 21A501   C   101           LXI     H,PCSTBL+6
00AC 220000   E   102           SHLD    PCSBUF
00AF C9           103           RET
                  104 ;
00B0 3E20         105 I2764:    MVI     A,20H   :    "      "      "
```

```
LOC  OBJ          SEQ          SOURCE STATEMENT

00B2 320000   E    106          STA     EPEND
00B5 3EFF          107          MVI     A,0FFH
00B7 320000   E    108          STA     PRTYPE
00BA 3E0F          109          MVI     A,15
00BC 320000   E    110          STA     PCTYPE
00BF 217501   C    111          LXI     H,PDRTBL+12
00C2 220000   E    112          SHLD    PDRBUF
00C5 21AB01   C    113          LXI     H,PCSTBL+12
00C8 220000   E    114          SHLD    PCSBUF
00CB C9            115          RET
                   116 ;
00CC 3E10          117 T2732A:  MVI     A,10H
00CE 320000   E    118          STA     EPEND
00D1 217B01   C    119          LXI     H,PDRTBL+18
00D4 220000   E    120          SHLD    PDRBUF
00D7 21B101   C    121          LXI     H,PCSTBL+18
00DA 220000   E    122          SHLD    PCSBUF
00DD C9            123          RET
                   124 ;
00DE 3E10          125 T2732:   MVI     A,10H    ;     "      "      "
00E0 320000   E    126          STA     EPEND
00E3 218101   C    127          LXI     H,PDRTBL+24
00E6 220000   E    128          SHLD    PDRBUF
00E9 21B701   C    129          LXI     H,PCSTBL+24
00EC 220000   E    130          SHLD    PCSBUF
00EF C9            131          RET
                   132 ;
00F0 3E08          133 T2716:   MVI     A,08H    ;     "      "      "
00F2 320000   E    134          STA     EPEND
00F5 218701   C    135          LXI     H,PDRTBL+30
00F8 220000   E    136          SHLD    PDRBUF
00FB 21BD01   C    137          LXI     H,PCSTBL+30
00FE 220000   E    138          SHLD    PCSBUF
0101 C9            139          RET
                   140 ;
0102 3E08          141 TM2716:  MVI     A,08H    ;     "      "      "
0104 320000   E    142          STA     EPEND
0107 218D01   C    143          LXI     H,PDRTBL+36
010A 220000   E    144          SHLD    PDRBUF
010D 21C301   C    145          LXI     H,PCSTBL+36
0110 220000   E    146          SHLD    PCSBUF
0113 C9            147          RET
                   148 ;
0114 3E10          149 TM2532:  MVI     A,10H    ;     "      "      "
0116 320000   E    150          STA     EPEND
0119 219301   C    151          LXI     H,PDRTBL+42
011C 220000   E    152          SHLD    PDRBUF
011F C9            153          RET
                   154 ;
0120 3E20          155 T68764:  MVI     A,20H    ;     "      "      "
0122 320000   E    156          STA     EPEND
0125 219901   C    157          LXI     H,PDRTBL+48
0128 220000   E    158          SHLD    PDRBUF
012B C9            159          RET
                   160 ;
```

```
LOC  OBJ          SEQ          SOURCE STATEMENT

                   161 :*****************************************************
                   162 :*
                   163 :*PROCEDURE:PGETSC
                   164 :*FUNCTION :GETS THAT WHICH SOCETS ARE  PROGRAMMED
                   165 :*INPUTS   :NONE
                   166 :*OUTPUTS  :NONE
                   167 :*CALLS    :GETCH. ECHO. MODYCS
                   168 :*DESTROYS :ALL
                   169 :
012C CD0000    E   170 PGETSC: CALL    GETCH
012F CD0000    E   171         CALL    ECHO
0132 79            172         MOV     A. C
0133 320000    E   173         STA     CSMODE
0136 214001    C   174         LXI     H. PGETBL
0139 CD4501    C   175         CALL    MDYSOC
013C CD4D01    C   176         CALL    MODYCS
013F C9            177         RET
                   178 :
0140 FE            179 PGETBL: DB      OFEH. OFCH. OF8H. OFOH. OEOH
0141 FC
0142 F8
0143 F0
0144 E0
                   180 :
0145 D630          181 MDYSOC: SUI     30H
0147 5F            182         MOV     E. A
0148 1600          183         MVI     D. 0
014A 19            184         DAD     D
014B 7E            185         MOV     A. M
014C C9            186         RET
                   187
                   188 :*****************************************************
                   189 :*
                   190 :*PROCEDURE:MODYCS
                   191 :*FUNCTION :CREATES A CS BUFFER NAMED AS PCMBUF FROM
                   192 :*          PCSBUF(CREATED BY EPROM TYPE) AND SOCHET
                   193 :*          INFORMATION: PCSBUF(6) & CSMASK
                   194 :*INPUTS   :PCSBUF(6) . (A)=CSMASK
                   195 :*OUTPUTS  :PCMBUF(6)
                   196 :*CALLS    :NONE
                   197 :*DESTROYS :A. C. HL. DE
                   198 :*
                   199 :
014D 47            200 MODYCS: MOV     B. A
014E 0E06          201         MVI     C. 6
0150 110000    E   202         LXI     D. PCMBUF
0153 2A0000    E   203         LHLD    PCSBUF
0156 7E            204 MODY01: MOV     A. M
0157 B0            205         ORA     B
0158 EB            206         XCHG
0159 77            207         MOV     M. A
015A 23            208         INX     H
015B 13            209         INX     D
015C EB            210         XCHG
```

```
LOC  OBJ          SEQ          SOURCE STATEMENT

015D 0D            211          DCR     C
015E C25601    C   212          JNZ     MODY01
0161 C9            213          RET
                   214 ;
                   215 ;*******************************************************
                   216 ;*
                   217 ;*PROCEDURE:DEL1M
                   218 ;*FUNCTION :PROVIDES 1MSEC DELAY
                   219 ;*INPUTS   :NONE
                   220 ;*OUTPUTS  :NONE
                   221 ;*CALLS    :NONE
                   222 ;*DESTROYS :A
                   223 ;*
0162 3EB2          224 DEL1M:  MVI     A,0B2H
0164 3D            225 LOOP1:  DCR     A
0165 C26401    C   226          JNZ     LOOP1
0168 C9            227          RET
                   228 ;
                   229 ;*******************************************************
                       ***
                   230 ;*
                   231 ;*TABLE   :PDRTBL
                   232 ;*FUNCTION:PROM DRIVER BIT DEFINITIONS
                   233 ;*ENTRIES :6 ENTRIES PER PROM TYPE
                   234 ;*              1 :READ STANDBY
                   235 ;*              2 :READ
                   236 ;*              3 :PROGRAM STANDBY
                   237 ;*              4 :PROGRAM PREPARATION
                   238 ;*              5 :PROGRAM
                   239 ;*              6 :PROGRAM VERIFY
                   240 ;*
0169 58            241 PDRTBL: DB  58H,18H,50H,41H,41H,01H      ;27256(INT)

016A 18
016B 50
016C 41
016D 41
016E 01
016F D0            242          DB  0D0H,90H,0D0H,0D2H,52H,92H   ;27128,2764(
                       )

0170 90
0171 D0
0172 D2
0173 52
0174 92
0175 D0            243          DB  0D0H,90H,0D0H,0C2H,42H,82H   ;27128,2764
                       )

0176 90
0177 D0
0178 C2
0179 42
017A 82
017B 10            244          DB  10H,18H,18H,12H,12H,18H      ;2732A
017C 18
017D 18
```

```
LOC  OBJ          SEQ          SOURCE STATEMENT

017E 12
017F 12
0180 18
0181 10           245          DB   10H.18H.18H.14H.14H.18H      :2732
0182 18
0183 18
0184 14
0185 14
0186 18
0187 50           246          DB   50H.10H.50H.54H.54H.14H      :2716
0188 10
0189 50
018A 54
018B 54
018C 14
018D 50           247          DB   50H.10H.50H.54H.54H.14H      :MCM2716
018E 10
018F 50
0190 54
0191 54
0192 14
0193 50           248          DB   50H.10H.50H.54H.14H.10H      :MCM2532
0194 10
0195 50
0196 54
0197 14
0198 10
0199 10           249          DB   10H.18H.18H.10H.14H.18H      :MCM68764
019A 18
019B 18
019C 10
019D 14
019E 18
                  250 :
                  251 :*****************************************************
                  252 :*
                  253 :*TABLE :PCSTBL
                  254 :*FUNCTION: PROM DRIVER CS TABLE
                  255 :*ENTRIES :6 ENTRIES PER PROM TYPE
                  256 :*          ALL ENTRIES SAME AS PDRTBL
                  257 :
019F FF           258 PCSTBL: DB       0FFH.00H.0FFH.0FFH.00H.0FFH   :27256(I
01A0 00
01A1 FF
01A2 FF
01A3 00
01A4 FF
01A5 FF           259          DB      0FFH.00H.0FFH.00H.00H.00H     :27128.27
                      INT)
01A6 00
01A7 FF
01A8 00
01A9 00
01AA 00
01AB FF           260          DB      0FFH.00H.0FFH.00H.00H.00H     :27128.2
```

```
 LOC  OBJ          SEQ          SOURCE STATEMENT

 LOC  OBJ          SEQ  NOR)
 01AC 00
 01AD FF
 01AE 00
 01AF 00
 01B0 00
 01B1 FF           261          DB      OFFH. 00H. OFFH. OFFH. 00H. 00H    :2732A
 01B2 00
 01B3 FF
 01B4 FF
 01B5 00
 01B6 00
 01B7 FF           262          DB      OFFH. 00H. OFFH. OFFH. 00H. 00H    :2732
 01B8 00
 01B9 FF
 01BA FF
 01BB 00
 01BC 00
 01BD FF           263          DB      OFFH. 00H. OFFH. 00H. OFFH. 00H    :2716
 01BE 00
 01BF FF
 01C0 00
 01C1 FF
 01C2 00
                   264 :
                   265 :*****************************************************
                   266 :
 01C3 0D           267 TERMSG: DB      0DH. 0AH. 'FUNC=' . 0
 01C4 0A
 01C5 46554E43
 01C9 3D
 01CA 00
                   268 :
                   269 :*****************************************************
                   270 :
                   271          END

UBLIC SYMBOLS
EL1M  C 0162    GETTYP C 0000    MDYSOC C 0145    MODYCS C 014D
CSTBL C 019F    PDRTBL C 0169    PGETSC C 012C

XTERNAL SYMBOLS
SMODE E 0000    ECHO   E 0000    EPEND  E 0000    GETCH  E 0000
ESOUT E 0000    PCMBUF E 0000    PCSBUF E 0000    PCTYPE E 0000
DRBUF E 0000    PRTYPE E 0000    TYPE   E 0000

SER SYMBOLS
SMODE E 0000    DEL1M  C 0162    ECHO   E 0000    EPEND  E 0000
ETCH  E 0000    GETTYP C 0000    I27128 C 007D    I27256 C 004A
2764  C 00B0    LOOP1  C 0164    MDYSOC C 0145    MESOUT E 0000
ODY01 C 0156    MODYCS C 014D    N27128 C 0066    N2764  C 0099
CMBUF E 0000    PCSBUF E 0000    PCSTBL C 019F    PCTYPE E 0000
DRBUF E 0000    PDRTBL C 0169    PGETBL C 0140    PGETSC C 012C
RTYPE E 0000    T2716  C 00F0    T2732  C 00DE    T2732A C 00CC
68764 C 0120    TERMSG C 01C3    TM2532 C 0114    TM2716 C 0102
```

YPE E 0000

SSEMBLY COMPLETE. NO ERRORS

```
480 MPP10A.001 NOOBJECT PRINT(:LP:)

IS-II 8080/8085 MACRO ASSEMBLER. V2.0          PPPROG    PAGE    1

LOC  OBJ          SEQ          SOURCE STATEMENT

                    1 $PAGEWIDTH(80)
                    2 :*************************************************
                    3 :*                                               *
                    4 :*         MULTIPURPOSE PROGRAMMER               *
                    5 :*              PROGRAM COMMEND                  *
                    6 :*                                               *
                    7 :* PROG.NO  :200-MM-10      DATE   :26.06.1984   *
                    8 :* FILE     :MPP10A.001     MODULE:PPPROG        *
                    9 :* AUTHOR   :E.DURMUS                            *
                   10 :*                                               *
                   11 :*************************************************
                   12 :*
                   13          NAME    PPPROG
                   14 :
                   15          PUBLIC  PCMND,PCOPY,PVERF,CRWAIT,GETADR
                   16 :
                   17          EXTRN   PGETSC,GETCH,ECHO,GETNM,MESOUT
                   18          EXTRN   ROMBEG,BUFEND,BUFBEG,CSMODE,CR
                   19          EXTRN   PROGAL,MODYCS,VERAL,STNDBY,MSGFIN
                   20          EXTRN   PMOTAL,VMOTAL,TYPE,SNDLED,WAIT
                   21 :
                   22          CSEG
                   23 :*************************************************
                   24 :*PROCEDURE :PCMND
                   25 :*FUNCTION  :PROGRAM EPROMS FROM BUFFER
                   26 :*INPUTS    :NONE
                   27 :*OUTPUTS   :NONE
                   28 :*CALLS     :PGETSC,GETCH,ECHO,GETNM,MESOUT
                   29 :*           PROGAL,MODYCS,VERAL
                   30 :*DESTROYS  :ALL
                   31 :*
0000 CD0000    E   32 PCMND:  CALL    PGETSC  :GET WHICH SOCETS ARE PROGRAMM
0003 CD0000    E   33         CALL    WAIT    :WAIT UNTIL CR IS PRESSED
0006 0E3E          34         MVI     C,')'   :DISPLAY PROMPT MSG.
0008 CD0000    E   35         CALL    ECHO
000B CDA000    C   36         CALL    GETADR  :GET PROM STR.ADR.& BUF.STR.EN
                   ADR.
000E 3A0000    E   37         LDA     CSMODE
0011 CD0000    E   38         CALL    SNDLED
                   39 :.....................................................
0014 3A0000    E   40         LDA     TYPE    :GET EPROM TYPE
0017 FE39          41         CPI     '9'     :CHECK THAT IS IT INTEL COMPAT
                   LE
0019 FA2500    C   42         JM      PCOPY   :JUMP IF YES
001C CD0000    E   43         CALL    PMOTAL  :ELSE MOTORALA PROGRAM IT
001F CD0000    E   44         CALL    VMOTAL  :VERIFY IT
0022 C38500    C   45         JMP     PSTN01  :PUT INTO STANDBY AND RETURN
                   46 :.....................................................
0025 CD0000    E   47 PCOPY:  CALL    PROGAL  :PROGRAM INTEL COMPATIBLE EPR

0028 3A0000    E   48 PVERF:  LDA     CSMODE  :FROM BUFFER CHECK HOW MANY E
                   MS
```

```
LOC  OBJ          SEQ         SOURCE STATEMENT

002B FE30           49          CPI     '0'     :ARE PROGRAMS. AND VERIFY ALL
                      ROMS
002D CA7800    C    50          JZ      L2      :WITH THE DATA IN THE BUFFER
0030 FE31           51          CPI     '1'
0032 CA6B00    C    52          JZ      L3
0035 FE32           53          CPI     '2'
0037 CA5E00    C    54          JZ      L4
003A FE33           55          CPI     '3'
003C CA5100    C    56          JZ      L5
003F FE34           57          CPI     '4'
0041 CA4400    C    58          JZ      L6
                    59 :
0044 3E34           60 L6:      MVI     A,'4'
0046 320000    E    61          STA     CSMODE
0049 3EEF           62          MVI     A,0EFH
004B CD0000    E    63          CALL    MODYCS
004E CD0000    E    64          CALL    VERAL
0051 3E33           65 L5:      MVI     A,'3'
0053 320000    E    66          STA     CSMODE
0056 3EF7           67          MVI     A,0F7H
0058 CD0000    E    68          CALL    MODYCS
005B CD0000    E    69          CALL    VERAL
005E 3E32           70 L4:      MVI     A,'2'
0060 320000    E    71          STA     CSMODE
0063 3EFB           72          MVI     A,0FBH
0065 CD0000    E    73          CALL    MODYCS
0068 CD0000    E    74          CALL    VERAL
006B 3E31           75 L3:      MVI     A,'1'
006D 320000    E    76          STA     CSMODE
0070 3EFD           77          MVI     A,0FDH
0072 CD0000    E    78          CALL    MODYCS
0075 CD0000    E    79          CALL    VERAL
0078 3E30           80 L2:      MVI     A,'0'
007A 320000    E    81          STA     CSMODE
007D 3EFE           82          MVI     A,0FEH
007F CD0000    E    83          CALL    MODYCS
0082 CD0000    E    84          CALL    VERAL
                    85 :
0085 CD0000    E    86 PSTNO1:  CALL    STNDBY  :PUT THE SOCETS INTO STANDBY
0088 3E35           87          MVI     A,35H
008A CD0000    E    88          CALL    SNDLED
008D 210000    E    89          LXI     H,MSGFIN : MODE DISPLAY END OF OPERATI

0090 CD0000    E    90          CALL    MESOUT   :MSG AND RETURN.
0093 C9             91          RET
                    92 :
                    93 :*************************************************
                    94 :*
0094 CD0000    E    95 CRWAIT:  CALL    GETCH
0097 FE00      E    96          CPI     LOW CR
0099 C29400    C    97          JNZ     CRWAIT
009C CD0000    E    98          CALL    ECHO
009F C9             99          RET
                   100 :*************************************************
                   101 :
```

```
_OC  OBJ          SEQ          SOURCE STATEMENT

00A0 0E03           102 GETADR: MVI     C.3
00A2 CD0000    E    103         CALL    GETNM
00A5 E1             104         POP     H
00A6 220000    E    105         SHLD    ROMBEG
00A9 E1             106         POP     H
00AA 220000    E    107         SHLD    BUFEND
00AD E1             108         POP     H
00AE 220000    E    109         SHLD    BUFBEG
00B1 C9             110         RET
                    111 ;***********************************************
                    112
                    113         END

BLIC SYMBOLS
WAIT C 0094     GETADR C 00A0     PCMND  C 0000     PCOPY  C 0025
ERF  C 0028

TERNAL SYMBOLS
FBEG E 0000     BUFEND E 0000     CR     E 0000     CSMODE E 0000
HO   E 0000     GETCH  E 0000     GETNM  E 0000     MESOUT E 0000
DYCS E 0000     MSGFIN E 0000     PGETSC E 0000     PMOTAL E 0000
OGAL E 0000     ROMBEG E 0000     SNDLED E 0000     STNDBY E 0000
PE   E 0000     VERAL  E 0000     VMOTAL E 0000     WAIT   E 0000

ER SYMBOLS
FBEG E 0000     BUFEND E 0000     CR     E 0000     CRWAIT C 0094
MODE E 0000     ECHO   E 0000     GETADR C 00A0     GETCH  E 0000
TNM  E 0000     L2     C 0078     L3     C 006B     L4     C 005E
     C 0051     L6     C 0044     MESOUT E 0000     MODYCS E 0000
GFIN E 0000     PCMND  C 0000     PCOPY  C 0025     PGETSC E 0000
OTAL E 0000     PROGAL E 0000     PSTN01 C 0085     PVERF  C 0028
MBEG E 0000     SNDLED E 0000     STNDBY E 0000     TYPE   E 0000
RAL  E 0000     VMOTAL E 0000     WAIT   E 0000

SEMBLY COMPLETE.  NO ERRORS
```

```
iBO MPP11A.OO1 NOOBJECT PRINT(:LP:)

:S-II 8080/8085 MACRO ASSEMBLER. V2.0          UTPROG   PAGE    1

.OC  OBJ          SEQ          SOURCE STATEMENT

                      1 $PAGEWIDTH(80)
                      2 :*****************************************************
                      3 :*                                                   *
                      4 :*              MULTIPURPOSE PROGRAMMER              *
                      5 :*              UTILITIES (PROGRAM MODE)             *
                      6 :*                                                   *
                      7 :*  PROG.NO:200-MM-11        DATE  :26.06.1984      *
                      8 :*  FILE   :MPP11A.OO1       MODULE:UTPROG          *
                      9 :*  AUTHOR :E.DURMUS                                 *
                     10 :*                                                   *
                     11 :*****************************************************
                     12 :
                     13          NAME    UTPROG
                     14 :
                     15          PUBLIC  PROGAL.PGPREP.PNOBYT.PINBYT
                     16          PUBLIC  PBPREP.PBPROG.PBVERF.SNDADR
                     17          PUBLIC  SNDDAT.GETDAT.SNMADR
                     18          PUBLIC  PMOTAL.SNDCS.SNDLED
                     19 :
                     20          EXTRN   ROMBEG.BUFEND.BUFBEG.PRTYPE.PCTYPE
                     21          EXTRN   BREAK.STNDBY.PDRBUF.PORTB1.DEL1M
                     22          EXTRN   PCOUNT.PCMBUF.PORTA1.PORTC1.ENLED
                     23          EXTRN   DISALL.ENAL.ENAH.DDROUT.ENDATO
                     24          EXTRN   DDRIN.ENDATI.ENCS.PPI1.DISERR
                     25
                     26          CSEG
                     27 :
                     28 :*****************************************************
                     29 :*
                     30 :*PROCEDURE:PROGAL
                     31 :*FUNCTION :PROGRAM EPROMS IDENTIFIED BY PCMND
                     32 :*INPUTS   :NONE
                     33 :*OUTPUTS  :NONE
                     34 :*CALLS    :PGPREP.PINBYT.BREAK.PNOBYT
                     35 :*DESTROYS :
                     36 :
0000 2A0000    E     37 PROGAL: LHLD    ROMBEG   :EPROM START ADR
0003 44              38         MOV     B.H      :...IN (BC)
0004 4D              39         MOV     C.L
0005 2A0000    E     40         LHLD    BUFEND   :BUFFER END ADR
0008 EB              41         XCHG             :....IN (DE)
0009 2A0000    E     42         LHLD    BUFBEG   :BUFFER START ADR IN (HL)
000C CD8500    C     43         CALL    PGPREP   :SET EPROMS IN STANDBY MODE
000F 3A0000    E     44         LDA     PRTYPE   :INTELLIGENT OR NORMAL
0012 B7              45         ORA     A
0013 CA3200    C     46         JZ      PAL00    :NORMAL
                     47 :.....................................
                     48 :       INTELLIGENT
                     49 :
0016 CDB800    C     50 PAL01:  CALL    PINBYT   :PROGRAM A BYTE
0019 D21F00    C     51         JNC     PNERR
001C CD0000    E     52         CALL    DISERR
```

```
LOC  OBJ          SEQ          SOURCE STATEMENT

001F 7D            53 PNERR:  MOV     A.L      :END OF PROGRAMMING
0020 BB            54         CMP     E        :(HL)=(DE)
0021 C22900    C   55         JNZ     PAL02    :IF NOT JUMP FOR PROGRAMMING
0024 7C            56         MOV     A,H      :NEXT BYTE
0025 BA            57         CMP     D
0026 CA3100    C   58         JZ      PAL03    :IF END JUMP FOR RETURN
0029 23            59 PAL02:  INX     H        :INCREMENT BUFFER ADR
002A 03            60         INX     B        :INCREMENT EPROM ADR
002B CD0000    E   61         CALL    BREAK    :ASK FOR BREAK
002E C31600    C   62         JMP     PAL01    :JUMP FOR NEXT BYTE
0031 C9            63 PAL03:  RET              :RETURN
                   64 :.....................................................
                   65 :       NORMAL
                   66
0032 CD9F00    C   67 PAL00:  CALL    PNOBYT   :SAME AS ABOVE PROCEDURE
0035 7D            68         MOV     A.L
0036 BB            69         CMP     E
0037 C23F00    C   70         JNZ     PAL04
003A 7C            71         MOV     A.H
003B BA            72         CMP     D
003C CA3100    C   73         JZ      PAL03
003F 23            74 PAL04:  INX     H
0040 03            75         INX     B
0041 CD0000    E   76         CALL    BREAK
0044 C33200    C   77         JMP     PAL00
                   78 :
                   79 :*************************************************
                   80 :*PROCEDURE:PMOTAL
                   81 :*FUNCTION :PROGRAM EPROMS
                   82 :*INPUTS   :NONE
                   83 :*OUTPUTS  :NONE
                   84 :*CALLS    :SNMADR,PMPREP,PMPRB,PMBPRG,SNDDAT
                   85 :*          BREAK
                   86 :*DESTROYS :A
                   87 :
                   88 PMOTAL:
0047 2A0000    E   89         LHLD    ROMBEG
004A 44            90         MOV     B.H
004B 4D            91         MOV     C.L
004C 2A0000    E   92         LHLD    BUFEND
004F EB            93         XCHG
0050 2A0000    E   94         LHLD    BUFBEG
0053 CD9100    C   95         CALL    PMPREP
0056 CD6C00    C   96 PM2:    CALL    PMOBYT
0059 7D            97         MOV     A.L
005A BB            98         CMP     E
005B C26300    C   99         JNZ     PM4
005E 7C           100         MOV     A,H
005F BA           101         CMP     D
0060 CA6B00    C  102         JZ      PM3
0063 23           103 PM4:    INX     H
0064 03           104         INX     B
0065 CD0000    E  105         CALL    BREAK
0068 C35600    C  106         JMP     PM2
006B C9           107 PM3:    RET
```

```
)C  OBJ          SEQ          SOURCE STATEMENT

                  108 :
                  109 :-----------------------------------------
                  110 :
)6C D5            111 PMOBYT: PUSH    D
)6D E5            112         PUSH    H
)6E CD7A01    C   113         CALL    SNMADR
)71 CD2801    C   114         CALL    PMBPRB
)74 CD3F01    C   115         CALL    PMBPRG
)77 E1            116         POP     H
)78 CD8D01    C   117         CALL    SNDDAT
)7B E5            118         PUSH    H
)7C CDE501    C   119         CALL    DEL50M
)7F CD2801    C   120         CALL    PMBPRB
)82 E1            121         POP     H
)83 D1            122         POP     D
)84 C9            123         RET
                  124 :
                  125 :*****************************************************
                  126 :*PROCEDURE:PGPREP
                  127 :*FUNCTION :PREPRATION FOR PROGRAMMING
                  128 :*INPUTS   :NONE
                  129 :*OUTPUTS  :NONE
                  130 :*CALLS    :PA3OUT.PMPREP.DELAY1
                  131 :*DESTROYS :A.HL
                  132 :
)085 E5           133 PGPREP: PUSH    H       :SAVE BUFFER STR ADR
)086 210200   E   134         LXI     H.PCMBUF+2
)089 CDB601   C   135         CALL    SNDCS   :SEND PROG STNBY CS CODE
)08C CD9100   C   136         CALL    PMPREP  :SEND PROG STNBY VPP CODE
)08F E1           137         POP     H
)090 C9           138         RET
                  139 :-----------------------------------------------------
                  140 :*PROCEDURE:PMPREP
                  141 :*FUNCTION :SEND PROG STANDBY VPP CODE
                  142 :*INPUTS   :NONE
                  143 :*OUTPUTS  :NONE
                  144 :*CALLS    :NONE
                  145 :*DESTROYS :A.HL
                  146 :
)091 E5           147 PMPREP: PUSH    H
)092 2A0000   E   148         LHLD    PDRBUF  :(HL)=POINT TO PDRTABLE ACCOR
                      G TO EPROM TYPE
)095 23           149         INX     H
)096 23           150         INX     H
)097 7E           151         MOV     A.M
)098 D300     E   152         OUT     LOW PORTB1
)09A CD0000   E   153         CALL    DEL1M
)09D E1           154         POP     H
)09E C9           155         RET
                  156 :
                  157 :*****************************************************
                  158 :*PROCEDURE :PNOBYT
                  159 :*FUNCTION  :PROGRAM A BYTE TO EPROM
                  160 :*INPUTS    :(BC):ADR OF EPROM
                  161 :*           (HL):ADR OF BUFFER
```

```
OC  OBJ          SEQ          SOURCE STATEMENT

                  162 :*OUTPUTS    :(BC):SAME
                  163 :*             (HL):SAME
                  164 :*CALLS      :SNDADR.PBPREP.PBPROG.SNDDAT.DEL50M
                  165 :*DESTROYS   :A
                  166 :
)09F D5           167 PNOBYT: PUSH    D        :SAVE EPROM END ADR
)0A0 E5           168         PUSH    H        :PUSH BUFFER START ADR
)0A1 CD5F01   C   169         CALL    SNDADR   :SEND ADR IN(BC) TO PROM
)0A4 CD1E01   C   170         CALL    PBPREP   :SET EPROM  TO PROG PREP MODE
)0A7 CD3501   C   171         CALL    PBPROG   :SET EPROM TO PROG MODE
)0AA E1           172         POP     H
)0AB CD8D01   C   173         CALL    SNDDAT   :SEND DATA TO EPROM
)0AE E5           174         PUSH    H
)0AF CDE501   C   175         CALL    DEL50M   :PROGRAMMING. WAIT 50Msec
)0B2 CD1E01   C   176         CALL    PBPREP   :SET PREP MODE
)0B5 E1           177         POP     H
)0B6 D1           178         POP     D        :RESTORE REGISTERS
)0B7 C9           179         RET              :RETURN
                  180
                  181 :*****************************************
                  182 :*
                  183 :*PROCEDURE:PINBYT
                  184 :*FUNCTION :PROGRAM A BYTE TO EPROM
                  185 :*INPUTS   :SME AS PNOBYT
                  186 :*
                  187 :*OUTPUTS  :    "
                  188 :*
                  189 :*CALLS    :SNADR.PBPREP.PBPROG.SNDDAT.GETDAT
                  190 :*           DEL1M.PVERF
                  191 :*DESTROYS :A
                  192 :
)0B8 D5           193 PINBYT: PUSH    D
)0B9 E5           194         PUSH    H
)0BA AF           195         XRA     A
)0BB 320000   E   196         STA     PCOUNT   :INITIALIZE PCOUNT TO ZERO
)0BE CD5F01   C   197         CALL    SNDADR
)0C1 CD1E01   C   198 L2:     CALL    PBPREP
)0C4 CD3501   C   199         CALL    PBPROG
)0C7 E1           200         POP     H
)0C8 CD8D01   C   201         CALL    SNDDAT
)0CB E5           202         PUSH    H
)0CC CD0000   E   203         CALL    DEL1M
)0CF CD1E01   C   204         CALL    PBPREP
)0D2 210000   E   205         LXI     H.PCTYPE
)0D5 3A0000   E   206         LDA     PCOUNT
)0D8 3C           207         INR     A
)0D9 320000   E   208         STA     PCOUNT
)0DC BE           209         CMP     M
)0DD CAF000   C   210         JZ      L1
)0E0 CD4A01   C   211         CALL    PBVERF
)0E3 CD9D01   C   212         CALL    GETDAT
)0E6 E1           213         POP     H
)0E7 BE           214         CMP     M
)0E8 CAEF00   C   215         JZ      L3
)0EB E5           216         PUSH    H
```

```
 C  OBJ          SEQ          SOURCE STATEMENT

)EC C3C100   C   217           JMP     L2
)EF E5           218 L3:      PUSH    H
)F0 CD1E01   C   219 L1:      CALL    PBPREP
)F3 CD3501   C   220           CALL    PBPROG
)F6 E1           221           POP     H
)F7 CD8D01   C   222           CALL    SNDDAT
)FA E5           223           PUSH    H
)FB CDF101   C   224           CALL    DELINT  :1ms*4*PCOUNT
)FE 210000   E   225           LXI     H.PCTYPE
101 3A0000   E   226           LDA     PCOUNT
104 BE           227           CMP     M
105 C21901   C   228           JNZ     L4
108 CD4A01   C   229           CALL    PBVERF
10B CD9D01   C   230           CALL    GETDAT
10E E1           231           POP     H
10F BE           232           CMP     M
110 CA1801   C   233           JZ      L5
113 E5           234           PUSH    H
114 37           235           STC
115 E1           236           POP     H
116 D1           237           POP     D
117 C9           238           RET
118 E5           239 L5:      PUSH    H
119 37           240 L4:      STC
11A 3F           241           CMC
11B E1           242           POP     H
11C D1           243           POP     D
11D C9           244           RET
                 245 :
                 246 :***********************************************
                 247 :*
                 248 :*PROCEDURE:PBPREP
                 249 :*FUNCTION :PUT EPROM IN THE PROG PREP MODE
                 250 :*INPUTS    :NONE
                 251 :*OUTPUTS   :NONE
                 252 :*CALLS     :SNDCS
                 253 :*DESTROYS :A.HL
                 254 :
11E 210300   E   255 PBPREP: LXI     H.PCMBUF+3
121 CDB601   C   256           CALL    SNDCS   :SEND PROG PREP CS CODE
124 CD2801   C   257           CALL    PMBPRB  :SEND PROG PREP VPP CODE
127 C9           258           RET
                 259 :-----------------------------------------------
12B 2A0000   E   260 PMBPRB: LHLD    PDRBUF
12B 23           261           INX     H
12C 23           262           INX     H
12D 23           263           INX     H
12E 7E           264           MOV     A.M
12F D300     E   265           OUT     LOW PORTB1
131 CD0000   E   266           CALL    DEL1M
134 C9           267           RET
                 268 :
                 269 :***********************************************
                 270 :*
                 271 :*PROCEDURE:PBPROG
```

```
LOC  OBJ          SEQ         SOURCE STATEMENT

                   272 :*FUNCTION :PUT EPROM IN THE PROGRAMMING MODE
                   273 :*INPUTS   :NONE
                   274 :*OUTPUTS  :NONE
                   275 :*CALLS    :SNDCS
                   276 :*DESTROYS :A.HL.DE
                   277 :
0135 210400    E   278 PBPROG: LXI     H.PCMBUF+4
0138 CDB601    C   279         CALL    SNDCS   :SEND PROG PROG CS CODE
013B CD3F01    C   280         CALL    PMBPRG  :SEND PROG PROG VPP CODE
013E C9            281         RET
                   282 :-----------------------------------------------
013F 2A0000    E   283 PMBPRG: LHLD    PDRBUF
0142 110400        284         LXI     D.0004
0145 19            285         DAD     D
0146 7E            286         MOV     A.M
0147 D300      E   287         OUT     LOW PORTB1
0149 C9            288         RET
                   289 :
                   290 :***********************************************
                   291 :*PROCEDURE:PBVERF
                   292 :*FUNCTION :SET EPROM IN THE VERIFY MODE
                   293 :*INPUTS   :
                   294 :*OUTPUTS  :
                   295 :*CALLS    :
                   296 :*DESTROYS :HL.DE.A
                   297 :
014A 210500    E   298 PBVERF: LXI     H.PCMBUF+5
014D CDB601    C   299         CALL    SNDCS   :SEND VERF CS CODE
0150 CD5401    C   300         CALL    PMBVRF  :SEND VERF VPP CODE
0153 C9            301         RET
                   302 :-----------------------------------------------
0154 2A0000    E   303 PMBVRF: LHLD    PDRBUF
0157 110500        304         LXI     D.0005
015A 19            305         DAD     D
015B 7E            306         MOV     A.M
015C D300      E   307         OUT     LOW PORTB1
015E C9            308         RET
                   309 :***********************************************
                   310 :*
                   311 :*PROCEDURE:SNDADR
                   312 :*FUNCTION :SEND ADDRESS TO EPROM IN (BC)
                   313 :*INPUTS   :(BC)=EPROM ADDRESS
                   314 :*OUTPUTS  :(BC)=EPROM ADDRESS
                   315 :*CALLS    :NONE
                   316 :*DESTROYS :A
                   317 :
015F 3E00      E   318 SNDADR: MVI     A.LOW DISALL
0161 D300      E   319         OUT     LOW PORTC1
0163 79            320         MOV     A.C
0164 D300      E   321         OUT     LOW PORTA1
0166 3E00      E   322         MVI     A.LOW ENAL
0168 D300      E   323         OUT     LOW PORTC1
016A 3E00      E   324         MVI     A.LOW DISALL
016C D300      E   325         OUT     LOW PORTC1
016E 78            326         MOV     A.B
```

```
.OC  OBJ          SEQ          SOURCE STATEMENT

)16F D300      E    327          OUT      LOW PORTA1
)171 3E00      E    328          MVI      A.LOW ENAH
)173 D300      E    329          OUT      LOW PORTC1
)175 3E00      E    330          MVI      A.LOW DISALL
)177 D300      E    331          OUT      LOW PORTC1
)179 C9             332          RET
                    333 :
                    334 :-----------------------------------------------
                    335 :*PROCEDURE:SNMADR
                    336 :*FUNCTION :SENDADR TO PROM(MOT) IN (BC)
                    337 :*INPUTS   :(BC) =EPROM ADR
                    338 :*OUTPUTS  :     "
                    339 :*CALLS    :SNDADR
                    340 :*DESTROYS :A
                    341 :
017A CD5F01    C    342 SNMADR: CALL     SNDADR  :SEND ADR EXCEPT A11
017D 78             343         MOV      A.B
017E E608           344         ANI      08H
0180 CA8501    C    345         JZ       PM1     :SEND ADR A11
0183 3EFF           346         MVI      A.0FFH
0185 210000    E    347 PM1:    LXI      H.PCMBUF
0188 77             348         MOV      M.A
0189 CDB601    C    349         CALL     SNDCS
018C C9             350         RET
                    351
                    352 :***********************************************
                    353 :*
                    354 :*PROCEDURE:SNDDAT
                    355 :*FUNCTION :SEND DATA ADDRESSED BY (HL) TO EPROM
                    356 :*INPUTS   :(HL)=ADDRESS OF THE DATA
                    357 :*OUTPUTS  :NONE
                    358 :*CALLS    :NONE
                    359 :*DESTROYS :A
                    360 :
018D 3E00      E    361 SNDDAT: MVI      A.LOW DISALL
018F D300      E    362         OUT      LOW PORTC1
0191 3E00      E    363         MVI      A.LOW DDROUT
0193 D300      E    364         OUT      LOW PORTC1
0195 7E             365         MOV      A.M
0196 D300      E    366         OUT      LOW PORTA1
0198 3E00      E    367         MVI      A.LOW ENDATO
019A D300      E    368         OUT      LOW PORTC1
019C C9             369         RET
                    370 :
                    371 :*******************************************
                    372 :*
                    373 :*PROCEDURE:GETDAT
                    374 :*FUNCTION :READ A BYTE FROM DATA BUFFER
                    375 :*INPUTS   :NONE
                    376 :*OUTPUTS  :(A)=DATA READ FROM EPROM
                    377 :*CALLS    :NONE
                    378 :*DESTROYS :A
                    379 :
019D 3E0E           380 GETDAT: MVI      A.0EH
019F D300      E    381         OUT      LOW PPI1
```

```
LOC  OBJ          SEQ          SOURCE STATEMENT

01A1 3E00      E   382          MVI     A,LOW DISALL
01A3 D300      E   383          OUT     LOW PORTC1
01A5 3E00      E   384          MVI     A,LOW DDRIN
01A7 D300      E   385          OUT     LOW PORTC1
01A9 3E00      E   386          MVI     A,LOW ENDATI
01AB D300      E   387          OUT     LOW PORTC1
01AD DB00      E   388          IN      LOW PORTA1
01AF F5            389          PUSH    PSW
01B0 3E0F          390          MVI     A,0FH
01B2 D300      E   391          OUT     LOW PPI1
01B4 F1            392          POP     PSW
01B5 C9            393          RET
                   394 ;
                   395 ;*********************************************
                   396 ;*PROCEDURE:SNDCS
                   397 ;*FUNCTION :SEND CS CODE TO EPROM
                   398 ;*INPUTS   :(HL)=ADRRESS OF CS CODE
                   399 ;*OUTPUTS  :     "
                   400 ;*CALLS    :NONE
                   401 ;*DESTROYS :A
                   402 ;
01B6 3E00      E   403 SNDCS:  MVI     A,LOW DISALL
01B8 D300      E   404         OUT     LOW PORTC1
01BA 7E            405         MOV     A,M
01BB D300      E   406         OUT     LOW PORTA1
01BD 3E00      E   407         MVI     A,LOW ENCS
01BF D300      E   408         OUT     LOW PORTC1
01C1 3E00      E   409         MVI     A,LOW DISALL
01C3 D300      E   410         OUT     LOW PORTC1
01C5 C9            411         RET
                   412 ;
                   413 ;*********************************************
                   414 ;*
                   415 ;*PROCEDURE :SNDLED
                   416 ;*FUNCTION  :TURNS ON/OFF PROGRAMMING LED
                   417 ;*INPUTS    :(A) = INDEX OF LEDTBL
                   418 ;*OUTPUTS   :NONE
                   419 ;*DESTROYS  :ALL
                   420 ;*CALLS     :NONE
                   421 ;
01C6 D630          422 SNDLED: SUI     30H
01C8 5F            423         MOV     E,A
01C9 3E00      E   424         MVI     A,LOW DISALL
01CB D300      E   425         OUT     LOW PORTC1
01CD 1600          426         MVI     D,0
01CF 21DF01    C   427         LXI     H,LEDTBL
01D2 19            428         DAD     D
01D3 7E            429         MOV     A,M
01D4 D300      E   430         OUT     LOW PORTA1
01D6 3E00      E   431         MVI     A,LOW ENLED
01D8 D300      E   432         OUT     LOW PORTC1
01DA 3E00      E   433         MVI     A,LOW DISALL
01DC D300      E   434         OUT     LOW PORTC1
01DE C9            435         RET
                   436 ;
```

```
LOC  OBJ          SEQ          SOURCE STATEMENT

01DF FE            437 LEDTBL: DB      OFEH.OFCH.OF8H.OF0H.0E0H.0FFH
01E0 FC
01E1 F8
01E2 F0
01E3 E0
01E4 FF
                   438
                   439 ;
                   440 ;***********************************************
                   441 ;*
                   442 ;*PROCEDURE :DEL50M
                   443 ;*FUNCTION  :50'MSEG DELAY
                   444 ;*INPUTS    :NONE
                   445 ;*OUTPUTS   :NONE
                   446 ;*CALLS     :NONE
                   447 ;*DESTROYS  :A
                   448 ;
01E5 3E32          449 DEL50M: MVI     A.50
01E7 F5            450 DEL501: PUSH    PSW
01E8 CD0000    E   451         CALL    DEL1M
01EB F1            452         POP     PSW
01EC 3D            453         DCR     A
01ED C2E701    C   454         JNZ     DEL501
01F0 C9            455         RET
                   456 ;*******************************************************
                   457 ;*
                   458 ;*PROCEDURE:DELINT
                   459 ;*FUNCTION :DELAY ACCORDING TO PCOUNT(1msec*4*PCOUNT)
                   460 ;*INPUTS   :NONE
                   461 ;*OUTPUTS  :NONE
                   462 ;*CALLS    :DEL1M
                   463 ;*DESTROYS :A
                   464 ;
01F1 3A0000    E   465 DELINT: LDA     PCOUNT
01F4 07            466         RLC
01F5 07            467         RLC
01F6 F5            468 DELIN1: PUSH    PSW
01F7 CD0000    E   469         CALL    DEL1M
01FA F1            470         POP     PSW
01FB 3D            471         DCR     A
01FC C2F601    C   472         JNZ     DELIN1
01FF C9            473         RET
                   474 ;
                   475 ;***********************************************
                   476 ;
                   477       END

PUBLIC SYMBOLS
PDAT  C 019D    PBPREP C 011E    PBPROG C 0135    PBVERF C 014A
PPREP C 0085    PINBYT C 00B8    PMOTAL C 0047    PNOBYT C 009F
POGAL C 0000    SNDADR C 015F    SNDCS  C 01B6    SNDDAT C 018D
POLED C 01C6    SNMADR C 017A

EXTERNAL SYMBOLS
BREAK  E 0000    BUFBEG E 0000    BUFEND E 0000    DDRIN  E 0000
```

```
DDROUT  E 0000    DELIM   E 0000    DISALL  E 0000    DISERR  E 0000
ENAH    E 0000    ENAL    E 0000    ENCS    E 0000    ENDATI  E 0000
ENDATO  E 0000    ENLED   E 0000    PCMBUF  E 0000    PCOUNT  E 0000
PCTYPE  E 0000    PDRBUF  E 0000    PORTAI  E 0000    PORTBI  E 0000
PORTCI  E 0000    PPII    E 0000    PRTYPE  E 0000    ROMBEG  E 0000
STNDBY  E 0000

USER SYMBOLS
BREAK   E 0000    BUFBEG  E 0000    BUFEND  E 0000    DDRIN   E 0000
DDROUT  E 0000    DELIM   E 0000    DELSOI  C 01E7    DELSOM  C 01E5
DELIN1  C 01F6    DELINT  C 01F1    DISALL  E 0000    DISERR  E 0000
ENAH    E 0000    ENAL    E 0000    ENCS    E 0000    ENDATI  E 0000
ENDATO  E 0000    ENLED   E 0000    GETDAT  C 019D    L1      C 00F0
L2      C 00C1    L3      C 00EF    L4      C 0119    L5      C 0118
LEDTBL  C 01DF    PAL00   C 0032    PAL01   C 0016    PAL02   C 0029
PAL03   C 0031    PAL04   C 003F    PBPREP  C 011E    PBPROG  C 0135
PBVERF  C 014A    PCMBUF  E 0000    PCOUNT  E 0000    PCTYPE  E 0000
PDRBUF  E 0000    PGPREP  C 0085    PINBYT  C 00B8    PM1     C 0185
PM2     C 0056    PM3     C 006B    PM4     C 0063    PMBPRB  C 0128
PMBPRG  C 013F    PMBVRF  C 0154    PMOBYT  C 006C    PMOTAL  C 0047
PMPREP  C 0091    PNERR   C 001F    PNOBYT  C 009F    PORTA1  E 0000
PORTB1  E 0000    PORTC1  E 0000    PPI1    E 0000    PROGAL  C 0000
PRTYPE  E 0000    ROMBEG  E 0000    SNDADR  C 015F    SNDCS   C 01B6
SNDDAT  C 018D    SNDLED  C 01C6    SNMADR  C 017A    STNDBY  E 0000

ASSEMBLY COMPLETE.  NO ERRORS
```

```
80 MPP20A.001 NOOBJECT PRINT(:LP:)

S-II 8080/8085 MACRO ASSEMBLER. V2.0          PPVERF    PAGE     1

.OC  OBJ           SEQ          SOURCE STATEMENT

                     1 $PAGEWIDTH(80)
                     2 :*****************************************************
                     3 :*                                                   *
                     4 :*           MULTIPURPOSE PROGAMMER                  *
                     5 :*               VERIFY COMMAND                      *
                     6 :*                                                   *
                     7 :* PROG.NO:200-MM-20          DATE  :26.06.1984 *
                     8 :* FILE    :MPP20A.001        MODULE:PPVERF     *
                     9 :* AUTHOR :E.DURMUS                                 *
                    10 :*****************************************************
                    11 :
                    12          NAME    PPVERF
                    13 :
                    14          PUBLIC  CCMND.VERAL.DISERR.VMOTAL.WAIT
                    15
                    16          EXTRN   TGETSC.GETCH.CR.ECHO.GETNM.ROMBEG
                    17          EXTRN   BUFEND.BUFBEG.STNDBY.MSGFIN.MESOUT
                    18          EXTRN   READBT.REMBYT.BREAK.CI.GETCM
                    19          EXTRN   TYPE.CSMODE.CO.SPCOUT.ADRD.NMOUT
                    20          EXTRN   GETADR
                    21 :
                    22          CSEG
                    23 :
                    24 :*****************************************************
                    25 :*PROCEDURE :VCMND
                    26 :*FUNCTION  :VERIFY SPECIFIED BUFFER DATA WITH PROM D
                    27 :*INPUTS    :NONE
                    28 :*OUTPUTS   :NONE
                    29 :*CALLS     :TGETSC.ECHO.GETNM.VERALL.STNDBY.MESOUT
                    30 :*DESTROYS   :ALL
                    31 :
0000 CD0000    E    32 CCMND:  CALL    TGETSC  :GET WHICH SOCET IS VERIFIED
0003 CDB800    C    33         CALL    WAIT    :WAIT UNTIL CR IS PRESSED
0006 0E3E           34         MVI     C.')'   :DISPLAY PROMPT CHARACTER
0008 CD0000    E    35         CALL    ECHO
000B CD0000    E    36         CALL    GETADR  :GET EPROM AND BUFFER ADRES
                    37 :.....................................
000E 3A0000    E    38         LDA     TYPE    :GET EPROM TYPE
0011 FE39           39         CPI     '9'     :CHECK IS IT MOTOROLA OR INTE
                       P.
0013 FA1C00    C    40         JM      VERINT  :JUMP IF INTEL COMPATIBLE.
0016 CD5900    C    41         CALL    VMOTAL  :ELSE VERIFY MOT. EPROM VIA E
                       ER
0019 C31F00    C    42         JMP     VSTNO1  :PUT INTO STANBY MODE AND RET
                    43 :.....................................................
                    44
001C CD2900    C    45 VERINT: CALL    VERAL
001F CD0000    E    46 VSTNO1: CALL    STNDBY
0022 210000    E    47         LXI     H.MSGFIN
0025 CD0000    E    48         CALL    MESOUT
0028 C9             49         RET
                    50 :
```

```
OC  OBJ          SEQ         SOURCE STATEMENT

                  51 :****************************************
                  52 :*
                  53 :*PROCEDURE:VERAL
                  54 :*FUNCTION :VERIFY EPROM(INTEL) WITH BUFFER
                  55 :*           SPECIFIED BY VCMND
                  56 :*INPUTS    :NONE
                  57 :*OUTPUTS   :NONE
                  58 :*CALLS     :READBT.DISERR.BREAK.STNDBY
                  59 :*DESTROYS  :ALL
                  60 :
)029 2A0000    E  61 VERAL:  LHLD    ROMBEG   :EPROM START ADR
)02C 44           62         MOV     B.H      :...IN (BC)
)02D 4D           63         MOV     C.L
)02E 2A0000    E  64         LHLD    BUFEND   :BUFFER END ADR
)031 EB           65         XCHG             :...IN (DE)
)032 2A0000    E  66         LHLD    BUFBEG   :BUFFER START ADR IN (HL)
)035 CD0000    E  67 VER1:   CALL    READBT   :READ A BYTE FROM EPROM
)038 BE           68         CMP     M        :COPMPARE IT WITH THE BUFFER
)039 CA4600    C  69         JZ      VER2     :SKIP IF MATCH
)03C CD0000    E  70         CALL    READBT   :ELSE READ AGAIN
)03F BE           71         CMP     M        :COMPARE
)040 CA4600    C  72         JZ      VER2     :SKIP IF MATCH
)043 CD8200    C  73         CALL    DISERR   :ELSE DISPLAY ERROR MSG
)046 7D           74 VER2:   MOV     A.L      :CHECK END OF VERIFICATION
)047 BB           75         CMP     E        :(DE)=(HL)
)048 C25000    C  76         JNZ     VER3     :SKIP IF NOT
)04B 7C           77         MOV     A.H
)04C BA           78         CMP     D
)04D CA5800    C  79         JZ      VER4     :IF END JMP
)050 03           80 VER3:   INX     B        :NOT END SO INCREMENT
)051 23           81         INX     H        :EPROM ADR AND BUFFER ADR
)052 CD0000    E  82         CALL    BREAK    :ASK FOR BREAK
)055 C33500    C  83         JMP     VER1
)058 C9           84 VER4:   RET              :END SO RETURN
                  85 :
                  86 :************************************************
                  87 :*
                  88 :*PROCEDURE:VMOTAL
                  89 :*FUNCTION :
                  90 :
)059 2A0000    E  91 VMOTAL: LHLD    ROMBEG
)05C 44           92         MOV     B.H
)05D 4D           93         MOV     C.L
)05E 2A0000    E  94         LHLD    BUFEND
)061 EB           95         XCHG
)062 2A0000    E  96         LHLD    BUFBEG
)065 CD0000    E  97 VM01:   CALL    REMBYT
)068 BE           98         CMP     M
)069 CA6F00    C  99         JZ      VM02
)06C CD8200    C 100         CALL    DISERR
)06F 7D          101 VM02:   MOV     A.L
)070 BB          102         CMP     E
)071 C27900    C 103         JNZ     VM03
)074 7C          104         MOV     A.H
)075 BA          105         CMP     D
```

```
)C  OBJ          SEQ          SOURCE STATEMENT

)76 CA8100   C    106          JZ      VM04
)79 03            107 VM03:    INX     B
)7A 23            108          INX     H
)7B CD0000   E    109          CALL    BREAK
)7E C36500   C    110          JMP     VM01
)81 C9            111 VM04:    RET
                  112
                  113 :*****************************************************
                  114 :*
                  115 :*PROCEDURE:DISERR
                  116 :*FUNCTION :DISPLAY VERIFICATION ERROR MESSAGE
                  117 :*INPUTS   :
                  118 :*OUPUTS   :
                  119 :*CALLS    :CO.SPCOUT.ADRD.MESOUT.NMOUT.WAIT
                  120 :*DESTROYS :
                  121 :
)82 C5            122 DISERR:  PUSH    B
)083 F5           123          PUSH    PSW
)084 E5           124          PUSH    H
)085 3A0000   E   125          LDA     CSMODE
)088 4F           126          MOV     C.A
)089 CD0000   E   127          CALL    CO
)08C CD0000   E   128          CALL    SPCOUT
)08F CD0000   E   129          CALL    ADRD
)092 21B000   C   130          LXI     H.MMSG
)095 CD0000   E   131          CALL    MESOUT
)098 E1           132          POP     H
)099 7E           133          MOV     A.M
)09A E5           134          PUSH    H
)09B CD0000   E   135          CALL    NMOUT
)09E 21B400   C   136          LXI     H.PMSG
)0A1 CD0000   E   137          CALL    MESOUT
)0A4 E1           138          POP     H
)0A5 F1           139          POP     PSW
)0A6 E5           140          PUSH    H
)0A7 CD0000   E   141          CALL    NMOUT
)0AA CDB800   C   142          CALL    WAIT
)0AD E1           143          POP     H
)0AE C1           144          POP     B
)0AF C9           145          RET
                  146 :
                  147 :*********************************************
                  148 :
)0B0 20423D       149 MMSG:    DB      ' B='.0
)0B3 00
)0B4 20453D       150 PMSG:    DB      ' E='.0
)0B7 00
                  151 :
                  152 :*********************************************
                  153 :*PROCEDURE: WAIT
                  154 :*FUNCTION :WAITS UNTIL CR IS PRESSED
                  155 :*INPUTS   :NONE
                  156 :*OUTPTS   :NONE
                  157 :*CALLS    :NONE
                  158 :*DESTROYS :A
```

```
DC  OBJ          SEQ          SOURCE STATEMENT

                  159 :
0B8 CD0000    E   160 WAIT:    CALL    GETCH
0BB FE1B          161         CPI     1BH
0BD CA0000    E   162         JZ      GETCM
0C0 FE00      E   163         CPI     LOW CR
0C2 C2B800    C   164         JNZ     WAIT
0C5 CD0000    E   165         CALL    ECHO
0C8 C9            166         RET
                  167 :
                  168 :*******************************************
                  169 :
                  170         END

LIC SYMBOLS
ND   C 0000    DISERR C 0082    VERAL  C 0029    VMOTAL C 0059
T    C 00B8

ERNAL SYMBOLS
D    E 0000    BREAK  E 0000    BUFBEG E 0000    BUFEND E 0000
     E 0000    CO     E 0000    CR     E 0000    CSMODE E 0000
D    E 0000    GETADR E 0000    GETCH  E 0000    GETCM  E 0000
NM   E 0000    MESOUT E 0000    MSGFIN E 0000    NMOUT  E 0000
DBT  E 0000    REMBYT E 0000    ROMBEG E 0000    SPCOUT E 0000
DBY  E 0000    TGETSC E 0000    TYPE   E 0000

ER SYMBOLS
RD   E 0000    BREAK  E 0000    BUFBEG E 0000    BUFEND E 0000
ND   C 0000    CI     E 0000    CO     E 0000    CR     E 0000
ODE  E 0000    DISERR C 0082    ECHO   E 0000    GETADR E 0000
CH   E 0000    GETCM  E 0000    GETNM  E 0000    MESOUT E 0000
SG   C 00BO    MSGFIN E 0000    NMOUT  E 0000    PMSG   C 00B4
ADBT E 0000    REMBYT E 0000    ROMBEG E 0000    SPCOUT E 0000
NDBY E 0000    TGETSC E 0000    TYPE   E 0000    VER1   C 0035
R2   C 0046    VER3   C 0050    VER4   C 0058    VERAL  C 0029
RINT C 001C    VM01   C 0065    VM02   C 006F    VM03   C 0079
04   C 0081    VMOTAL C 0059    VSTN01 C 001F    WAIT   C 00B8

SEMBLY COMPLETE.  NO ERRORS
```

```
 0 MPP30A.001 NOOBJECT PRINT(:LP:)

-II 8080/8085 MACRO ASSEMBLER. V2.0          PPTRAN    PAGE     1

IC  OBJ           SEQ          SOURCE STATEMENT

                    1 $PAGEWIDTH(80)
                    2 :************************************************
                    3 :*                                              *
                    4 :*        MULTIPURPOSE PROGRAMMER               *
                    5 :*           TRANSFER COMMAND                   *
                    6 :*                                              *
                    7 :* PROG.NO:200-MM-30       DATE  :26.06.1943 *
                    8 :* FILE    :MPP30A.001     MODULE:PPTRAN      *
                    9 :* AUTHOR :E.DURMUS                           *
                   10 :*                                              *
                   11 :************************************************
                   12 :
                   13          NAME    PPTRAN
                   14 :
                   15          PUBLIC  TCMND.TRANAL.TGETSC
                   16 :
                   17          EXTRN   GETCH.CR.ECHO.GETNM.BUFBEG
                   18          EXTRN   ROMEND.ROMBEG.STNDBY.MSGFIN
                   19          EXTRN   MESOUT.READBT.REMBYT.BREAK
                   20          EXTRN   CSMODE.MODYCS.TYPE.WAIT
                   21          EXTRN   MDYSOC
                   22 :
                   23          CSEG
                   24 :************************************************
                   25 :*
                   26 :* PROCEDURE:TCMD
                   27 :* FUNCTION :TRANSFER SPECIFIED EPROM CONTENT
                   28 :*             TO BUFFER
                   29 :* INPUTS   :NONE
                   30 :* OUTPUTS  :NONE
                   31 :* CALLS    :GETNM.RDPREP.READBT.BREAK.RSTNBY.MESOUT
                   32 :* DESTROYS :ALL
                   33 :* NOTES    :NONE
                   34 :
0000 CD5A00    C   35 TCMND:  CALL    TGETSC
0003 CD0000    E   36         CALL    WAIT
0006 0E3E          37         MVI     C.')'
0008 CD0000    E   38         CALL    ECHO
000B 0E03          39         MVI     C.3
000D CD0000    E   40         CALL    GETNM
0010 E1            41         POP     H
0011 220000    E   42         SHLD    BUFBEG
0014 E1            43         POP     H
0015 220000    E   44         SHLD    ROMEND
0018 E1            45         POP     H
0019 220000    E   46         SHLD    ROMBEG
001C CD2900    C   47         CALL    TRANAL
001F CD0000    E   48         CALL    STNDBY
0022 210000    E   49         LXI     H.MSGFIN
0025 CD0000    E   50         CALL    MESOUT
0028 C9            51         RET
                   52 :
```

```
)C  OBJ          SEQ          SOURCE STATEMENT

                  53 :*********************************************
                  54 :*
                  55 :*PROCEDURE:TRANAL
                  56 :*FUNCTION :TRANSFER EPROM TO BUFFER
                  57 :*INPUTS   :NONE
                  58 :*OUTPUTS  :NONE
                  59 :*CALLS    :READBT. REMBYT. BREAK
                  60 :*DESTROYS :ALL
                  61 :
)29 2A0000    E   62 TRANAL: LHLD    ROMBEG   :EPROM START ADR
)2C 44            63         MOV     B. H     :....IN (BC)
)2D 4D            64         MOV     C. L
)2E 2A0000    E   65         LHLD    ROMEND   :EPROM  END ADR
)31 EB            66         XCHG             :....IN (DE)
)32 2A0000    E   67         LHLD    BUFBEG   :BUFFER START ADR IN (HL)
)35 3A0000    E   68 TRAN1:  LDA     TYPE
)38 FE39          69         CPI     '9'      :CHECK THAT IT IS INTEL/MOT
)3A FA4300    C   70         JM      TRAN4
)3D CD0000    E   71         CALL    REMBYT   :MOT SO READ A BYTE BY MOT REA
)40 C34600    C   72         JMP     TRAN5
)43 CD0000    E   73 TRAN4:  CALL    READBT   :INTEL SO READ A BYTE BY INT R
                      D
)46 77            74 TRAN5:  MOV     M. A     :PUT DATA BYTE TO BUFFER
)47 79            75         MOV     A. C
)48 BB            76         CMP     E        :CHECK ED OF OPERATION(BC=DE)
)49 C25100    C   77         JNZ     TRAN2    :IF NOT CONTINUE
)4C 78            78         MOV     A. B
)4D BA            79         CMP     D
)4E CA5900    C   80         JZ      TRAN3    :ELSE RETUR
)51 23            81 TRAN2:  INX     H        :INCREMENT BUFFER ADR
)52 03            82         INX     B        :INCREMET EPROM ADR
)53 CD0000    E   83         CALL    BREAK    :ASK FOR BREAK
)56 C33500    C   84         JMP     TRAN1    :CONTINUE
)59 C9            85 TRAN3:  RET
                  86 :
                  87 :*********************************************
                  88 :*
                  89 :*PROCEDURE:TGETSC
                  90 :*FUNCTION :OBTAIN THAT WHICH SOCKET IS TRANSFERRED
                  91 :*INPUTS   :NONE
                  92 :*OUTPUTS  :NONE
                  93 :*CALLS    :GETCH. ECHO. MODYCS
                  94 :*DESTROYS :ALL
                  95 :
)5A CD0000    E   96 TGETSC: CALL    GETCH
)5D CD0000    E   97         CALL    ECHO
)60 79            98         MOV     A. C
)61 320000    E   99         STA     CSMODE
)64 216E00    C  100         LXI     H. TGETBL
)67 CD0000    E  101         CALL    MDYSOC
)6A CD0000    E  102         CALL    MODYCS
)6D C9           103         RET
                 104 :
)6E FE           105 TGETBL: DB      OFEH. OFDH. OFBH. OF7H. OEFH
)6F FD
```

```
OC  OBJ          SEQ          SOURCE STATEMENT

070 FB
071 F7
072 EF
                  106 :
                  107 :*****************************************************
                  108 :
                  109          END

LIC SYMBOLS
ND   C 0000    TGETSC C 005A    TRANAL C 0029

ERNAL SYMBOLS
AK   E 0000    BUFBEG E 0000    CR     E 0000    CSMODE E 0000
D    E 0000    GETCH  E 0000    GETNM  E 0000    MDYSOC E 0000
OUT  E 0000    MODYCS E 0000    MSGFIN E 0000    READBT E 0000
BYT  E 0000    ROMBEG E 0000    ROMEND E 0000    STNDBY E 0000
E    E 0000    WAIT   E 0000

R SYMBOLS
AK   E 0000    BUFBEG E 0000    CR     E 0000    CSMODE E 0000
D    E 0000    GETCH  E 0000    GETNM  E 0000    MDYSOC E 0000
OUT  E 0000    MODYCS E 0000    MSGFIN E 0000    READBT E 0000
BYT  E 0000    ROMBEG E 0000    ROMEND E 0000    STNDBY E 0000
ND   C 0000    TGETBL C 006E    TGETSC C 005A    TRAN1  C 0035
N2   C 0051    TRAN3  C 0059    TRAN4  C 0043    TRAN5  C 0046
NAL  C 0029    TYPE   E 0000    WAIT   E 0000

EMBLY COMPLETE.  NO ERRORS
```

```
30 MPP31A.001 NOOBJECT PRINT(:LP:)

8-II 8080/8085 MACRO ASSEMBLER. V2.0          UTREAD    PAGE     1

OC  OBJ          SEQ          SOURCE STATEMENT

                   1 $PAGEWIDTH(80)
                   2 :************************************************
                   3 :*                                              *
                   4 :*       MULTIPURPOSE PROGRAMMER                *
                   5 :*        UTULITIES(READ MODE)                  *
                   6 :*                                              *
                   7 :* PROG.NO:200-MM-31       DATE  :26.06.1984  *
                   8 :* FILE    :MPP31A.001     MODULE:UTREAD      *
                   9 :* AUTHOR  :E.DURMUS                          *
                  10 :*                                              *
                  11 :************************************************
                  12 :
                  13          NAME    UTREAD
                  14 :
                  15          PUBLIC  STNDBY,READBT,REMBYT,RBPREP
                  16          PUBLIC  RMBPRB,RBREAD,RMBRED
                  17 :
                  18          EXTRN   DISALL,PORTC1,PORTB1,PORTA1
                  19          EXTRN   SNDADR,SNMADR,GETDAT,PCMBUF
                  20          EXTRN   SNDCS,PDRBUF
                  21 :
                  22          CSEG
                  23 :************************************************
                  24 :*
                  25 :* PROCEDURE:STNDBY
                  26 :* FUNCTION :SET EPROM IN STANDBY MODE
                  27 :* INPUTS   :NONE
                  28 :* OUTPUTS  :NONE
                  29 :* CALLS    :NONE
                  30 :* DESTROYS :A
                  31 :* NOTES    :NONE
                  32 :
0000 3E00      E  33 STNDBY: MVI     A,LOW DISALL
0002 D300      E  34         OUT     LOW PORTC1
0004 3E28         35         MVI     A,28H
0006 D300      E  36         OUT     LOW PORTB1
0008 3EFF         37         MVI     A,0FFH
000A D300      E  38         OUT     LOW PORTA1
000C 3EE8         39         MVI     A,0E8H
000E D300      E  40         OUT     LOW PORTC1
0010 3E00      E  41         MVI     A,LOW DISALL
0012 D300      E  42         OUT     LOW PORTC1
0014 AF           43         XRA     A
0015 D300      E  44         OUT     LOW PORTA1
0017 3E17         45         MVI     A,17H
0019 D300      E  46         OUT     LOW PORTC1
001B C9           47         RET
                  48 :
                  49 :************************************************
                  50 :*
                  51 :*PROCEDURE:READBT
                  52 :*FUNCTION :READ A BYTE FROM EPROM
```

```
 IC  OBJ           SEQ          SOURCE STATEMENT

                    53 :*INPUTS    :(BC)=EPROM START ADR
                    54 :*           (HL)=BUFFER START ADR
                    55 :*OUTPUTS   :(A)DATA READ FROM EPROM
                    56 :*CALLS     :SNDADR. PBPREP, RBREAD. GETDAT
                    57 :*DESTROYS  :(A)
                    58 :
 01C E5             59 READBT: PUSH    H
 01D CD0000    E    60         CALL    SNDADR   :SEND EPROM ADR
 020 CD4400    C    61         CALL    RBPREP   :PUT EPROM TO READ PREP MODE
 023 CD5500    C    62         CALL    RBREAD   :PUT EPROM INTO READ MODE
 026 CD0000    E    63         CALL    GETDAT   :READ A BYTE FROM EPROM
 029 F5             64         PUSH    PSW      :SAVE IT
 02A CD4400    C    65         CALL    RBPREP   :PUT EPROM INTO PREP MODE
 02D F1             66         POP     PSW      :RESTORE DATA
 02E E1             67         POP     H
 02F C9             68         RET              :RETURN
                    69 :
                    70 :*********************************************
                    71 :*
                    72 :*PROCEDURE:REMBYT
                    73 :*FUNCTION  :READ A BYTE FROM EPROM(MOT)
                    74 :*INPUTS    :SAME AS READBT
                    75 :*OUTPUTS   :   "
                    76 :*CALLS     :SNMADR. RMBPRB. RMBRED. GETFAT
                    77 :*DESTROYS  :A
                    78 :
 030 E5             79 REMBYT: PUSH    H
 031 CD0000    E    80         CALL    SNMADR   :SAME AS READBT
 034 CD4E00    C    81         CALL    RMBPRB
 037 CD5F00    C    82         CALL    RMBRED
 03A CD0000    E    83         CALL    GETDAT
 03D F5             84         PUSH    PSW
 03E CD4E00    C    85         CALL    RMBPRB
 041 F1             86         POP     PSW
 042 E1             87         POP     H
 043 C9             88         RET
                    89 :
                    90 :*********************************************
                    91 :*
                    92 :*PROCEDURE:RBPREP
                    93 :*FUNCTION  :PUT EPROM IN THE READ STANDBY MODE
                    94 :*INPUTS    :NONE
                    95 :*OUTPUTS   :NONE
                    96 :*CALLS     :SNDCS
                    97 :*DESTROYS  :A. HL
                    98 :
 044 210000    E    99 RBPREP: LXI     H. PCMBUF
 047 CD0000    E   100         CALL    SNDCS    :SEND READ PREP MODE CS CODE
 04A CD4E00    C   101         CALL    RMBPRB   :SEND READ PREP MODE VPP CODE
 04D C9            102         RET
                   103 :-------------------------------------------
 04E 2A0000    E   104 RMBPRB: LHLD    PDRBUF
 051 7E            105         MOV     A. M
 052 D300      E   106         OUT     LOW PORTB1
 054 C9            107         RET
```

```
OC  OBJ          SEQ         SOURCE STATEMENT

                  108 :
                  109 :***********************************************
                  110 :*
                  111 :*PROCEDURE:RBREAD
                  112 :*FUNCTION :PUT EPROM IN THE READ MODE
                  113 :*INPUTS   :NONE
                  114 :*OUTPUTS  :NONE
                  115 :*CALLS    :SNDCS
                  116 :*DESTROYS :A.HL
                  117 :
055 210100    E   118 RBREAD: LXI     H.PCMBUF+1 :SEND READ MODE CS CODE
058 CD0000    E   119         CALL    SNDCS
05B CD5F00    C   120         CALL    RMBRED  :SEND READ MODE VPP CODE
05E C9            121         RET
                  122 :-----------------------------------------------
05F 2A0000    E   123 RMBRED: LHLD    PDRBUF
062 23            124         INX     H
063 7E            125         MOV     A.M
064 D300      E   126         OUT     LOW PORTB1
066 C9            127         RET
                  128 :
                  129 :***********************************************
                  130 :
                  131         END

BLIC SYMBOLS
PREP C 0044    RBREAD C 0055    READBT C 001C    REMBYT C 0030
BPRB C 004E    RMBRED C 005F    STNDBY C 0000

TERNAL SYMBOLS
SALL E 0000    GETDAT E 0000    PCMBUF E 0000    PDRBUF E 0000
RTA1 E 0000    PORTB1 E 0000    PORTC1 E 0000    SNDADR E 0000
DCS  E 0000    SNMADR E 0000

ER SYMBOLS
SALL E 0000    GETDAT E 0000    PCMBUF E 0000    PDRBUF E 0000
RTA1 E 0000    PORTB1 E 0000    PORTC1 E 0000    RBPREP C 0044
READ C 0055    READBT C 001C    REMBYT C 0030    RMBPRB C 004E
BRED C 005F    SNDADR E 0000    SNDCS  E 0000    SNMADR E 0000
NDBY C 0000

SEMBLY COMPLETE.  NO ERRORS
```

```
30 MPP40A.001 NOOBJECT PRINT(:LP:)

8-II 8080/8085 MACRO ASSEMBLER. V2.0          PPCOMP    PAGE     1

OC  OBJ           SEQ          SOURCE STATEMENT

                    1 $PAGEWIDTH(80)
                    2 :*****************************************************
                    3 :*                                                   *
                    4 :*           MULTIPURPOSE PROGRAMMER                 *
                    5 :*                CHECK COMMAND                      *

                    6 :*                                                   *
                    7 :* PROG.NO:200-MM-40      DATE    :26.05.1984       *
                    8 :* FILE    :MPP40A.001    MODULE  :PPCOMP           *
                    9 :* AUTHOR  :E.DURMUS                                 *
                   10 :*                                                   *
                   11 :*****************************************************
                   12 :
                   13          NAME    PPCOMP
                   14 :
                   15          PUBLIC  ECMND
                   16 :
                   17          EXTRN   TGETSC.GETCH.ECHO.READBT.REMBYT
                   18          EXTRN   CR.LF.EPEND.TYPE.STNDBY.MESOUT
                   19          EXTRN   WAIT
                   20
                   21          CSEG
                   22 :*
                   23 :*****************************************************
                   24 :*PROCEDURE  :ECMND
                   25 :*FUNCTION   :CHECK FOR EPROM IS ERASED OR NOT
                   26 :*INPUTS     :NONE
                   27 :*OUTPUTS    :NONE
                   28 :*DESTROYS   :ALL
                   29 :*CALLS      :CI.CO.RDPREP.READBT.RSTNBY.MESOUT
0000 CD0000    E   30 ECMND:  CALL    TGETSC
0003 CD0000    E   31         CALL    WAIT
                   32 :
0006 010000        33         LXI     B.0
0009 3A0000    E   34 C6:     LDA     TYPE
000C FE39          35         CPI     '9'
000E FA1700    C   36         JM      C3
0011 CD0000    E   37         CALL    REMBYT
0014 C31A00    C   38         JMP     C5
0017 CD0000    E   39 C3:     CALL    READBT
001A FEFF          40 C5:     CPI     0FFH
001C C22D00    C   41         JNZ     C2
001F 03            42         INX     B
0020 3A0000    E   43         LDA     EPEND
0023 B8            44         CMP     B
0024 C20900    C   45         JNZ     C6
0027 214500    C   46         LXI     H.YERMSG
002A C33000    C   47         JMP     C4
002D 213700    C   48 C2:     LXI     H.NERMSG
0030 CD0000    E   49 C4:     CALL    STNDBY
0033 CD0000    E   50         CALL    MESOUT
0036 C9            51         RET
```

```
.OC  OBJ          SEQ          SOURCE STATEMENT

                   52 ;
                   53 ;************************************************
                   54 ;
)037 OD            55 NERMSG: DB      ODH,OAH,' NOT ERASED',0
)038 OA
)039 204E4F54
)03D 20455241
)041 534544
)044 00
                   56 ;
)045 OD            57 YERMSG: DB      ODH,OAH,'ERASED',0
)046 OA
)047 45524153
)04B 4544
)04D 00
                   58 ;
                   59 ;************************************************
                   60 ;
                   61         END

BLIC SYMBOLS
MND  C 0000

TERNAL SYMBOLS
     E 0000    ECHO   E 0000    EPEND  E 0000    GETCH  E 0000
     E 0000    MESOUT E 0000    READBT E 0000    REMBYT E 0000
NDBY E 0000    TGETSC E 0000    TYPE   E 0000    WAIT   E 0000

ER SYMBOLS
     C 002D    C3     C 0017    C4     C 0030    C5     C 001A
     C 0009    CR     E 0000    ECHO   E 0000    ECMND  C 0000
END  E 0000    GETCH  E 0000    LF     E 0000    MESOUT E 0000
RMSG C 0037    READBT E 0000    REMBYT E 0000    STNDBY E 0000
ETSC E 0000    TYPE   E 0000    WAIT   E 0000    YERMSG C 0045

SEMBLY COMPLETE.  NO ERRORS
```

```
180 MPP50A.001 NOOBJECT PRINT(:LP:)

:S-II 8080/8085 MACRO ASSEMBLER. V2.0          MMLOAD    PAGE     1

.OC  OBJ           SEQ          SOURCE STATEMENT

                     1 $PAGEWIDTH(80)
                     2 :*****************************************************
                     3 :*                                                   *
                     4 :*          MULTIPURPOSE PROGRAMMER                  *
                     5 :*          MPP TO MDS DATA LOADING                  *
                     6 :*                                                   *
                     7 :* PROG. NO:200-MM-50          DATE    :26.06.1984 *
                     8 :* FILE    :MM050A.001         MODULE :MMLOAD      *
                     9 :* AUTHOR  :E.DURMUS                                *
                    10 :*                                                   *
                    11 :*****************************************************
                    12 :
                    13          NAME    MMLOAD
                    14 :
                    15          PUBLIC  BCMD
                    16 :
                    17          EXTRN   GETNM.CO
                    18 :
                    19          CSEG
                    20 :*****************************************************
                    21 :*PROCEDURE:BCMD
                    22 :*FUNCTION :ENCODE HEX DATA INTO HEXADECIMAL FORMAT
                    23 :*           AND WRITE ON THE MDS SYSTEM
                    24 :*INPUTS    :NONE
                    25 :*OUTPUTS   :NONE
                    26 :*CALLS     :GETNM.CO.HILO.WBYTE.WADR
                    27 :*DESTROYS  :ALL
                    28 :*
0000 0E02           29 BCMD:    MVI     C.2       :GET ADDRESS RANGE
0002 CD0000    E    30          CALL    GETNM
0005 D1             31          POP     D         :DE=HIGH ADDRESS
0006 E1             32          POP     H         :HL=LOW ADDRESS
                    33 WR0:
0007 0E3A           34          MVI     C.':'     :EMIT RECORD MARK
0009 CD0000    E    35          CALL    CO
000C 011000         36          LXI     B.16      :INITIALIZE B:=0. C:=16
                    37 :-------------------------------------------------
000F E5             38          PUSH    H         :SAVE HL
0010 04             39 WR1:     INR     B         :INCREMENT RECORD LENGHT
0011 0D             40          DCR     C
0012 CA1B00    C    41          JZ      WR2       :TERMINATE ON COUNT OF 16 BY
0015 CD9200    C    42          CALL    HILO      :OR END OF RANGE
0018 D21000    C    43          JNC     WR1       :WHICHEVER OCCURS FIRST
                    44 :-------------------------------------------------
                    45 WR2:                       :OUTPUT A DATA RECORD
001B E1             46          POP     H         :RESTORE HL=LOW ADDRESS
001C D5             47          PUSH    D         :SAVE HIGH ADDRESS
001D 1600           48          MVI     D.0       :INITIALIZE CHECKSUM=0
001F 78             49          MOV     A.B       :A=RECORD LENGHT
0020 CD7200    C    50          CALL    WBYTE     :EMIT RECORD LENGHT
0023 CD6A00    C    51          CALL    WADR      :EMIT HL:=LOW ADDRESS
0026 AF             52          XRA     A
```

```
 OC  OBJ          SEQ          SOURCE STATEMENT

 027 CD7200   C    53           CALL    WBYTE   :EMIT RECORD TYPE(DATA)
                   54 WR3:
 02A 7E            55           MOV     A.M     :FETCH  DATA
 02B CD7200   C    56           CALL    WBYTE   :EMIT   IT
 02E 23            57           INX     H       :INCREMENT MEMORY ADDRESS
 02F 05            58           DCR     B       :DECREMENT COUNT
 030 C22A00   C    59           JNZ     WR3     :LOOP UNTIL ENTIRE RECORD HAS
 033 AF            60           XRA     A       :          BEEN OUTPUT
 034 92            61           SUB     D       :D CONTAINS RUNNING CHECKSUM
 035 CD7200   C    62           CALL    WBYTE   :EMIT CHECKSUM:=-D
 038 D1            63           POP     D       :RESTORE DE:=HIGH ADDRESS
 039 2B            64           DCX     H       :BACK UP MEMORY POINTER
 03A 0E0D          65           MVI     C.0DH   :PUNC CARRIAGE RETURN
 03C CD0000   E    66           CALL    CO
 03F 0E0A          67           MVI     C.0AH   :PUNCH LINE FEED
 041 CD0000   E    68           CALL    CO
 044 CD9200   C    69           CALL    HILO    :TEST FOR TERMINATION
 047 D20700   C    70           JNC     WR0     :IF NOT DONE FORM NEXT RECORD
                   71 :.....................................................
                   72 :          END OF FILE RECORD
                   73 :
 04A 0E3A          74           MVI     C.':'   :EMIT RECORD MARK
 04C CD0000   E    75           CALL    CO
 04F 3E00          76           MVI     A.0     :RECORD LENGHT:=0
 051 CD7200   C    77           CALL    WBYTE   :EMIT IT
 054 210000        78           LXI     H.0     :LOAD ADDRESS:=0
 057 CD6A00   C    79           CALL    WADR    :EMIT IT
 05A 3E01          80           MVI     A.01    :RECORD TYPE=1 (END OF FILE)
 05C CD7200   C    81           CALL    WBYTE   :EMIT IT
 05F 0E0D          82           MVI     C.0DH   :CARRIAGE RETURN
 061 CD0000   E    83           CALL    CO      :EMIT IT
 064 0E0A          84           MVI     C.0AH   :LINE FEED
 066 CD0000   E    85           CALL    CO      :EMIT IT
 069 C9            86           RET
                   87 :
                   88 :*****************************************************
                   89 :*
                   90 :*PROCEDURE:WADR
                   91 :*FUNCTION :PUNCH CONTENTS OF HL IN HEX ON MDS
                   92 :*INPUTS   :HL:=8 BIT LOAD ADDRESS
                   93 :*OUTPUTS  :
                   94 :*CALLS    :WBYTE
                   95 :*DESTROYS :A
                   96 :*
 06A 7C            97 WADR:   MOV     A.H     :A:=MSB OF LOAD ADDRESS
 06B CD7200   C    98         CALL    WBYTE   :EMIT FRAMES 3&4
 06E 7D            99         MOV     A.L     :A:=LSB OF LOAD ADDRESS
 06F C37200   C   100         JMP     WBYTE   :EMIT FRAMES 5&6
                  101 :*****************************************************
                  102 :*
                  103 :*PROCEDURE:WBYTE
                  104 :*FUNCTION :PUNCH A BYTE TO MDS AS 2 ASCII CHARACTERS
                  105 :*INPUTS   :A=BYTE TO BE CONVERTED, D=RUNNING CHECKSUM
                  106 :*OUTPUTS  :D=UPDATED CHECKSUM
                  107 :*CALLS    :CONV.CO
```

```
LOC  OBJ          SEQ         SOURCE STATEMENT

                   108 ;*DESTROYS ;A.F.D.E
                   109 ;
                   110 WBYTE:
0072 5F            111         MOV     E.A     ;SAVE BYTE TO BE CONVERTED
0073 0F            112         RRC             ;    IN E-REG
0074 0F            113         RRC
0075 0F            114         RRC
0076 0F            115         RRC             ;LOOK ONLY AT 4 MSB OF THE BYT
0077 CD8800    C   116         CALL    CONV    ;CONVERT IT TO 1 ASCII CHARAC.
007A CD0000    E   117         CALL    CO      ;PUNCH  IT
007D 7B            118         MOV     A.E     ;NOW LOOK ONLY AT 4 LSB OF BYT
007E CD8800    C   119         CALL    CONV    ;CONVERT IT TO ONE ASCII CHARA
0081 CD0000    E   120         CALL    CO      ;PUNCH  IT
0084 7B            121         MOV     A.E
0085 82            122         ADD     D       ;UPDATA THE RUNNING CHECKSUM
0086 57            123         MOV     D.A     ;STORE IT BACK IN THE D REG
0087 C9            124         RET
                   125 ;
                   126 ;*****************************************************
                   127 ;*
                   128 ;*PROCEDURE:CONV
                   129 ;*FUNCTION :CONVERT 4 BIT HEX VALUE TO ASCII CHARACTER
                   130 ;*INPUTS   :A:=HEX DIGIT(0....F)
                   131 ;*OUTPUTS  :C:=ASCII    (30H....39H.41H...46H)
                   132 ;*CALLS    :NONE
                   133 ;*DESTROYS :A.C
                   134 ;*
                   135 CONV:
0088 E60F          136         ANI     0FH ;ONLY 4 LSB ARE SIGNIFICANT.SO MAS
                       4 MSB
008A C690          137         ADI     90H     ;SET UP A REG SO THAT A-F CAUS
                       CARRY
008C 27            138         DAA
008D CE40          139         ACI     40H     ;ADD IN CARRY AND ADJUST UPPER
                       IPPLE
008F 27            140         DAA
0090 4F            141         MOV     C.A     ;STORE CONVERTED RESULT IN C-
0091 C9            142         RET
                   143 ;
                   144 ;*****************************************************
                   145 ;*
                   146 ;*PROCEDURE:HILO
                   147 ;*FUNCTION :COMPARE HL WITH DE
                   148 ;*INPUT    :ADDRESS VALUES IN HL AND DE
                   149 ;*OUTPUTS  : IF HL(= DE THEN CARRY 0
                   150 ;*            IF HL)  DE THEN CARRY 1
                   151 ;*CALLS    :NONE
                   152 ;*DESTROYS :A.F.HL
                   153 ;*
0092 23            154 HILO:   INX     H       ;INCREMENT HL ADDRESS
0093 7C            155         MOV     A.H     ;TEST FOR HL=0
0094 B5            156         ORA     L       ;ZERO BIT SET IF H=L=0
0095 37            157         STC             ;CARRY:=1
0096 C8            158         RZ
```

```
)C  OBJ          SEQ          SOURCE STATEMENT

)97 7B            159          MOV     A,E     :DE-HL. SET/RESET CARRY
)98 95            160          SUB     L       :(LSB OF HIGH ADR)-(LSB OF LOW
                  DR)
)99 7A            161          MOV     A,D
)9A 9C            162          SBB     H       :(MSB OF HIGH ADR)-(MSB OF LOW
                  DR)
)9B C9            163          RET
                  164 :
                  165 :*************************************************
                  **
                  166 :
                  167          END

LIC SYMBOLS
D   C 0000

ERNAL SYMBOLS
    E 0000    GETNM  E 0000

R SYMBOLS
D   C 0000    CD     E 0000    CONV   C 0088    GETNM  E 0000
D   C 0092    WADR   C 006A    WBYTE  C 0072    WR0    C 0007
    C 0010    WR2    C 001B    WR3    C 002A

EMBLY COMPLETE.  NO ERRORS
```

```
30 MPP60A.001 NOOBJECT PRINT(:LP:)

8-II 8080/8085 MACRO ASSEMBLER. V2.0          PPLOAD    PAGE     1

DC   OBJ           SEQ          SOURCE STATEMENT

                     1 $PAGEWIDTH(80)
                     2 :*****************************************************
                     3 :*                                                   *
                     4 :*              MULTIPURPOSE PROGRAMMER              *
                     5 :*              MDS TO MPP DATA LOADING              *
                     6 :*                                                   *
                     7 :* PROG.NO:200-MM-60      DATE:26.06.1984            *
                     8 :* FILE    :MM060.001     MODULE:PPLOAD              *
                     9 :* AUTHOR :E.DURMUS                                  *
                    10 :*                                                   *
                    11 :*****************************************************
                    12 :
                    13           NAME    PPLOAD
                    14 :
                    15           PUBLIC  LCMD
                    16 :
                    17           EXTRN   GETNM.ADRD
                    18           EXTRN   MESOUT
                    19
                    20           DSEG
                    21 :*
                    22 :*****************************************************
                    23 :*
0018                24 USART    EQU     018H
0019                25 USARTC   EQU     USART+1
00C0                26 BUFBEG   EQU     0C0H
0002                27 HBUF:    DS      2
0002 0000           28 CODEAD:  DW      0
0002                29 RBYTE:   DS      2
                    30 :
                    31           CSEG
                    32 :
                    33 :*****************************************************
                    34 :*PROCEDURE :LCMD
                    35 :*FUNCTION  :READS AN HEX FILE FROM AN EXTERNAL
                    36 :*           SERIAL DEVICE AND LOADS IT INTO THE
                    37 :*           LOCATIONS SPECIFIED BY THE ADDRESS FIELS
                    38 :*           IN THE HEXADECIMAL RECORDS
                    39 :*INPUTS    :NONE
                    40 :*OUTPUTS   :NONE
                    41 :*CALLS     :RMDS.BYTE1.MESOUT ADRD
                    42 :*DESTROYS  :ALL
                    43 :
0000 0E01           44 LCMD:    MVI     C,1
0002 CD0000    E    45           CALL    GETNM
0005 E1             46           POP     H
0006 220400    D    47           SHLD    RBYTE    :GET LOCATION ADR.
0009 CDA700    C    48           CALL    INUSRT   :INITIALIZE USART
000C CD8500    C    49 LABEL1:  CALL    RMDS     :READ AN ASCII CHR.FROM
000F FE3A           50           CPI     ':'      :MDS AND COMPARE IT WITH':'
0011 C20C00    C    51           JNZ     LABEL1
0014 0E00           52           MVI     C.0      :INITIALIZE CHECKSUM
```

```
LOC  OBJ          SEQ          SOURCE STATEMENT

0016 CD6300   C    53           CALL    BYTE1    :GET THE FIRST HEX BYTE
0019 47            54           MOV     B,A      :(RECORD LENGTH) IN (B)
001A CD6300   C    55           CALL    BYTE1    :GET THE SECOND BYTE AND
001D 67            56           MOV     H,A      :THIRD BYTE (CODE ADR)
001E CD6300   C    57           CALL    BYTE1    :IN (HL)
0021 6F            58           MOV     L,A
0022 220200   D    59           SHLD    CODEAD   :SAVE IT
0025 CD6300   C    60           CALL    BYTE1    :GET RECORD TYPE
0028 CA2C00   C    61           JZ      LABEL2   :IF DATA LINE SKIP
002B C9            62           RET
002C CD6300   C    63 LABEL2:   CALL    BYTE1    :GET DATA
002F E5            64           PUSH    H        :SAVE CODE ADR
0030 F5            65           PUSH    PSW      :SAVE DATA
0031 EB            66           XCHG             :GET CODE ADR. INTO DE
0032 2A0400   D    67           LHLD    RBYTE    :GET LOAD ADR
0035 7B            68           MOV     A,E
0036 95            69           SUB     L
0037 6F            70           MOV     L,A
0038 7A            71           MOV     A,D
0039 9C            72           SBB     H
003A C6C0          73           ADI     BUFBEG
003C 67            74           MOV     H,A
003D F1            75           POP     PSW
003E 77            76           MOV     M,A      :STORE DATA INTO BUFFER
003F E1            77           POP     H
0040 23            78           INX     H        :INCREMENT MEMORY ADR
0041 05            79           DCR     B        :DECREMENT RECORD LENGT COUNT
0042 C22C00   C    80           JNZ     LABEL2   :IF NOT END OF RECORD.GET NEX
                      ATA
0045 CD6300   C    81           CALL    BYTE1    :READ .CHECKSUM BYTE
0048 79            82           MOV     A,C
0049 FE00          83           CPI     0
004B CA0C00   C    84           JZ      LABEL1   :EXIT IF CHECKSUM IS OK
004E 21BB00   C    85           LXI     H,MSGCSE:WRITE CHECKSUM EROR
0051 CD0000   E    86           CALL    MESOUT   :MESSAGE
0054 2A0200   D    87           LHLD    CODEAD
0057 CD0000   E    88           CALL    ADRD
005A 21D800   C    89           LXI     H,MSGEND
005D CD0000   E    90           CALL    MESOUT
0060 C30C00   C    91           JMP     LABEL1
                   92 :*
                   93 :*****************************************************
                   94 :*
                   95 :*PROCEDURE  :BYTE1
                   96 :*FUNCTION   :BYTE1 READS 2 ASCII CHARACTERS FROM MDS
                   97 :*            CONVERTS THE CHARACTERS TO ONE HEADECIMA
                   98 :*            CHARACTER AND UPDATES CHECKSUM
                   99 :*INPUTS     :(HL)=POINTER TO BUFFER
                  100 :*           :(A)=VALUE IN BINARY
                  101 :*           :(C)=UPDATED VALUE OF CHECKSUM
                  102 :*CALLS      :RMDS,SUB1,SUB2
                  103 :*DESTROYS   :NONE
0063 E5           104 BYTE1:  PUSH    H
0064 C5           105         PUSH    B
0065 51           106         MOV     D,C      :SAVE CHECKSUM IN REG D
```

```
LOC  OBJ          SEQ          SOURCE STATEMENT

0066 210000   D    107          LXI     H.HBUF
0069 CD8500   C    108          CALL    RMDS    :READ ONE ASCII CHR.FROM MDS
006C 77            109          MOV     M.A
006D 23            110          INX     H
006E CD8500   C    111          CALL    RMDS    :READ NEXT ASCII CHR.
0071 77            112          MOV     M.A
0072 2B            113          DCX     H
0073 2B            114          DCX     H
0074 CD9500   C    115          CALL    SUB1    :GET UPPER BYTE
0077 47            116          MOV     B.A
0078 CD9D00   C    117          CALL    SUB2    :GET COWER BYTE
007B B0            118          ORA     B
007C 47            119          MOV     B.A     :SAVE DATA IN REG B
007D 82            120          ADD     D       :UPDATE CHECKSUM
007E 57            121          MOV     D.A     :SAVE CHECKSUM IN REG D
007F 78            122          MOV     A.B
0080 B7            123          ORA     A
0081 C1            124          POP     B
0082 4A            125          MOV     C.D     :GET CHECKSUM INTO REG C
0083 E1            126          POP     H
0084 C9            127          RET
                   128 :*
                   129 :*****************************************************
                       ***
                   130 :*PROCEDURE :RMDS
                   131 :*FUNCTION  :READS ASCII CHARACTER FROM MDS
                   132 :*INPUTS    :NONE
                   133 :*OUTPUTS   :(A)=ASCII CHARACTER
                   134 :*CALLS     :NONE
                   135 :*DESTROYS  :
                   136 :*
0085 CD8B00   C    137 RMDS:   CALL    LCI
0088 E67F          138         ANI     7FH
008A C9            139         RET
                   140 :*
                   141 :*****************************************************
                   142 :*
                   143 :*PROCEDURE :LCI
                   144 :*FUNCTION  :WAIT AND RECEIVE A CHARACTER FROM MDS
                   145 :*INPUTS    :NONE
                   146 :*OUTPUTS   :(A)=CHARACTER FROM MDS
                   147 :*CALLS     :NONE
                   148 :*DESTROYS  :A
                   149 :*
008B DB19          150 LCI:    IN      LOW USARTC
008D E602          151         ANI     002H
008F CA8B00   C    152         JZ      LCI
0092 DB18          153         IN      LOW USART
0094 C9            154         RET
                   155 :*
                   156 :*****************************************************
                   157 :*
                   158 :*PROCEDURE :SUB1
                   159 :*FUNCTION  :CONVERT ASCII-HEX CHARACTER TO UPPER NI
                   160 :*INPUTS    :(HL)=POINTS TO CHARACTER 1
```

```
LOC  OBJ           SEQ          SOURCE STATEMENT

                    161 :*OUTPUTS    :(A)=HEX VALUE IN UPPER NIBBLE (00-F0)
                    162 :*CALLS      :SUB2
                    163 :*DESTROYS   :A
0095 CD9D00    C    164 SUB1:    CALL     SUB2
0098 07             165          RLC
0099 07             166          RLC
009A 07             167          RLC
009B 07             168          RLC
009C C9             169          RET
                    170 :*
                    171 :*****************************************************
                        **
                    172 :*PROCEDURE :SUB2
                    173 :*FUNCTION   :CONVERT ASCII-HEX CHARACTER TO LOWER NI
                    174 :*INPUTS     :(HL)=POINTS TO CHARACTER 2
                    175 :*OUTPUTS    :(A)=HEX VALUE IN LOWER NIBBLE (00-0F)
                    176 :*CALLS      :NONE
                    177 :*DESTROYS   :A
009D 23             178 SUB2:    INX      H
009E 7E             179          MOV      A,M
009F D630           180          SUI      '0'
00A1 FE0A           181          CPI      10
00A3 F8             182          RM
00A4 D607           183          SUI      7
00A6 C9             184          RET
                    185 :
                    186 :*****************************************************
                    187 :*
                    188 :*PROCEDURE:INUSRT
                    189 :*FUNCTION:INITIALIZE COMPUTER INTERFACE USART
                    190 :
00A7 AF             191 INUSRT: XRA      A
00A8 D319           192          OUT      LOW USARTC
00AA D319           193          OUT      LOW USARTC
00AC D319           194          OUT      LOW USARTC
00AE 3E40           195          MVI      A,40H
00B0 D319           196          OUT      LOW USARTC
00B2 3ECF           197          MVI      A,0CFH
00B4 D319           198          OUT      LOW USARTC
00B6 3E25           199          MVI      A,25H
00B8 D319           200          OUT      LOW USARTC
00BA C9             201          RET
                    202 :*
                    203 :*****************************************************
                        **
                    204 :*
00BB 0D             205 MSGCSE: DB       0DH,0AH,'CHECKSUM ERROR (LINE ADD.='
00BC 0A
00BD 43484543
00C1 4B53554D
00C5 20455252
00C9 4F522028
00CD 4C494E45
00D1 20414444
00D5 2E3D
```

```
LOC  OBJ          SEQ          SOURCE STATEMENT

00D7 00
00D8 3E            206 MSGEND: DB      ')',0DH,0
00D9 0D
00DA 00
                   207 ;*
                   208 ;*****************************************************
                       **
                   209 ;*
                   210      END
```

```
JBLIC SYMBOLS
CMD    C 0000

XTERNAL SYMBOLS
DRD    E 0000     GETNM   E 0000     MESOUT E 0000

SER SYMBOLS
DRD    E 0000     BUFBEG A 00C0     BYTE1  C 0063     CODEAD D 0002
ETNM   E 0000     HBUF   D 0000     INUSRT C 00A7     LABEL1 C 000C
ABEL2  C 002C     LCI    C 008B     LCMD   C 0000     MESOUT E 0000
SGCSE  C 00BB     MSGEND C 00D8     RBYTE  D 0004     RMDS   C 0085
UB1    C 0095     SUB2   C 009D     USART  A 0018     USARTC A 0019

SSEMBLY COMPLETE.  NO ERRORS
```

```
3M80 MPP70A.001 NOOBJECT PRINT(:LP:)

3IS-II 8080/8085 MACRO ASSEMBLER. V2.0          GPPROG    PAGE     1

LOC  OBJ          SEQ          SOURCE STATEMENT

                     1 $PAGEWIDTH(80)
                     2 :*****************************************************
                     3 :*                                                   *
                     4 :*         MULTIPURPOSE PROGRAMMER                   *
                     5 :*           GANG PROGRAM COMMEND                    *
                     6 :*                                                   *
                     7 :* PROG.NO :200-MM-70      DATE  :26-06-1984        *
                     8 :* FILE    :MPP70A.001     MODULE:PPPROG            *
                     9 :* AUTHOR  :E.DURMUS                                 *
                    10 :*                                                   *
                    11 :*****************************************************
                    12 :*
                    13          NAME    GPPROG
                    14 :
                    15          PUBLIC  ACMND.MGTSOC
                    16 :
                    17          EXTRN   MODYCS.BUFBEG.ROMBEG.EPEND.ROMEND
                    18          EXTRN   TRANAL.GETCH.CR.ECHO.MESOUT.BUFEND
                    19          EXTRN   PCOPY.CSMODE.SNDLED.MDYSOC.WAIT
                    20 :
                    21          CSEG
                    22 :*****************************************************
                    23 :*PROCEDURE :MCMND
                    24 :*FUNCTION  :PROGRAM EPROM
                    25 :*INPUTS    :NONE
                    26 :*OUTPUTS   :NONE
                    27 :*CALLS     :MODYCS.MGTSOC.TRANAL.GETCH.ECHO.MESOUT
                    28 :*DESTROYS  :ALL
                    29 :*
0000 3EFE           30 ACMND:  MVI     A.0FEH  :TRANSFER MASTER EPROM TO BU
0002 CD0000    E    31         CALL    MODYCS
0005 210080         32         LXI     H.8000H
0008 220000    E    33         SHLD    BUFBEG
000B 210000         34         LXI     H.0
000E 220000    E    35         SHLD    ROMBEG
0011 3A0000    E    36         LDA     EPEND
0014 3D             37         DCR     A
0015 67             38         MOV     H.A
0016 2EFF           39         MVI     L.0FFH
0018 220000    E    40         SHLD    ROMEND
001B CD0000    E    41         CALL    TRANAL
                    42 :
001E CD4F00    C    43         CALL    MGTSOC  :GET WHICH EPROMS ARE PROGRA
0021 CD0000    E    44         CALL    WAIT
0024 0E3E           45         MVI     C.')'
0026 CD0000    E    46         CALL    ECHO
0029 216900    C    47         LXI     H.GPRMSG
002C CD0000    E    48         CALL    MESOUT
                    49 :
002F 210080         50         LXI     H.8000H :PROGRAM  SLAVE EPROMS FROM
                       ER
0032 220000    E    51         SHLD    BUFBEG
```

```
LOC  OBJ          SEQ          SOURCE STATEMENT

0035 3A0000   E     52           LDA     EPEND
0038 3D             53           DCR     A
0039 57             54           MOV     D.A
003A 1EFF           55           MVI     E.OFFH
003C 19             56           DAD     D
003D 220000   E     57           SHLD    BUFEND
0040 210000         58           LXI     H.0
0043 220000   E     59           SHLD    ROMBEG
0046 3A0000   E     60           LDA     CSMODE
0049 CD0000   E     61           CALL    SNDLED
004C C30000   E     62           JMP     PCOPY
                    63 :
                    64 :************************************************
                    65 :*
                    66 :*PROCEDURE:MGTSOC
                    67 :*FUNCTION :OBTAINS WHIC EPROMS ARE PROGRAMMED
                    68 :*INPUTS   :NONE
                    69 :*OUTPUTS  :NONE
                    70 :*CALLS    :GETCH.ECHO.MODYCS
                    71 :*DESTROYS :ALL
                    72 :
004F CD0000   E     73 MGTSOC: CALL     GETCH
0052 CD0000   E     74         CALL     ECHO
0055 79             75         MOV      A.C
0056 3C             76         INR      A
0057 320000   E     77         STA      CSMODE
005A 3D             78         DCR      A
005B 216500   C     79         LXI      H.MGETBL
005E CD0000   E     80         CALL     MDYSOC
0061 CD0000   E     81         CALL     MODYCS
0064 C9             82         RET
                    83 :
0065 FD             84 MGETBL: DB       OFDH.OF9H.OF1H.OE1H
0066 F9
0067 F1
0068 E1
                    85 :
                    86 :************************************************
                    87 :
0069 47414E47       88 GPRMSG: DB       'GANG PROG'.0
006D 20504524F
0071 47
0072 00
                    89 :
                    90         END

UBLIC SYMBOLS
CMND   C 0000     MGTSOC C 004F

XTERNAL SYMBOLS
UFBEG E 0000     BUFEND E 0000     CR      E 0000     CSMODE E 0000
CHO   E 0000     EPEND  E 0000     GETCH   E 0000     MDYSOC E 0000
ESOUT E 0000     MODYCS E 0000     PCOPY   E 0000     ROMBEG E 0000
OMEND E 0000     SNDLED E 0000     TRANAL  E 0000     WAIT   E 0000

SER SYMBOLS
```

```
MND   C 0000    BUFBEG E 0000    BUFEND E 0000    CR      E 0000
MODE  E 0000    ECHO   E 0000    EPEND  E 0000    GETCH   E 0000
RMSG  C 0069    MDYSOC E 0000    MESOUT E 0000    MGETBL  C 0065
TSOC  C 004F    MODYCS E 0000    PCOPY  E 0000    ROMBEG  E 0000
MEND  E 0000    SNDLED E 0000    TRANAL E 0000    WAIT    E 0000

SEMBLY COMPLETE.  NO ERRORS
```

```
M80 MPP80A.001 NOOBJECT PRINT(:LP:)

IS-II 8080/8085 MACRO ASSEMBLER. V2.0          GPPVER    PAGE     1

LOC  OBJ          SEQ          SOURCE STATEMENT

                     1 $PAGEWIDTH(80)
                     2 :*************************************************
                     3 :*                                               *
                     4 :*         MULTIPURPOSE PROGRAMMER               *
                     5 :*           GANG PROGRAM VERIFY                 *
                     6 :*                                               *
                     7 :* PROG.NO :200-MM-80      DATE  :26.06.1943     *
                     8 :* FILE    :MPP80A.001     MODULE:PPPROG         *
                     9 :* AUTHOR  :E.DURMUS                            *
                    10 :*                                               *
                    11 :*************************************************
                    12 :*
                    13          NAME    GPPVER
                    14 :
                    15          PUBLIC  BCMND
                    16 :
                    17          EXTRN   MODYCS.BUFBEG.BUFEND.ROMBEG.ROMEND
                    18          EXTRN   EPEND.TRANAL.GETCH.ECHO.CSMODE
                    19          EXTRN   PVERF.WAIT
                    20 :
                    21          CSEG
                    22 :*************************************************
                    23 :*PROCEDURE  :BCMND
                    24 :*FUNCTION   :VERIFY MASTER EPROM WIT THE SLAVES
                    25 :*INPUTS     :NONE
                    26 :*OUTPUTS    :NONE
                    27 :*CALLS      :MODYCS.TRANAL.GETCH.ECHO
                    28 :*DESTROYS   :ALL
                    29 :*
0000 3EFE           30 BCMND:   MVI     A.0FEH  :TRANSFER MASTER EPROM CONTEN
0002 CD0000    E    31          CALL    MODYCS  :TO BUFFER
0005 210080         32          LXI     H.8000H :BUFFER STR ADR=C000H
0008 220000    E    33          SHLD    BUFBEG
000B 210000         34          LXI     H.0     :EPROM STR ADR=0000H
000E 220000    E    35          SHLD    ROMBEG
0011 3A0000    E    36          LDA     EPEND   :CALCULATE EPROM END ADR FRON
0014 3D             37          DCR     A       :TYPE
0015 67             38          MOV     H.A     :EPROM END ADR EQU (B)=TYPE -
0016 2EFF           39          MVI     L.0FFH  :(C)=0FFH
0018 220000    E    40          SHLD    ROMEND
001B CD0000    E    41          CALL    TRANAL  :TRANSFER MASTER EPROM TO BUF
                    42 :
001E CD0000    E    43          CALL    GETCH   :GET WHICH EPROMS ARE VERIFI
0021 CD0000    E    44          CALL    ECHO    :WITH THE MASTER
0024 79             45          MOV     A.C
0025 3C             46          INR     A       :UPTADE FLAGS
0026 320000    E    47          STA     CSMODE
                    48 :
0029 CD0000    E    49          CALL    WAIT
002C 210080         50          LXI     H.8000H :BUFFER STR ADR=C000H
002F 220000    E    51          SHLD    BUFBEG
0032 3A0000    E    52          LDA     EPEND
```

```
LOC  OBJ          SEQ          SOURCE STATEMENT

0035 3D             53          DCR     A
0036 57             54          MOV     D.A
0037 1EFF           55          MVI     E.OFFH
0039 19             56          DAD     D
003A 220000   E     57          SHLD    BUFEND  :BUFFER END ADR=C000+TYPE -1.
003D 210000         58          LXI     H.0
0040 220000   E     59          SHLD    ROMBEG  :EPROM STR ADR=0H
0043 C30000   E     60          JMP     PVERF   :VERIFY ALL OF THEM
                    61 :
                    62 :*************************************************
                    63 :
                    64          END

BLIC SYMBOLS
MND   C 0000

TERNAL SYMBOLS
FBEG E 0000    BUFEND E 0000    CSMODE E 0000    ECHO   E 0000
END  E 0000    GETCH  E 0000    MODYCS E 0000    PVERF  E 0000
MBEG E 0000    ROMEND E 0000    TRANAL E 0000    WAIT   E 0000

ER SYMBOLS
MND   C 0000    BUFBEG E 0000    BUFEND E 0000    CSMODE E 0000
HO    E 0000    EPEND  E 0000    GETCH  E 0000    MODYCS E 0000
ERF   E 0000    ROMBEG E 0000    ROMEND E 0000    TRANAL E 0000
IT    E 0000

SSEMBLY COMPLETE.  NO ERRORS
```

```
S-II OBJECT LINKER V3.0 INVOKED BY:
):LINK :F4:MON85A.OBJ, :F4:MON85B.OBJ, :F4:MON85C.OBJ, :F4:MON85D.OBJ, &
:4:MON85E.OBJ, :F4:MON85G.OBJ, :F4:MON85H.OBJ, :F4:MON85I.OBJ, :F4:MPP00A.OBJ,
:4:MPP01A.OBJ, :F4:MPP10A.OBJ, :F4:MPP11A.OBJ, :F4:MPP20A.OBJ, :F4:MPP30A.OBJ,
:4:MPP31A.OBJ, :F4:MPP40A.OBJ, :F4:MPP70A.OBJ, :F4:MPP80A.OBJ, &
:4:MPP60A.OBJ TO :F4:MPPMON.LNK MAP PRINT(:LP:)

( MAP OF MODULE MPPMON
ITEN TO FILE :F4:MPPMON.LNK
JLE IS NOT A MAIN MODULE

MENT INFORMATION:
RT    STOP LENGTH REL NAME

              F04H   B  CODE
               7CH   B  DATA

JT MODULES INCLUDED:
4:MON85A.OBJ(MPP85A)
4:MON85B.OBJ(MPP85B)
4:MON85C.OBJ(MPP85C)
4:MON85D.OBJ(MPP85D)
4:MON85E.OBJ(MPP85E)
4:MON85G.OBJ(MPP85G)
4:MON85H.OBJ(MPP85H)
4:MON85I.OBJ(MPP85I)
4:MPP00A.OBJ(PPMAIN)
4:MPP01A.OBJ(UTMAIN)
4:MPP10A.OBJ(PPPROG)
4:MPP11A.OBJ(UTPROG)
4:MPP20A.OBJ(PPVERF)
4:MPP30A.OBJ(PPTRAN)
4:MPP31A.OBJ(UTREAD)
4:MPP40A.OBJ(PPCOMP)
4:MPP70A.OBJ(GPPROG)
4:MPP80A.OBJ(GPPVER)
4:MPP60A.OBJ(PPLOAD)

ESOLVED EXTERNAL NAMES:
MD - REFERENCED IN :F4:MON85A.OBJ(MPP85A)
MND - REFERENCED IN :F4:MPP00A.OBJ(PPMAIN)
```

```
)-11 OBJECT LOCATER V3.0 INVOKED BY:
):LOCATE :F4:MPPMON.LNK TO :F4:MPPMON.LOC DATA(07000H) CODE(0000H) MAP &
JBLICS PRINT(:LP:) COLUMNS(2)

3OL TABLE OF MODULE MPPMON
) FROM FILE :F4:MPPMON.LNK
ITTEN TO FILE :F4:MPPMON.LOC

JE TYPE SYMBOL                                  VALUE TYPE SYMBOL

1H PUB  PORTA1                                  0009H PUB  PORTA2
2H PUB  PORTB1                                  000AH PUB  PORTB2
3H PUB  PORTC1                                  000BH PUB  PORTC2
0H PUB  PPI1                                    0010H PUB  USART1
0H PUB  CR                                      0008H PUB  DDRIN
CH PUB  DDROUT                                  0008H PUB  DISALL
9H PUB  ENAH                                    0009H PUB  ENAL
9H PUB  ENCS                                    0000H PUB  ENDATI
4H PUB  ENDATO                                  0028H PUB  ENLED
AH PUB  LF                                      00D2H PUB  ERROR
7H PUB  EXIT                                    00A4H PUB  GETCM
0H PUB  MONINI                                  0114H PUB  RTAB
3H PUB  ACMD                                    0133H PUB  DCMO5
CH PUB  DCMD                                    012CH PUB  DSUB
AH PUB  FCMD                                    0156H PUB  GCMD
DH PUB  ICMD                                    01A3H PUB  DCMD
9H PUB  SCMD                                    01FBH PUB  TCMD
EH PUB  ECMD                                    045CH PUB  ADRD
CH PUB  CI                                      035EH PUB  CO
6H PUB  CROUT                                   03F8H PUB  ECHO
1H PUB  GETCH                                   0418H PUB  GETNM
8H PUB  MESOUT                                  0461H PUB  NMOUT
CH PUB  PTSER                                   035CH PUB  SPCOUT
6H PUB  TBLSER                                  0533H PUB  BREAK
3H PUB  CNVBN                                   049CH PUB  FRET
EH PUB  GETHX                                   04CEH PUB  HILO
0H PUB  RSTTF                                   04B6H PUB  SRET
4H PUB  STHFO                                   04FAH PUB  STHLF
6H PUB  VALDG                                   0529H PUB  VALDL
7H PUB  DISCLR                                  0534H PUB  DISREF
4H PUB  KEYBOR                                  0661H PUB  INTTIM
2H PUB  ERRMSG                                  0674H PUB  MPPMON
CH PUB  MSGFIN                                  0864H PUB  DEL1M
2H PUB  GETTYP                                  0847H PUB  MDYSOC
FH PUB  MODYCS                                  08A1H PUB  PCSTBL
BH PUB  PDRTBL                                  082EH PUB  PGETSC
1H PUB  CRWAIT                                  096DH PUB  GETADR
DH PUB  PCMND                                   08F2H PUB  PCOPY
5H PUB  PVERF                                   0B1CH PUB  GETDAT
DH PUB  PBPREP                                  0AB4H PUB  PBPROG
9H PUB  PBVERF                                  0A04H PUB  PGPREP
7H PUB  PINBYT                                  09C6H PUB  PMOTAL
EH PUB  PNOBYT                                  097FH PUB  PROGAL
EH PUB  SNDADR                                  0B35H PUB  SNDCS
CH PUB  SNDDAT                                  0B45H PUB  SNDLED
9H PUB  SNMADR                                  0B7FH PUB  CCMND
1H PUB  DISERR                                  0BA8H PUB  VERAL
8H PUB  VMOTAL                                  0C37H PUB  WAIT
8H PUB  TCMND                                   0CA2H PUB  TGETSC
1H PUB  TRANAL                                  0CFFH PUB  RBPREP
0H PUB  RBREAD                                  0CD7H PUB  READBT
BH PUB  REMBYT                                  0D09H PUB  RMBPRB
AH PUB  RMBRED                                  0CBBH PUB  STNDBY
2H PUB  ECMND                                   0D70H PUB  ACMND
```

```
29H PUB  LCMD                                   703EH PUB  BRKFL
36H PUB  LSAVE                                  7030H PUB  MSTAK
38H PUB  PSAVE                                  7030H PUB  SCRTH
3AH PUB  SSAVE                                  703CH PUB  TEMP
3DH PUB  TEMP1                                  7045H PUB  DCLCNT
48H PUB  DISBUF                                 7058H PUB  DISCNT
5DH PUB  KEYBUF                                 7059H PUB  KEYFLG
5EH PUB  BUFBEG                                 7060H PUB  BUFEND
75H PUB  CSMODE                                 7074H PUB  EPEND
8BH PUB  PCMBUF                                 7073H PUB  PCOUNT
89H PUB  PCSBUF                                 7072H PUB  PCTYPE
86H PUB  PDRBUF                                 7071H PUB  PRTYPE
82H PUB  ROMBEG                                 7064H PUB  ROMEND
88H PUB  TYPE
SATISFIED EXTERNAL(0) BCMD
SATISFIED EXTERNAL(1) SCMND
FERENCE TO UNSATISFIED EXTERNAL(0) AT 00F4H
FERENCE TO UNSATISFIED EXTERNAL(1) AT 06FBH

MORY MAP OF MODULE MPPMON
AD FROM FILE :F4:MPPMON.LNK
ITTEN TO FILE :F4:MPPMON.LOC
DULE IS NOT A MAIN MODULE

ART   STOP LENGTH REL NAME

00H  0F03H   F04H  B  CODE
04H  0F0FH    CH   B  STACK
00H  707BH    7CH  B  DATA
7CH  F6BFH  8644H  B  MEMORY
```

BIBLIOGRAPHY

1. Intel Component Data Catalog, 1982.
2. Intel Peripheral Design Handbook, 1979.
3. Motorola Memory Data Manual, 1981-1982
4. Motorola LS TTL Data Manual, 19781.
5. Hitachi Memory Data Manual, 1982.
6. Intellec Series II Microcomputer Development System Boot/Monitor Listing, INTEL Corporation.
7. INTEL 8080/8085 Assembly Language Programming Manual.
8. 8080/8085 Software Design, Larsen,Titus.
9. 8080 Microcomputer Interfacing and Programming, second edition by Peter R. Rony.
10. National TTL Data Book, 1976.
11. National Interface Data Book, 1978.
12. National Linear Data Book, 1978.
13. Texas Instruments The Transistor and Diode Data Book,1973.
14. Texas Instrument The Interface Circuits Data Manual.
15. Integrated Electronics, Millman Halkias.